JIPAM WEB SITE
Dynamic Database System

by
Trevor McGuckin

A four (4) Unit Research Project/Thesis for partial fulfillment of the degree of Master of Science (Mathematics & Computer Science)

January 2001
School of Communications and Informatics
Faculty of Engineering and Science
This thesis contains no material that has been accepted for the award of any other degree or diploma in any University or Tertiary Institute. To the best of my knowledge and belief it contains no material previously published or written by another person, except where reference is made in the text of the thesis.

SIGNATURE ...........................................................................

NAME Trevor McGuckin

DATE 5/2/2001
ABSTRACT

The research project was framed around the need to convert a static page web site into a database driven web site using Microsoft technologies Active Server Pages and SQL server 7.

The methodology used for the software development was the object oriented use case approach of Jacobson.

Although a single software application was originally thought to be the best design strategy, the solution which emerged as being the preferred framework resulted in two software applications being developed.

Thus, the JIPAM Web Site Dynamic Database System is comprised of an application developed in ASP to allow online access to the journal and a second application to maintain the contents of the database which was developed in C++.
ACKNOWLEDGEMENTS

I would like to thank Steven Young, my supervisor, for his patience during the first semester when work commitments affected the amount of time I could find to work on this thesis.

I would also like to dearly thank my wife, Eileen Day. Firstly, I would like to thank her for her editorial effort in particular the work involved in making this thesis look very professional. Secondly, I would like to thank her for her involvement in testing the software towards the end of the development phase. Finally, I would like to thank her for her patience, support and encouragement with this undertaking.
# TABLE OF CONTENTS

Disclaimer ................................................................. ii
Abstract ....................................................................... iii
Acknowledgements ......................................................... iv

## Chapter 1. INTRODUCTION
1.1. The Nature of the Research Problem ............................ 1
1.2. Research Objectives .................................................. 2
1.3. Research Methodology ............................................... 2
   1.3.1. Requirements Model ......................................... 4
   1.3.2. Analysis Model ................................................. 4
   1.3.3. Design Model .................................................. 4
   1.3.4. Implementation Model ....................................... 4
   1.3.5. Test Model ................................................... 4
1.4 Thesis Outline ......................................................... 5

## CHAPTER 2. WEB ENVIRONMENT
2.1 Common Gateway Interface (CGI) ................................ 7
2.2 Internet Server Application Programming Interface (ASAPI) 8
2.3 Active Server Pages (ASP) .......................................... 9
   2.3.1 Visual Basic ASP Environment ............................ 10
   2.3.2 File Uploads .................................................... 12
2.4 Web State ............................................................... 12
   2.4.1 Cookies .......................................................... 13
2.5 Security ............................................................... 14
   2.5.1 Authenticating Users .......................................... 15

## Chapter 3. REQUIREMENTS MODEL
3.1. Problem Definition .................................................. 17
3.2. Scope of the Project ................................................ 17
3.3. Stakeholders ........................................................ 18
3.4. Use Cases ........................................................... 20
   3.4.1. General ......................................................... 20

*JIPAM Web Site: Dynamic Database System*
3.5. ACTOR — GENERAL USER .................................................. 20
3.5.1. USE CASE 1 — Collect Statistics ..................................... 21
3.5.2. USE CASE 2 — Search JIPAM ........................................... 21
3.5.3. USE CASE 3 — Subscribe to JIPAM ................................. 23
3.5.4. USE CASE 4 — Display Current Volume/Issue ................. 23
3.5.5. USE CASE 5 — Display Articles Yet to be Published ........ 24
3.5.6. USE CASE 6 — Display List of Editors .............................. 24
3.5.7. USE CASE 7 — Provide Feedback ...................................... 24
3.5.8. USE CASE 8 — Static Pages ........................................... 24

3.6. ACTOR — SUBSCRIBER .......................................................... 25
3.6.1. USE CASE 9 — Login as a JIPAM Subscriber .................... 25
3.6.2 USE CASE 10 Change Subscriber Details .......................... 26

3.7. ACTOR — AUTHOR ................................................................. 26
3.8. ACTOR — EDITOR ................................................................. 27

3.9. ACTOR — ADMINISTRATOR .................................................. 27
3.9.1. USE CASE 11 — Login as Administrator .......................... 27
3.9.2. USE CASE 12 — Change Default Values ............................ 27
3.9.3. USE CASE 13 — Create a New Article .............................. 28
3.9.4. USE CASE 14 — Change the Article Details ...................... 30
3.9.5. USE CASE 15 — Add a Subscriber ..................................... 31
3.9.6. USE CASE 16 — Change a Subscriber’s Details ................. 31
3.9.7. USE CASE 17 — Delete a Subscriber .................................. 32
3.9.8. USE CASE 18 — Update a Subscriber’s Subscription Date .... 32
3.9.9. USE CASE 19 — Create a New Volume/Issue ..................... 32
3.9.10. USE CASE 20 — Add Articles to a New Volume/Issue ....... 33
3.9.11. USE CASE 21 — Retrieve Usage Statistics ....................... 33
3.9.12. USE CASE 22 — Add a New Institution ........................... 34
3.9.13. USE CASE 23 — Modify an Institution’s Details ............... 34

CHAPTER 4. ANALYSIS MODEL 35

4.1. Use Case Analysis ............................................................... 35
4.1.1. Collect/Retrieve Usage Statistics ....................................... 35
4.1.2. Search JIPAM ................................................................. 36
4.1.3. Subscribe to JIPAM/Login as a JIPAM Subscriber/ 
      Change Subscriber Details .............................................. 37
4.1.4. Display Current Volume/Issue ......................................... 38
4.1.5. Display Articles to be Published .................................... 38
4.1.6. Display List of Editors .................................................. 39
4.1.7. Provide Feedback ........................................................ 39
4.1.8. Static Pages ............................................................... 40
4.1.9. Change Default Values .................................................. 40
4.1.10. Create an Article/Change Article Details ........................ 41
4.1.11. Add/Delete/Change a Subscriber’s Details ...................... 41
4.1.12. Update a Subscriber’s Subscription Date ........................ 42
4.1.13. Create New Volume/Issue .............................................. 42
4.1.14. Add Articles to a New Volume/Issue ............................... 42
4.1.15. Add/Change an Institution ............................................ 43
4.2. Objects and their Responsibilities
   4.2.1. Interface Objects
   4.2.2. Entity Objects
   4.2.3. Control Objects

4.3. Subsystems
   4.3.1. Viewing Subsystem
   4.3.2. Maintenance Subsystem

CHAPTER 5. CONSTRUCTION AND DESIGN

5.1. Analysis Model Entities
   5.1.1. Database Tables
   5.1.2. Default Values
   5.1.3. Statistic Tables

5.2. Viewing Subsystem
   5.2.1. Final Design
   5.2.2. Security
   5.2.3. Interface Objects
   5.2.4. Control Objects

5.3. Maintenance Subsystem
   5.3.1. Administration Program Users Guide
   5.3.2. Database Issues
   5.3.3. Final Design
   5.3.4. Security
   5.3.5. Interface Objects
   5.3.6. Control Objects

5.4. Testing

CHAPTER 6. SUMMARY

REFERENCES

APPENDIX A — Installation CD-ROM
APPENDIX B — SQL Server 7 Programming Code
APPENDIX C — Web Server ASP Code
APPENDIX D — Administration Program C++ Code
APPENDIX E — Administration Program Guide
List of Figures

Figure 1  Jacobson's Five Basic Models (1992, p.113) 3
Figure 2  Authentication with BuiltIn Browser/Server Methods (Homer 1999, p.202) 16
Figure 3  JIPAM Web Site — Stakeholders and Goals 19
Figure 4  Collect/Retrieve Usage Statistics 35
Figure 5  Search JIPAM 36
Figure 6  Subscribe/login to JIPAM 37
Figure 7  Current Volume/Issue 38
Figure 8  Articles Yet to be Published (New Papers) 38
Figure 9  List of Editors 39
Figure 10  Provide Feedback 39
Figure 11  Static Pages 40
Figure 12  Change Default Values 40
Figure 13  Create/Change an Articles Details 41
Figure 14  Add/Delete/Change a Subscriber's Details 41
Figure 15  Update a Subscriber's Subscription Date 42
Figure 16  Create a New Volume/Issue 42
Figure 17  Add Articles to New Volume/Issue 43
Figure 18  Add/Change an Institution 43
Figure 19  Block Diagram of the Viewing Subsystem 47
Figure 20  Block Diagram of the Maintenance Subsystem 47
Figure 21  Database Scheme 49
Figure 22  Usage Statistics Database Tables 51
Figure 23  Viewing Subsystem Final Design 53
Figure 24  Relationship Between the Menu Structure and the ASP Pages 54
Figure 25  Maintenance Subsystem Final Design 61
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Scope of the JIPAM Web Site Dynamic Database System</td>
<td>18</td>
</tr>
<tr>
<td>Table 2</td>
<td>Responsibilities of the Interface Objects</td>
<td>44</td>
</tr>
<tr>
<td>Table 3</td>
<td>Responsibilities of the Entity Objects</td>
<td>45</td>
</tr>
<tr>
<td>Table 4</td>
<td>Responsibilities of the Control Objects</td>
<td>46</td>
</tr>
<tr>
<td>Table 5</td>
<td>Association Between ASP Page Names and Menu Page Names</td>
<td>56</td>
</tr>
<tr>
<td>Table 6</td>
<td>Viewing Subsystem - Interface Objects</td>
<td>57</td>
</tr>
<tr>
<td>Table 7</td>
<td>Viewing Subsystem - Control Objects</td>
<td>58</td>
</tr>
<tr>
<td>Table 8</td>
<td>Maintenance Subsystem Interface Objects</td>
<td>62</td>
</tr>
<tr>
<td>Table 9</td>
<td>Maintenance Subsystem Control Objects</td>
<td>63</td>
</tr>
</tbody>
</table>
1. **INTRODUCTION**

The School of Communication and Informatics within the Victoria University Faculty of Engineering and Science publish an academic journal titled, Journal of Inequalities in Pure and Applied Mathematics (JIPAM). In common with many academic other journals, both within the scientific community and also more broadly across numerous disciplines, the management board of JIPAM have chosen to utilize the opportunities provided by the Internet in terms of dissemination and delivery.

*Scientific publishing on the World Wide Web makes it possible to disseminate new information to a wide audience around the world in a matter of minutes* (Weintraub, 2000, p.57).

*Electronic access and distribution of information have acquired a momentum which will continue to grow and thrive in the future with emerging technologies* (Bandyopadhyay, 1999 p.14).

*Users, across the board, are becoming more savvy. Although the rise in online users varies widely across academic areas, they're all becoming increasingly demanding and knowledgeable* (Fletcher, 1999, p.117).

The first issue of JIPAM was produced in both online and the more traditional paper-based format. According to the Managing Editor, John Roumeliotis, ‘E-publishing lets us compete with major publishing houses at a fraction of the cost. It’s a more flexible medium, with no loss in quality’ (2000). The first generation of the JIPAM e-journal appeared online in early 2000 and it was developed on the basis of static HTML files on a Unix platform. The web site contained basic content with little to no functionality.

1.1. **THE NATURE OF THE RESEARCH PROBLEM**

As part of their strategic planning for the continual development of JIPAM, the Managerial Board wished to take fuller advantage of more of the unique publishing opportunities that the Internet provides for e-journals such as global access combined with the functionality of different levels of access, speed of publication and reduced maintenance costs.
Several problem areas that were directly related to the static nature of the site’s content were identified and are outlined below.

- Maintenance was a particular issue because of the growing number of hyperlinks required to maintain the static content. This leads to an associated problem concerning the effort required to ensure high levels of consistency were maintained throughout the numerous links.
- It was not possible for visitors to the JIPAM site to undertake online searches.
- A desire to be able to track the peer review process of new articles from the initial receipt of the draft article through to its publication.
- Although access to the full text of each article published in JIPAM was currently free, this may change in the future with the application of a fee for full access articles. Thus, the web application needed the functionality of a process whereby users were required to register (or subscribe) before gaining access to the full text articles.

1.2. RESEARCH OBJECTIVES

The overall objective of this research project was to redevelop the current first generation JIPAM e-journal web site into a dynamic web site addressing all of the above problem areas. In broad terms, the research project seeks to …

- create a database framework in which JIPAM articles can be stored, searched on and retrieved, and to also
- develop an ASP front-end to provide dynamic access online to the JIPAM database.

1.3. RESEARCH METHODOLOGY

Unified Modeling Language (UML) formed the underlying methodology used to develop the JIPAM Web Site Dynamic Database System. The UML has evolved from the work of Booch, Jacobson and Rumbaugh and it is a language for specifying, visualising, constructing and documenting software systems. It provides a ‘unified industry-standard graphical language that enables ones to develop software models’ (Vadaparty 2000).

Kobryn has said that UML ‘is the software industry’s dominant modeling language’. He goes on to say that ‘UML is not only a de factor modeling language standard; it is fast becoming a de jure standard’ (Kobryn 1999, p.29).
It also represents a collection of the best engineering practices that have proven successful in the modelling of large and complex systems. However, it is important to note that UML focuses on a standard modelling language, not a standard process. In fact, one of the advantages of UML is that it is independent of particular programming languages and development processes.

Although UML is evolving into a common and standardised methodology, some of the ideas and principles of each of the three originators (Booch, Jacobson and Rambaugh) remain unique.

Jacobson’s ideas about UML, as published in his 1992 book: ‘Object-Oriented Software Engineering’ became the dominant methodological thinking used (with only minor amendments) in this project. Jacobson’s view of UML is that system development is model building (with object modelling the central technique). ‘The complexity needs to be handled in an organized way’ and ‘by introducing the complexity gradually in a specific order in successive models, we are able to manage the system complexity’ (Jacobson 1992, p.113). Each of the models focus on a certain aspect of the system, see Figure 1 below.

![Figure 1. Jacobson's Five Basic Models (1992, p.113)](image)

Hence, the requirements model is the starting point and it is derived from the system’s requirement specification.
1.3.1. Requirements Model

In terms of the methodology followed in this research project, the requirements model was derived from discussions held with both John Roumeliotis (Managing Editor of JIPAM) and Stephen Young (member of the JIPAM management team and thesis supervisor). Use cases were then developed based on their stated requirements that formed the requirements model.

1.3.2. Analysis Model

The analysis model was effectively built directly from the use cases specified in the requirements model where the behaviour specified in the use case descriptions is distributed among three object types (interface, entity and control). The aim of the model is to form a logical and maintainable structure for the system.

1.3.3. Design Model

In Jacobson’s methodology, the design phase is where the analysis model is translated into a model that will operate in the real world. Typically at this point, objects are determined. However, as ASP was the implementation language specified by the client for this project and as it does not support objects, the implementation was completed using the modules derived from the analysis model.

1.3.4. Implementation Model

The implementation model is the actual programming code which is the chosen solution to the problem as defined by the requirements model.

1.3.5. Test Model

Jacobson defines the test model as being ‘developed to support the verification of the developed system’ (1992, p.114). His methodology was followed through user tests to ensure that the system behaved in the same manner as defined in the use cases.
1.4.  THEESIS OUTLINE

This is a very project-based research thesis and extensive documentation of the modeling and development processes involved in the second generation of the JIPAM Web site has been included throughout the thesis.

Chapter one has introduced the research by presenting the nature and scope of the problem being investigated and the methodology followed.

Chapter two defines the Web environment and the constraints and challenges associated with developing a database-driven and dynamic site architecture.

Chapter three begins the redevelopment process using Jacobson’s object-oriented methodology by outlining the requirements model.

Chapter four moves to the analysis model in Jacobson’s development process. Two subsystems emerged from the modeling process — a viewing subsystem and a maintenance subsystem.

Chapter five discusses the design, construction and implementation issues associated with the development of the next generation of the JIPAM web site. Constraints in the application building process are discussed.

Chapter six draws the project to a close and compares the final status of the JIPAM web site with the objectives established in Chapter one.

Also included are the following five appendices.

Appendix A — CD-Rom containing the Second Generation JIPAM Web System

Appendix B — Programming code to set up the database

Appendix C — Programming code associated with implementing the web site

Appendix D — Programming code to implement the maintenance program

Appendix E — Administration Program Guide (50 pages).
2. **WWW Environment**

The World Wide Web (WWW) is an electronic environment where an Internet user runs an application program on their machine called a browser to obtain information from a information provider called a web server (host).

The WWW uses the hypertext transfer protocol (HTTP) which is an 'application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems' (RFC1945). It is based on a request/response paradigm. This paradigm implies a client/server relationship in that a client (request) retrieves information from a server (response). Two versions of the HTTP protocol are in use. HTTP/1.0 protocol has been used since 1990 while HTTP/1.1 is only a recent addition.

HTTP/1.0 does not sufficiently take into consideration the effects of hierarchical proxies, caching, the need for persistent connections, and virtual hosts. In addition, the proliferation of incompletely-implemented applications calling themselves "HTTP/1.0" has necessitated a protocol version change in order for two communicating applications to determine each other's true capabilities (RFC2068).

In the main, it does not matter which of the two protocols are used for the JIPAM web system enhancement project because the IIS server interprets the messages and provides access to the information in the request headers. However, the version 1.1 protocol (RFC2068) is useful for determining information about users of the JIPAM web site.

Once the browser has the address of the site (or a document on the site), a request for the specified document is sent to the site. The layout of the message sent to the host is defined in the HTTP protocol. The host interprets the message and sends a response as an HTML stream that the browser processes. The way that the browser responds to the return message from the host depends on the content of the response message. The response is generally displayed in the client's window but there are conditions when this will not occur. Sometimes, the browser will attempt to save the response as a file or it may start another program that will interpret the response coming from the host.
The layout of the response message is also defined in the HTTP protocol as well as a number of supplementary RFC’s that provide for additional features in the response message.

In the early days of the web, all document content was static. When a request for an individually identified document was received by a host, the document was located and returned back to the client as the response message. This approach was fine for content that did not change very much but it did not allow interactive responses nor did it provide functionality except for static linkages to other web pages. Even though it does have limitations, it is still used today for sites that are only changed very rarely.

However, when information provided by a host needs to be updated or when information is sent from the client to the host for processing, an alternate mechanism was required.

2.1. Common Gateway Interface (CGI)

The first attempt at processing dynamic content occurred when the common gateway interface (CGI) was defined. The CGI interface allowed programs to run on the server allowing the creation of a HTML page that would reflect the result of executing the requested program.

In the CGI interface, the host creates a number of environment variables containing information about the request. If additional information is sent from the client, it is written to the standard input device. The host then starts executing the requested program. The program examines the environment variables and reads the standard input device if appropriate and then writes its reply to the standard output device. The host collects the reply from the output device and returns it to the browser. Both these devices behave in the same way in both a UNIX and a WINDOWS operating environment which means the CGI interface was common to hosts running in either environments.

By using the CGI interface, the host server could interact with third party programs and as a result the nature of the Web changed dramatically. Now content, could be
dynamically created. One such example involves database programs responding to interactive queries of its content.

2.2. **Internet Server Application Programming Interface (ISAPI)**

When Microsoft released their web server program, they also introduced the Internet Server Application Programming Interface (ISAPI) as an alternative mechanism to run third party programs.

They did this because they considered that the CGI process was not very efficient in using resources on the web server running in the Windows operating environment. One example of the inefficiency generally cited is that of a web server that receives a large number of requests for exactly the same program to be run. To understand why this might be a problem requires some knowledge of the way programs run in the Windows Win32 environment. In this environment, there are two quite distinct approaches to starting another program.

- The first approach is to start the other program as a thread. A thread runs in the same address space as the starting process and has access to all its variable and environmental settings. The only additional memory the new thread requires is for its own program stack.

- The other approach is to create a complete instance of the other program. This approach uses the most system resources as each program instance requires its own memory allocation for all its needs including code space, variables, environmental settings and program stack space.

Starting a program as a thread requires far less memory than starting a program as a separate instance. Further, it also takes longer to start a separate process than it does to start a separate thread. One disadvantage of threads is that they must be part of the starting program. This requires the thread programs to be written as dynamic link libraries (DLLs). However, once the DLL is loaded in memory, it remains there until the host program unloads it. As a result, the next time it needs to run, very little time is required for the thread program to start.
Returning to the inefficient scenario cited earlier, it can be seen that it would be much more efficient in a Windows operating environment to run third party programs as threads and therefore gain resource savings on the host.

The Internet server application interface ISAPI was developed as a result of these issues.

Filters were another facility that came with the introduction of ISAPI on Win32 platforms. 'An ISAPI filter is a custom DLL that is in the same address space as the web server and is called by the web server in response to every HTTP request ... The filter then instructs the web server on how to handle the request. ISAPI thus allows you to customize your web server's response to specific types of user requests' (Keyton Weissinger 1999, p.5).

### 2.3. Active Server Pages (ASP)

Active server pages is a Microsoft technology that leverages the installable filter aspect of their ISAPI interface. When a browser requests a file that has an ASP file extension, the web server passes the requested document to the active server page filter ASP.DLL. The ASP.DLL routines process the passed file to produce HTML output that the web server then passes back to the browser.

The ASP file can contain both program code and HTML formatting code. The program code is generally a form of the visual basic code popular on Microsoft platforms. It is interesting to note that the code does not have to be visual basic. It could also be Perl, for example, or any other scripted (interpreted) language. The language chosen depends on how the ASP filter is setup, that is, what scripting language interpreter DLL the ASP DLL should load to process the incoming request. The default setup of the visual basic scripting language will be the only one considered for the JIPAM web system enhancement.

ASP is also not an object oriented language. While it supports the use of objects, these objects are Active X objects installed into the operating system. There is no language support for directly specifying objects using ASP.
2.3.1. The Visual Basic ASP Environment

An ASP page is a visual basic program that can also contain HTML formatting code. The visual basic code in an ASP page is separated from the HTML code by beginning and ending markers, <%% and %%%> respectively. The ASP script interpreter reads the file and executes the visual basic code. When it comes to HTML formatting code, it writes the HTML directly to the output message. It then continues on with the visual basic code program sequence until it reaches either the end of the page or a message terminating the output page.

When a host site is configured to use the visual basic scripting language, there are a number of components/objects (see list below) created automatically in that environment which are accessible by direct reference in ASP pages.

- Application Object
- ObjectContext Object
- Request Object
- Response Object
- Server Object
- Session Object
- Application Object

An IIS web site has a hierarchial directory structure which consists of a base directory called the WWW root directory. A chain of children directories are under this directory and when they are combined they form the web site.

An application in terms of the IIS web server consists of all the documents/files that are accessible to an Internet user from a particular directory and includes all the contents of that directory and its subdirectories. The particular directory is marked by the presence of a file named GLOBAL.ASA.

The Application Object is initialised by the IIS as a result of the presence of the GLOBAL.ASA file the moment the first Internet user accesses a document from directory containing the GLOBAL.ASA file or any of its children. The IIS also runs the application ON_Start event code if that code is present in the file. This event code
can itself initialise variables and other objects that should be visible to all users of the application and persist for the duration of the Application Object. The GLOBAL.ASA file can also contain ON_End event code which is run when the Application Object is stopped. The Application Object can be stopped from the Microsoft Management Console but is more generally stopped when IIS is shut down.

The ObjectContext Object was made available in version 2.0 of Active Server Pages. It enables the concept of a database transaction to be implemented by allowing a transaction script to be created. Thus, it is possible to create one code section of ASP code that removes a record from a database table while another adds a record to another table. If both sections of code complete successfully, the database will be updated permanently with both changes. If either or both fail, the transaction will be rolled back, that is the database would be returned to the state which it was in before either of the two code sections commenced. This transaction support in IIS is based on Microsoft Transaction Server technology which is built into the IIS.

The Request Object is created for every request message that an Internet user sends to the ASP application. It allows program access to the HTTP request header and body without the ASP application having to parse the incoming message.

The Response Object is created to represent the HTML message that will be returned to the Internet user. For every Request Object created, there is a corresponding Response Object. Like the Request Object, it allows programs access to the HTTP header and body of the return message.

The Server Object represents IIS itself and exists for the life of the web server. It provides miscellaneous functions available to all applications and the CreateObject method is the most important. It allows ASP applications to use third party components as though they were part of the ASP environment themselves. A number of these components are provided when the IIS program is installed. The ActiveX Data Object (ADO) is the most important of the components for the JIPAM Web System Enhancement because it allows access to databases.

The Session Object is only created if the ASP application makes reference to the object directly. Its role is to hold data variables that are applicable to an individual
The Session Object for an individual user is identified by its SessionID value. This value is a 32 bit number that the web server generates when the object is created. The SessionID value is sent as a temporary cookie to the client’s browser. The cookie exists only until the client closes their browser software.

2.3.2. File Uploads

When creating a dynamic web program, it would be of benefit if clients could send data from their machines to the web server as this feature would enable clients to update databases dynamically. The functionality to send data from the client machine is built into the HTML language with its <FORM> and </FORM> tags. The original specification for the input element tag included a variety of field formats but it did not include files.

Since then, the HTTP definition and the HTML specification has been extended to allows files to be sent to the web server as part of the request message. This specification is described in RFC1867.

This feature has not been provided as part of the ASP DLL. The extraction of file data from the request message has been left to other third party suppliers to implement.

2.4. Web State

As discussed previously, HTTP is based on a request/response paradigm which unfortunately has a negative side effect.

After a web server receives a message, it formulates and sends back a response. It then promptly forgets that it is has just processed a message. When the server next receives a request (which could, quite possibly. come from the sender of the previous request) and it subsequently responds, it also forgets about that message as well. This sequence of the web server receiving, responding and forgetting is continuous which means that there is no history of previous messages maintained. Thus, it is said that the web server does not maintain state or is stateless.
Being stateless can cause problems. In the scenario where a web site requires customers to pay for access rights, there needs to be some mechanism to identify users who are registered customers. This is a simple example because users could be required to identify themselves as registered customers when they first connect to the site. However, as the web server is stateless and does not remember its message history, it will not recognise the next message as being from the same registered customer. Instead, the incoming message is seen by the server as a new message and the customer must again identify themselves to gain access to the site. The need to continuously identify as registered customers would quickly irritate customers.

However, there is a mechanism in the HTTP 1.0 protocol that provides for persistent connections between the web server and the client software (RFC 1945). Its use is limited though.

The Hypertext Transfer Protocol 1.0 allows client browsers to send Keep-Alive messages to proxy servers. These messages basically tell the proxy server to maintain an open connection with the requesting client. However, these connection requests are often unrecognised by the proxy server. This problem in the proxy server results in a hung connection between the proxy server and the requested web server. In a nutshell, maintaining connections with web servers is prone to error and thus unreliable in HTTP 1.0, still by far the protocol most commonly used by client browsers (Keyton Weissinger 1999, p.122).

A mechanism was needed to provide state information between the sending one page and the next to a client.

2.4.1. Cookies

Netscape Communications Corp developed the term 'cookies' to refer to their Persistent Client State Mechanism that they saw as a way of overcoming the lack of state in the HTTP protocol.

The concept behind cookies is quite simple. They are small pieces of information that the web server could store on the client's machine and are name/value pairs created by the server and included in the body of the response message. A client's browser would then include these name/value pairs in the body of any request message that was then sent to the server originating the message. In this way, the
server could maintain appropriate historical information associated with requests from one client.

However, as well as the name/value pairs associated with a cookie, there are also additional properties included that determine the way the client’s machine should process them.

- **Expires** which tells the browser when it can delete a cookie from the client’s machine. If no date is specified, the cookie lasts for the duration of the current HTTP session.

- **Domain** which tells the client when it should provide the cookie back to the web server. The domain of the web server must agree with the domain setting in the cookie. This ensures that a cookie provided by one web server does not get included in messages to another web server.

- **Path** which tells the client which application on the web server the cookie belongs to. This also ensures that unrelated applications running on the same web server do not receive cookies from another application.

- **Secure** which tells the client browser to provide the cookie if the browser and the web server are using the secure HTTP protocol.

There are two associated issues that have a bearing on the use of cookies. One is that at the current time, all browsers do not support cookies. The second issue relates to the ability for web users to disable cookies in their web browsers. If cookies are disabled in this way, the issue of state arises again.

### 2.5. Web Security

*Security is an issue that most people would just as soon ignore – whether it is on the Web, in your car, at home, or anywhere else. The essence of security is keeping things safe: in the case of the Web, that means information. Safety is defined as many things: keeping the wrong people out, letting the right people in, allowing only certain people to enter or modify certain data elements, determining conditions under which security procedures can be modified, and on and on and on. In practical terms, security often seems to be a barrier or roadblock erected between the user who wants information and the information provider who wants people to have information (Feiler 1999)*

The web server software IIS runs in an NT environment so it is necessary to understand the relation between the web server security system and the NT security system.
In the NT operating system, everything that a user accesses and uses is controlled by the NT security system, for example, a user must be first logged on to gain access to the desktop in a NT environment. The user needs to enter a user account and a password that the security system checks before access is allowed. The level of access is determined by the permissions the NT administrator allocated directly to that user or the groups to which that user has been assigned.

This degree of security is fine if information or services provided on the machine is restricted to a group of individual users. In the web environment, information is available to the global community and allocating individual user names and passwords is not feasible.

An alternative could be to use a guest account but as Homer indicates ‘guest (or anonymous) accounts are generally not a good idea within Windows NT itself, and are usually shunned by network administrators because they can provide opportunities for hackers to log onto the local machine, and then attack the network’ (1998, p.188).

IIS overcomes this issue by creating a user account (IUSR_machinename) with a randomly generated password that it uses when it is installed. When a Internet user requests information from the web server, the web server retrieves the information from the NT environment using the IUSR_machinename account. Thus, the access that IIS is granted through this account can be controlled in the same way that other users of NT are controlled. In the initial setting, this user is made a member of the Guest group and therefore has access to the same resources that the Guest group has.

2.5.1. Authenticating Users

There are situations where anonymous access using the IUSR_machinename account does not provide access to all resources on the NT machine. Alternatively, access to certain parts of the web site may need to be restricted to certain users (for example, those that pay a fee).
There are two ways to restrict or enhance access on the web site. The first, which has already been discussed, concerns the allocation of an individual user account. The second way is to have user programs running under IIS. These programs control access by displaying web pages which accept usernames and passwords.

Homer discusses the first option in great detail and his diagram reproduced above summaries the logical flow of messages that pass between the web server and the user’s browser when trying to access resources not available to the IUSR_machinename account. Most of the time these messages are transparent to the user, especially when using Microsoft’s browser (Version 3 onwards) which has support for the NT security system built in.

The second method is the simplest to implement and it is probably the method of choice when the level of security provided by the NT security system is not needed or the NT security system does not provide the degree of control required. The second scenario (where the NT security system does not provide the degree of control required) arises when database access is involved. The NT security system will determine whether a user has rights to access the resource (database) but after access is given, it is not concerned with the user’s actions regarding the data within the database. In a dynamic web environment, this level of security needs to be provided by the underlying database software.
3. REQUIREMENTS MODEL

The requirements model is built from the perspective of the user and it comprises use cases and their associated actors. A component of the object-oriented approach is the need to be clear about the system boundary with an emphasis on being explicit about the scope and goals of the system.

3.1. PROBLEM DEFINITION

Victoria University publish an electronic journal JIPAM (Journal of Inequities in Pure and Applied Mathematics). Currently, the journal is hosted on a Unix platform and consists of static HTML pages. Each time a new article or volume is placed on the site, a significant maintainence effort is required to update the site. In addition, the site does not provide search capabilities on the article contents because of its static nature.

The editor of the journal wants the site to be easier to maintain. This is to be accomplished by migrating the site to a database solution thereby allowing search capabilities to be provided. He also wants to restrict access to the full text of articles published in JIPAM by requiring users to login before they are able to retrieve the full text. He has a long-term objective of charging users a fee for such access (e.g. access to the full text of the journal articles online).

A further requirement is to use only minimal Javascript as a number of Web users still have relatively primitive browsers. This last requirement also applies to the level of HTML to be used.

The editor also required Microsoft SQL 7 to be used as the target database with Microsoft Internet Information server providing the web functionality.

3.2. SCOPE OF THE PROJECT

It is critical to be clear about the boundaries of the system. Table 1 defines the scope of the JIPAM Web Site Dynamic Database System (Phase 1).
<table>
<thead>
<tr>
<th>YES</th>
<th>NO</th>
<th>ITEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td></td>
<td>Storing user (subscriber) details.</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td>Searching the Journal using a range of criteria.</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td>Accessing the full text of published articles</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td>Storing all published articles</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td>Storing all unpublished articles while undergoing peer review</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td>Maintaining the system</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td>Storing Editor details</td>
</tr>
<tr>
<td>×</td>
<td></td>
<td>Tracking the peer review process for unpublished articles.</td>
</tr>
<tr>
<td>×</td>
<td></td>
<td>Charging a subscription fee to access full text articles</td>
</tr>
</tbody>
</table>

*Table1. Scope of the JIPAM Web Site Dynamic Database System*

In determining these boundaries, all of the user’s requirements have been considered to be within scope except for the last item, that is, the tracking of the peer review process of unpublished articles. This functionality has been considered beyond the range of the current project and it remains to be developed and implemented as a future enhancement of the system.

### 3.3. STAKEHOLDERS

There are six groups of people interested in the system. These groups are defined as being stakeholders in the system and they are described as follows.

1. **General Users** are people from the global community who wish to either browse or search the JIPAM web site as a source of information relating to current mathematics research. As well as browsing through the table of contents for each issue of the journal, they may also be interested in searching the journal using a range of defined criteria (such as author name and/or keyword). The search results provide a summary of specific articles complete with an abstract. However, to access the full text, general users must be registered as subscribers.

2. **Subscribers** are general users who want access to the full text of articles.

3. **Authors** are professionals within the mathematics discipline who have undertaken research and who now seek to publish their research findings.
4. **Editors** are also professionals in the field of mathematics who are generally recognised by others in the international maths community as being leaders in the field.

5. **The JIPAM Web System Administrator** is a member of the Department of Communication and Informatics, Victoria University and has administrator responsibilities as part of their functional duties at the university.

6. **The Managerial Board** seeks to disseminate research results quickly to the international maths community using an efficient and up-to-date publication process which maximises the delivery benefits of Internet technology.

The term 'stakeholders' refers to actual roles that individuals and/or groups play in their interactions with the JIPAM web system. A person can be a member of more than one of these stakeholders' groups, for instance, the author of a specific journal article would most likely also be a subscriber and they may also undertake the role of journal editor. Five of these six groups are direct users of the system with the Managerial Board being primarily involved with strategic concerns rather than operational issues. Figure 3 below provides an overview of the JIPAM Web Site stakeholders and their goals.

Figure 3. JIPAM Web System — Stakeholders and Their Goals
3.4. **USE CASES**

'Use cases provide a way of describing the external view of the system and its interactions with the outside world' (Fowler nd, http://www2.awl.com/cseng/titles/0-201-89542-0/techniques/useCase.htm). According to Jacobson a use case is 'a behaviourly related sequence of transactions' performed by a user 'in a dialogue with the system' (1992, p.127).

Each interaction with the system performed by a user will result in a use case scenario. The combination of all use case scenarios defines the behaviour of the system from the user's point of view.

3.4.1. **General**

In considering use cases for this system, five actors (as described in 3.4) have been defined.

- General User
- Subscriber
- Author
- Editor
- System Administrator

The URL for the JIPAM web site will be generally available to all five actors with some restrictions, that is access to the full text of specific articles will only be available to those general users who login as subscribers.

Access to administration functions will also be restricted with the menu selection for administration actions only becoming available after the System Administrator has logged in as a subscriber.

3.5. **ACTOR — GENERAL USER**

The group 'General Users' includes everyone who visits the JIPAM Journal web site.
3.5.1. **USE CASE 1— Collect Statistics**

Very basic statistics will be collected from everyone who visits the site. The statistics will be extracted from the request header provided by the general user’s browser.

Typical data items could include the visitor’s language, their country of origin and the type of browser the visitor is using.

3.5.2. **USE CASE 2 — Search JIPAM**

General users need to be able to search the JIPAM Journal site using the following five search criteria.

- Search on the author’s name
- Word/s in the article title
- Keyword/s
- Classification code/s
- Any word/s in the article abstract itself

To search, general users will be presented with an entry screen where they enter the search criteria (either the whole text, or part thereof). There will be a submit button on the screen which, when selected, will initiate a search of the database. The results of the search (based on the criteria entered) will then be displayed on the screen.

The search results will appear in the form of a list of the titles of all the articles held in the JIPAM database that match the search criteria specified. In addition to the article titles, the relevant author’s name/s and the volume/issue details are also listed. There will be a limit on the number of entries that the search returns on a single screen to prevent the whole database being displayed.

Each of the three items in the search results screen will be hyperlinks to more detailed information and they are discussed separately below.

3.5.2.1. **Article Title**

The article title will be a hyperlink to more detail about the article (the article details which provides the following information).

- Article title
• Article author details (the name and contact details which contain a hyperlink to their email address and also their individual web page, if one is available)
• Author’s institution (their affiliation)
• Date article draft received
• Date article draft accepted for publication
• Accepting editor’s name
• Abstract
• Keyword/s
• Classification code/s

If the full text is currently available (that is, it has either been published or is currently awaiting inclusion in the next volume/issue to be published), hyperlinks to it will be provided. The full text will be available as PDF files in two formats: one for viewing on the screen and one for printing.

However, if the user is not logged in, the hyperlinks will not be displayed. Instead, instructions on how to subscribe or login will be provided (see Section 3.5.3).

3.5.2.2. Author Name

The author’s name will be a hyperlink to their publication list. The details of the author (the name and contact details which contain a hyperlink to their email address and also their individual web page, if one is available) will be provided.

A list of all articles (showing the article title and volume/issue details) associated with this author will be displayed. The article title will be a hyperlink to the article details (see above) and the volume/issue details will be hyperlinked to the table of contents for that specific volume/issue.

3.5.2.3. Volume/Issue

The volume/issue will be a hyperlink to the table of contents for the selected volume/issue of JIPAM. The publication number details (volume number and issue number) will be shown followed by a list of all articles in the publication. The article list will show the article title (a hyperlink to the article details) and the article author (a hyperlink to all the articles written by that author, e.g. their publication list).
A list of all of the JIPAM volume/issues currently published will also be provided at the end of the list of articles. Each publication number will be hyperlinked to its table of contents.

3.5.3. **USE CASE 3 — Subscribe to JIPAM**

To access the full text of articles published in JIPAM, general users must become subscribers. This will be achieved through the login function which displays a login screen and information about the subscription process.

If a general user has not previously registered as a subscriber, there will be a hyperlink within the information displayed on the login screen. This hyperlink displays an input screen with a submit button which the general user will be required to complete. Some of the data will be compulsory (shown with a asterisk) while other data will be optional. The compulsory data will be the user’s email address (which is also their login name) and a password.

The optional details include:

- Name
- Physical address
- Institution (their affiliation)
- Individual web page address

There will be a submit button on this screen which the general user will select after they have entered all the required details.

The details entered will then be checked to confirm that all compulsory fields have been completed. The email address will be unpacked to see if it is a valid email address. The details will be entered into the database and the user will be logged in.

3.5.4. **USE CASE 4 — Display Current Volume/Issue**

General users will be able to access details of the current volume/issue. A list of the contents of the current volume/issue showing both the article title and author/s will be displayed. Each article title will contain a hyperlink to the article details (see 3.5.2.1.) while each author name (see 3.5.2.2.) will also contain a hyperlink to all the articles written by that author (their publication list).
A list of all other volume/issues currently published will also be provided at the end of the list of articles. Each publication number will be hyperlinked to its table of contents screen.

3.5.5. **USE CASE 5 — Display Articles yet to be Published**

A list of all articles (article title and author/s) that have been accepted but not yet published in JIPAM will be available to general users. Each article title will contain a hyperlink to the article details (see 3.5.2.1.) while each author name (see 3.5.2.2.) will also contain a hyperlink to all the articles written by that author (their publication list).

3.5.6. **USE CASE 6 — Display List of Editors**

A list of all of the editors of JIPAM in alphabetical order will be available to general users. The editor’s name and their institution details will be provided followed by either one or two symbols; the first symbol (an envelope) will be a hyperlink to the editor’s email address while if the other symbol (a globe) is shown, it will be a hyperlink to the editor’s individual web page. The second symbol will only be displayed if the editor has their own individual web page.

3.5.7. **USE CASE 7 — Provide Feedback**

By using a text entry form, general users will have the facility to provide feedback to the JIPAM Administrator. The data entered by the user will be stored in the database. An email facility to contact the JIPAM Administrator will also be provided.

3.5.8. **USE CASE 8 — Static Pages**

There are several activities (as listed below) that a general user can perform that will provide access to information about JIPAM which are based on static HTML pages.

- *Aims and Scope* will describe the aims and scope of JIPAM.
- *Contact* will indicate how to contact the JIPAM editorial team.
- *Copyright* will discuss copyright issues for JIPAM.
- *Management* will list the members of the JIPAM management board.

---

*Requirements Model*  
*JIPAM Web Site: Dynamic Database System*  
24
• *PDF Files* will detail the format of full text articles provided for publication in JIPAM.

• *What's New* will provide the latest information about JIPAM.

### 3.6. ACTOR — SUBSCRIBER

Subscribers are general users who have additional access rights which allows them to download the full text of JIPAM articles. Note that general users will need to be logged in as a subscriber before they can access the full text.

### 3.6.1. USE CASE 9 — Login as a JIPAM Subscriber

General users need to login as subscribers before they can access the full text of articles. A login screen is displayed with information about subscribing (see 3.5.3). There will also be an input area for them to enter their login details plus an associated login button to complete the login process. To login, subscribers will need to enter their username (which is their email address) and their password.

As part of the validation process, the subscriber's details will be checked to ensure that their subscription to JIPAM has not lapsed (that is, the annual date is still current). Given that in Phase 1 of the system development, there will not be any subscription charge, the renewal date will be updated automatically.

The login process will either be successful or not — both outcomes are now discussed in more detail.

#### 3.6.1.1. Login is Successful

If the login details entered are valid, the subscriber will receive a confirmation message that they are logged in. At the same time, there will be a hyperlink provided to allow subscribers to update their details, if they feel it necessary to do so.

Once logged in, subscribers then have access to the full text of articles that have been published in JIPAM or that are awaiting publication.

#### 3.6.1.2. Login is Unsuccessful

If the login details are not recognised as being valid, the subscriber will be advised that they cannot be logged in. Another input form will then be provided for them to
try again. For times where they have forgotten their password, this form will also contain a hyperlink where they can request their password to be emailed to them. Another hyperlink will also be provided to enable them to become a subscriber.

A count of the number of times that a general user attempts to login as a subscriber will be maintained for each session. A successful login will reset the default count for the current session. The system will only allow three attempts to login. If the general user is still unsuccessful after their third try, no further attempts to login (and thus gain access to the full text) will be allowed during the current session.

3.6.2. **USE CASE 10 — Change Subscriber Details**

This option will only be available if the subscriber has already successfully logged in. Their current subscription information will be displayed and they will then be able to change the following details.

- Email address (also their username)
- Password
- Name
- Physical address
- Institution (their affiliation)
- Individual web page address

Once the changes have been entered, the subscriber selects the submit button which will update the JIPAM database. A message will then be displayed that confirms the changes were successfully carried out.

3.7. **ACTOR — AUTHOR**

Authors submit draft articles to JIPAM for publication which then undergo an editorial review to determine acceptability for publication in JIPAM.

Tracking this editorial review process will be outside the scope of Phase 1 of this project.
3.8. **ACTOR — EDITOR**

Are responsible for reviewing draft articles and applying the relevant publications standards to the contents of all volumes/issues of JIPAM. Tracking this editorial review process will be outside the scope of Phase 1 of this project.

3.9. **ACTOR — ADMINISTRATOR**

Provision needs to be made for administration and maintenance of the JIPAM system. Thus, there is an actor known as the JIPAM Administrator.

3.9.1. **USE CASE 11 — Login as Administrator**

The JIPAM Administrator must login as a subscriber before any system administration and/or maintenance can be done. If the correct login details are entered and subscriber has administrator rights, an extra menu option will be available to allow database maintenance activities.

3.9.2. **USE CASE 12 — Change Default Values**

An entry form will be displayed that shows the existing default values and these values can be changed by simply entering new values into the form and clicking the accept button.

The actual system defaults are now discussed in greater detail.

3.9.2.1. **Grace Period**

Although Phase 1 of the system has been developed based on the premise that subscription will be free, some functionality towards moving to a charge basis needs to be provided within Phase 1. Hence, subscriber registrations will provide full access rights for a defined period of time (12 calendar months from the initial subscription date).

The grace period is the amount of time after the annual renewal date that the customer has available to them to renew their subscription. If they do not do so, their subscription details will be deleted from the database. In Phase 1 of the project, the renewal date will be automatically updated.
3.9.2.2. JIPAM Administrator Email Address

To facilitate communication with JIPAM administrator, there is facility within Phase 1 for the system to automatically generate an email message addressed to the administrator. Thus, the default email address for the administrator must be known to the system.

3.9.2.3. Default Login Count

As a security measure, there will be a limit to the number of times (in one session) that a general user can attempt to login as a subscriber (to gain access to the full text). The default value within the Phase 1 is set to a maximum of three attempts.

If the count is exceeded, the user will be then locked out for the duration of their current session.

3.9.2.4. Maximum Search Result Count

There will be a limit on the number of records that will be displayed as a result of a search. Within Phase 1, the default value is 100 records.

3.9.3. USE CASE 13 — Create a New Article

New articles for JIPAM will be created in a number of stages.

3.9.3.1. Basic Information About the Article

Firstly, a data input screen will be displayed for the administrator to enter basic data about the article (see the following list).

- Article number
- Article title
- Date article draft received
- Date article draft accepted for publication
- Accepting editor’s name
- Abstract
- Available flag (provided to ensure that an article does not appear in article lists until all details about the article are acceptable to the administrator.)
When these fields have been filled out, an accept button will add these details about the new article into the database. The system will validate the article number and the two dates.

The administrator will then be presented with a series of screens that will accept the following data elements for a new article.

- Authors
- Keywords
- Math codes
- Full text PDF files

3.9.3.2. Authors

A screen will be presented to the administrator showing a list of all the subscribers who are also authors within the JIPAM database. The administrator will then select the author/s for the new article in the required order. A select button will update the article on the database with these authors.

3.9.3.3. Keywords

A screen will be presented to allow the administrator to enter keywords for the new article into the database. When the accept button is selected, the keywords will be written to the database.

3.9.3.4. Math Codes

A screen will be presented to allow the administrator to enter the math codes for the new article into the database. When the accept button is selected, the math codes will be written to the database.

3.9.3.5. Full Text PDF Files

A screen will be presented to allow the administrator to select the full text PDF files for each of the two formats. When a file is selected, its contents will be written to the database.
3.9.4. USE CASE 14 — Change the Article Details

There are a number of reasons why it may be necessary for the administrator to have the facility to change the details of an article in the database. For instance, there may have been a mistake in one of the data entries when it was created. Also, an article may have originally been flagged as unavailable. In such a case, the unavailable flag will need to be changed.

The administrator will have access to a screen that displays a list of all of the articles in the database. The administrator then selects the article that needs modification.

An entry form will display all of the basic details relevant to the selected article that can be changed (see list below).

- Article no.
- Article title
- Date article draft received
- Date article draft accepted for publication
- Accepting editor
- Abstract
- Available flag

The existing entries are shown for each field and they can be changed by simply typing in the new data and then clicking the submit button.

The administrator will then be presented with a series of screens to allow changes to be made to other data elements associated with the article.

3.9.4.1. Authors

A screen will be presented to the administrator that will display the existing authors on an article as well as a list of all other authors with the JIPAM system. The administrator will be able to remove an existing author, select an author from the main list and reorder the authors in the selected list. When completed, the accept button will write the changes to the database.
3.9.4.2. **Keywords**

A screen will be presented to the administrator that will display the existing keywords. The administrator will be able to add new keywords or delete existing keywords from this list. When completed, the accept button will write the changes to the database.

3.9.4.3. **Math Codes**

A screen will be presented to the administrator that will display the existing math codes. The administrator will be able to add new math codes or delete existing math codes from this list. When completed, the accept button will write the changes to the database.

3.9.4.4. **Full Text PDF Files**

A screen will be presented to the administrator which will show the existing PDF files assigned to the article. The administrator will be able to delete existing formats. They will also be able to select a PDF file for any of the two formats not already in the database. When a file is selected, the contents of the file will be written to the database.

3.9.5. **USE CASE 15 — Add a Subscriber**

There may also be times where it is necessary for the administrator to manually add a subscriber to the database, for instance, the web subscription process may be unavailable for whatever reason.

The administrator will enter the subscriber’s details into an input form (see 3.5.3. for more details about the data required). When submitted, various details will be validated and then the details will be written to the database.

3.9.6. **USE CASE 16 — Change a Subscriber’s Details**

Subscriber details may also need to be changed by the administrator at various times. In particular, the administrator needs to be able to change the role of subscribers, for instance, a subscriber may become an author or an editor.
A screen listing all the JIPAM subscribers will be displayed to the administrator who then selects the subscriber whose details need to be changed. An input screen is then displayed which provides the subscriber’s existing details. However, in addition to the fields that a general user completes as part of the subscription process, the following additional fields are also displayed.

- Institution
- Role
- Renewal date
- Start date

The administrator makes the necessary changes and the updated data is then written to the database.

3.9.7. **USE CASE 17 — Delete a Subscriber**

It is possible that the administrator will need to delete a subscriber before the automatic process associated with the end of the subscription period.

A list of the subscribers is displayed to the administrator who then selects the subscriber they wish to delete. The subscriber’s details are displayed for the administrator to confirm the deletion.

3.9.8. **USE CASE 18 — Update a Subscriber’s Subscription Date**

The renewal dates for all subscribers will be checked to identify those that have expired. In Phase 1, there is no fee being charged for subscription so all renewal dates will be automatically updated if they have used the system in the last three months. If they have not used the system in the preceding six months, they will be removed from the system.

3.9.9. **USE CASE 19 — Create a New Volume/Issue**

The administrator will need to be able to create new volume/issues on an ongoing basis. An input screen containing the following text boxes will be available to the administrator.
A submit button will be provided and the date entered will be validated and a new entry will be made into the database.

**3.9.10. USE CASE 20 — Add Articles to a New Volume/Issue**

Once articles are accepted for publication, they are flagged as new papers. When appropriate, these new papers need to be added to the contents of the next volume/issue to be published (the new volume/issue).

The administrator will have access to a screen which displays a list of the articles associated with the new volume. In addition, the list of articles that have not yet been associated with an issued volume are also displayed. The administrator will be able to select those articles that will be included in the new volume/issue. This process will change the status of these specific articles from being unpublished to published.

The administrator will be able to also manage the order that the articles appear in the table of contents of the new volume/issue through this screen.

When the selection and order of the articles is completed, the database will be updated with the table of contents of the new volume/issue.

**3.9.11. USE CASE 21 — Retrieve Usage Statistics**

At some point, the administrator will want to examine the statistics the system gathers automatically from users to the site. A screen will be presented to the administrator that lists the statistics available to them. The administrator will select the desired statistic and a further screen will be presented to them showing the statistic data gathered.
3.9.12. **USE CASE 22 — Add a New Institution**

Because a list of institutions is maintained within the database, there needs to be facility for the administrator to add new institutions to the list. Thus, there is an input screen available to the administrator to complete with the following details.

- Name of the institution
- Full address
- Telephone no.
- Fascimile no.

There will be a submit button on the screen and when it is selected, the entered details will be validated and a new entry made to the database.

3.9.13. **USE CASE 23 — Modify an Institution’s Details**

It is possible that the details of a specific institution may change, e.g. their telephone or fascimile number. The administrator has access to a screen which lists all the institutions in the database. They then select the institution whose details they wish to change.

A second screen will appear that will show the details in the database for the selected institution. The administrator can modify the selected fields and when completed the updated details will be written to the database.
4. **ANALYSIS MODEL**

The next stage of the Jacobson visual modeling process is the analysis model which aims to form a logical and maintainable structure for the system. It is derived from an analysis of each use case followed by a distribution of the behavior of each use case into three object types (interface, control, entity) and the interactions between them.

- **Interface objects** are responsible for functionality that is directly dependent on the system environment. As described by Jacobson et al, 'The task of an interface object is to translate those events in the system that the actor is interested into something that can be presented to the actor' (1992, p.176).

- **Entity objects** are used to model the information that the system will maintain over a period of time (Jacobson et al, 1992, p.184). Generally, this period of time will be longer than the length of time that is covered by the use cases. This means that these objects represent the persistent data of the system.

- **Control objects** are typically used to model all behaviour that cannot be placed into either of the other two objects.

4.1. **USE CASE ANALYSIS**

Although there are 22 use cases, several have been combined during the analysis stage in instances where they have some common elements. Thus, the following 15 diagrams were created from analysis of the use cases.

4.1.1. **Collect/Retrrieve Usage Statistics**

Figure 4 concerns the statistics gathered in relation to users of the JIPAM web site and incorporates Use Case No. 1 and 21.

![Figure 4. Collect/Retrieve Usage Statistics](image-url)
The remaining 14 diagrams have been created on similar principles with the next two diagrams representing the search functionality and the subscribe/login functionality which are the most complex activities within the system.

4.1.2. Search JIPAM

Figure 5 represents the search functionality of the JIPAM web system as outlines in Use Case No.2. The entity objects within this diagram (i.e., Default Values and Database Tables) are shown more than once in order to reduce the number of interconnections and therefore keep the diagram simple.

According to the use case specification, each of the three interface objects (i.e., Articles Details, Author Publication List and Volume Contents List) would know of each other. The relationships that each of these interfaces would have with each other’s control object does not appear in the diagram in order to ensure the diagram itself remains clear and simple.

![Search JIPAM Diagram](image)
4.1.3. Subscribe to JIPAM/Login as a JIPAM Subscriber/Change Subscriber Details

Similarly, the requirement for users to register as subscribers and then login as a subscriber before gaining access to the full text articles has meant that the analysis model shown as Figure 6 below is quite complex. The following use cases have been incorporated into one analysis diagram to condense the number of separate analysis diagrams.

- Use Case 3. Subscribe to JIPAM
- Use Case 9. Login as a JIPAM subscriber
- Use Case 10. Change subscriber details

![Diagram of Subscribe/login to JIPAM](image)

Figure 6. Subscribe/login to JIPAM
4.1.4. Display Current Volume/Issue

Figure 7 below is the analysis model derived from Use Case No. 4.

![Image of analysis model for current volume/issue](image)

*Figure 7. Current Volume/Issue*

4.1.5. Display Articles to be Published

Figure 8 below is the analysis diagram developed from Use Case No. 5.

![Image of analysis diagram for articles to be published](image)

*Figure 8. Articles Yet to be Published (New Papers)*
4.1.6. Display List of Editors

Figure 9 below is the analysis diagram developed from Use Case No.6.

![Figure 9. List of Editors](image)

4.1.7. Provide Feedback

Figure 10 below is the analysis diagram developed from Use Case No.7.

![Figure 10. Provide Feedback](image)
4.1.8. Static Pages

There are several existing static pages defined in the requirement model. Figure 11 below is the analysis diagram developed from Use Case No.8 which details all the static pages on the site.

![Image of Static Pages Diagram]

**Figure 11. Static Pages**

4.1.9. Change Default Values

The remaining seven analysis diagrams relate to use cases where the actor is the JIPAM administrator.

Firstly, the system will require several default values to be maintained by the administrator. See Figure 12 below. These default values relate to Use Case No. 12 and refer to values such as the administrator’s email address (for feedback etc). Other default values will include maximum counts for various user activities (e.g. the number of login attempts possible and the number of search results which will be displayed on one screen).

![Image of Change Default Values Diagram]

**Figure 12. Change Default Values**
4.1.10. Create an Article/Change Article Details

The next analysis diagram (Figure 13) represents two use cases (i.e. Use Case No's 13 and 14) where the administrator creates or changes the details about an article.

![Figure 13. Create/Change an Articles Details](image)

4.1.11. Add/Delete/Change a Subscriber's Details

The next analysis diagram represents three use cases (i.e. Use Case No's 15, 16 and 17) where the administrator may need to add, delete or change a subscriber's details.

![Figure 14. Add/Delete/Change a Subscriber's Details](image)
4.1.12. Update a Subscriber’s Subscription Date

To meet the client’s requirement to provide the functionality of being able to charge for access to full text articles at some stage in the future, it will be necessary to maintain time based data. Figure 15 shows the analysis diagram relating to Use Case No. 18.

![Figure 15. Update a Subscriber’s Subscription Date](image)

4.1.13. Create New Volume/Issue

It will be necessary for the administrator to create new volumes/issues of JIPAM on an on-going basis, i.e. Use Case No. 19 which is the basis for the following analysis diagram (Figure 16 refers).

![Figure 16. Create a New Volume/Issue](image)

4.1.14. Add Articles to a New Volume/Issue

After the new volume/issue has been created (Figure 16 above), the administrator next builds up its table of contents by adding articles as they are received from the Accepting Editors.

Use Case No. 20 refers and the analysis diagram is shown as Figure 17 on the following page.
4.1.15. Add/Change an Institution

The JIPAM administrator also needs to be able to add or change details regarding the institutions that general users and subscribers are affiliated with (Use Case No's 22 and 23 refers).

The analysis diagram is shown as Figure 18 below.

4.2. OBJECTS AND THEIR RESPONSIBILITIES

A list of all three objects types used in the analysis combined with a brief description of their roles and responsibilities is provided next. As Jacobson et al states, ‘we should state explicitly which object is responsible for which behaviour in the use case’ (1992, p.175).
4.2.1. Interface Objects

Actors communicate with the system via the interface objects. Details of the responsibilities associated with each of the interface objects is provided in Table 2.

<table>
<thead>
<tr>
<th>INTERFACE OBJECTS</th>
<th>RESPONSIBILITIES</th>
<th>SUBSYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aims And Scope Screen</td>
<td>Screen displaying the aims and scope of JIPAM</td>
<td>View</td>
</tr>
<tr>
<td>Article Assignment Form</td>
<td>Input form for assigning an article to a volume</td>
<td>Mtce</td>
</tr>
<tr>
<td>Article Details</td>
<td>Screen showing all details about an article</td>
<td>View</td>
</tr>
<tr>
<td>Article Input Form</td>
<td>Input form for accepting data for a new article</td>
<td>Mtce</td>
</tr>
<tr>
<td>Article Listing</td>
<td>Screen displaying the list of articles</td>
<td>Mtce</td>
</tr>
<tr>
<td>Article Titles List</td>
<td>Screen that lists article titles</td>
<td>View</td>
</tr>
<tr>
<td>Article Update Form</td>
<td>Input menu form that provides choices for additional data for an article</td>
<td>Mtce</td>
</tr>
<tr>
<td>Author’s Information Screen</td>
<td>Screen displaying the information advising author’s about JIPAM publication requirements</td>
<td>View</td>
</tr>
<tr>
<td>Author’s Input Form</td>
<td>Entry form for assigning authors to a article</td>
<td>Mtce</td>
</tr>
<tr>
<td>Author Publication List</td>
<td>Screen listing details about an author as well as all their articles</td>
<td>View</td>
</tr>
<tr>
<td>Contact Screen</td>
<td>Screen for display how to contact JIPAM editorial team</td>
<td>View</td>
</tr>
<tr>
<td>Copyright Screen</td>
<td>Screen for discussing the copyright issues</td>
<td>View</td>
</tr>
<tr>
<td>Default Values Form</td>
<td>Input form for entering and maintaining the system default values</td>
<td>Mtce</td>
</tr>
<tr>
<td>Editor List</td>
<td>Screen displaying the list of editors</td>
<td>View</td>
</tr>
<tr>
<td>Email Feedback</td>
<td>Email form to the administrator concerning feedback</td>
<td>View</td>
</tr>
<tr>
<td>Email Password</td>
<td>Email form to a subscriber advising them of their password</td>
<td>View</td>
</tr>
<tr>
<td>Feedback Input Form</td>
<td>Input form that accepts feedback data</td>
<td>View</td>
</tr>
<tr>
<td>Keyword Input Form</td>
<td>Input form that assigns keywords to an article</td>
<td>Mtce</td>
</tr>
<tr>
<td>Login Accepted Form</td>
<td>Form that indicates that login was successful</td>
<td>View</td>
</tr>
<tr>
<td>Login Form</td>
<td>Form that accepts the login details of subscribers</td>
<td>View</td>
</tr>
<tr>
<td>Management Screen</td>
<td>Screen that will list the members of the JIPAM management board</td>
<td>View</td>
</tr>
<tr>
<td>Math Codes Input Form</td>
<td>Input form that assigns maths codes to an article</td>
<td>Mtce</td>
</tr>
<tr>
<td>Menu Selection</td>
<td>The functionality for providing a choice to the user</td>
<td>Both</td>
</tr>
<tr>
<td>New Article Listing</td>
<td>Screen displaying a list of unpublished articles</td>
<td>View</td>
</tr>
<tr>
<td>New Institution Form</td>
<td>Input form for entering data relating to an institution</td>
<td>Mtce</td>
</tr>
</tbody>
</table>
## Table 2. Responsibilities of the Interface Objects

<table>
<thead>
<tr>
<th>INTERFACE OBJECTS</th>
<th>RESPONSIBILITIES</th>
<th>SUBSYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Subscriber Form</td>
<td>Administrator’s input form for entering all data relating to a subscriber</td>
<td>Mtce</td>
</tr>
<tr>
<td>PDF Files Input Form</td>
<td>Input form that assigns PDF files to an article</td>
<td>Mtce</td>
</tr>
<tr>
<td>PDF Files Screen</td>
<td>Screen which will detail the format of full text articles files</td>
<td>View</td>
</tr>
<tr>
<td>Renew Subscriber Menu</td>
<td>Interface for the administrator to process subscriber’s renewal dates</td>
<td>Mtce</td>
</tr>
<tr>
<td>Search Form</td>
<td>Input form that accepts data from a user on which to perform a search</td>
<td>View</td>
</tr>
<tr>
<td>Statistic Report</td>
<td>Report displaying statistic data collected from user’s browsers</td>
<td>View</td>
</tr>
<tr>
<td>Subscribe Form</td>
<td>Form that accepts a user’s details so they can subscribe</td>
<td>View</td>
</tr>
<tr>
<td>Subscription Successful</td>
<td>Form that advises a user that their subscription was successful</td>
<td>View</td>
</tr>
<tr>
<td>Update Institution Form</td>
<td>Input form for updating data relating to an institution</td>
<td>Mtce</td>
</tr>
<tr>
<td>Update User Details Form</td>
<td>Form a user completes to update their details</td>
<td>View</td>
</tr>
<tr>
<td>Volume Contents List</td>
<td>Screen showing all articles on a published JIPAM volume</td>
<td>View</td>
</tr>
<tr>
<td>Volume Input Form</td>
<td>Input form for entering data relating to a new volume</td>
<td>Mtce</td>
</tr>
<tr>
<td>What’s New Screen</td>
<td>Screen which provides the latest information about JIPAM</td>
<td>View</td>
</tr>
</tbody>
</table>

## 4.2.2. Entity Objects

Entity objects are used to model the information handled by the system over a period of time. Table 3 below lists the JIPAM system entity objects and their responsibilities.

## Table 3. Responsibilities of the Entity Objects

<table>
<thead>
<tr>
<th>ENTITY OBJECTS</th>
<th>RESPONSIBILITIES</th>
<th>SUBSYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Tables</td>
<td>Manages persistent data for the system</td>
<td>Both</td>
</tr>
<tr>
<td>Default Values</td>
<td>Manages non-database persistent data for the system</td>
<td>Both</td>
</tr>
<tr>
<td>Statistic Tables</td>
<td>Manages data collected from users’ browsers</td>
<td>Both</td>
</tr>
</tbody>
</table>
4.2.3. Control Objects

Control objects contain behaviour which is not naturally placed in either interface or entity objects. Table 4 lists the control objects used in the JIPAM system and their associated responsibilities.

<table>
<thead>
<tr>
<th>CONTROL OBJECTS</th>
<th>RESPONSIBILITIES</th>
<th>SUBSYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Article Detail Formatter</td>
<td>Implementation of the rules to extract and format article details from the database</td>
<td>View</td>
</tr>
<tr>
<td>Article Input Validator</td>
<td>Implementation of the rules that ensure that an article's data is valid</td>
<td>Mtce</td>
</tr>
<tr>
<td>Author Formatter</td>
<td>Implementation of the rules to extract and format an author's details from the database</td>
<td>View</td>
</tr>
<tr>
<td>Editor Formatter</td>
<td>Implementation of the rules that extract editor details from the database</td>
<td>View</td>
</tr>
<tr>
<td>Feedback Process</td>
<td>Implementation of the rules that process feedback data</td>
<td>View</td>
</tr>
<tr>
<td>Institution Validator</td>
<td>Implementation of the rules that ensure an institution's data is valid</td>
<td>Mtce</td>
</tr>
<tr>
<td>Login Validator</td>
<td>Implementation of the rules that govern login to the system</td>
<td>View</td>
</tr>
<tr>
<td>Subscriber Validator</td>
<td>Implementation of the rules that ensure a subscriber's data is valid</td>
<td>Both</td>
</tr>
<tr>
<td>Renew All Subscribers</td>
<td>Implementation of the rules governing subscriber's renewal dates</td>
<td>Mtce</td>
</tr>
<tr>
<td>Search Process</td>
<td>Implementation of the rules that perform searches on the database tables</td>
<td>View</td>
</tr>
<tr>
<td>Statistic Formatter</td>
<td>Implementation of the rules that extract data from the statistic tables</td>
<td>View</td>
</tr>
<tr>
<td>Unpublished Formatter</td>
<td>Implementation of the rules to extract and format unpublished article details from the database</td>
<td>View</td>
</tr>
<tr>
<td>Volume Formatter</td>
<td>Implementation of the rules to extract and format a JIPAM volume's details from the database</td>
<td>View</td>
</tr>
<tr>
<td>Volume Validator</td>
<td>Implementation of the rules that ensure the data describing a volume is valid</td>
<td>Mtce</td>
</tr>
</tbody>
</table>

Table 4. Responsibilities of the Control Objects

4.3. SUBSYSTEMS

Objects derived by the analysis phase may be grouped together into subsystems to provide a clearer overview of the system. Analysis of the JIPAM Web System...
indicated that two subsystems are required. Tables 2, 3 and 4 provide more detailed information about the relationships between the objects and the two subsystems.

4.3.1. Viewing Subsystem

Although these groupings are quite arbitrary, Jacobson et al says they should be ‘coupled to only one actor, since changes are usually caused by actors’ (1992, p.196). In contrast, within the JIPAM system, the first subsystem is derived from the general user and the subscriber actors because they are closely related and the subscriber login function just provides general users with additional access rights (access to the full text articles. ‘The division into subsystems should also be based on the functionality of the system’ (Jacobson et al, 1992, p.196).

The statistical gathering component has been included in the viewing subsystem because the data collected arises from actions associated with these two groups of actors. Figure 19 below represents a block diagram of the viewing subsystem.

- Collect/retrieve usage statistics
- Search JIPAM
- Subscribe/login/change subscriber details
- Display current volume/issue
- Display articles to be published
- Display list of editors
- Provide feedback
- Static Pages

Figure 19. Block Diagram of the Viewing Subsystem

4.3.2. Maintenance Subsystem

The actor, JIPAM Administrator, is the basis of the other subsystem, called maintenance. (see Figure 20 below).

- Change default values
- Create/change article details
- Add/delete/change a subscriber’s details
- Create new volume/issue
- Add articles to a new volume/issue
- Add/change an institution

Figure 20. Block Diagram of the Maintenance Subsystem
5. CONSTRUCTION AND DESIGN

According to Jacobson et al, 'the design model will further refine the analysis model in the light of the actual implementation environment' (1992, p.204). This model is used because the ideal behaviour of the analysis model is constrained by operating factors (the real world).

The original intention of this project was to build a web application providing all the additional functionality required of the system. As prototyping was carried out, it was discovered that the ASP software that was specified did not directly support one of the customer's key requirements. Namely, the customer required the full text PDF files to be stored in the database instead of by referencing the files residing on the web server.

There are two possible solutions which allow a file to be loaded into the database. The first involves sending a HTML message to the web server encoded so that the ASP application would communicate with a process running on the client's machine that would return the file directly to the server ASP page. The second method involves embedding the contents of the file in the HTML message. The file is then extracted from the message on the web server.

Neither solution is supported directly by the ASP software and therefore third-party components are required. As these components would have to be purchased (at some considerable cost), it was necessary to find an alternate solution. The solution chosen was to build the administrator subsystem as a stand-alone NT application where the NT application would write directly into the database.

5.1. ANALYSIS MODEL ENTITIES

The analysis phase identified that the JIPAM Web Site: Dynamic Database System could be categorised into two subsystems (Viewing and Maintenance). However, the entity objects have some commonality across both subsystems and are considered in further detail next.
5.1.1. Database Tables

All of the data items needed to support the application excluding statistic data items were extracted from the existing web site. Using standard normalising processes, the database scheme shown in figure 21 was developed.

Figure 21. Database Scheme
The only tricky relationship in the displayed scheme is that between an institution_id and an author-article combination. When an author publishes an article, the author does so as member of an particular institution. At some point later in their academic career they may move to another institution and publish additional papers. The first relationship needs to maintained forever and is why the field institution_id is stored in the ArticleAuthors table.

Specific backup strategies have not been included with this application because it is considered that backups would be managed by the Database Administrator in an SQL Production Environment.

5.1.2. Default Values

The attribute values of this data entity are relatively static in value. Consequently, they are implemented as in memory variables available to the viewing subsystem. They are created and initialsed when the JIPAM viewing subsystem is first accessed by the IIS web server. The relevent code is included in the global.ASA module (see below).

```csharp
Application.Contents.Item("EmailAdmin") = "EmailAdmin"
Application.Contents.Item("MaxLines") = 30
Application.Contents.Item("LatestVolumId") = 1
Application.Contents.Item("LoginAttempts") = 4
```

5.1.3. Statistic Tables

The design for implementing the statistic component has been based on Chapters 8 and 9 of Homer’s book, ‘Professional ASP Techniques for Webmasters’ (1998, p327–425). Not all the techniques discussed in these chapters are used, only those related to the capture and display of information that is extracted from the web browser request header. The code provided in the book was modified and incorporated into the JIPAM viewing subsystem.

The database tables needed to support the process of gathering and reporting on visitor statistics are shown in figure 22. Note that there are no relationships between any of the tables as they are populated under the control of a SQL batch job.
Every time a user accesses the JIPAM web site, a new session object is created by the web server provided they do not already have a current session object. As part of the creation of the session object, a record is written to the Sessions database table. The code for writing the record is contained in the session’s on_start procedure that resides in the global.ASA file in the root JIPAM directory and is as follows.

```
strSQL = "INSERT INTO Sessions 
  (EventDateTime, EventType, URL, Referer, RemoteHost, UserAgent  
   , UALanguage, UserID, HostIP) " 
  VALUES (GetDateO,'New Session', "
  Request.ServerVariables("URL") ", "
  Request.ServerVariables("HTTP_REFERER") ", "
  Request.ServerVariables("REMOTE_HOST") ", "
  Request.ServerVariables("HTTP_USER_AGENT") & ", "
  Request.ServerVariables("HTTP_ACCEPT_LANGUAGE") & ", "
  & CLng(Session.SessionID) & ", "
  Request.ServerVariables("LOCAL_ADDR") & ")"
WriteSessionData strSQL
```

The record written to the database contains information extracted from the users’ request header and it is raw data. The raw data needs to be summarised into other tables before it can be analysed and reported on. The following two database procedures are provided to support this summarisation process. The actual code is shown as Appendix B.
• **SummariseSession.SQL** which summaries the data
• **DeleteOldRecords.SQL** removes the summarised records from the Sessions table as well as removing all records over 26 weeks old from the other statistical tables in the database.

As part of the database setup procedure, an SQL job called Weekly JIPAM Session Update2 is created to run these two procedures once a week. The HTML code for displaying the contents of the summarisation tables is in module TrafficReports.ASP (Appendix C).

### 5.2. VIEWING SUBSYSTEM

This subsystem was designed to run from an Internet browser. One of the customer’s requirements was to use reduce the amount of Javascript used as much as possible because a number of their customers still had primitive browsers that did not support its use.

The requirement to support primitive browsers also had an affect on the menu design of the HTML pages. As HTML tables are supported in nearly all browsers, it was decided to use HTML tables in preference to HTML frames. Each HTML page is assigned a name and the menu that is displayed to a user depends on both the page name and the user’s role in the system. In Phase 1, only the Administrator role has an addition menu selection.

#### 5.2.1. Final Design

In the final design, each page returned from the IIS server is constructed from the combination of four files. The main file is the file shown as being the body of the HTML page in the figure. This file uses the IIS facility of Server-Side includes to incorporate the three files that incorporate the HTML forming the document header, the menu sidebar and the document footer.

This approach was chosen for the following two reasons.

- To promote a common look to all JIPAM web pages
- To reduce future maintenance needs by incorporating all common look facilities in one place.
The final design for the viewing subsystem is shown in Figure 23.

![Figure 23. Viewing Subsystem Final Design](image)

The HTML code produced by each of the modules listed in figure 23 is placed inside a HTML data cell. The `<TD>` and `</TD>` HTML data cell pair are output as the last statement of file common/menu.inc and the first statement of file common/footer.INC.

The relationship between the buttons on the menu bar and the ASP modules is shown in Figure 24 (shown on the following page).

The menu facility on each web page in Phase 1 of the JIPAM web system uses buttons to control the actions that can be performed by users. The same menu items
are provided as hyperlink markers in the footer. The state of both menus is controlled by the asp variable, PageName which is initialised by each asp page. The value set for the variable PageName determines whether the button for that page is displayed as being up or displayed as being down. It also determines whether the corresponding hyperlink marker in the footer is turned on or off.

As an example, the ASP code for displaying the list of editors model is shown below.

```aspx
<% if PageName="EDITORS" then %>
SRC="/jipam/images/Editors-down.gif "&nbsp;</TD>
<% else  %>
SRC="/jipam/images/Editors-up.gif USEMAP="#FPEditors"&nbsp; </TD>
<% endif%>

The corresponding code in the footer is shown next.

```aspx
if PageName<>"EDITORS" then%
<NOPR>[&nbsp;<A HREF="Editors.asp">Editors&nbsp;</A>]</NOPR>
<% else %>
<NOPR>[&nbsp;Editors&nbsp;]</NOPR>
<% end if %>
```
Note that the ASP module names are the external file name whereas the menu page name is the value set to identity those files internally within the system. Table 5 below details the ASP files names and their associated menu page names.

<table>
<thead>
<tr>
<th>ASP MODULE (FILE NAMES)</th>
<th>MENU PAGE NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accepted.asp</td>
<td>ACCEPTED</td>
</tr>
<tr>
<td>AimsScope.asp</td>
<td>SCOPE</td>
</tr>
<tr>
<td>Authors.asp</td>
<td>AUTHORS</td>
</tr>
<tr>
<td>Contact.asp</td>
<td>CONTACT</td>
</tr>
<tr>
<td>Copyright.asp</td>
<td>COPYRIGHT</td>
</tr>
<tr>
<td>Default.asp</td>
<td>MAIN</td>
</tr>
<tr>
<td>DispArticle.asp</td>
<td>DISPLAYART</td>
</tr>
<tr>
<td>DispAuthTitle.asp</td>
<td>DISPLAYAUTH</td>
</tr>
<tr>
<td>DispVolume.asp</td>
<td>DISPLAYVOL</td>
</tr>
<tr>
<td>DoSearch.asp</td>
<td>DOSEARCH</td>
</tr>
<tr>
<td>DownArticle.asp</td>
<td>noname</td>
</tr>
<tr>
<td>Editors.asp</td>
<td>EDITORS</td>
</tr>
<tr>
<td>Feedback.asp</td>
<td>FEEDBACK</td>
</tr>
<tr>
<td>GraphicLang.asp</td>
<td>TRAFFICLAN</td>
</tr>
<tr>
<td>GraphicOpSys.asp</td>
<td>TRAFFICOPS</td>
</tr>
<tr>
<td>GraphicUaType.asp</td>
<td>TRAFFICUAT</td>
</tr>
<tr>
<td>Login.asp</td>
<td>LOGON</td>
</tr>
<tr>
<td>LogonPass.asp</td>
<td>LOGON</td>
</tr>
<tr>
<td>Management.asp</td>
<td>MANAGEMENT</td>
</tr>
<tr>
<td>ManagementPhotos.asp</td>
<td>MANAGEMENT</td>
</tr>
<tr>
<td>ModLogon.asp</td>
<td>MODLOGON</td>
</tr>
<tr>
<td>Pdf.asp</td>
<td>PDF</td>
</tr>
<tr>
<td>Search.asp</td>
<td>SEARCH</td>
</tr>
<tr>
<td>ShowArtImg.asp</td>
<td>NONAME</td>
</tr>
<tr>
<td>ShowLog.asp</td>
<td>TRAFFICLOG</td>
</tr>
<tr>
<td>Subscribe.asp</td>
<td>SUBSCRIBE</td>
</tr>
<tr>
<td>TrafficReports.asp</td>
<td>TRAFFIC</td>
</tr>
<tr>
<td>WhatsNew.asp</td>
<td>WHATSNEW</td>
</tr>
</tbody>
</table>

*Table 5. Association Between ASP Page Names and Menu Page Names*
The ASP modules with a page name of 'noname' in the table are not ASP pages called by the menu system. These ‘noname’ pages provide support to other ASP pages to complete actions that enable those calling pages to be completed correctly.

5.2.2. Security

Security is one of the major issues in using a database in the web environment. In order to restrict the access that general users have, a separate database user (INTERNET) is created in the setup process for the JIPAM application. This user is only given execute rights to the database and therefore can only execute database procedures for which they have been given permission. This means that, even if an Internet user found out the userid and password of the created database user, they could not see what tables exist in the database or the structure of any database table.

A further security feature involves the use of the Session object provided in IIS. All details relating a general user/subscriber including their login status are kept on the web server. While this increased the memory used by individual users, it has meant that messages to and from the web server were reduced in size. It has also meant that user information would not be returned to a user, thereby preventing possible fraudulent activities.

The initialization of the data items relating to security occurs when the Session object is created for a user. The creation of a Session object for individual JIPAM users occurs whenever a JIPAM web page is accessed. The variables initial code is in the Session on_start procedure which is contained in the global.ASA file.

5.2.3. Interface Objects

The viewing subsystem interface objects and the ASP modules on which they are implemented is show in Table 6. An ASP module may implement more than one interface object.
5.2.4. Control Objects

The ASP pages that implement the control objects are shown in Table 7. Also shown are the SQL database procedures that assist in implementing the control objects.
Where there is no entry in the Related SQL Procedures column in Table 7 above, the database access was constructed directly in SQL using code provided within Chapters 8 and 9 of Homer’s book (1998).

5.3. MAINTENANCE SUBSYSTEM

The maintenance program was developed to run as a stand-alone NT program and it was written in the C++ language using the Borland Builder Program. The development environment provided by this program is very similar to Microsoft Visual Basic environment. An application program is developed in this environment using pre-supplied components that are inserted into the containers that provide the underlying structure. The entire program is built using these components and C++ objects that are developed to support the user’s application.

5.3.1. Administration Program Users Guide

Jacobson et al says that ‘documentation of the system should be produced by the developers’ but that ‘manuals, both for maintenance and for users, should be written by people with special skills for this’ (1992, p.462). Their views on documentation have been followed as guiding principles for the maintenance subsystem, however, significant effort has been made to develop effective maintenance documentation.
Thus, a detailed manual titled ‘JIPAM Administration Program Guide’ has been produced. It is structured into seven chapters as listed below.

- Chapter 1 Introduction
- Chapter 2 Administration Program Installation
- Chapter 3 Subscribers
- Chapter 4 Articles
- Chapter 5 Volumes/Issues
- Chapter 6 Institutions
- Chapter 7 Statistics

Although normally this guide would be bound and presented as a separate document for ease of use by the administrator, it has been collated within this thesis as Appendix E.

5.3.2. Database Issues

The database objects provided in the environment, for example classes based on TTable, TQuery and TDBGrid make it a simple matter to view and scroll through the database tables so this approach was chosen as the underlying methodology for accessing individual records in the database. These database objects connect to the database using the ODBC interface.

In addition, a single variable was created for every field in the database and it controls the size of any input field associated with the associated database field. This strategy was used to reduce future maintenance effort if the size of the database fields ever need to be changed. The variable definitions appear in the module that contain the form on which the associated field is input.

A number of issues arose while developing the maintenance subsytem. The first issue related to the database viewing grid that formed the starting point for all activities. When an individual record was modified in a dialog box, the changed record was not displayed with the updated details in the viewing grid. This was eventually traced to the fact that the tables were relational tables accessed through the ODBC pipeline and they were cached in memory as they were retrieved in the underlying query. Despite researching as many different ways as possible to overcome this issue including making the underlying query supporting the viewing grid, the issue
remains. It is only overcome by reissuing the query against the database. All viewing
grids support filter and sort operations which create a new query on the underlying
database table when they are changed.

The second issue concerned the size of variable length text fields, in particular the
article abstract field. Despite the abstract field being created with a 6,144 variable
character size, the viewing grid and the memo edit box saw the field as a 255
character field. The problem was eventually traced to the ODBC driver. Even when
accessing the abstract field directly through the ODBC driver, only the first 255
characters were being accessed. The solution was to cast the abstract field as text in
the query that retrieved the abstract field from the database. This resulted in the
correct ‘type’ being returned which made the ODBC driver behave correctly.

The third issue was caused by the Borland database driver. This driver sits above the
ODBC API and acts as an interface to their database objects, e.g. their database table
object (TTable). Because the PDF files were going to be stored in the database as
BLOBS (Binary Large ObjectS) and the Borland database objects supplied a BLOB
object, the writing of the PDF files to the database should have been easy.

This was not the case.

The Borland database driver was truncating the PDF file at the size defined by the
‘BLOB SIZE’ parameter in their database administrator. Despite their documentation
which suggested that there should be no problem with using their BLOB object, the
only solution found was to write the PDF files directly to the database using the
ODBC API.

Because of these issues, there were a number of database activities which were not
implemented as database procedures. Instead, some were implemented as straight
SQL code.

5.3.3. Final Design

The relationship between the main menu items in the maintenance program and the
C++ modules is shown in Figure 25 on the following page. The main menu items are
implemented as push buttons on the main program form. When selected, each of the
buttons brings up a screen showing the contents of the underlying table.

Figure 25. Maintenance Subsystem Final Design

5.3.4. Security

Additional security (other than that provided by the NT operating system) has not
been built into the maintenance program. As the program needs to be installed for
each individual user, they will have rights to use and execute the program. The
administration program installation process is fully detailed in the Administration
Program Guide (Appendix E).

During the setup of the database (see Appendix B - SetUp.SQL which provides the
relevant code), a database login id (JPADMN with password JPADMN) is created
for the user of the maintenance subsystem. This login id has administration rights to
the database.
5.3.5. Interface Objects

The implementation of all interface objects was completed using dialog box forms. In most cases, all use cases had their own dialog box but some interfaces were implemented using the same dialog box. This approach was chosen to reduce the amount of code required to implement the display aspects of the dialog box. Where the same dialog box was used, the behaviour of the box was dependent on the interface being shown at that time.

A number of interface objects other than those specified by the use case analysis were developed. They were provided to make the process more efficient. An example of this is the module that creates an author from a subscriber when adding authors to an article. Instead of closing that dialog box, opening the modify subscriber dialog box, making the subscriber an author, and then re-opening the original dialog box, the process can be done while still holding the original dialog box open.

The maintenance subsystem interface objects and their relevant C++ modules are shown in Table 8 below.

<table>
<thead>
<tr>
<th>INTERFACE OBJECT</th>
<th>C++ MODULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Article Assignment Form</td>
<td>DlgAddArtToVolume</td>
</tr>
<tr>
<td>Article Input Form</td>
<td>DlgModArticles</td>
</tr>
<tr>
<td>Article Update Form</td>
<td>DlgModArticles</td>
</tr>
<tr>
<td>Author's Input Form</td>
<td>DlgAddAuthArt</td>
</tr>
<tr>
<td>Keyword Input Form</td>
<td>DlgKeyWords</td>
</tr>
<tr>
<td>Math Codes Input Form</td>
<td>DlgKeyWords</td>
</tr>
<tr>
<td>New Institution Form</td>
<td>DlgModInstitution</td>
</tr>
<tr>
<td>New Subscriber Form</td>
<td>DlgAddUser</td>
</tr>
<tr>
<td>PDF Files Input Form</td>
<td>DlgArticleDownloads</td>
</tr>
<tr>
<td>Update Institution Form</td>
<td>DlgModInstitution</td>
</tr>
<tr>
<td>Volume Input Form</td>
<td>DlgNewVolume</td>
</tr>
</tbody>
</table>

Table 8. Maintenance Subsystem Interface Objects
The interface item ‘Renew Subscriber Menu’ has been implemented in this version as SQL database job 'Weekly JIPAM Session Update2'. This job automatically runs each week summarising the Sessions database table into the other statistic tables.

5.3.6. Control Objects

The implementation of the control objects and their associated C++ modules in the maintenance subsystem are shown in the following table (Table 9).

<table>
<thead>
<tr>
<th>CONTROL OBJECT</th>
<th>C++ MODULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Article Input Validator</td>
<td>DlgModArticles</td>
</tr>
<tr>
<td>Institution Validator</td>
<td>DlgModInstitution</td>
</tr>
<tr>
<td>Subscriber Validator</td>
<td>DlgAddUser</td>
</tr>
<tr>
<td>Volume Validator</td>
<td>DlgNewVolume</td>
</tr>
</tbody>
</table>

Table 9. Maintenance Subsystem Control Objects

5.4. TESTING

Testing was undertaken in parallel through all stages of construction to ensure that the use cases were successfully completed.

As Jacobson et al specified, ‘each use case is initially tested separately, from an external viewpoint. These tests are thus based on the requirements model. When all use cases have been tested separately, the entire system is tested as a whole’ (1992, p.333).
6. **Summary**

The research problem involved the redevelopment of the existing JIPAM Web Site from a site that was based on static content pages to one that was based on database technology. In addition, it was a client requirement that the new web site was to be based on Active Server Pages technology delivered from an SQL7 database.

In arriving at a solution to this problem, it was necessary to investigate the technologies proposed by the client to explore both how they should be used as well as their limitations. Discussions on these issues were covered in Chapter 2.

The methodology chosen to arrive at a software solution was the use case driven approach of Jacobson et al (1992). Discussions on this methodology and how the methodology applied to the solution were covered in Chapter 3 through to Chapter 5.

All of the research objectives set out in the introduction were achieved. A dynamic web site has been created for the JIPAM Managerial Board. It is based on a database framework and it uses an ASP front-end to provide customers with dynamic access to the journal articles held in the database. It also solves a number of problems that affected the existing JIPAM site.

In summary, the software solution which is being delivered …

- reduces the maintenance effort by automatically providing links to other documents
- provides a search facility for visitors so that they can query articles on a number of criteria fields
- restricts access to the full text articles to only those visitors that subscribe and subsequently logon to the site

It did not however solve the problem of tracking the peer review process of articles nor did the subscription process include charging for site access.
After discussions with my academic supervisor, it was decided that the research project should focus on developing an extensive and effective foundational platform to meet the basic requirements.

The facility to track the review process and the facility to charge a fee to access are therefore left to those responsible for the next generation of the JIPAM Web Site.

Another potential future enhancement involves the provision of a more complex search engine to query the database. In phase 1 of the JIPAM Web Site development process, a simple search process is provided with the search being performed on a single field at a time. As the JIPAM web site matures, it may be beneficial to allow logical searching amongst multiple fields.
REFERENCES


RFC 1945
RFC 2068
RFC 1867
APPENDIX A

JIPAM WEB SITE
Dynamic Database System

Installation CD-ROM

INSTALL CD IS UNAVAILABLE — INTELLECTUAL PROPERTY RIGHTS APPLY
APPENDIX B

JIPAM WEB SITE
Dynamic Database System
SQL Server 7
Programming Code
APPENDIX B - MODULE SETUP.SQL
CREATE TABLE ArticleAuthors (Article_ID char(6) Foreign Key references Articles(Article_ID), User_ID int Foreign Key references Users(User_ID), AuthorOrder smallint Default 1, Institution_id smallint Foreign Key references Institutions(Institution_id))
GO

CREATE TABLE ArticleKeywords (Article_ID char(6) Foreign Key references Articles(Article_ID))
GO

CREATE TABLE ArticleMathCodes (Article_ID char(6) Foreign Key references Articles(Article_ID))
GO

CREATE TABLE ArticlesInFull (Article_ID char(6) Foreign Key references Articles(Article_ID), PrintType tinyint Foreign Key references PrintFileTypes(PrintType), TheArticle Image)
GO

CREATE TABLE VolumeArticles (Article_ID char(6) Foreign Key references Articles(Article_ID), Volume_ID smallint Foreign Key references Volumes(Volume_ID), ArticleNO tinyint)
GO

CREATE TABLE ArticleAbstractImages (Article_ID char(6) Foreign Key references Articles(Article_ID), ImageNumber int not null, ImageSource varchar(255) not null, ImageType char(3) not null, ImageAltText varchar(255) not null, ArticleImage Image)
GO

CREATE TABLE Feedback (Feedback_ID smallint IDENTITY(1,1) PRIMARY KEY, FeedbackDateTime smallDateTime DEFAULT (Getdate()),

GO

CREATE VIEW Authors AS
Select a.Article_ID, a.AuthorOrder, a.User_ID, b.DisplayName, b.Surname, b.FirstName, b.OtherNames, a.Institution_ID, b.EmailAddress, b.WebAddress
from ArticleAuthors a JOIN USERS b on a.User_ID = b.User_ID
GO

CREATE VIEW ArticlesInVolume AS
Select Article_ID, ArticleNO, a.Volume_ID, Volume_NO, Issue_NO, ISSUEYEAR
from VolumeArticles a INNER JOIN VOLUMES b ON a.Volume_ID = b.Volume_ID
GO

CREATE VIEW TitlesInVolume AS
Select a.Article_ID, a.ArticleTitle, a.ArticleNO, c.AuthorOrder, c.User_ID, c.DisplayName, b.Volume_ID, b.Volume_NO, b.Issue_NO, b.IssueYear
from articles a INNER JOIN Authors c ON a.Author_ID = c.Author_ID
GO
```
ON s.Article_ID = b.Article_ID
and s.Available = 'Y'
GO

****** Object: View TitlesUnPublished ******
if exists (Select table_name from Information_Schema.views
where table_name = 'TitlesUnPublished')
DROP VIEW TitlesUnPublished
GO

CREATE VIEW TitlesUnPublished
AS
select a.Article_id ,a.ArticleTitle,b.UserID,b.DisplayName,b.AuthorOrder,
a.ReceiptDate
from articles a inner join Authors b
on a.Article_id = b.article_id
where a.published='N' and a.available = 'Y'
GO

****** Object: View EditorDetails ******
if exists (Select table_name from Information_Schema.views
where table_name = 'EditorDetails')
DROP VIEW EditorDetails
GO

CREATE VIEW EditorDetails
AS
select a.user_id,a.FoundingFlag,
b.FirstName,b.OtherInit,b.surname,b.webAddress,b.EmailAddress,b.DisplayName,
c.AddressLine1,c.AddressLine2,c.AddressLine3,c.AddressLine4,c.AddressLine5
from Editors a inner join users b
on a.user_id = b.user_id
inner join Institutions c
on b.Institution_ID = c.Institution_ID
where a.ActiveFlag='Y'
GO

****** Object: View EditorList ******
if exists (Select table_name from Information_Schema.views
where table_name = 'EditorList')
DROP VIEW EditorList
GO

CREATE VIEW EditorList
AS
select a.user_id,b.DisplayName,b.surname
from Editors a inner join users b
on a.user_id = b.user_id
GO

****** Object: View AuthorDetails ******
if exists (Select table_name from Information_Schema.views
where table_name = 'AuthorDetails')
DROP VIEW AuthorDetails
GO

CREATE VIEW AuthorDetails
AS
select a.user_id,
b.FirstName,b.OtherInit,b.surname,b.webAddress,b.EmailAddress,
c.AddressLine1,c.AddressLine2,c.AddressLine3,c.AddressLine4,c.AddressLine5
```

```
from (Select distinct user_id from ArticleAuthors) a
inner join users b
on a.user_id = b.user_id
inner join Institutions c
on b.Institution_ID = c.Institution_ID
GO

****** Preload static table values ******
Insert into PrintFileTypes (PrintShortDesc,HtmlDescription,ImageFile) Values ('PDF SCREEN', 'APPLICATION/PDF', 'viewpanel.jpg')
Insert into PrintFileTypes (PrintShortDesc,HtmlDescription,ImageFile) Values ('PDF PRINTER', 'APPLICATION/PDF', 'BSQ0883A.gif')
GO

USE JIPAM

****** Statistic tables Code copied from Profession ASP Techniques
For Webmasters ******

****** Object: Table [dbo].[CountrySummary] Script Date: 21/12/1999 14:23:29
******
if exists (Select * from sysobjects where id = object_id(N'['dbo].[CountrySummary']")
and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[CountrySummary]
GO

****** Object: Table [dbo].[LangCodes] Script Date: 21/12/1999 14:23:29 ******
if exists (Select * from sysobjects where id = object_id(N'['dbo].[LangCodes']")
and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[LangCodes]
GO

****** Object: Table [dbo].[Referees] Script Date: 21/12/1999 14:23:29 ******
if exists (Select * from sysobjects where id = object_id(N'['dbo].[Referees']")
and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[Referees]
GO

****** Object: Table [dbo].[Referers] Script Date: 21/12/1999 14:23:29
******
if exists (Select * from sysobjects where id = object_id(N'['dbo].[Referers']")
and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[Referers]
GO

****** Object: Table [dbo].[SessionSummary] Script Date: 21/12/1999 14:23:29
******
if exists (Select * from sysobjects where id = object_id(N'['dbo].[SessionSummary']")
and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[SessionSummary]
GO

****** Object: Table [dbo].[SessionTargetSummary] Script Date: 21/12/1999
14:23:29 ******
if exists (Select * from sysobjects where id = object_id(N'['dbo].[SessionTargetSummary']")
and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[SessionTargetSummary]
GO
```

**APPENDIX B - MODULE SETUp.SQL**
GRANT REFERENCES, SELECT, INSERT, DELETE, UPDATE ON [dbo].[UserAgentSummary]
TO [iJampLog]
GO

GRANT REFERENCES, SELECT, INSERT, DELETE, UPDATE ON [dbo].[WeekSummary] TO
[iJampLog]
GO

-------
*

/* insert into WeekSummary (YearNumber,WeekNumber,WSessions )*/
/* Values ( DATEPART(year,GetDate()),DATEPART(week,GetDate()) ) = 1,0)*/
-------

insert into WeekSummary (YearNumber,WeekNumber,WSessions )
Values ( 2000, 25,5)

INSERT INTO langcodes (LangText, LangCode) VALUES ('Afrikaans', 'af')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Arabic (U.A.E.)', 'ar-ae')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Arabic (UAE)', 'ar-ae')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Arabic Egypt', 'ar-eg')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Arabic (Iraqi)', 'ar-ig')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Arabic (Jordan)', 'ar-jo')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Arabic (Kuwait)', 'ar-kw')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Arabic (Lebanon)', 'ar-lb')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Arabic Libya', 'ar-ly')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Arabic (Morocco)', 'ar-ma')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Arabic (Saudi Arabia)', 'ar-sa')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Arabic (Syria)', 'ar-sy')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Arabic (Tunisia)', 'ar-tn')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Arabic (Yemen)', 'ar-ye')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Belarusian', 'be')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Bulgarian', 'bg')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Catalan', 'ca')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Czech', 'cs')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Danish', 'da')
INSERT INTO langcodes (LangText, LangCode) VALUES ('German (unspecified)', 'de')
INSERT INTO langcodes (LangText, LangCode) VALUES ('German (Australia)', 'de-au')
INSERT INTO langcodes (LangText, LangCode) VALUES ('German (Switzerland)', 'de-ch')
INSERT INTO langcodes (LangText, LangCode) VALUES ('German (Lichtenstein)', 'de-li')
INSERT INTO langcodes (LangText, LangCode) VALUES ('German (Luxembourg)', 'de-lu')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Greek', 'el')
INSERT INTO langcodes (LangText, LangCode) VALUES ('English (unspecified)', 'en')
INSERT INTO langcodes (LangText, LangCode) VALUES ('English (Australian)', 'en-au')
INSERT INTO langcodes (LangText, LangCode) VALUES ('English (Belgium)', 'en-be')
INSERT INTO langcodes (LangText, LangCode) VALUES ('English (Canada)', 'en-ca')
INSERT INTO langcodes (LangText, LangCode) VALUES ('English (British)', 'en-gb')
INSERT INTO langcodes (LangText, LangCode) VALUES ('English (Jamaica)', 'en-ja')
INSERT INTO langcodes (LangText, LangCode) VALUES ('English (New Zealand)', 'en-nz')
INSERT INTO langcodes (LangText, LangCode) VALUES ('English (United States)', 'en-us')
INSERT INTO langcodes (LangText, LangCode) VALUES ('English (South Africa)', 'en-za')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Spanish (unspecified)', 'es')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Spanish (Argentina)', 'es-ar')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Spanish (Australia)', 'es-au')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Spanish (Chile)', 'es-cl')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Spanish (Colombia)', 'es-co')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Spanish (Costa Rica)', 'es-cr')

APPENDIX B - MODULE SetUp.sql
INSERT INTO langcodes (LangText, LangCode) VALUES ('Sutu', 'su')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Sami (Lappish)', 'sa')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Thai', 'th')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Tswana', 'tn')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Turkish', 'tr')
GO
INSERT INTO langcodes (LangText, LangCode) VALUES ('Tsonga', 'ts')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Ukrainian', 'uk')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Urdu', 'ur')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Venda', 've')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Vietnamese', 'vi')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Xhosa', 'xh')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Chinese (PRC)', 'zh-cn')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Chinese (Hong Kong)', 'zh-hk')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Chinese (Taiwan)', 'zh-tw')
INSERT INTO langcodes (LangText, LangCode) VALUES ('Zulu', 'zu')
GO
APPENDIX B - MODULE SETUP.sql
```sql
MODULE NAME - PROCEDURE.SQL

USE JPFM

/*** Internet only procedures **/**

/*** Object: Procedure AddFeedback **/**
if exists (select name from sysobjects where name = 'AddFeedback' and type = 'P')
drop procedure AddFeedback
GO

CREATE PROCEDURE AddFeedback @Message varchar(255) = 'SUGGESTION',
    @Subject varchar(255) = 'OTHER',
    @Origin varchar(255) = ' ',
    @Email varchar(255) = ' ',
    @Name varchar(255) = ' ',
    @Phone varchar(255) = ' ',
    @Comments varchar(1024),
    @Admin varchar(255)
AS
BEGIN
    DECLARE @feedbackId int,
    INSERT INTO Feedback (Message, Subject, Origin, Email, Name, Phone, Comments, Name, Admin)
    VALUES(@Message, @Subject, @Origin, @Email, @Name, @Phone, @Comments, @Admin, @Admin) IF len(@Comments) > 10
        BEGIN
            SELECT @FeedbackId = max(FeedbackId) FROM Feedback
        SET @CMD = 'SELECT * FROM Feedback where FeedbackId = ' + @FeedbackId
        EXEC xp_sendmail @recipients = @Admin, @subject = 'JPFM Feedback'
    END
END
GO

/*** Object: Procedure AddUserShort **/**
if exists (select name from sysobjects where name = 'AddUserShort' and type = 'P')
drop procedure AddUserShort
GO

/*** return code meanings **
1 Email address already exists
200 User_id for new entry

CREATE PROCEDURE AddUserShort @user varchar(255),
    @password varchar(255)
AS
    IF EXISTS (SELECT * FROM Users WHERE EmailAddress = @user)
        RETURN 1
    ELSE
        BEGIN
            INSERT INTO Users (EmailAddress, EncryptionPassword, FirstName, OtherIniti, Surname, StartDate, RenewalDate, LastAccessed)
            VALUES(@user, @password, @FirstName, @OtherIniti, @Surname, @webAddress, @affiliation, @RoleId, @UserId)
            SELECT EmailAddress, FirstName, OtherIniti, Surname, RoleId, UserId
            FROM Users WHERE EmailAddress = @user AND EncryptionPassword = @password
        END
END
GO

/*** Object: Procedure AddUserFull **/**
if exists (select name from sysobjects where name = 'AddUserFull' and type = 'P')
drop procedure AddUserFull
GO

/*** return code meanings **
1 Email address already exists
100 User_id for new entry

CREATE PROCEDURE AddUserFull @Email varchar(255),
    @password varchar(32),
    @FirstName varchar(255) = '',
    @OtherIniti varchar(255) = '',
    @SName varchar(255) = '',
    @PhoneNumber varchar(255) = '',
    @EmailAddress varchar(255) = '',
    @OrderId int = 0
AS
    DECLARE @UserOut int
    BEGIN
        SELECT @Out = COUNT(*) FROM Users
        INSERT INTO Users (EmailAddress, EncryptionPassword, FirstName, OtherIniti, Surname, StartDate, RenewalDate, LastAccessed)
        VALUES(@user, @password, @FirstName, @OtherIniti, @Surname, @StartDate, @EndDate, @LastAccessed)
        ELSE
            SELECT EmailAddress, FirstName, OtherIniti, Surname, StartDate, RenewalDate, LastAccessed
            FROM Users WHERE EmailAddress = @user AND EncryptionPassword = @password
        END
END
GO

/*** Object: Procedure DisplayAuthors **/**
if exists (select name from sysobjects where name = 'DisplayAuthors' and type = 'P')
DROP PROCEDURE DisplayAuthors
GO
CREATE PROCEDURE DisplayAuthors @ArticleId char(6) = '000_00',
    @maxCount int = 100
AS
    SET ROWCOUNT @maxCount
    SELECT b.DisplayName, b.EmailAddress, b.WebAddress
    FROM Authors a, Articles b
    WHERE a.AuthorId = b.AuthorId AND ArticleId = @ArticleId;
```
c smallDesc, c.AddressLine1, c.AddressLine2, c.AddressLine3, c.AddressLine4, c.AddressLine5
from ArticleAuthors a, Users b, Institutions c
where a.article_ID = #intArticleID
and a.UserID = b.UserID
and a.Institution_ID = c.Institution_ID
order by c.Institution_ID
SET ROWCOUNT 0
GO

/****** Object: Procedure DispArtDetails ******/
if exists (select name from sysobjects where name = 'DispArtDetails' and type = 'P')
DROP PROCEDURE DispArtDetails
GO
CREATE PROCEDURE DispArtDetails
@intArticleID char(6) = '000_00',
@maxrowCount int = 100
AS
SET ROWCOUNT @maxrowCount
select article_ID, ArticleTitle, ReceiveDate, AcceptDate, Editor_ID, Abstract
from Articles where Article_ID = @intArticleID
SET ROWCOUNT 0
GO

/****** Object: Procedure DispArtDownloads ******/
if exists (select name from sysobjects where name = 'DispArtDownloads' and type = 'P')
DROP PROCEDURE DispArtDownloads
GO
CREATE PROCEDURE DispArtDownloads
@intArticleID char(6) = '000_00',
@maxrowCount int = 100
AS
SET ROWCOUNT @maxrowCount
select article_ID, a.PrintType, b.PrintDesc, b.ImageFile
from ArticlesInFull a inner join PrintFileTypes b on a.PrintType = b.PrintType
where a.article_ID = @intArticleID
SET ROWCOUNT 0
GO

/****** Object: Procedure DispArtKeywords ******/
if exists (select name from sysobjects where name = 'DispArtKeywords' and type = 'P')
DROP PROCEDURE DispArtKeywords
GO
CREATE PROCEDURE DispArtKeywords
@intArticleID char(6) = '000_00',
@maxrowCount int = 100
AS
SET ROWCOUNT @maxrowCount
select Keyword
from ArticleKeywords
where article_ID = @intArticleID
SET ROWCOUNT 0
GO

/****** Object: Procedure DispAuthor ******/
if exists (select name from sysobjects where name = 'DispAuthor' and type = 'P')
DROP PROCEDURE DispAuthor
GO
CREATE PROCEDURE DispAuthor
@intUserID int = 99,
@maxrowCount int = 100
AS
SET ROWCOUNT @maxrowCount
select a.Display_name, a.FirstName, a.OtherInit, a.Surname,
a.EmailAddress, a.WebAddress,
c.smallDesc, c.AddressLine1, c.AddressLine2, c.AddressLine3, c.AddressLine4, c.AddressLine5
from ArticleAuthors a, Users b, Institutions c
where a.article_ID = #intArticleID
and a.UserID = b.UserID
and a.Institution_ID = c.Institution_ID
order by c.Institution_ID
SET ROWCOUNT 0
GO

/****** Object: Procedure DISPAUTHART ******/
if exists (select name from sysobjects where name = 'DISPAUTHART' and type = 'P')
DROP PROCEDURE DISPAGE AUTHART
GO
CREATE PROCEDURE DISPAUTHART
@intUserID int = 99,
@maxrowCount int = 100
AS
SET ROWCOUNT @maxrowCount
select ArticleTitle, Volume_No, Issue_No, IssueYear from TitlesInVolume
where User_id = #intUserID
SET ROWCOUNT 0
GO

/****** Object: Procedure DispArtDownloadList ******/
if exists (select name from sysobjects where name = 'DispArtDownloadList' and type = 'P')
DROP PROCEDURE DispArtDownloadList
GO
CREATE PROCEDURE DispArtDownloadList
@intUserID int = 1
AS
select c.article_id, c.user_id, c.Institution_id, c.AuthorOrder,
d.Display_name, g.ArticleTitle,
h.Volume_id, h.Volume_No, h.Issue_No, h.IssueYear
from
(select a.article_id, a.user_id, a.Institution_id, a.AuthorOrder
from ArticleAuthors a,
| select article_id from ArticlesInFull
| where article_id = @article_id) as Articles
left outer join ArticlesInVolume h on h.article_id = a.article_id
left outer join ArticlesAvailability g on g.Available = 'Y'
inner join users d on c.user_id = d.user_id
order by c.article_id, c.AuthorOrder
SET ROWCOUNT @maxrowCount
GO

/****** Object: Procedure DispArtMathCodes ******/
if exists (select name from sysobjects where name = 'DispArtMathCodes' and type = 'P')
DROP PROCEDURE DispArtMathCodes
GO
CREATE PROCEDURE DispArtMathCodes
@intArticleID char(6) = '000_00',
@maxrowCount int = 100
AS
SET ROWCOUNT @maxrowCount
select MathCode
from ArticleMathCodes
where article_ID = @intArticleID
SET ROWCOUNT 0
GO

APPENDIX B - Module Procedure.sql
Select ImageType, AnImage from ArticleAbstractImages where ArticleId = @s_ArticleId and ImageNumber = @i_ImageNumber
end
GO

//***/ Object: Procedure GetUserDetails *****
//***/ Security the password must be supplied to extract the details *****
if exists (select name from sysobjects where name = 'GetUserDetails' and type = 'P')
DROP PROCEDURE GetUserDetails
GO
Create PROCEDURE GetUserDetails
@intUserId int = 1,
@strOldPass varchar(30) = ''
AS
select DisplayName, FirstName, OtherInit, Surname, EmailAddress, WebAddress, Affiliation
from Users
where UserId = @intUserId and UserPassword = @strOldPass
GO

//***/ Object: Procedure ModUserLong *****
//***/ Security the password must be supplied to extract the details *****
if exists (select name from sysobjects where name = 'ModUserLong' and type = 'P')
drop procedure ModUserLong
GO
**** return code meanings ****
1 User_id does not exist
0 record updated
*****
CREATE PROCEDURE ModUserLong @i_UserId int,
@s_OldPass varchar(32),
@s_email varchar(128),
@s_pword varchar(32),
@s_Firstname varchar(60) = '',
@s_OtherInit varchar(20) = '',
@s_Surname varchar(128) = '',
@s_webaddress varchar(128) = '',
@s_affiliation varchar(60) = '',
@s_DisplayName varchar(128) = ''
AS
BEGIN
IF NOT EXISTS (SELECT * FROM Users WHERE User_id = @i_UserId and UserPassword = @s_OldPass)
Select User_id FROM Users WHERE User_id = @i_UserId and UserPassword = @s_OldPass
ELSE
BEGIN
Update users set EmailAddress = @s_email,
UserPassword = @s_pword,
FirstName = @s_Firstname,
OtherInit = @s_OtherInit,
Surname = @s_Surname,
WebAddress = @s_webaddress,
Affiliation = @s_affiliation,
DisplayName = @s_DisplayName
where User_id = @i_UserId
Select User_id FROM Users WHERE User_id = @i_UserId
END
END
GO

***** Object: Procedure OrderEditorDetails *****

if exists (select name from sysobjects where name = 'OrderEditorDetails' and type = 'P')
DROP PROCEDURE OrderEditorDetails
GO
Create PROCEDURE OrderEditorDetails
AS
select user_id, Displayname
from EditorList
order by displayname
GO

****** Object: Procedure SearchAbstracts ******
if exists (select name from sysobjects where name = 'SearchAbstracts' and type = 'P')
DROP PROCEDURE SearchAbstracts
GO
Create PROCEDURE SearchAbstracts
@strAbstract varchar(60) = 'ABCD%',
@maxrowcount int = 100
AS
SET ROWCOUNT @maxrowcount
select article_id, ArticleTitle, User_id, DisplayName, Volume_id, Volume_NO, Issue_NO, IssueYear from TitlesInVolume
where article_id in
(Select Article_ID from Articles where Abstract like @strAbstract )
SET ROWCOUNT 0
GO

****** Object: Procedure SearchAuthors ******
if exists (select name from sysobjects where name = 'SearchAuthors' and type = 'P')
DROP PROCEDURE SearchAuthors
GO
Create PROCEDURE SearchAuthors
@strDisplayName varchar(60) = 'ABCD%',
@maxrowcount int = 100
AS
SET ROWCOUNT @maxrowcount
select article_id, ArticleTitle, User_id, DisplayName, Volume_id, Volume_NO, Issue_NO, IssueYear from TitlesInVolume
where User_id in
(Select User_ID from Authors where DisplayName like @strDisplayName )
SET ROWCOUNT 0
GO

****** Object: Procedure SearchKeyWords ******
if exists (select name from sysobjects where name = 'SearchKeyWords' and type = 'P')
DROP PROCEDURE SearchKeyWords
GO
Create PROCEDURE SearchKeyWords
@strKeyword varchar(60) = 'ABCD%',
@maxrowcount int = 100
AS
SET ROWCOUNT @maxrowcount
select article_id, ArticleTitle, User_id, DisplayName, Volume_id, Volume_NO, Issue_NO, IssueYear from TitlesInVolume
where article_id in
(Select Article_ID from ArticleKeyWords where KeyWord like @strKeyword )
SET ROWCOUNT 0
GO

****** Object: Procedure SendUserPassword ******
if exists (select name from sysobjects where name = 'SendUserPassword' and type = 'P')
drop procedure SendUserPassword
GO
CREATE PROCEDURE SendUserPassword
@user varchar(128) as
IF NOT EXISTS (SELECT * FROM Users WHERE EmailAddress = @user)
Select 1
ELSE
BEGIN
DECLARE @strPass varchar(32)
select @strPass=UspSendmail @recipient = @user, @subject = 'JIPAM Password', @message = @strPass
select 0
END
GO

****** Object: Procedure UnPubArticles ******
if exists (select name from sysobjects where name = 'UnPubArticles' and type = 'P')
DROP PROCEDURE UnPubArticles
GO
Create PROCEDURE UnPubArticles
GO

APPENDIX B - MODULE PROCEDURE.SQL
CREATE PROCEDURE ValidateUser
    @s_user varchar(128), @s_password varchar(32) AS
BEGIN
    IF EXISTS (SELECT * FROM Users WHERE EmailAddress = @s_user AND UserPassword = @s_password)
        BEGIN
            DECLARE @RenDate date, @ExpDate date
            SELECT @RenDate = RenewalDate from Users WHERE EmailAddress = @s_user
            IF @RenDate < @ExpDate
                UPDATE Users Set RenewalDate = dateadd(mm,12,@RenDate) WHERE EmailAddress = @s_user
            end
            UPDATE Users Set LastAccessed = dateadd(mm,12,@ExpDate) WHERE EmailAddress = @s_user
        END
    SELECT EmailAddress, FirstName, OtherInit1, Surname, UserRole, User_ID FROM Users WHERE EmailAddress = @s_user AND UserPassword = @s_password
END
GO

CREATE PROCEDURE Volumelist
    @intVolumeID int = 1
AS
select Volume_id, VOLUME_NO, ISSUE_NO, ISSUEYear
from VOLUMES
where Volume_id <> @intVolumeID
GO

Grant Execute On AddFeedbackTo internet
Grant Execute On AddUserFull TO internet
Grant Execute On AddUserShort TO internet
Grant Execute On DispArtDetails to internet
Grant Execute On DispArtDownloads to internet
Grant Execute On DispArtKeyWords to internet
Grant Execute On DispArtMathCodes to internet
Grant Execute On DispAuthorTo internet
Grant Execute On DispAuthorTo internet
Grant Execute On DispAuthArtList to internet
Grant Execute On DispVolArticles to internet
Grant Execute On FullEditorList to internet
Grant Execute On GetArticleFull to internet
Grant Execute On GetArticleInMags to internet
Grant Execute On GetUserDetails to internet
Grant Execute On ModEerLong TO internet
Grant Execute On OrderEditorDetails to internet

GO

GO

/***** Object: Procedure AddUserFullAdmin *******/

if exists (select name from sysobjects where name = 'AddUserFullAdmin' and type = 'P')
drop procedure AddUserFullAdmin
GO

/**** return code meanings ****
1 Email address already exists
100 User_id for new entry

CREATE PROCEDURE AddUserFullAdmin
    @s_email varchar(128),
    @s_password varchar(32),
    @s_FullName varchar(60) = '','
    @s_UserName varchar(20) = '','
    @s_Surname varchar(128) = '','
    @s_Webaddress varchar(128) = '','
    @s_Affiliation varchar(60) = '','
    @sUserRole smallint = 0,
    @s_InstutionID smallint = 1 AS
DECLARE @useridOut intger
Select @useridOut = 1
IF EXISTS (SELECT * FROM Users WHERE EmailAddress = @s_email)
    RETURN 1
ELSE
    BEGIN
    INSERT INTO Users (EmailAddress, UserPassword, FirstName, OtherInit1, Surname, WebAddress, Affiliation, UserRole, Institution_ID, StartDate, RenewalDate, LastAccessed)
    VALUES(@s_email, @s_password, @s_FullName, @s_UserName, @s_Surname, @s_Webaddress, @s_Affiliation, @sUserRole, @s_InstutionID, getdate(), dateadd(month,12,getdate()),getdate())
    END
GO

/***** Object: Procedure ModArticle *******/
if exists (select name from sysobjects where name = 'ModArticle' and type = 'P')
drop procedure ModArticle
GO

APPENDIX B – MODULE PROCEDURE.sql
CREATE PROCEDURE ModArticle
   @s_ArticleId char(6)='000000',
   @s_ArticleTitle varchar(255),
   @s_ReceiveDate varchar(32),
   @s_AcceptDate varchar(32),
   @i_Editor_Id int,
   @s_Abstract varchar(6144),
   @s_Published char(1),
   @s_Available char(1)
AS
SET DATEFORMAT dmy
IF NOT EXISTS (SELECT Article_Id FROM Articles WHERE Article_Id = @s_ArticleId)
   BEGIN
      UPDATE Articles set
         ArticleTitle = @s_ArticleTitle,
         ReceiveDate = @s_ReceiveDate,
         AcceptDate = @s_AcceptDate,
         Editor_Id = @i_Editor_Id,
         Abstract = @s_Abstract,
         Published = @s_Published,
         Available = @s_Available
      WHERE Article_Id = @s_ArticleId
   RETURN 0
END
GO

CREATE PROCEDURE AddArticle
   @s_ArticleId char(6)='000000',
   @s_ArticleTitle varchar(255),
   @s_ReceiveDate varchar(32),
   @s_AcceptDate varchar(32),
   @i_Editor_Id int,
   @s_Abstract varchar(6144),
   @s_Published char(1),
   @s_Available char(1)
AS
SET DATEFORMAT dmy
BEGIN
   INSERT INTO Articles (ArticleId, ArticleTitle, ReceiveDate, AcceptDate, Editor_Id, Abstract, Published, Available)
   VALUES (@s_ArticleId, @s_ArticleTitle, @s_ReceiveDate, @s_AcceptDate, @i_Editor_Id, @s_Abstract, @s_Published, @s_Available)
END
GO

CREATE PROCEDURE ModVolume
   @i_VolumeNo smallint,
   @s_IssueNo smallint,
   @s_IssueDate varchar(20)
AS
DECLARE @intVolumeId integer
SET DATEFORMAT dmy
BEGIN
   IF EXISTS (SELECT Volume_id FROM Volumes WHERE Volume_No = @i_VolumeNo and ISSUE_No = @s_IssueNo and ISSUEDate = @s_IssueDate)
      BEGIN
      RETURN 0
      END
ELSE
   BEGIN
      INSERT INTO Volumes (Volume_No, ISSUE_No, ISSUEDate, ISSUEYear)
      VALUES (@i_VolumeNo, @s_IssueNo, @s_IssueDate)
   END
END
GO

CREATE PROCEDURE GetNextArticleId
   @s_ArticleId varchar(4) output
AS
BEGIN
   DECLARE @article_id_id int
   DECLARE @s_ArticleId char(4)
   SET @s_ArticleId = cast(datepart(year, getdate()) as char(4))
   SET @s_ArticleId = @s_ArticleId + year(2) + substr('t', 3, 1) as string from articles where article_id like @s_ArticleId
   SET @article_id_id = article_id = 1001
   SET @s_ArticleId = substr(cast(article_id as char(4), 2, 1) + _ + substr('t', 3, 2)
END
GO

APPENDIX B - MODULE PROCEDURE.SQL
IF Not EXISTS (SELECT Volume_id FROM VOLUMES WHERE Volume_id = #1_VolumeId )
  Return 0
ELSE
BEGIN
  Update VOLUMES set
  Volume_No = #1_VolumeNo,
  ISSUE_No=#1_ISSUE_No,
  ISSUEYear=#1_ISSUEYear,
  ISSUEDate=#1_ISSUEDate
  where Volume_id = #1_VolumeId
END
END
GO

****** Object:  Procedure AddInstitution ******
if exists (select name from sysobjects where name = 'AddInstitution' and type = 'P')
  drop procedure AddInstitution
GO

CREATE PROCEDURE AddInstitution
    @#_SmallDesc varchar(60),
    @#_AddressLine1 varchar(255),
    @#_AddressLine2 varchar(255),
    @#_AddressLine3 varchar(255),
    @#_AddressLine4 varchar(255),
    @#_Telephone varchar(30),
    @#_Facsimile varchar(30)
AS
DECLARE @intInstitutionId integer

SET DATEFORMAT day
BEGIN
IF exists (SELECT Institution_ID FROM Institutions WHERE SmallDesc = @#_SmallDesc)
  Return 0
ELSE
BEGIN
  INSERT INTO Institutions
  (SmallDesc,AddressLine1,AddressLine2,AddressLine3,AddressLine4,AddressLine5,Telephone,Facsimile)
  VALUES (@#_SmallDesc, @#_AddressLine1, @#_AddressLine2, @#_AddressLine3,
  @#_AddressLine4, @#_AddressLine5, @#_Telephone, @#_Facsimile)
END
RETURN @intInstitutionId
END
END
GO

****** The procedures for the statistic tables (DeleteOldRecords and SummarizeSession) are taken from the book Professional ASP Techniques for Webmasters ******

****** Object:  Stored Procedure dbo.DeleteOldRecords ******
if exists (select * from sysobjects where id = object_id(N'[dbo].[DeleteOldRecords]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
  drop procedure [dbo].[DeleteOldRecords]
GO

****** Object:  Stored Procedure dbo.SummarizeSession******
if exists (select * from sysobjects where id = object_id(N'[dbo].[SummarizeSession]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
  drop procedure [dbo].[SummarizeSession]
GO

SET QUOTED_IDENTIFIER OFF
SET ANSI_NULLS ON
GO

****** Object:  Stored Procedure dbo.DeleteOldRecords ******
CREATE PROCEDURE DeleteOldRecords
AS
DECLARE @rowsupdated int
BEGIN TRANSACTION
DELETE
FROM CountrySummary
WHERE (YearNumber = DATEDATE(year, GETDATE()) AND TWeekNumber < (DATEDATE(week, GETDATE()) - 25))
OR (YearNumber < DATEDATE(year, GETDATE()) AND TWeekNumber < (DATEDATE(week, GETDATE()) + 26))
APPENDIX B - MODULE PROCEDURE.SQL

IF (@ERROR <> 0) GOTO on_error
DELETE FROM RefererSummary WHERE @YearNumber = DATEPART(year, GETDATE()) AND @WeekNumber < [DATEPART(week, GETDATE()) + 26]
IF (@ERROR <> 0) GOTO on_error
DELETE FROM SessionTargetSummary WHERE @YearNumber = DATEPART(year, GETDATE()) AND @WeekNumber < [DATEPART(week, GETDATE()) + 26]
IF (@ERROR <> 0) GOTO on_error
DELETE FROM UserAgentSummary WHERE @YearNumber = DATEPART(year, GETDATE()) AND @WeekNumber < [DATEPART(week, GETDATE()) + 26]
IF (@ERROR <> 0) GOTO on_error
SELECT 'Complete Deletion' COMMIT TRANSACTION RETURN (0) on_error: SELECT 'Error - deletion aborted.' ROLLBACK TRANSACTION RETURN(1)
GO
SET QUOTED_IDENTIFIER OFF SET ANSI_NULLS ON GO
GRANT EXECUTE ON [dbo].[DeleteOldRecords] TO [jipanLog]
GO
SET QUOTED_IDENTIFIER ON SET ANSI_NULLS ON GO
/***** Object: Stored Procedure dbo.SummariseSession Script Date: 27/04/98 09:22:37 ******
CREATE proc SummariseSession as
DECLARE @ToYear int
DECLARE @ToWeek int
DECLARE @ThisYear int
DECLARE @ThisWeek int
DECLARE @Sessions int
BEGIN TRANSACTION
SELECT @ToYear = DATEPART(year, GETDATE()), @ToWeek = DATEPART(week, GETDATE())
SELECT @ThisYear = MAX(@YearNumber) FROM WeekSummary
SELECT @ThisWeek = MAX(@WeekNumber) FROM WeekSummary
WHERE @YearNumber = @ThisYear
SELECT @ThisWeek = @ThisWeek + 1 IF (@ThisWeek > 53)
BEGIN
SELECT @ThisWeek = 1
SELECT @ThisYear = @ThisYear + 1
END
WHERE @ThisYear = @ToYear AND @ThisWeek < @ToWeek OR (@ThisWeek < @ToYear)
BEGIN
SELECT @Sessions = COUNT(EventDateTime)
FROM Sessions WHERE DATEPART(year, EventDateTime) = @ThisYear AND DATEPART(week, EventDateTime) = @ThisWeek AND EventType = 'New Session'
IF (@ERROR <> 0) GOTO on_error
INSERT INTO WeekSummary SELECT @ThisYear, @ThisWeek, ISNULL(@Sessions, 0)
IF (@ERROR <> 0) GOTO on_error
SELECT @ThisWeek = @ThisWeek + 1 IF (@ThisWeek > 53)
BEGIN
SELECT @ThisWeek = 1
SELECT @ThisYear = @ThisYear + 1
END
/***** start loop over again *****/ END
INSERT INTO RefererSummary SELECT RefYear = MAX(DATEPART(year, EventDateTime)), RefWeek = MAX(DATEPART(week, EventDateTime)), RefUrl=MAX( CASE WHEN CHARINDEX('?', Referrer) > 5 THEN SUBSTRING(Referrer, 1, CHARINDEX('?') - 1), Referrer) - 1 ELSE Referrer END), RefCount=COUNT(Referrer), RefHostIP = MAX(HostIP) FROM Sessions WHERE DATEDIFF(week, EventDateTime, GETDATE()) > 0 AND Referrer IS NOT NULL AND Referrer <> '' AND CHARINDEX('jipan', Referrer) = 0 AND CHARINDEX('file', Referrer) = 0 GROUP BY CASE WHEN CHARINDEX('?', Referrer) > 5 THEN SUBSTRING(Referrer, 1, CHARINDEX('?', Referrer) - 1) ELSE Referrer END, DATEDIFF(week, EventDateTime), DATEPART(week, EventDateTime), HostIP
IF (@ERROR <> 0) GOTO on_error
INSERT INTO SessionTargetSummary

SELECT TarYear = MAX(DATEPART(year, EventDateTime)),
    TarWeek = MAX(DATEPART(week, EventDateTime)),
    TarText = MAX(Text),
    TarCount = COUNT(Text),
    TarHostIP = MAX(IP)
FROM Sessions
WHERE DATEDIFF(week, EventDateTime, GETDATE()) > 0
AND URL IS NOT NULL
AND URL = '"
GROUP BY URL,
    DATEPART(year, EventDateTime),
    DATEPART(week, EventDateTime),
    HostIP
IF (@@ERROR <> 0) GOTO on_error

INSERT INTO UserAgentSummary
SELECT UAYear = MAX(DATEPART(year, EventDateTime)),
    Uaweek = MAX(DATEPART(week, EventDateTime)),
    UText = MAX(Text),
    UCount = COUNT(UserAgent),
    UHostIP = MAX(IP)
FROM Sessions
WHERE DATEDIFF(week, EventDateTime, GETDATE()) > 0
AND UserAgent IS NOT NULL
AND UserAgent = '"
GROUP BY UserAgent,
    DATEPART(year, EventDateTime),
    DATEPART(week, EventDateTime),
    HostIP
IF (@@ERROR <> 0) GOTO on_error

INSERT INTO CountrySummary
SELECT CoYear = MAX(DATEPART(year, EventDateTime)),
    CoWeek = MAX(DATEPART(week, EventDateTime)),
    CoText = MAX(Text),
    CoCount = COUNT(UA LANGUAGE),
    CoHostIP = MAX(IP)
FROM Sessions
WHERE DATEDIFF(week, EventDateTime, GETDATE()) > 0
AND UA LANGUAGE IS NOT NULL
AND UA LANGUAGE = '"
GROUP BY UA LANGUAGE,
    DATEPART(year, EventDateTime),
    DATEPART(week, EventDateTime),
    HostIP
IF (@@ERROR <> 0) GOTO on_error

DELETE
FROM Sessions
WHERE DATEDIFF(week, EventDateTime, GETDATE()) > 0
IF (@@ERROR <> 0) GOTO on_error

COMMITS TRANSACTION
SELECT 'Summary processed.'
RETURN (1)

on_error:
ROLLBACK TRANSACTION
SELECT 'Error - summary aborted.' RETURN (1)
GO
SET QUOTED_IDENTIFIER OFF
SET ANSI_NULLS ON
GO
GRANT EXECUTE ON [dbo].[SummariseSession] TO [JipamLog]
GO

**** Create a Job to automatically run statistic table summarisation and clean up

USE MDDB
**** must be in this database *****/
if exists (Select * from sysjobs where name = 'Weekly JIPAM Session Update2')
begin
exec sp_delete_job @job_name = 'Weekly JIPAM Session Update2'
end
EXEC sp_add_job @job_name = 'Weekly JIPAM Session Update2',
    @enabled = 1,
    @description = 'weekly update of summary tables'
EXEC sp_add_jobstep @job_name = 'Weekly JIPAM Session Update2',
    @step_name = 'Delete records older than 26 weeks',
    @subsystem = 'TSQL',
    @database_name = 'JIPAM',
    @command = 'exec DeleteOldRecords',
    @on_success_action = 3,
    @retry_attempts = 5,
    @retry_interval = 5
EXEC sp_add_jobstep @job_name = 'Weekly JIPAM Session Update2',
    @step_name = 'Weekly Update of Summary Tables',
    @subsystem = 'TSQL',
    @database_name = 'JIPAM',
    @command = 'exec SummariseSession',
    @retry_attempts = 5,
    @retry_interval = 5
EXEC sp_add_jobschedule @job_name = 'Weekly JIPAM Session Update2',
    @name = 'UpdatedSession',
    @freq_type = 4,
    @freq_interval = 1,
    @active_start_time = 130000
APPENDIX C

JIPAM WEB SITE
Dynamic Database System
Web Server ASP Code
APPENDIX C - MODULE ACCEPTED.ASP
MODULE NAME – AIMSSCOPE.ASP

<? Language=VBScript ?>
<HTML>
<meta NAME="robots" CONTENT="noindex, nofollow">
<!-- #include virtual="/jipam/common/common.inc" -->
<HEAD>
<TITLE>Aims and Scope</TITLE>
</HEAD>
<Dim PageName=
PageName="SCOPE"
',//
<!-- #include virtual="/jipam/common/Mainheader.inc" -->
<!-- #include virtual="/jipam/common/menu.inc" -->
<HL ALIGN="center">
<FONT COLOR="#FF0000">Aims and Scope</FONT>
</HL>
<div align="left">
<table border="0" cellpadding="17" cellspacing="0" width="1008">
<tr>
<td width="1008">
<HL ALIGN="center" color="#FF0000">Aims and Scope</HL>
<p ALIGN="left">The peer-reviewed electronic Journal of Inequalities in Pure and
Applied Mathematics (JIPAM) aims to foster and develop further growth in
all areas of mathematics relating to inequalities. The journal accepts
high quality papers containing original research results, survey
articles of exceptional merit, short letters and notes. The journal
recognizes the need for papers to be published in a timely fashion and
therefore papers will be refereed within 60100 working days of
submission. Refereed papers will be refereed within 45 working
days of submission. The journal encourages <a href="/authors.asp#Submissions">submissions</a>
in electronic form, typed form or clearly handwritten English. To
encourage clear and succinct papers, the journal will award, annually, a
<br><br>
<HL ALIGN="center" color="#FF0000">Best-Published Paper</HL>
The <a href="/jipam.vu.edu.au/BestMimWeb.html">
<HL ALIGN="center" color="#FF0000">Editor-in-Chief</HL></a>
together with a panel of international experts will judge the published
papers.</p>
<p ALIGN="left">JIPAM will initially be published twice a year in March
and October.</p>
</td>
</tr>
</table>
</div>
</html>
<!-- #include virtual="/jipam/common/Footer.inc" -->

APPENDIX C – MODULE AIMSSCOPE.ASP
**MODULE NAME - COMMON.INC**

```vbnet
<
Dim strConnection
Dim strConnLogs
Dim MaxLines
Dim MathCodeUrl
Dim LatestVoluId
LatestVoluId=Application.Contents.Item("LatestVoluId")
strConnection = Application.Contents.Item("strConnection")
strConnLogs = Application.Contents.Item("strConnLogs")
MaxLines=Application.Contents.Item("MaxLines")

Function IsAdmin()
    IsAdmin = false
    If Session("UserID")>0 and Session("UserRole") >= 4 then IsAdmin=true
end Function

Function IsloggedIn()()
    IsloggedIn=false
    If Session("UserID")>0 then IsloggedIn=true
end Function

>
```

**APPENDIX C - MODULE COMMON.INC**
Module Name - CONTACTUs.asp

<html><head>
<meta name="robots" content="noindex, nofollow">
</head>
<body>

Dim PageName
PageName = "CONTACT"

<!-- include virtual="/jipan/common/Include.inc" -->
<!-- include virtual="/jipan/common/menu.inc" -->
<blockquote>

<table border="0" cellpadding="0" cellspacing="0" bgcolor="#FFFFFF"

<td width="100%">
<align=center><font color="#FF0000">Contact</font></align>
<table border="0" cellpadding="0" cellspacing="0" bgcolor="#FFFFFF"

<td width="100%">
<align=center><font color="#FF0000">Contact</font></align>
<table border="0" cellpadding="0" cellspacing="0" bgcolor="#FFFFFF"

</table>
</table>
</td>
</table>
</blockquote>
</body></html>
<p align="left">All copies, paper or electronic, or other use of the work must include an indication of the copyright ownership and a full citation of the journal source. Please refer to requests for all uses not included above, including the authorization of third parties to reproduce or otherwise use all or part of the work (including figures and tables) to:</p>

<p align="left">Assistant Editor &nbsp;&nbsp; Editor &nbsp;&nbsp; </p>

<p align="left">Journal of Inequalities in Science and Mathematics &nbsp;&nbsp; Applied Mathematics &nbsp;&nbsp; School of Mathematics &nbsp;&nbsp; Informatics (POL) &nbsp;&nbsp;</p>

<p align="left">Victoria University of Technology &nbsp;&nbsp; PO Box 14628 &nbsp;&nbsp; Melbourne, Victoria 8001, AUSTRALIA</p>

<p align="left">&nbsp;&nbsp; City MC &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nte
MODULE_NAME - DEFAULT.ASP

QUALITY, JIPAM will offer an annual <A HREF="authors.asp">prize</A> to the best article.</P>
</LI>
</UL>
<P ALIGN="left">Finally, we hope you enjoy reading the articles in this first issue and we ask that you keep JIPAM in mind when considering your next publication.</P>
</LI>
</UL>
</BODY></HTML>

APPENDIX C - MODULE DEFAULT.ASP
Module Name - DISPARTICLE.ASP

```vbscript
Dim strArticleID
Dim strEditorName
Set oRs = nothing
strArticleID=Request.QueryString("AI")
if len(strArticleID) < 1 then Response.Write("000_00")
'create and open a connection to the database
Set oCon = Server.CreateObject("ADODB.Connection")
oCon.Open strConnection
'create a command and set the properties
Set oCmd = Server.CreateObject("ADODB.Command")
Set oCmd = Server.CreateObject("ADODB.Command")
oCmd.ActiveConnection = oCon
oCmd.CommandType = 4
'stored procedure
oCmd.CommandText = "DispArtDetails" 'procedure name
logRecAffected = 0
'execute the command, supplying the parameters, and get the result
Set oRs = oCmd.Execute(imgRecAffected, Array(strArticleID, MaxLines))
if oRs.BOF then
  Response.Write("The article requested is not yet
published." <!--<CENTER>-->"
  Response.End
end if
'get article details
strArticleTitle = oRs.Fields("ArticleTitle")
strReceivedDate = oRs.Fields("ReceiveDate")
strAcceptDate = oRs.Fields("AcceptDate")
intEditorID = oRs.Fields("Editor_ID")
strAbstract = ExpandAbstractString(oRs.Fields("Abstract"))
'need to expand the abstract for images
'...
</CENTER></b2>
</p>
</table WIDTH="804" BORDER="0" />
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</body>
</html>
```

Appendix C - Module DISPARTICLE.ASP
if not oRs3.EOF then
  strKeyword = oRs3.Fields("Keyword")
  Response.Write("<table WIDTH="604">\TR|<TR>\TABLE|<P>|<P>|\TR>");
  oRs3.MoveNext
  do while not oRs3.EOF
    strKeyword = oRs3.Fields("Keyword")
    Response.Write("<TR|<TR>\TABLE|<P>|<P>|\TR>");
    oRs3.MoveNext
  loop
    Response.Write("<I|</I>|</TR|</TR>|</TABLE|<P>|</P>");
  end if
oRs3.Close
Set oRs3 = Nothing
Set oCmd3 = nothing

's math codes
Set oCmd4 = Server.CreateObject("ADODB.Command")
oCmd4.ActiveConnection = oCon
oCmd4.CommandType = 4 'stored procedure
oCmd4.CommandText = "DispArtMathCodes"  'procedure name
Set oRs4 = oCmd4.Execute( lngRecsAffected, Array (strArticleId, MaxLines))

if not oRs4.EOF then
  strMathCode = oRs4.Fields("MathCode")
  Response.Write("<TR|<TR>\TABLE|<P>|<P>|\TR>");
  oRs4.MoveNext
  do while not oRs4.EOF
    strMathCode = oRs4.Fields("MathCode")
    Response.Write("<TR|<TR>\TABLE|<P>|<P>|\TR>");
    oRs4.MoveNext
  loop
    Response.Write("<I|</I>|</TR|</TR>|</TABLE|<P>|</P>");
  end if
oRs4.Close
Set oRs4 = Nothing
Set oCmd4 = nothing

'Down loads
Set oCmd5 = Server.CreateObject("ADODB.Command")
oCmd5.ActiveConnection = oCon
oCmd5.CommandType = 4 'stored procedure
oCmd5.CommandText = "DispArtDownloads"  'procedure name
Set oRs5 = oCmd5.Execute(lngRecsAffected, Array (strArticleId, MaxLines))

if not oRs5.EOF then
  <hr>
  <p align="left">Download this article :<br>
  <table border="0" width="1008" cellpadding="5">\TR|<TR>
</p>
</td width="504">\TR>
</table>
</p>
</td>
</tr>
</table>

APPENDIX C - MODULE DispArticle.asp
APPENDIX C - MODULE DISPA RTICLE.ASP
 MODULE NAME - DISPAuthTitle.ASP

<? Language=VBScript ?>
<HTML>
<meta NAME="robots" CONTENT="noindex, nofollow">
<!-- #include virtual="/jiapm/common/common.inc" -->
<HEAD>
<title>Display Details of Author JIPAM</title>
</head>

Dim PageName
PageName="DISPLAYAUTH"

<!-- #include virtual="/jiapm/common/Mainheader.inc" -->
<!-- #include virtual="/jiapm/common/menu.inc" -->

<center>
<div>
< CPLIST>
<div>
< CTITLE>
<div>
< A HREF="/DISPARTICLE.ASP?AI=4-strArticleId">4-strArticleTitle</A>

if len(strVolume) > 0 then
, <A HREF="/DISVOLUME.ASP?V1=4-strVolume&amp;V2=4-stringYear">4-stringYear</A>
else
Response.Write("Not yet published.")
end if
</div>
</div>
</div>
</CTITLE>
</div>
</center>

Dim intUserID
intUserID=Request.QueryString("UI")
if len(intUserID) < 1 then intUserID=1 'this will fail as user_id start at 101
'create and open a connection to the database
Set oCon = Server.CreateObject("ADODB.Connection")
oCon.Open strConnectionString
'create a command and set the properties
Set oCmd = Server.CreateObject("ADODB.Command")
oCmd.ActiveConnection = oCon
oCmd.CommandType = 4 'stored procedure
oCmd.CommandText = "DispAuthfArtical" 'procedure name
if oCmdAffected>0 then
execute the command, supplying the parameters, and get the result
Set oRs = oCmd.ExecutelngRecAffected, Array(intUserID, MaxLines)
if oRs.EOF then
Response.Write("<p>The requested author is not available.</p>!"

<!-- #include virtual="/jiapm/common/Footer.inc" -->
</div>
</ center>
</body>
</HTML>
if strDisplayName = Orsl.Fields("Displayname") then 'the author being listed
strAuthorList=strAuthorList & strDisplayName
else
strAuthorList=strAuthorList & "<A HREF=DISPAUTHTITLE.ASP?User_id= & Orsl.Fields("User_id") & "> & Orsl.Fields("Displayname") & "</A>
end if
else ' a new record
DisplayAnArticle
if strDisplayName = Orsl.Fields("Displayname") then 'the author being listed
strAuthorList=strDisplayName
else
strAuthorList=strAuthorList & "<A HREF=DISPAUTHTITLE.ASP?User_id= & Orsl.Fields("User_id") & "> & Orsl.Fields("Displayname") & "</A>
end if
strArticleId = Orsl.Fields("Article_id")
strArticleTitle=Orsl.Fields("ArticleTitle")
intVolumeNo = Orsl.Fields("Volume_id")
strVolume = Orsl.Fields("Volume_MG")
strIssue = Orsl.Fields("issue_MG")
strIssueYear = Orsl.Fields("IssueYear")
Orsl.MoveNext
loop
DisplayAnArticle
</UL>
</TD>
</TR>
</TABLE>
<!-- #include virtual="/jipem/common/Footer.inc" -->

APPENDIX C - MODULE DISPAUTHTITLE.ASP
MODULE NAME - DISPVolume.asp

<T> Language=VBScript >
<T>
<META NAME="robots" CONTENT="noindex, nofollow">
<!-- #include virtual="/jipam/common/common.inc" -->
<HEAD>
<"title">Display articles in volume</title>
</HEAD>
<HR>
<% Dim PageName
PageName="DISPLAYVOL*"
-->
<!-- #include virtual="/jipam/common/Mainheader.inc" -->
<!-- #include virtual="/jipam/common/menu.inc" -->
<%
Dim intVolId
intVolId = Request.QueryString("V")
if intVolId < 1 or intVolId > LatestVolId then
intVolId = LatestVolId
end if
'create and open a connection to the database
Set oCon = Server.CreateObject("ADO(Connection)"
OCon.Open strConnection
'create a command and set the properties
Set oCmd = Server.CreateObject("ADOCommand")
oCmd.ActiveConnection = oCon
oCmd.CommandType = 4 ' stored procedure
oCmd.CommandText = "DispVolArticles" 'procedure name
lngRecsAffected = 0
'execute the command, supplying the parameters, and get the result
Set ors = ocmd.Execute(lngRecsAffected, Array(intVolId, Machines))
if ors.EOF then 'this should never happen but ....
<P>
Sorry, the volume requested has not been published</P>
</%>"#include virtual="/jipam/common/Footer.inc" -->
<%
Response.End
end if
<%
<"TD WIDTH="100%" VALIGN="top" ALIGN="left"><H ALIGN="center"><FONT COLOR="#FF0000"></FONT>volume <%# ors.Fields("Volume_NO") %>:
<%# ors.Fields("Volume_YEAR") %>
</H></TD>
<"DIV ALIGN="center">
<"TABLE BORDER=1 CELLPADDING=5 WIDTH="100%" CELLSIZING="15">
<"TR>
<% intArticleNo = 1
strAuthorList = ""
strAuthorId = ""
<% while not ors.EOF
strAuthor = ors.Fields("DisplayName")
strAuthorId = ors.Fields("User_ID")
if intArticleNo = ors.Fields("ArticleNO") then
just another author
else
if len(strAuthorList) then '
complete the previous article
<"TD ALIGN="center"><%!strAuthorList%>
</TD>
<% strAuthorList = "" 
strAuthorId = "" 
strAuthor = "" 
end if
<% intArticleNo = ors.Fields("ArticleNO")
strArticleTitle = ors.Fields("ArticleTitle")
strArticleId = ors.Fields("Article_ID")
<% end if
<% if of new article
ors.Next
loop
Set ors=nothing
Set ocmd=nothing
'complete author list
<% strAuthorList = " & ors.Fields("Author") & " & strAuthorId & 
strAuthor & 
end if
<% if LatestVolId > 1 then
<"BR>
<"DIV ALIGN="center">
<NOBR>Other Volumes:<BR@qqasp>\<HR>
Set oCmd = Server.CreateObject("ADOCommand")
oCmd.ActiveConnection = oCon
oCmd.CommandType = 4 ' stored procedure
oCmd.CommandText = "VOLumeList" 'procedure name
lngRecsAffected = 0
'execute the command, supplying the parameters, and get the result
Set ors = oCmd.Execute(lngRecsAffected, Array(intVolId))
Do While NOT ors.EOF
strVolume = "V " & ors.Fields("Volume_NO") 
" & ors.Fields("Issue_NO") 
" & ors.Fields("Issue_YEAR")
%# ors.Fields("Issue_YEAR")%>
<% end if
</DIV>
</DIV>
</DIV>
</DIV>

APPENDIX C - MODULE DISPVolume.asp
<% Language=VBScript %>
"HTML"
<meta NAME="robots" CONTENT="noindex,nofollow">
<!-- #include virtual="/jijpan/common/common.inc" -->
<!--#HEAD-->
<title>Perform Search or JIPAM</title>
</head>
<body>
<p>Dim PageName
PageName="DOSEARCH"
</p>
<!-- #include virtual="/jijpan/common/Mainheader.inc" -->
<!-- #include virtual="/jijpan/common/menu.inc" -->
<!-- #include virtual="/jijpan/common/search.inc" -->
<!-- #include virtual="/jijpan/common/footer.inc" -->
<p>Dim strSearch
Dim intOption
Dim strSearchA
Dim strExecuteOption
Dim lngRecAffected

strSearch=Request.Form("TxTextSearch")
intOption=Request.Form("RdoSearch")
if Len(strSearch) < 1 then
strSearch = "ABCD"
end if
if intOption < 0 or intOption > 5 then
intOption=1
end if
strSearch = ";"+strSearch+";"
Select Case intOption
Case 1
strExecuteOption="Authors"
strExecuteOption=SearchAuthors"
Case 2
strExecuteOption="Title"
strExecuteOption=SearchTitles"
Case 3
strExecuteOption="Keyword"
strExecuteOption=SearchKeyWords"
Case 4
strExecuteOption="Math Code"
strExecuteOption=SearchMathCode"
Case 5
strExecuteOption="Abstracts"
strExecuteOption=SearchAbstracts"
Case Else
intOption=1
strExecuteOption="Authors"
strExecuteOption=SearchAuthors"
end Select
</p>
<p>&lt;H1 ALIGN="center"&gt;
&lt;FONT COLOR="#FF0000">Select Search &lt;/FONT&gt;&lt;FONT COLOR="#FF0000">&lt;/FONT&gt;
results &lt;/H1&gt;
</p>
<p>create and open a connection to the database
Set oCon = Server.CreateObject("ADODB.Connection")</p>
</body>
</html>
strVolume="<A HREF=DISPVOLUME.ASP?VI=" & intVolumeNo & ">" & strVolume &"</A>"
end if
Orr.MoveNext
loop
' complete the outstanding record
%
</TR>
<TD WIDTH="70%">&nbsp;&nbsp;<%strArticleTitle%></TD>
<TD WIDTH="20%">&nbsp;&nbsp;<%strAuthorList%></TD>
<TD WIDTH="10%">&nbsp;&nbsp;%strVolume%</TD>
</TR>
</table>

<!-- #include virtual="/jipam/common/Footer.inc" -->

APPENDIX C - MODULE DoSEARCH.ASP
Module Name - DownArticle.asp

```vbscript
<%@ Language=VBScript %>
<%  'Response.buffer=true
%>
<-- include virtual="/jipam/common/common.inc" -->
<-- include virtual="/jipam/common/error.inc" -->
<%  ' this process returns a binary file
Dim PageName
Dim strArticleId
Dim intPrintType
Sub TermMess(Mess)
%>
<HTML>
<meta NAME="robots" CONTENT="noindex,nofollow">
<HEAD>
<TITLE>Down Map/TITLE>
</HEAD>
<-- include virtual="/jipam/common/Mainheader.inc" -->
<-- include virtual="/jipam/common/menu.inc" -->
</HEAD><ct>
<-- include virtual="/jipam/common/Footer.inc" -->
<%  Response.End
end sub

strArticleId = Request.QueryString("Ai")
intPrintId = Request.QueryString("PI")
if not IsLoggedIn() or len(strArticleId) <> 6 or intPrintId < 1 then
  TermMess "Sorry, you cannot download the requested article because you have not
logged on."
else Response.End
end if
On Error Resume Next
Err.Clear()
' create and open a connection to the database
Set oCon = Server.CreateObject("ADODB.Connection")
if Err.number > 0 then TermMess "Unable to connect to database(1)"
Err.Clear ()
oCon.Open strConnection
if Err.number > 0 then TermMess "Unable to connect to database(2)"
' create a command and set the properties
Set oCmd = Server.CreateObject("ADODB.Command")
if Err.number > 0 then TermMess "Unable to connect to database(3)"
oCmd.ActiveConnection = oCon
oCmd.CommandType = 4 ' stored procedure
oCmd.CommandText = "GetArticleFull" procedure name
lngRecsAffected = 0
arrParams=Array(strArticleId, intPrintId)
execute the command, supplying the parameters, and get the result:
Err.Clear ()
Set oRs = oCmd.Execute(lngRecsAffected, arrParams )
if Err.number > 0 then TermMess "Unable to connect to database(4)"
if oRs.EOF then
  TermMess "Sorry, the requested article is not available for download."
else Response.End
end if
strHtmlResponse=oRs.Fields("HtmlDescription")
Response.ContentType = strHtmlResponse
```

' Response.Flush() Set adoFildBlob=adoFsDlBlob. ActualSize
lngBlockCount=lngFieldDataLength / 4096
lngRemainingData=lngFieldDataLength mod 4096
' Response.Buffer = false
for lngCounter=1 to lngBlockCount
Response.BinaryWrite (adoFildBlob.GetChunk(4096))
next
if lngRemainingData > 0 then
  Response.BinaryWrite (adoFildBlob.GetChunk(lngRemainingData))
end if
adoFildBlob.Close
set adoFildBlob = nothing
set oRs=nothing
set oCmd=Nothing
Response.End
```
Appendix C - Module Editors.asp
<%@ Language="VBScript" %>
<HTML>
<meta NAME="robots" CONTENT="noindex,nofollow">
<!-- INCLUDE virtual="/jipam/common/common.inc" -->
</HEAD>
<TITLE>Feedback</TITLE>
</HEAD>

' Dim PageName
PageName = "FEEDBACK"

' <!-- INCLUDE virtual="/jipam/common/MainHeader.inc" -->
' <!-- INCLUDE virtual="/jipam/common/menu.inc" -->

<h1 align="center"><font color="#000000">Feedback</font></h1>

<%= @theMess %>

</div>

</div>

</div>

</div>

</div>

</body>
</html>
MODULE NAME - GLOBAL.ASA

<SCRIPT LANGUAGE=VBScript RUNAT=Server>

' You can add special event handlers in this file that will get run automatically when
'special Active Server Pages events occur. To create these handlers, just create a
'subroutine with a name from the list below that corresponds to the event you want to
'use. For example, to create an event handler for Session_OnStart, you would put the
'following code into this file (without the comments):
'Sub Session_OnStart
'****Put your code here **
'End Sub

'EventName Description
'Session_Onstart Runs the first time a user runs any page in your application
'Session_OnEnd Runs when a user's session times out or quits your application
'Application_OnStart Runs once when the first page of your application is run for
'the first time by any user
'Application_onEnd On Error Resume Next

Application.OnStart() -m SQL = "INSERT INTO Sessions (GetDate(),EventType) VALUES
(GetDate(), 'Application Start')"
WriteSessionData strSQL

Application. Contents. Item("EmailAdmin") = "EmailAdmin"
Application. Contents. Item("strConnection") = "DSN=JIPAM; UID=Internet; PWD=Internet; DATABASE
"SE=JIPAM"
Application. Contents. Item("strConnLog") = "DSN=JIPAM; UID=JipamLog; PWD=JipamLog; DATABASE
"JIPAM"
Application. Contents. Item("MaxLines") = 30
Application. Contents. Item("LatestVolume") = 1
Application. Contents. Item("LoginAttempts") = 4

End Sub

Sub Session_onstart()
' The variables to indicate a successful login
Dim UserLogin
Dim UserID
Dim UserRole
Dim Locked
Dim UserPass
Dim LoginCount
UserRole=0
Locked=False
User1d=0
UserPass="" LoginCount=Application. Contents. Item("LoginAttempts")

On Error Resume Next
strSQL = "INSERT INTO Sessions "

' (GetDate(), EventType, URL, Referer, RemoteHost, UserAgent, ULDAP, UserID, HostIP)
' -
' VALUES (GetDate(), 'New Session', '',
' & Request. ServerVariables("URL") & ",", "
' & Request. ServerVariables("HTTP_REFERER") & ",", "
' & Request. ServerVariables("REMOTE_HOST") & ",", "

WriteSessionData strSQL
End Sub

Sub Application_onEnd()
On Error Resume Next
strSQL = "UPDATE Sessions (GetDate(),EventDateTime,EventType, VALUES
(GetDate(), 'Application End')"
WriteSessionData strSQL
End Sub

Sub Session_onEnd()
On Error Resume Next
strSQL = "UPDATE Sessions (GetDate(),EventType, UserID) "

End Sub

Sub WriteSessionData(sql)
On Error Resume Next
Set oConn = Server.CreateObject("ADODB.Connection")
oConn. open "DSN=JIPAM; UID=JipamLog; PWD=JipamLog;"
oConn. Execute SQL
Set oConn = Nothing

End Sub

</SCRIPT>

APPENDIX C - MODULE GLOBAL.ASA
$\text{Response.Write "(<TD>"$}
$\text{For i=1 To 3}$
$\text{If oRs.EOF Then}$
$\text{Response.Write "(<TD>&nbsp;&lt;/TD&gt;&lt;TD&gt;&nbsp;&lt;/TD&gt;&lt;TD&gt;&nbsp;&lt;/TD&gt;"}$
$\text{Else}$
$\text{Response.Write "(<TD NOWRAP ALIGN=RIGHT>" & oRs("LangCount") & "</TD>&nbsp;&lt;/TD&gt;&lt;TD NOWRAP&gt;&nbsp;&lt;/TD&gt;&lt;TD NOWRAP&gt;&nbsp;&lt;/TD&gt;"}$
$\text{End If}$
$\text{oRs.MoveNext}$
$\text{Next}$
$\text{Response.Write "</TD>" & vbCrLf}$
$\text{Loop}$
$\text{Set oConn = Nothing}$
$\text{)}</TABLE></CENTER></BR>
$\text{<!-- Include virtual="/jipam/common/Footer.inc" -->}$

**APPENDIX C - MODULE GRAPHICLANG.ASP**
&lt;%@LANGUAGE=VBScript%&gt;
&lt;% Server.ScriptTimeout = 600 %&gt;
&lt;!-- Include virtual="/jppam/common/common.inc" --&gt;
&lt;%
binIE4 = False
strUA = Request.ServerVariables("HTTP_USER_AGENT")
intMSIE = instr(strUA, "MSIE ") + 5
If intMSIE &gt; 5 Then
  If CInt(Mid(strUA, intMSIE, Instr(intMSIE, strUA, ",") - intMSIE)) &gt;= 4 Then
    binIE4 = True
End If
End If

&lt;HTML&gt;
&lt;META NAME="robots" CONTENT="noindex,nofollow">
&lt;HEAD&gt;
&lt;TITLE&gt;Results of your traffic query&lt;/TITLE&gt;
&lt;/HEAD&gt;
&lt;Dim PageName
PageName="TRAFFICOS"

&lt;!-- Include virtual="/jppam/common/Mainheader.inc" --&gt;
&lt;!-- Include virtual="/jppam/common/menu.inc" --&gt;
&lt;%
If Not IsAdmin() then
  Response.Write("&lt;b&gt;Sorry, this function is not available &lt;/b&gt;")
End If

&lt;!-- Include virtual="/jppam/common/Footer.inc" --&gt;
&lt;%
Response.End
End if

&lt;IL ALIGN="center"&gt;
&lt;FONT COLOR="#FF0000"&gt;&lt;/FONT&gt;raphical &lt;FONT COLOR="#FF0000"&gt;&lt;/FONT&gt;erceing &lt;FONT COLOR="#FF0000"&gt;&lt;/FONT&gt;ystems &lt;/HL&gt;
&lt;%
QOT = Chr(34)
strWhere = " WHERE " + Request.Form("criteria")
strDisplay = Request.Form("display")

&lt;SPAN&gt;Breakdown of operating systems for visitors to our site&lt;/SPAN&gt;&lt;/SPAN&gt;&lt;/SPAN&gt;&lt;/P&gt;

On Error Resume Next
Set oConn = Server.CreateObject("ADODB.Connection") ' create a connection
Set oSql = Server.CreateObject("ADODB.Recordset") ' create a recordset
oConn.Open strConnLogos 'open connection using value in include file
Function getOSTypeCount(strUADetail)
strWhere = strWhere & strUADetail
strSQL = "SELECT HitCount,SUM(SystemCount) FROM UserAgentSummary" & strUAWhere
oRs.Open strSQL oConn, 3, 1 ' open recordset using adOpenDynamic, adLockReadOnly
If Err.Number &lt; 0 Then
  Response.Write("&lt;FONT FACE="" & QOT & "Arial" & QOT & "</SIZE=3&gt;&lt;b&gt;Sorry, the database is not accessible at present.&lt;/b&gt;&lt;/FONT&gt;")
  Exit Function
End If

&lt;!-- Include virtual="/jppam/common/Footer.inc" --&gt;

APPENDIX C - Module GRAPHICOPSys.ASP
```html
<TR><TD ALIGN="LEFT">
</TD></TR>
<TR align="left">
&wbr; Count: <B>&nbsp;</B> &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbs
</TD></TR></TABLE>

<%-- #include virtual="/japan/common/Footer.inc" -->

APPENDIX C – MODULE GRAPHICOPSYST.MAPS
```
**APPENDIX C – MODULE GRAPHICUATYPE.ASP**

```plaintext
If len(strUDetailNotIn2) Then strUWhere = strUWhere & ' AND CHARINDEX("' & strUDetailNotIn2 & '", ItemText) > 0'
strSQL = "SELECT HitCount, SUM(ItemCount) FROM UserAgentSummary" & strUWhere
  oks.open strSQL, oConn, 3, 1 'open recordset using edgedb&txt, " admiText
  If err.Number <> 0 Then
    Response.Write "Beg the database is not accessible at present!"<BR>
  <BR><-- include virtual="/jipam/common/Footer.inc" -->
  Response.End
End If
IngResult = 0
oks.MoveFirst
If Not oks.EOF Then ingResult = oks("HitCount")
If ingResult <> "0" Or IsMulti(ingResult) Then ingResult = 0
oks.Close
getUATypeCount = ingResult
End Function
intTotal = getUATypeCount("", 0, "")
intIE3 = getUATypeCount("InternetExplorer"; 3, 40, ",")
intIE4 = getUATypeCount("InternetExplorer"; 4, 40, ",")
intIE5 = getUATypeCount("InternetExplorer"; 5, 40, ",")
intNav2 = getUATypeCount("Mozilla/1.0", 1, "MSIE", "Opera")
intNav3 = getUATypeCount("Mozilla/1.1", 1, "MSIE", "Opera")
intNav4 = getUATypeCount("Mozilla/1.2", 1, "MSIE", "Opera")
intNav5 = getUATypeCount("Mozilla/1.5", 1, "MSIE", "Opera")
intOpera = getUATypeCount("Opera", 0, ",")
intOther = intTotal - intIE3 - intIE4 - intIE5 - intNav2 - intNav3 - intNav4 - intNav5 - intOpera
If intOther < 1 Then intOther = 1
intTotal = intTotal + intIE4 + intIE5 + intNav2 + intNav3 + intNav4 + intNav5 + intOpera + intOther
Set oConn = Nothing
<BR>
If binIE4 Then
  **** create the pie chart ***
  lineCnt = 3
  idegPos = 0
  iRectTop = -110
  <BR><TABLE STYLE="height:400; width:700;" >
    <TR><TD>
      <OBJECT ID="Pie1" STYLE="position:relative; height:230; width:300; top:10; left:10;"
        CLASSID="CLSID:36933CD2-7ACD-11D0-8905-00C04F9833E6">
        <PARAM NAME="Line001" VALUE="SetLineColor(0, 0, 0)"/>
        <PARAM NAME="Line002" VALUE="SetLineStyle(0)"/>
        <PARAM NAME="Line003" VALUE="SetFillColor(192, 192, 192)"/>
        <PARAM NAME="Line004" VALUE="oval(-30, -30, 200, 200, 0)"/>
        <PARAM NAME="Line005" VALUE="Rect(100, 10, CStr(iRectTop) + 5)"/>
        <PARAM NAME="Line006" VALUE="SetLineStyle(0)"/>
        <PARAM NAME="Line007" VALUE="SetFillColor(128, 128, 0)"/>
        <PARAM NAME="Line008" VALUE="Pie(-140, -100, 200, 200, CStr(iDegPos)-pos)"/>
        <PARAM NAME="Line009" VALUE="Rect(95, CStr(iRectTop) + 20, 15, 0)"/>
    </OBJECT>
    <BR>
```
```
*** create the table of values ***
Response.Write "<TR><TD>Netscape Navigator 2.x </TD><TD ALIGN=RIGHT><B><font size=-1>&nbsp;</font></B></TD><TD ALIGN=RIGHT><font size=-1>&nbsp;</font></TD></TR>
Response.Write "<TR><TD>Netscape Navigator 3.x </TD><TD ALIGN=RIGHT><B><font size=-1>&nbsp;</font></B></TD><TD ALIGN=RIGHT><font size=-1>&nbsp;</font></TD></TR>
Response.Write "<TR><TD>Netscape Navigator 4.x </TD><TD ALIGN=RIGHT><B><font size=-1>&nbsp;</font></B></TD><TD ALIGN=RIGHT><font size=-1>&nbsp;</font></TD></TR>
Response.Write "<TR><TD>Internet Explorer 3.x </TD><TD ALIGN=RIGHT><B><font size=-1>&nbsp;</font></B></TD><TD ALIGN=RIGHT><font size=-1>&nbsp;</font></TD></TR>
Response.Write "<TR><TD>Internet Explorer 4.x </TD><TD ALIGN=RIGHT><B><font size=-1>&nbsp;</font></B></TD><TD ALIGN=RIGHT><font size=-1>&nbsp;</font></TD></TR>
Response.Write "<TR><TD>Internet Explorer 5.x </TD><TD ALIGN=RIGHT><B><font size=-1>&nbsp;</font></B></TD><TD ALIGN=RIGHT><font size=-1>&nbsp;</font></TD></TR>
Response.Write "<TR><TD>Opera (all versions) </TD><TD ALIGN=RIGHT><B><font size=-1>&nbsp;</font></B></TD><TD ALIGN=RIGHT><font size=-1>&nbsp;</font></TD></TR>
Response.Write "<TR><TD>Other User Agents </TD><TD ALIGN=RIGHT><B><font size=-1>&nbsp;</font></B></TD><TD ALIGN=RIGHT><font size=-1>&nbsp;</font></TD></TR>"
<% Language=VBScript %>
<HTML>
<meta NAME="robots" CONTENT="noindex, nofollow">
<!-- INCLUDE virtual="/jipam/common/mainheader.inc" -->
<!-- INCLUDE virtual="/jipam/common/mainmenu.inc" -->
<!-- INCLUDE virtual="/jipam/common/common.inc" -->
<HEAD>
<TITLE>Login</TITLE>
</HEAD>
<!-- INCLUDE virtual="/jipam/common/mainheader.inc" -->
<!-- INCLUDE virtual="/jipam/common/mainmenu.inc" -->
<style type="text/css">
@import url("/jipam/common/common.css");
</style>
<body bgcolor="#ffffff" link="#000000" vlink="#000000">
  <div class="content">
  <!-- BEGIN CONTENT -->
  <h1>Login</h1>
  <form action="/login.asp" method="POST">
    <center>
      <table>
        <tr>
          <td>Email Address</td>
          <td><input type="text" name="email"></td>
        </tr>
        <tr>
          <td>Password</td>
          <td><input type="password" name="password"></td>
        </tr>
        <tr>
          <td>Submit</td>
          <td><input type="submit" value="Login"></td>
        </tr>
      </table>
    </center>
  </form>
  <!-- END CONTENT -->
  </div>
</body>
</HTML>
```html
<!-- Language=VBScript -->
<!-- #include virtual="/jipam/common/common.inc" -->
<!-- #include virtual="/jipam/common/Mainheader.inc" -->
<!-- #include virtual="/jipam/common/menu.inc" -->
strUserName = Request.QueryString("UI")
if len(strUserName) < 1 then
<!-- Your Email address is invalid. -->
else
<!-- include virtual="/jipam/common/Footer.inc" -->
Response.End
end if
'create and open a connection to the database
Set oCon = Server.CreateObject("ADODB.Connection")
oCon.Open strConnection
'create a command and set the properties
Set cmd = Server.CreateObject("ADODB.Command")
oCmd.ActiveConnection = oCon
oCmd.CommandType = 4 'stored procedure
oCmd.CommandText = "SendUserPassWord" 'procedure name
lngRecsAffected = 0
arrParams = Array(strUserName)
'execute the command, supplying the parameters, and get the result
Set oRs = oCmd.Execute(lngRecsAffected,arrParams)
<!-- Your password has been sent to your email address. -->
<!-- include virtual="/jipam/common/Footer.inc" -->
```
**Module Name** - **MainHeader.inc**

```html
<BODY BACKGROUND="/jpam/images/bgbeige2.gif">
<TBODY>
<tr>
<td align="LEFT" table width="100%" align="LEFT">
<tr>
<td width="300" align="LEFT">
</td>
</tr>
</tbody>
</table>
</body>
</html>
```

**Appendix C - Module MainHeader.inc**
APPENDIX C - MODULE MANAGEMENT.ASP
```vbscript
' Module Name - ModLogon.asp

<!---% Language=VBScript -->
<!---% MODULE -->
<!---% Include virtual="/jipam/common/common.inc" -->
<!---% Include virtual="/jipam/common/error.inc" -->
<!---% HEAD -->
<TITLE>Site Map</TITLE>
</HEAD>

<BODY>
<%= Include virtual="/jipam/common/Menu.inc" %>
<%=
If Session("Locked") = 1 Or Session("UserID") > 101 Then
Response.Write("<H2>Sorry this facility is not available.</H2>"")
Else
Response.Write("""
' We should now check to see if we are updating the data rather than retrieving it
On Error Resume Next
Dim strEmailAddress
Dim strPassWord
Dim strFirstName
Dim strOtherInitials
Dim strSurname
Dim strWebAddress
Dim strAffiliation
Dim strDisplayName
Dim intResult
'
' Create and open a connection to the database
Err.Clear()
Set oCon = Server.CreateObject("ADODB.Connection")
If Err.number > 0 Then
Terminatemessage("Cannot connect to Database(1)")
Set oCon = Nothing
Else
Set oCon = Server.CreateObject("ADODB.Command")
If Err.number > 0 Then
Terminatemessage("Cannot connect to Database(2)")
Else
Set oCon = Server.CreateObject("ADODB.Connection")
If Err.number > 0 Then
Terminatemessage("Cannot connect to Database(3)")
End If
End If
If Err.number = 4 Then
' Stored procedure
If Request.Form.Count > 1 Then
This is the second pass
On Error Resume Next
strEmailAddress = trim(Request.Form.Item("E")
strPassWord = trim(Request.Form.Item("W")
strFirstName = trim(Request.Form.Item("F")
strOtherInitials = trim(Request.Form.Item("T")
strSurname = trim(Request.Form.Item("S")
strWebAddress = trim(Request.Form.Item("W")
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
End If
```
strOtherInit=trim(oRs.Fields("OtherInit"))
strSurname=trim(oRs.Fields("Surname"))
strWebAddress=trim(oRs.Fields("WebAddress"))
strAffiliation=trim(oRs.Fields("Affiliation"))
strDisplayName=trim(oRs.Fields("DisplayName"))

<!-- #include virtual="/jipam/common/Subscribe.inc" -->
<!-- #include virtual="/jipam/common/Footer.inc" -->
MODULE NAME - SEARCH.ASP

<!-- Language=VBScript -->
<!-- HTML -->
<!-- meta NAME="robots" CONTENT="noindex,nofollow"> -->
<!-- include virtual="/jipam/common/common.inc" -->
<!-- TITLE>Search Page for JIPAM</TITLE> -->
</HEAD>
<!-- Dim PageName -->
PageName="SEARCH"
<!-- include virtual="/jipam/common/Mainheader.inc" -->
<!-- include virtual="/jipam/common/menu.inc" -->
<!-- Dim strSearch -->
Dim intOption
strSearch=""
intOption=1
<!-- H1 ALIGN="center"><FONT -->
<!-- COLOR="#FF0000">" -->
<!-- SEARCH.search. -->
<!-- FONT COLOR="#FF0000">" -->
<!-- JIPAM <FONT COLOR="#FF0000">" -->
</H1>
<!-- p -->
<!-- include virtual="/jipam/common/SEARCH.inc" -->
<!-- include virtual="/jipam/common/Footer.inc" -->

APPENDIX C - MODULE SEARCH.ASP
<center>
<form METHOD="POST" ACTION="DoSearch.asp">
<p>Enter the text you would like to search for<input TYPE="TEXT" NAME="txtSearch" VALUE=""></p>
<p>Where would you like the search performed</p>
<table WIDTH="50k">
<tr>
<td>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
<@ Language=VBScript>
<!-- #include virtual="/jipan/common/common.inc" -->
<!-- #include virtual="/jipan/common/error.inc" -->
<% 'this process returns a binary file
Dim PageName
PageName="NONAME"
Dim strArticleId
Dim iImageNumber
strArticleId = Request.QueryString("AI")
iImageNumber = Request.QueryString("IM")
On Error Resume next
'Step 2: grab the picture from the database
Dim oConn,oCmd, oRs
'create and open a connection to the database
Set oConn = Server.CreateObject("ADODB.Connection")
if Err.number > 0 then Response.End
oConn.Open strConnectionString
if Err.number > 0 then Response.End
'create a command and set the properties
Set oCmd = Server.CreateObject("ADODB.Command")
if Err.number > 0 then Response.End
oCmd.ActiveConnection = oConn
oCmd.CommandType = 4 'stored procedure
oCmd.CommandText = "GetArticleImage" 'procedure name
lngRecsAffected=0
'execute the command, supplying the parameters, and get the result
Set oRs = oCmd.Execute(lngRecsAffected, Array(strArticleId, iImageNumber))
if Err.number > 0 then Response.End
if oRs.EOF then Response.end
end if
strImageType = oRs.Fields("ImageType")
Set adoFldBlob=Rs("AnImage")
lngFieldDataLength= adoFldBlob.ActualSize
lngBlockCount=lngFieldDataLength / 4096
lngRemainingData=lngFieldDataLength mod 4096
Response.ContentType = "image/" + strImageType
for lngCounter=1 to lngBlockCount
    Response.BinaryWrite (adoFldBlob.GetChunk(4096))
next
if lngRemainingData > 0 then
    Response.BinaryWrite (adoFldBlob.GetChunk(lngRemainingData))
end if
adoFldBlob.Close
Set adoFldBlob=nothing
'Clean up...
Set oRs.Close
Set oRs = Nothing
Set oCmd = Nothing
oConn.Close
Set oConn = Nothing
%>

APPENDIX C - MODULE SHOWARTIMG.ASP
<% Language=VBScript %>
<%}
<% QUOT = Chr(34)
strQuery = Request.Form("query")
strCriteria = Request.Form("criteria")
strWhere = "";
strSessNT = "".
If Len(strCriteria) > 1 Then
    strSessNT = " WHERE " & strCriteria
End If.
Select Case strQuery
    Case "refer";
        strSQL = "SELECT Count, SUM(ItemCount), Referring_Page=MAX(ItemText) FROM ReferersSummary & strSessNT & " GROUP BY ItemText ORDER BY SUM(ItemCount) DESC"
        Case "refcount";
            strSQL = "SELECT Hits=SUM(ItemCount), Target_Page=MAX(ItemText) FROM SessionTargetSummary & strSessNT & " GROUP BY ItemText ORDER BY SUM(ItemCount) DESC"
        Case "agents";
            strSQL = "SELECT Hits=SUM(ItemCount), UserAgent=MAX(CASE WHEN CHARINDEX('"', ItemText) > 10 THEN SUBSTRING(ItemText, 1, CHARINDEX('"', ItemText)) ELSE ItemText END) FROM UserAgentSummary & strSessNT & " GROUP BY WHEN CHARINDEX('"', ItemText) > 10 THEN SUBSTRING(ItemText, 1, CHARINDEX('"', ItemText)) ELSE ItemText END ORDER BY SUM(ItemCount) DESC"
        Case "lang";
            strSQL = "SELECT Hits=SUM(ItemCount), Language=MAX(ItemText) FROM CountrySummary & strSessNT & " GROUP BY ItemText ORDER BY SUM(ItemCount) DESC"
End Select
<%%>
<%%>
<% Dim PageName
PageName="TRAFFICLOG";
<%>
<% End If
If Not IsAdmin() or len(strQuery) < 1 then
    Response.Write"<B>Sorry, this function is not available</B>"
<%>
<% Response.Write"<B>Sorry, this function is not available</B>"
<% Response.End
end if
<%>
<% "ALIGN=center"
<% Select Case strQuery
APPENDIX C - MODULE SHOWLOG.ASP
Do While Not oRs1.EOF
    Response.Write "<TR>" & vbCrLf
    For Each objItem In oRs1.Fields
        Response.Write "<TD nowrap align=" & quot & "left" & quot & "">" & objItem.value & " &nbsp; & quot & "/>" & vbCrLf
    Next
    Response.Write "</TR>" & vbCrLf
    oRs1.MoveNext
Loop
Set oRs = Nothing
Set oRs1 = Nothing
Set oConn = Nothing
</TABLE>
<!--[include virtual="/jspam/common/Footer.inc" -->
<% Dim PageName
PageName = "SUBSCRIBE"
%>

<!-- Include virtual="\//jpin/\common/\Mainheader.inc" -->
<!-- Include virtual="/\jpin/\common/\menu.inc" -->

Sub TerminateWithMessage(Mess)
  If len(Mess) > 0 Then Response.Write(Mess)
End Sub

Function MakeSureNotNull(inputStr)
  If inputStr Is Nothing Then
    MakeSureNotNull = ""
  Else
    MakeSureNotNull = inputStr
  End If
End Function

<% if Request.Form.Count > 1 then 'this is the a second or greater pass %>
  If Session("LoginCount") < 0 Then
    Session("Locked") = true
    Message = "<b>Sorry, You have been locked out due to exceeding the number of login attempts.</b>"
  End If
End If

'create and open a connection to the database
Set oCon = Server.CreateObject("ADODB.Connection")
'if err.number > 0 then TerminateWithMessage "Sorry, Cannot connect to database(1)"
'oCon.Open strConnection
'if err.number > 0 then TerminateWithMessage "Sorry, Cannot connect to database(2)"
Err.Clear() 'create a command and set the properties
Set oCmd = Server.CreateObject("ADODB.Command")
'if err.number > 0 then TerminateWithMessage "Sorry, Cannot connect to database(3)"
'oCmd.ActiveConnection = oCon
'oCmd.CommandType = 4 'stored procedure
'oCmd.CommandText = "AddUserFull" 'procedure name
lngRecsAffected = 0
arrParams = Array(strEmailAddress, strPassword, strFirstName, strOtherInit, strSurname, strWebAddress, strAffiliation, intUser)
'execute the command, supplying the parameters, and get the result
Set oRs = oCmd.Execute(lngRecsAffected, arrParams)
If (Err.number > 0) Then TerminateWithMessage "Sorry, Cannot execute database procedure"
If Not oRs.EOF Then 'successful login
  intResult = 1
End If
 '<int result contains the user_id value
Session("UserID") = oRs.Fields("UserID")
Session("UserLogon") = oRs.Fields("UserLogon")
Session("UserPass") = strPassword 'kept as parameter for procedures
GetUsersDetails, and ModUsers
Session("UserRole") = oRs.Fields("UserRole") ' this is default for all new users
Messaage = "Thank you for registering. When logging in in future please use your E-Mail address " + Session("UserLogon") + ""</p></p>"
End If
Response.End
else
intResult = 1
end if
'we fall out here if the email address is in use '
"intResult > 0 for this scenario

APPENDIX C - MODULE_SUBSCRIBE.ASP
end if
if intResult < 1 then 'a new subscription
while access to JIPAM is current free, you must subscribe to download full text
articles as PDF files. To subscribe to JIPAM for twelve months, complete the
subscription form
areas marked * must be completed.</p><br><br>
else
<br>
<p>The E-Mail address you entered is already registered. If you have previously
registered please attempt to <a href="login.asp">login</a> instead. </p>
end if
</p>
</div>
</div>

</body>
</html>

<!-- include virtual="/jipam/common/Subscribe.inc" -->

<!-- include virtual="/jipam/common/Footer.inc" -->

APPENDIX C - MODULE SUBSCRIBE.ASP
APPENDIX C - MODULE SUBSCRIBE.INC

```javascript
function validatePrompt(ctrl, mess) |
alert(mess);
ctrl.focus();
return;
}
function ValidateEmail(form) |
Ctrl = form.E1;
off = Ctrl.value.indexOf('@', 0);
if (Ctrl.value == '' || off < 2) |
validatePrompt(Ctrl, "Enter a valid email address")
return(false);
return (true);
}
function ValidatePass(form) |
p1 = form.E2.value;
p2 = form.E3.value;
if (p1 != p2) |
validatePrompt(form.E2, "Your password are not the same")
return(false);
return (true);
}
function ValidateAllEntries(form) |
if (!ValidateEmail(form)) return;
if (!ValidatePass(form)) return;
form.submit();
return;
}</SCRIPT>

<FORM METHOD="post" ACTION="\<t:PageName>.asp" NAME="Register">
|<p>
|<center>
|<TABLE WIDTH="90%">
|<TR>
|<TD WIDTH="171">&nbsp;E-Mail Address *</TD>
|<TD WIDTH="343"><INPUT NAME="E1" SIZE="50" MAXLENGTH="128" VALUE="<c:strMailAddress"></TD>
|<TR>
|<TD WIDTH="171">&nbsp;Password *</TD>
|<TD WIDTH="343"><INPUT TYPE="password" NAME="E2" SIZE="15" MAXLENGTH="32" VALUE="<c:strPassword"></TD>
|<TR>
|<TD WIDTH="171">&nbsp;Password Again *</TD>
|<TD WIDTH="343"><INPUT TYPE="password" NAME="E3" SIZE="15" MAXLENGTH="32" VALUE="<c:strPassword"></TD>
|<TR>
|<TD WIDTH="171">&nbsp;First Name</TD>
|<TD WIDTH="343"><INPUT NAME="E4" SIZE="30" MAXLENGTH="60" VALUE="<c:strFirstName"></TD>
|<TR>
|<TD WIDTH="171">&nbsp;Other Initials</TD>
|<TD WIDTH="343"><INPUT NAME="E5" SIZE="15" MAXLENGTH="20" VALUE="<c:strOtherInitials"></TD>
|<TR>
|<TD WIDTH="171">&nbsp;Surname</TD>
|<TD WIDTH="343"><INPUT NAME="E6" SIZE="50" MAXLENGTH="128" VALUE="<c:strSurname"></TD>
|<TR>
|<TD WIDTH="171">&nbsp;Your Web address</TD>
|<TD WIDTH="343"><INPUT NAME="E7" SIZE="50" MAXLENGTH="128" VALUE="<c:strWebAddress"></TD>
|<TR>
|<TD WIDTH="171">&nbsp;Institution/Affiliations</TD>
|<TD WIDTH="343"><INPUT NAME="E8" SIZE="35" MAXLENGTH="60" VALUE="<c:strAffiliations"></TD>
|<TR>
|<TD WIDTH="171">&nbsp;Userrole</TD>
|<TD WIDTH="343">if Session("Userrole") > 0 then |
|<TD WIDTH="171">&nbsp;Your Display Name</TD>
|<TD WIDTH="343"><INPUT NAME="E9" SIZE="50" MAXLENGTH="128" VALUE="<c:strDisplayName"></TD>
|<TR>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">if End if |
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
|<TD WIDTH="343">&nbsp;</TD>
|<TD WIDTH="171">&nbsp;</TD>
```````
APPENDIX C – MODULE TRAFFIC REPORTS.ASP
<HTML>
  <meta NAME="robots" CONTENT="noindex,nofollow">
  <HEAD>
    <TITLE>Whats New</TITLE>
  </HEAD>
  <BODY>
  <!-- Include virtual="/ipam/common/Mainheader.inc" -->
  <!-- Include virtual="/ipam/common/menu.inc" -->
  <!-- Text for Whats New -->
</HTML>
APPENDIX D

JIPAM WEB SITE
Dynamic Database System
Administration Program
C++ Code
# Module Name - DLGAddArtToVolume

```c
#include <vcl.h>
#pragma hdate

#include "DLGAddArtToVolume.h"

// NR I have used the field institution in the TForm to indicate
// whether the article has been moved between listboxes
// If it has been moved then the record behind it will need to be updated

#pragma resource "*.dfm"
TFrmAddArtToVolume *TFrmAddArtToVolume;

_fastcall TFrmAddArtToVolume::TFrmAddArtToVolume(TComponent* AOwner)
{ TForm(AOwner); }

_fastcall TFrmAddArtToVolume::TFrmAddArtToVolume()
{ ClearObjects(lbxAssigned);
  ClearObjects(lbxAvailable);
  //
  void_fastcall TFrmAddArtToVolume::ClearObjects(TListbox* tbl)
  for [int i=0; i<tbl->items->Count; ++i]
    ArticleObj* tob = { ArticleObj* }; tbl->items->Objects[i];
    delete tob;
  //

  void_fastcall TFrmAddArtToVolume::SetUp(const AnsiString dbname, int volume_id)
  DatabaseName = dbname;
  Volume_ID = volume_id;
  QryGeneral->DatabaseName = dbname;
  LoadListBox(lbxAssigned);
  LoadListBox(lbxAvailable);
  FixPositionIndex();
  //

  void_fastcall TFrmAddArtToVolume::LoadListBox(TListbox* tblBox)
  const AnsiString Query;
  if (tblBox == lbxAssigned)
    Query = "Select article_ID,ArticleTitle from Articles where Published='Y' and Available='Y';";
    else
    Query = "Select article_ID,ArticleTitle,ArticleNo from Titles in Volume where Volume_ID=";
    Query = IntToStr(Volume_ID);
    try
      QueryGeneral->Close();
      QueryGeneral->SQL->Clear();
      QueryGeneral->SQL->Add( Query);
      try
        QueryGeneral->Open();
        while (! QryGeneral->EOF)
        {
          AnsiString Title = QueryGeneral->FieldByName("ArticleTitle")->AsString;
          //
```
void __fastcall TFormAddArtToVolume::OKBtnClick(TObject *Sender)
{
    // AssiString Query;
    TSession * ts = Sessions->FindSession(QryGeneral->SessionName);
    TDatabase * td = ts->FindDatabase(DatabaseName);
    try {
        Query = "Delete from VolumeArticles where Volume_id = ";
        Query = IntToStr(Volume_ID) + Query;
        QryGeneral->Close();
        QryGeneral->SQL->Clear();
        QryGeneral->SQL->Add(Query);
        QryGeneral->ExecuteSQL();
        QryGeneral->SQL->Add("Insert into VolumeArticles(Article_ID,Volume_ID,ArticleNO) ");
        QryGeneral->SQL->Add("VALUES (:Article_ID ,:Volume_ID ,:ArticleNO) ");
        QryGeneral->Prepare();
        QryGeneral->ParamByName("Article_ID ")->AsInteger = Article_ID;
        for (int i = 0; i < LbxAssigned->Items->Count; i++)
        {?
            ArticleObj * tob = (ArticleObj *) LbxAssigned->Items->Objects[i];
            QryGeneral->ParamByName("Article_ID ")->AsString = tob->GetArticleId();
            QryGeneral->ExecuteSQL();
        }
        QryGeneral->UnPrepare();
    }
    catch (Exception e)
    {
        ShowMessage(e.Message);
        OKBtn->Enabled = false;
    }
}

void __fastcall TFormAddArtToVolume::BTNUpAssignedClick(TObject *Sender)
{
    // adjust the movement buttons
    BTNUpAssigned->Enabled = false;
    BTNDownAssigned->Enabled = false;
    BTNUpAvailable->Enabled = false;
    BTNDownAvailable->Enabled = false;
    if (LbxAssigned->Items->Count > 0 && LbxAssigned->ItemIndex > 0) {
        BTNUpAssigned->Enabled = true;
        if (LbxAssigned->ItemIndex > 0) // if not first item
            BTNUpAssigned->Enabled = true;
        if (LbxAssigned->ItemIndex < LbxAssigned->Items->Count - 1) // not last item
            BTNDownAssigned->Enabled = true;
        if (LbxAvailable->Items->Count > 0 && LbxAvailable->ItemIndex > 0) {
            BTNUpAvailable->Enabled = true;
        } else {
            BTNUpAvailable->Enabled = true;
        }
    } else {
        BTNUpAvailable->Enabled = true;
    }
}

void __fastcall TFormAddArtToVolume::LbxAssignedClick(TObject *Sender)
{
    FixPositionIndex();
}

void __fastcall TFormAddArtToVolume::LbxUpAssignedClick(TObject *Sender)
{
    // reorder current lbxitem up
    int index = LbxAssigned->ItemIndex;
    if (LbxAssigned->Items->Move(index, index - 1))
    {
        LbxAssigned->ItemIndex = index - 1;
        FixPositionIndex();
        OKBtn->Enabled = true;
    }
}

void __fastcall TFormAddArtToVolume::LbxDownAssignedClick(TObject *Sender)
{
    // reorder current lbxitem down
    int index = LbxAssigned->ItemIndex;
    if (LbxAssigned->Items->Move(index, index + 1))
    {
        LbxAssigned->ItemIndex = index + 1;
        FixPositionIndex();
        OKBtn->Enabled = true;
    }
}

APPENDIX D - MODULE DLGADDARTTOVOLUME
LbxAssigned->ItemIndex = index1;
FixPositionIndex();
OKBtn->Enabled = true;

//_____________________________________________________________________

void __fastcall TfrmAddArtToVolume::BtnUpAvailableClick(TObject *Sender)
{
    // move from available to assigned
    // move items between list boxes
    int index;
    const char *old;
    ArticleObj *tob;

    index = LbxAvailable->ItemIndex;
    old = LbxAvailable->Items->Strings[index];
    tob = (ArticleObj*) LbxAvailable->Items->Objects[index];
    tob->SetMoved(); // flag moved
    LbxAvailable->Items->Delete(index);
    if (LbxAvailable->Items->Count - 1 > index)
        LbxAvailable->ItemIndex =
        LbxAssigned->Items->IndexOf(old);
    else
        LbxAssigned->ItemIndex =
        LbxAssigned->ItemIndex - index;
    // put string in other box
    LbxAssigned->Items->AddObject(old, tob);
    LbxAvailable->ItemIndex =
    LbxAvailable->Items->IndexOf(old);
    FixPositionIndex();
    OKBtn->Enabled = true;
}

//_____________________________________________________________________

void __fastcall TfrmAddArtToVolume::BtnDownAvailableClick(TObject *Sender)
{
    // move from assigned to available
    // move items between list boxes
    int index;
    const char *old;
    ArticleObj *tob;

    index = LbxAssigned->ItemIndex;
    old = LbxAssigned->Items->Strings[index];
    tob = (ArticleObj*) LbxAssigned->Items->Objects[index];
    tob->SetMoved(); // flag moved
    LbxAssigned->Items->Delete(index);
    if (LbxAssigned->Items->Count - 1 > index)
        LbxAssigned->ItemIndex =
        LbxAssigned->ItemIndex + index;
    else
        LbxAssigned->ItemIndex =
        LbxAssigned->ItemIndex + index;
    // put string in other box
    LbxAvailable->Items->AddObject(old, tob);
    LbxAvailable->ItemIndex =
    LbxAvailable->Items->IndexOf(old);
    FixPositionIndex();
    OKBtn->Enabled = true;
}

//_____________________________________________________________________

APPENDIX D - MODULE DLGADDARTTOVOLUME
---

APPENDIX D - MODULE DLGADDAUTHART
void __fastcall TFormDlgAddAuthArt::BtnUpAssignedClick(TObject *Sender)
{
    // adjust the movement buttons
    BttUpAssigned->Enabled = false;
    BttDownAssigned->Enabled = true;
    BttUpAvailable->Enabled = false;
    BttDownAvailable->Enabled = true;
    if (LbxAssigned->Items->Count > 0 && LbxAssigned->ItemIndex >= 0) {
        BttDownAssigned->Enabled = true;
        if (LbxAssigned->ItemIndex > 0) // if not first item
            BttUpAssigned->Enabled = true;
        if (LbxAssigned->ItemIndex < LbxAssigned->Items->Count - 1) // not last item
            BttDownAvailable->Enabled = true;
    }
    if (LbxAvailable->Items->Count > 0 && LbxAvailable->ItemIndex >= 0) {
        BttUpAvailable->Enabled = true;
    }
}

void __fastcall TFormDlgAddAuthArt::BtnUpAvailableClick(TObject *Sender)
{
    // move from available to assigned
    int indx;
    AnsiString old;
    TFormObject *tab;
    // moving from available to assigned
    indx = LbxAvailable->ItemIndex;
    old = LbxAvailable->Items->Strings[indx];
    tab = (TFormObject*) LbxAvailable->Items->Objects[indx];
    LbxAvailable->Items->Delete(indx);
    if (LbxAvailable->Items->Count - 1 >= indx)
        LbxAvailable->ItemIndex = indx;
    else
        LbxAvailable->ItemIndex = indx - 1;
    // put string in other box
    LbxAssigned->Items->AddObject(old, tab);
    LbxAssigned->ItemIndex = LbxAssigned->Items->IndexOf(old);
    FixPositionIndex();
    OKBtn->Enabled = true;
}

void __fastcall TFormDlgAddAuthArt::BtnDownAvailableClick(TObject *Sender)
{
    // move from assigned to available
    int indx;
    AnsiString old;
    TFormObject *tab;
    indx = LbxAssigned->ItemIndex;
    old = LbxAssigned->Items->Strings[indx];
    tab = (TFormObject*) LbxAssigned->Items->Objects[indx];
    LbxAssigned->Items->Delete(indx);
    if (LbxAssigned->Items->Count - 1 >= indx)
        LbxAssigned->ItemIndex = indx;
    else
        LbxAssigned->ItemIndex = indx - 1;
    // put string in other box
    LbxAvailable->Items->AddObject(old, tab);
    LbxAvailable->ItemIndex = LbxAvailable->Items->IndexOf(old);
    FixPositionIndex();
    OKBtn->Enabled = true;
}

void __fastcall TFormDlgAddAuthArt::FixPositionIndex()
{
MODULE NAME - DLGADDUSER

DLGADDUSER.H

// ------------------------------------------------------------------------
#define DlgAddUserH
#define DlgAddUserM
// ------------------------------------------------------------------------
#include <tlctrrs.hpp>
#include <tlcbuttons.hpp>
#include <tlccontrols.hpp>
#include <tlcforms.hpp>
#include <tlcgraphics.hpp>
#include <tlcclasses.hpp>
#include <tlcwindows.hpp>
#include <tlcsystem.hpp>
#include <tlc.hpp>
#include <tlcdatabas.hpp>
#include <tlcmask.hpp>
// ------------------------------------------------------------------------
class TDlgFmAddUser : public TForm
{__published:
  TButton *BtnAdd;
  TButton *CancelButton;
  TLabel *TLabel1;
  TLabel *TLabel2;
  TLabel *TLabel3;
  TStringsProc *SProcAddUser;
  TLabel *TLabel4;
  TLabel *TLabel5;
  TLabel *TLabel6;
  TLabel *TLabel7;
  TLabel *TLabel8;
  TLabel *TLabel9;
  TLabel *TLabel10;
  TLabel *TLabel11;
  Query *QryGeneral;
  void __fastcall BrmAddClick(TObject *Sender);
  void __fastcall EdEmailAddressChange(TObject *Sender);
private:
  int UserID;
  bool SetInProgress;
  bool InitEditFields();
public:
  virtual __fastcall TDlgFmAddUser(TComponent *Owner);
  void SetUp(const AnsiStrings DbName);
};
// ------------------------------------------------------------------------
extern PACKAGE TDlgFmAddUser *DlgFmAddUser;
// ------------------------------------------------------------------------
#endif

DLGADDUSER.CPP

// ------------------------------------------------------------------------
#include <tlc.hpp>
#pragma ideproto
#include "TlpmCommon.hpp"
#include "DlgAddUser.hpp"
// ------------------------------------------------------------------------
#include "sysCommon.sys"
#include "SysAddUser.hpp"
// ------------------------------------------------------------------------
// Size Testor = 1234567801234567890123456789012345678901234567890
// char* EDPassword = "112345678901234567890123456789012345678901234567890";
extern const int LengthEmailAddress = 128;
extern const int LengthPassword = 32;
extern const int LengthFirstName = 40;
extern const int LengthOtherInitns = 50;
extern const int LengthSurName = 128;
extern const int LengthDisplayEmailName = 128;
extern const int LengthWebAddress = 128;
extern const int LengthAffiliation = 60;
// ------------------------------------------------------------------------
fastcall TDlgFmAddUser::TDlgFmAddUser(TComponent *Owner)
{ TForm(Owner);
  SetInProgress = true;
} // ------------------------------------------------------------------------
void TDlgFmAddUser::SetUp(const AnsiStrings DbName)
{ EdPassword = EDPassword;
  EdEmailAddress = MaxLength = LengthEmailAddress;
  EdPassword = MaxLength = LengthPassword;
  EdFirstName = MaxLength = LengthFirstName;
  EdOtherInitns = MaxLength = LengthOtherInitns;
  EdSurName = MaxLength = LengthSurName;
  EdWebAddress = MaxLength = LengthWebAddress;
  EdAffiliation = MaxLength = LengthAffiliation;
  EdDisplayEmailName = MaxLength = LengthDisplayEmailName;
  SProcAddUser = DatabaseName = DbName;
  QueryGeneral = DatabaseName = DbName;
  // load the institution list
  QueryGeneral = SQL = "Select SmallDesc from Institutions";
  QueryGeneral = Open();
  while (!QueryGeneral->Xof())
    { CbInstitution->Items->Add;
      QueryGeneral->FieldByName("SmallDesc") = AsString;
      QueryGeneral->Next();
    }
  QueryGeneral = Close();
  SetInProgress = false;
  InitEditFields();
} // ------------------------------------------------------------------------
void __fastcall TDlgAddUser::BtnAddClick(TObject *Sender)
{ 
  LblError->Caption="";
  SprocAddUser->Params->Items[1]->AsString = EdEmailAddress->Text;
  SprocAddUser->Params->Items[2]->AsString = EdPassword->Text;
  SprocAddUser->Params->Items[3]->AsString = EdFirstName->Text;
  SprocAddUser->Params->Items[4]->AsString = EdOtherInitials->Text;
  SprocAddUser->Params->Items[5]->AsString = EdSurName->Text;
  SprocAddUser->Params->Items[6]->AsString = EdWebAddress->Text;
  SprocAddUser->Params->Items[7]->AsString = EdAffiliation->Text;
  SprocAddUser->Params->Items[8]->AsInteger = CbxUserRole->ItemIndex;
  // QryGeneral->SQL->Clear();
  AnsiString Query = "Select Institution_ID from Institutions where SmallDesc='" + CbxInstitution->Items->Strings[CbxInstitution->ItemIndex] + ";"
  QryGeneral->SQL->AddQuery(QryGeneral, Query);
  QryGeneral->Open();
  int instID = QryGeneral->FieldByName("Institution_ID")->AsInteger;
  QryGeneral->Close();
  SprocAddUser->Params->Items[9]->AsInteger = instID;
  SprocAddUser->Prepare();
  SprocAddUser->Execute();
  User_Id = SprocAddUser->Params->Items[0]->AsInteger;
  if (User_Id > 99) 
  { LblError->Caption = "Added " + EdEmailAddress->Text + " as User " + User_Id; 
  } else 
  { LblError->Caption = "Error - Email Address already exists" ; 
  }
  //
  //
  //

void __fastcall TDlgAddUser::EdEmailAddressChange(TObject *Sender)
{ if (EdEmailAddress->Text.Length() > 0) StnAdd->Enabled=true; else StnAdd->Enabled=false; 
  //
  //

void __fastcall TDlgAddUser::CbxInstitutionChange(TObject *Sender)
{ if (SetUpInProgress) return;
  EdAffiliation->Text = CbxInstitution->Items->Strings[CbxInstitution->ItemIndex];
  //
  //

void TDlgAddUser::InitEditFields()
{ EdEmailAddress->Text="";
  EdPassword->Text="";
  EdFirstName->Text="";
  EdOtherInitials->Text="";
  EdSurname->Text="";
  EdWebAddress->Text="";
  EdAffiliation->Text="NONE";
  EdDisplayName->Text="";
  CbxInstitution->Items->Caption = "";
  LblError->Caption="";
  //

APPENDIX D - MODULE DLGADDUSER
MODULE NAME - DLGARTICLEDOWNLOADS

DLGARTICLEDOWNLOADS.H

#ifndef DLGARTICLEDOWNLOADS_H
#define DLGARTICLEDOWNLOADS_H
    #include <vclSystem.hpp>
    #include <vclWindows.hpp>
    #include <vclSysUtils.hpp>
    #include <vclClasses.hpp>
    #include <vclGraphics.hpp>
    #include <vclStdCtrls.hpp>
    #include <vclForms.hpp>
    #include <vclButtons.hpp>
    #include <vclExtCtrls.hpp>
    #include <DB.hpp>
    #include <DBTables.hpp>
    #include <Dialog.hpp>
    #include <CmxCtrls.hpp>

#define ODCOVER 0x0300
#define ODCOVER 0x0300
#define sqlextr.hpp

class TDglFrmDownloads : public TForm
{ // Nott Published:
    TButton *BtnAdd;
    TButton *BtnCancel;
    TButton *BtnDelete;
    TListBox *ListBox;
    TQuery *qryArticlesInFull;
    TLabel *Label1;
    TLabel *Label2;
    TProgress *Progressbar;
    void_fastcall BtnAddClick(TObject *Sender);
    void_fastcall ListBoxClick(TObject *Sender);
    void_fastcall LbxAvailableClick(TObject *Sender);
    void_fastcall LbxDeleteClick(TObject *Sender);
}

private:
    TFormType *hnt;
    AnsiString DatabaseName;
    AnsiString ArticleName;
    void SetUpListBoxes();
    bool_fastcall CheckStatementRetResult(SQIAReturn retcode);
    bool_fastcall WriteData(int filehandle, int PrintType);
    void_fastcall ClearObjects(TListBox *Tbl);

public:
    virtual_fastcall TDglFrmDownloads(TComponent *Owner);
    virtual_fastcall ~TDglFrmDownloads();
    void SetUp(AnsiString articleNo, const AnsiString dname);
};

extern PACKAGE TDglFrmDownloads *TDglFrmDownloads;
#ifndef DLGARTICLEDOWNLOADS_CPP
#define DLGARTICLEDOWNLOADS_CPP

#include <vcl.h>
#pragma hdrstop

#include "DLGARTICLEDOWNLOADS.h"
#include "VclCommon.h"
#include <cont.h>
#include <io.h>

#pragma resource "*.dfm"

TDglFrmDownloads::TDglFrmDownloads()
{  TForm(Owner)  
    ArticleNo=0;
}

_fastcall TDglFrmDownloads::TDglFrmDownloads(TComponent *Owner)
  {
    ArticleNo=0;
    
    ClearObjects(LbxUsed);
    ClearObjects(LbxAvailable);
  }

void_fastcall TDglFrmDownloads::ClearObjects(TListBox *Tbl)
for (int i=0; i<Tbl->Items->Count; i++)
  {PrintTypeEx* p = (PrintTypeEx*) Tbl->Items->Objects[i];
    delete p;
  }

    void

TDglFrmDownloads::SetUp(AnsiString articleNo, const AnsiString dname) {
    ArticleName=articleNo;
    DatabaseName=dname;
    QryArticlesInFull->DatabaseName=dname;
    SetUpListBoxes();
}

    void

void

TDglFrmDownloads::SetUpListBoxes() {
  LbxDelete->Enabled = false;
  BtnAdd->Enabled = false;

  AnsiString Query = "Select PrintType, PrintShortDesc from PrintFileTypes"
  * where printType in ("
  * Select PrintType from ArticlesInFull where Article_ID = ";
  Query = "Article No + ";

  QryArticlesInFull->SQL->Clear();
  QryArticlesInFull->SQL->Add(Query);
  ClearObjects(LbxUsed);
  ClearObjects(LbxAvailable);
  LbxUsed->Clear();
  LbxAvailable->Clear();
  try {

APPENDIX D - MODULE DLGARTICLEDOWNLOADS

if (rcode == SQL_INVALID_HANDLE)  
    throw Exception("ODBC Invalid handle");
// error
SQLGetDiagRec((SQLHANDLE_STMT, stmt, 1, SQLSTATE, &NativeError, Msg,  
    sizeof(Msg), &MsgLen));
throw Exception(char* Msg);
}

bool _fastcall TDlgFrmDownloads::WriteData(int filehandle, int PrintType) {
#define MAX_DATA_LEN 4096
char* Data[MAX_DATA_LEN];
long FileLength = filelength(filehandle);
ProgressBar->Max = FileLength;
ProgressBar->Visible = true;
Update();
SQLINTEGER cbPDFFile = 0;
SQLPOINTER pToken;
SQLRETURN retcode;

char* Query = "INSERT INTO ARTICLESINFULL (ARTICLE_ID, PRINTTYPE, ThmAreamle) VALUES";

retcode = SQLPrepare(stmt, Query, SQL_NTS);
if (!retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO)  
    CheckStatementRetResult(retcode); // return failure
/* Bind the parameters. For parameter 3, pass */
/* the parameter number in ParameterValueTr instead of a buffer */
/* address. */
cbPDFFile = SQL_LEN_DATA_AT_EXEC(FileLength);
CheckStatementRetResult(SQLBindParameter(stmt, 1, SQL_PARAM_INPUT, SQL_C_CHAR,  
    SQL_CHAR, 6, 0, ArticleNo_c_str, ArticleNo.c_str(), 0, 0));
CheckStatementRetResult(SQLBindParameter(stmt, 2, SQL_PARAM_INPUT, SQL_C_SHORT,  
    SQL_SMALLINT, 0, 0, 4PrintType, 0, 0));
CheckStatementRetResult(SQLBindParameter(stmt, 3, SQL_PARAM_INPUT,  
    SQL_C_BINARY, SQL_LONGVARBINARY,  
    0, 0, SQLPOINTER 3, 0, &cbPDFFile));
/* Set values so data for parameter 3 will be */
/* passed at execution. Note that the length parameter in */
/* the macro SQL_LEN_DATA_AT_EXEC is 0. This assumes that */
/* the driver returns "M" for the SQL_NEED_LONG_DATA_LEN */
/* information type in SQLGetInfo. */

retcode = SQLExecute(stmt);
if (!retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO  
    || retcode == SQL_NEED_DATA)  
    CheckStatementRetResult(retcode); // return failure
/* For data-at-execution parameters, call SQLParamData to */
/* get the parameter number set by SQLBindParameter. */
/* Call InitUserData. Call GetUserData and SQLGetData */
/* repeatedly to get and put all data for the parameter. */
/* Call SQLParamData to finish processing this parameter */

while (rcode == SQL_NEED_DATA)  
    retcode = SQLParamData(stmt, 3pToken);
if (rcode == SQL_NEED_DATA)  
    SQLINTEGER cbData, DataLeft;
DataLeft = FileLength;
}

while (true) {
    cbData = DataLeft > MAX_DATA_LEN ? MAX_DATA_LEN : DataLeft;
    if (read(filehandle,Data,cbData) < 0)  
        throw Exception("Unable to read from file");
    SQLPutData(stmt, Data, cbData);
    DataLeft -= cbData;
    ProgressBar->Position += cbData;
    Update();
    if (DataLeft <= 0) break;
}
CheckStatementRetResult(retcode); // test for failure
return true; // flag that we were successful

APPENDIX D - MODULE DLGARTICLEDOWNLOADS
MODULE NAME - DLGCHANGEAUTHORINSTITUTION

#include <vcl/system.hpp>
#include <vcl/Win32.hpp>
#include <vcl/Units.hpp>
#include <vcl/Classes.hpp>
#include <vcl/Graphics.hpp>
#include <vcl/Forms.hpp>
#include <vcl/Controls.hpp>
#include <vcl/Buttons.hpp>
#include <vcl/StdCtrls.hpp>
#include <vcl/Intrins.h>
#include <vcl/TStrings.h>

class TForm: public TForm
{

protected:

private:

public:

virtual __fastcall TFormChangeAuthorInstitution(TComponent* Owner);
virtual __fastcall TFormChangeAuthorInstitution();

__fastcall TFormChangeAuthorInstitution(TComponent* Owner);
```cpp
TMob->SetInstitution(tob->GetInstitution_id());

void __fastcall TFormChangeAuthorInstitution::SetUpListBox()
{
    TMObject* tob;
    int institution_id;
    ClearObjects(lstInstitution);
    lstInstitution->Items->Clear();
    try {
        QryGeneral->Open();
        while (!((QryGeneral->EOF) ||
            QryGeneral->FieldByName("Institution_ID") != -1))
        {
            tob = new TMObject(0, institution_id, 0);
            lstInstitution->Items->AddObject(
                QryGeneral->FieldByName("Institution_ID") == tob->GetInstitution_id());
            QryGeneral->Next();
        }
        // set the item index for the current value
        for (int i=0;i<lstInstitution->Items->Count;i++)
        {
            if (tob->GetInstitution_id() == Tob->GetInstitution_id())
            lstInstitution->ItemIndex = i;
            break; // out of for loop
        }
    } catch (Exception e) {
        ShowMessage(e.Message);
        lstInstitution->Enabled = false;
        OKBtn->Enabled = false;
    }
    QryGeneral->Close();
}

void __fastcall TFormChangeAuthorInstitution::BtnAddInstitutionClick(TObject *Sender)
{
    TFormModInstitution* dlg = new TFormModInstitution(this);
    try {
        dlg->Setup(DatabaseName, false, 0);
        if (dlg->ShowModal() == mrOK) SetUpListBox();
        delete dlg;
    } catch (Exception e) {
        ShowMessage(e.Message);
        delete dlg;
    }
}
```

APPENDIX D - MODULE DLGCHANGEAUTHORINSTITUTION
APPENDIX D - MODULE DLGINSTITUTIONS
void __fastcall TFormInstitutions::BtnDeleteClick(TObject *Sender)
{
    // delete
    int institutionId = TblInstitutions->FindByName("Institution_ID")->AsInteger;
    if (institutionId == 1) {
        ShowMessage("Default institution cannot be deleted");
        return;
    }
    try {
        qryGeneral->Close();
        qryGeneral->SQL->Clear();
        AnsiString Query="Delete from Institutions where Institution_ID=

        Query += IntToStr(institutionId);
        qryGeneral->SQL->AddQuery(Query);
        qryGeneral->ExecuteSQL();
        TblInstitutions->Refresh();
        DBGridInstitutions->Refresh();
    }
    catch (Exception e) {
        ShowMessage(e.Message);
    }
}

MODULE NAME - DLGKEYWORDS

#include <vcl\system.hpp>
#include <vcl\Windows.hpp>
#include <vcl\Classes.hpp>
#include <vcl\Graphics.hpp>
#include <vcl\StdCtrls.hpp>
#include <vcl\Forms.hpp>
#include <vcl\Controls.hpp>
#include <vcl\Buttons.hpp>
#include <vcl\ExtCtrls.hpp>
#include <DB.hpp>

class TForm: public TForm
{
_published:
  TButton *OKBtn;
  TButton *CancelButton;
  TPanel *Bevel;
  TLabel *LblKeyName;
  TLabel *EdKeyWord;
  TLabel *LbxKeywords;
  TLabel *LbxFieldNames;
  TLabel *LbxArticleIds;
  TLabel *FormCaption;
  TLabel *MspCaption;
  TLabel *FormCaption2;
  TLabel *MspCaption2;
  TBitBtn *BntDelete;
  TBitBtn *BntAdd;
  TBitBtn *BntChange;
  TQuery *QryGeneral;

  void __fastcall EdKeyWordChange(TObject *Sender);
  void __fastcall LbxKeywordsClick(TObject *Sender);
  void __fastcall LbxFieldNamesClick(TObject *Sender);
  void __fastcall LbxArticleIdsClick(TObject *Sender);
  void __fastcall OKBtnClick(TObject *Sender);
private:
  AnsiString SessionName;
  AnsiString DatabaseName;
  AnsiString ArticleId;
  bool KeyWords;
  AnsiString KeyFieldNames;
  AnsiString TableName;
public:
  virtual __fastcall TFormChangeKeywords(TComponent* AOwner);
  void __fastcall SetUp(const AnsiString sessname,
                       const AnsiString dbname,
                       const AnsiString article_id,
                       bool keywords)
  {
  }
  //---

void __fastcall TFormChangeKeywords::TDKeyWordChange(TObject *Sender)
  {
  if (EdKeyWord->Text.Length() > 0)
    BntAdd->Enabled=true;
  else
    BntAdd->Enabled=false;
  }

}
void __fastcall TFormChangeKeyWords::BtnAddClick(TObject *Sender)
{
    int index = LbxKeywords->Items->IndexOf (EdKeyWord->Text);
    if (index >= 0) {
        ShowMessage("Word is already in list");
        return;
    }
    LbxKeywords->Items->Add(EdKeyWord->Text);
    EdKeyWord->Text = "");
    BtnAdd->Enabled = false;
    OKBtn->Enabled = true;
}

//--------------------------------------------------------------------------

void __fastcall TFormChangeKeyWords::LbxKeywordsClick(TObject *Sender)
{
    BtnChange->Enabled = true;
    LbxDelete->Enabled = true;
}

//--------------------------------------------------------------------------

void __fastcall TFormChangeKeyWords::BtnChangeClick(TObject *Sender)
{
    int index = LbxKeywords->ItemIndex;
    EdKeyWord->Text = LbxKeywords->Items->Strings[index];
    LbxKeywords->Items->Delete (index);
    BtnChange->Enabled = false;
    LbxDelete->Enabled = false;
    OKBtn->Enabled = true;
}

//--------------------------------------------------------------------------

void __fastcall TFormChangeKeyWords::BtnDeleteClick(TObject *Sender)
{
    int index = LbxKeywords->ItemIndex;
    LbxKeywords->Items->Delete (index);
    BtnChange->Enabled = false;
    LbxDelete->Enabled = false;
    OKBtn->Enabled = true;
}

//--------------------------------------------------------------------------

void __fastcall TFormChangeKeyWords::OKBtnClick(TObject *Sender)
{
    // this is where we update the keywords
    AnsiString Query;
    TSession ts = Sessions->FindSession(SessionName);
    TDatabase *td = ts->FindDatabase(DatabaseName);
    td->StartTransaction();
    TRY
    {
        Query = "Delete from " + TableName + " where article_id=";
        Query += ArticleID + ")
        QueryGeneral->Close();
        QueryGeneral->Close();
        QueryGeneral->SQL->Clear();
        QueryGeneral->SQL->Add (Query);
        QueryGeneral->Prepare();
        QueryGeneral->ParamsByName["Article_ID"]->AsString = ArticleID;
        for (int i=0; i<LbxKeywords->Items->Count; i++)
        {
            AnsiString kw = LbxKeywords->Items->Strings[i];
            QueryGeneral->ParamsByName[KeyFieldName]->AsString = kw;
        }
        QueryGeneral->ExecSQL();
        QueryGeneral->UnPrepare();
    }
    CATCH
    {
        ShowMessage(e.Message);
    }
}

//--------------------------------------------------------------------------

APPENDIX D — MODULE DLGKEYWORDS
MODULE NAME - DLGMODARTICLES

DLGMODARTICLES.H

ifndef DLGMODARTICLES_H
#define DLGMODARTICLES_H

#include <vc1\System.h>
#include <vc1\Windows.h>
#include <vc1\SysUtils.h>
#include <vc1\Classes.h>
#include <vc1\Graphics.h>
#include <vc1\StdCtrls.h>
#include <vc1\Forms.h>
#include <vc1\Controls.h>
#include <vc1\Buttons.h>
#include <vc1\ExtCtrls.h>
#include <Db.h>
#include <DBTables.h>
#include <DBCtrls.h>
#include <Mask.h>
#include <Menus.h>

#define DBCONNECT 0x0300
#define DBSCANNER 0x0300

class TDBModifyArticle : public TForm
{
  __published:
  TQuery *QryGeneral;
  TTable proc *SProcModify;
  TTable proc *SProcAdd;
  TPopupMenu *PopupMenu;
  TMenuItem *MenuInsertImage;
  TMenuItem *MenuChangeImage;
  TPanel *Panel1;
  TButton *BntModify;
  TButton *CancelBtn;
  TPanel *Panel2;
  TMemo *MemoAbstract;
  TLabel *Label1;
  TLabel *Label2;
  TLabel *Label3;
  TLabel *Label4;
  TLabel *Label5;
  TLabel *Label6;
  TLabel *Label7;
  TLabel *Label8;
  TLabel *Label9;
  TLabel *Label10;
  TLabel *Label11;
  TLabel *Label12;
  TLabel *Label13;
  TLabel *Label14;
  TStringList *MyImageList;
  char *ImgText;
  char *ImgId;
  int ImageStart;
  int ImageEnd;
  int ImageLen;
  int ImageMarkerLength;
  virtual void_fastcall GetImageDetails(int act=0);
  SQLHENV hEnv;
  SQLHDBC hDBC;
  SQLHSTMT hStmt;
  SQLRETURN retcode;
  void_fastcall AddImageRecord(TMArticleImageObject *tm);
  void_fastcall DeleteImageRecord(TMArticleImageObject *tm);
  void_fastcall CheckStatementResult(.SQLRETURN retcode);
  void_fastcall WriteImageRecord(int filehandle, TMArticleImageObject *tm);
  void_fastcall UpdateAbstractField();
  void_fastcall GetAbstractField();
  void_fastcall ReadAbstractField();

public:
  virtual __fastcall TDBModifyArticle(TComponent *Owner);
  void_fastcall ~TDBModifyArticle();
  void SetDB(TDataSource &ds, const AnsiString &dbName, bool modify);
};

#endif

DLGMODARTICLES.CPP

#include <vc1.h>
#include <vc1\clipbrd.h>

#pragma once
#include <fcntl.h>
#include <io.h>
#include "PipeCommon.h"
#include "DLGModArticles.h"
#include "DIBSelectImage.h"
#include "DBPipe.h"

#define MAXABSTRACTSIZE 6000

APPENDIX D - MODULE DLGMODARTICLES
void __fastcall TDlgModifyArticle::EdAcceptDateExit(TObject *Sender)
{
LbLError->Caption = "";
OKBtn->Enabled = false;
if (EdAcceptDate->Text.Trim().Length() > 1 )
    try
        { 
    TDateTime text(EdAcceptDate->Text);
    catch (Exceptions E) 
        { 
LbLError->Caption = "The Accept Date is not a valid date";
    display message
    EdAcceptDate->SetFocus();
    return;
        }
    OKBtn->Enabled = true;
    }
//_______________________________________________________________________________

void __fastcall TDlgModifyArticle::EdReceiveDateExit(TObject *Sender)
{
// check to see if valid date
LbLError->Caption = "";
OKBtn->Enabled = false;
if (EdReceiveDate->Text.Trim().Length() < 1 )
    try
        { 
    TDateTime text(EdReceiveDate->Text);
    catch (Exceptions E) 
        { 
LbLError->Caption = "The Receive date is not valid";
    display message
    EdReceiveDate->SetFocus();
    return;
        }
    OKBtn->Enabled = true;
    }
//_______________________________________________________________________________

bool __fastcall TDlgModifyArticle::CursorInsideImageMarker()
{
char* ImMarker="#";
char* TxtStrt = MemoAbstract->Text.c_str();
char* Cursor = TxtStrt + MemoAbstract->SelStart;
// this is starts at i = 0 means an empty string
if (*Cursor == 0) return false;
char* AltTxt;
// a non empty abstract exists
TxtStrt = strsrc(TxtStrt, ImMarker); 
while (TxtStrt)
{
// is the start less than this
if (Cursor <= ImStrt) // outside
    return false;
TxtStrt = strsrc(TxtStrt, ImMarker);
if (TxtStrt == 0) // invalid image pair marker
    return false;
// work out
if (Cursor <= ImEnd) ! // we have found it
    return alt text stuff
AltTxt = strchr(TxtStrt, '#');
AltTxtStart = 0;
if (AltTxt < ImEnd) { //alt txt found
    AltTxtStart = AltTxt - MemoAbstract->Text.c_str()+2;
    AltTxtLen = ImEnd - AltTxt-1;
    return true;
} else 
    return false;
    }

void __fastcall TDlgModifyArticle::PnuAbstractPopup(TObject *Sender)
{
// check to see if we can enable the change image
PnuChangeImage->Enabled = true;
else
    PnuInsertImage->Enabled = true;

//_______________________________________________________________________________

void __fastcall TDlgModifyArticle::PnuInsertImageClick(TObject *Sender)
{
// insert menu
GetImageDetails();
}
//_______________________________________________________________________________

void __fastcall TDlgModifyArticle::PnuChangeImageClick(TObject *Sender)
{
// change image
GetImageDetails();
}
//_______________________________________________________________________________

TStrings * ts; 
int i; 
T MA rticleImageOn* tob; 
T FrmSelectImage + Dlg;
try
    { 
    Dlg = new TFormSelectImage(this);
    catch(Exceptioins E) { ShowMessage(e.Message); return ;
    // before loading db with imagelist need to check to see if alttext has changed
    AnsiString AltText;
    int itemno;
    if (act == 1) 
        { 
    image = MemoAbstract->Text.Substring(ImageStart+2, ImageLen);
    itemno = MyImageList->IndexOf(image);
    if (itemno >= 0 && AltTxtStart ) { this is item no we have on file
    // update the alt text from the memo box
    AltText = MemoAbstract->Text.Substring(AltTxtStart, AltTextLen );
    tob =(TMArticleImageOn*) MyImageList->Objects[itemno];

APPENDIX D — MODULE DLGMODARTICLES
void_fastcall TDlgModifyArticle::UpdateAbstractFile(TMemAbstractData* &tio)
{

// Get the parameter number set by SQLBindParameter.
// Call GetUserData and SQLPutData
// repeatedly to get and put all data for the parameter.

while (retcode == SQLEND{DATA})
{

CheckStatementRetResult(retcode); // return failure

if (read(filehandle, DataLeft, cbData) < 0)

throw Exception("Unable to read from file");

cbData = cbData;
DataLeft = cbData;
}
return; // flag that we were successful
}

void_fastcall
TDlgModifyArticle::UpdateAltText(TMArticleImageCB) *tio
{

Assuming Query = "Update ArticleAbstractImages set ImageAltText = ";
Query = tio->AltText + ", where article_id = ";
Query = Query + Tstr(tio->ImageNumber);
}

ModifyGeneral->Close();
ModifyGeneral->SQL->Clear();
ModifyGeneral->SQL->Add(Query);
ModifyGeneral->SQL->ExecSQL();
}

void_fastcall TDlgModifyArticle::DeleteImageRecord(TMArticleImageCB) *tio
{

Assuming Query = "Delete from ArticleAbstractImages where article_id = ";
Query = Query + Tstr(tio->ImageNumber);
}

ModifyGeneral->Close();
ModifyGeneral->SQL->Clear();
ModifyGeneral->SQL->Add(QUERY);
ModifyGeneral->SQL->ExecSQL();
}

TCursor temp = Screen->Cursor;
Screen->Cursor = cNoHourClass;
try {

WriteAbstractField();
}
finally {

SQLFreeStmt(screen, SQ_CLOSE);
Screen->Cursor = temp;
}

void__fastcall TDlgModifyArticle::WriteAbstractField()
{

#define MAX_DATA_LEN 4096

SQLINTEGER cbPDFFile=0;
SQLPOINTER pToken;
SQLRETURN retcode;

 GryGeneral->Close();

Assuming Query = "Update articles set abstract= ? where article_id = ";
Query = &ArticleId->Text + "";

checkStatementResult(retcode); // return failure

/* Bind the parameters. For parameter 3, pass */
/* the parameter number in ParameterValuePtr instead of a buffer */
/* address. */
cbPDFFile = SQL_L TypeDescriptor->TextLength();

CheckStatementRetResult(retcode); // return failure

/* The macro SQL_L TYPE_DESCRIPTOR is 0. This assumes that */
/* passed at execution. Note that the length parameter in */
/* for the SQL_N獐_DATA AT EXEC is 0. This assumes that */
/* the driver returns "N" for the SQL_N獐_DATA_LEN */
/* information type in SQLGetInfo. */

retcode = SQLExecute(screen);
if (!retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO )

CheckStatementRetResult(retcode); // return failure

/* Get the parameter number set by SQLBindParameter. */
/* Call GetUserData. Call GetUser and SQLPutData */
/* repeatedly to get and put all data for the parameter. */
/* Call SQLParamData to finish processing this parameter */

while (retcode == SQL_NEED_DATA)
{

CheckParamData(screen, &pToken);

if (retcode == SQL_NEED_DATA) {

SQLINTEGER cbData, DataLeft;

set Left = MemoAbstract->TextLength();

char* Data = MemoAbstract->Text.c_str();

while (true) {

cbData = DataLeft > MAX_DATA_LEN ? MAX_DATA_LEN : DataLeft;
}

APPENDIX D - MODULE DLG MOD ARTICLES
```c
// The function to retrieve the abstract field from the database.

#include <windows.h>
#include <sql.h>
#include <sqlext.h>

void_fastcall TD1gModifyArticle::GetAbstractField() {
    // Check if the connection is successful.
    if (dataRetCode == SQL_SUCCESS)
        // Get the abstract data from the database.
        SQLGetData(hstmt, 1, SQL_C_CHAR, &dataRetCode, &dataSize, &data);
    else
        // Return if the connection is not successful.
        return;
}
```

**APPENDIX D - MODULE DLGMODARTICLES**
// when update is clicked
try {
    if (ProcModifying) {
        SProcModify->Params->Items[1]->AsString = EdSmallDesc->Text;
        SProcModify->Params->Items[2]->AsString = EdAddress1->Text;
        SProcModify->Params->Items[3]->AsString = EdAddress2->Text;
        SProcModify->Params->Items[4]->AsString = EdAddress3->Text;
        SProcModify->Params->Items[5]->AsString = EdAddress4->Text;
        SProcModify->Params->Items[6]->AsString = EdAddress5->Text;
        SProcModify->Params->Items[7]->AsString = EdAddress6->Text;
        SProcModify->Params->Items[8]->AsString = EdFascimile->Text;
        SProcModify->Params->Items[9]->AsInteger = institution_id;
        SProcModify->Prepare();
        SProcModify->ExecProc();
    }
}
int res = SProcModify->Params->Items[0]->AsInteger;
if (res < 0) {
    LblError->Caption = "Small Desc already exists in table";
    EdSmallDesc->SetFocus();
    return;
}
if (res == 0) {// should never happen
    LblError->Caption = "The institution was deleted elsewhere";
    EdSmallDesc->SetFocus();
    return;
}
else {//add a record
    SProcAdd->Params->Items[1]->AsString = EdSmallDesc->Text;
    SProcAdd->Params->Items[2]->AsString = EdAddress1->Text;
    SProcAdd->Params->Items[3]->AsString = EdAddress2->Text;
    SProcAdd->Params->Items[4]->AsString = EdAddress3->Text;
    SProcAdd->Params->Items[5]->AsString = EdAddress4->Text;
    SProcAdd->Params->Items[6]->AsString = EdAddress5->Text;
    SProcAdd->Params->Items[7]->AsString = EdAddress6->Text;
    SProcAdd->Params->Items[8]->AsString = EdFascimile->Text;
    SProcAdd->Prepare();
    SProcAdd->ExecProc();
    if (SProcAdd->Params->Items[0]->AsInteger < 1) {
        LblError->Caption = "Small Desc already exists in table";
        EdSmallDesc->SetFocus();
        return;
    }
    ModalResult = mrOK;
    return;
} catch (Exception & e) {
    ShowMessage(e.Message);
}

/*-------*/

void_fastcall TForm1::EdSmallDescChange(TObject *Sender) {
    BtnUpdate->Enabled = true;
}
```
#include "ipamscommon.h"
#include "DlgNewVolume.h"

#pragma resource "*.dfm"

TfrmNewVolume *FrmNewVolume;

fastcall TfrmNewVolume::TfrmNewVolume(TComponent *Owner)
    : TForm(Owner)
    {
    AmModifying=false;
    }

fastcall void __fastcall TfrmNewVolume::Setup(const AnsiString dname, TMVOLObject *tvb,
        bool ammodifying)
    {
    DatabaseName=dname;
    TMV tvb;
    AmModifying=ammodifying;
    EdVolNo->Text = IntToStr(tvb->VOLUME_NO);
    EdIssueNo->Text = IntToStr(tvb->ISSUE_NO);
    EdYear->Text = IntToStr(tvb->ISSUEYEAR);
    EditDate->Text = tvb->DateOfIssue.FormString("dd/mm/yy");
    BntUpdate->Enabled=false;
    }

fastcall void __fastcall TfrmNewVolume::EdVolumeNoChange(TObject *Sender)
    {
    BntUpdate->Enabled=true;
    }

fastcall void __fastcall TfrmNewVolume::EdIssueDateExit(TObject *Sender)
    {
    try {
    TDateTime v(EDateDate->Text);
    } catch (Exceptions e) { ShowMessage("Invalid date");
    EditDate->SetFocus();
    }
    }

fastcall void __fastcall TfrmNewVolume::OkBtnClick(TObject *Sender)
    {
    try {
    if (AmModifying) {
    SprocMod->Params->Items[1]->AsInteger = StrToInt(EdVolNo->Text.Trim());
    SprocMod->Params->Items[2]->AsInteger = StrToInt(EdIssueNo->Text.Trim());
    SprocMod->Params->Items[3]->AsInteger = StrToInt(EdYear->Text.Trim());
    SprocMod->Params->Items[4]->AsInteger = EdDate->Text.Trim();
    SprocMod->Params->Items[5]->AsInteger = tvb->VOLUME_ID;
    SprocMod->Prepare();
    SprocMod->ExecProc();
    }
    if (SprocMod->Params->Items[0]->AsInteger < 1) { // should never happen
    ShowMessage("Volume with ID " + IntToStr(tvb->VOLUME_ID) + " No longer exists");
    return;
    }
    else // add a record
    SprocAdd->Params->Items[1]->AsInteger = StrToInt(EdVolNo->Text.Trim());
    }
```
SprocAdd->Params->Items[2]->AsInteger = StrToInt(EditNo->Text.Trim());
SprocAdd->Params->Items[3]->AsInteger = StrToInt(EditYear->Text.Trim());
SprocAdd->Params->Items[4]->AsString = EditIssueDate->Text.Trim();
SprocAdd->Prepare();
SprocAdd->ExecProc();
int vol_id = SprocAdd->Params->Items[0]->AsInteger;
if (vol_id < 1) // should never happen
    ShowMessage("Volume already exists - not added");
    return;
}
ModalResult = mrOk;
return;
}
catch (Exception & e) {
    ShowMessage(e.Message);
    return;
}

//-------------------------------------------------------

APPENDIX D - MODULE DLGNEWVOLUME
void __fastcall TFormSelectImage::EdAltTextExit(TObject *Sender)
{
    TMArticleImageObj *tob = (TMArticleImageObj *)lbxAvailable->Items->Objects[ 
        lbxAvailable->ItemIndex];
    if (tob->AltText != EdAltText->Text) { 
        tob->AltText = EdAltText->Text; 
        tob->AltTextChange = true; 
    }
} //-------------------------------------------------------------------------------

APPENDIX D - MODULE DLGSELECTIMAGE
MODUL NAME - DLGSELECTNEWAUTHOR

DLGSELECTNEWAUTHOR.H

#include <vcl.h>
#include <vclSystem.h>
#include <vclWindows.h>
#include <vclForms.h>
#include <vclClasses.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclObjects.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclControls.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclObjects.h>
#include <vclClasses.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>
#include <vclControls.h>
#include <vclGraphics.h>
#include <vclSysUtils.h>
#include <vclStrings.h>
#include <vclStrings.h>
#include <vclDialogs.h>
#include <vclGUI.h>

APPENDIX D - MODULE DLGSELECTNEWAUTHOR
void_fastcall T FrmSelectNewAuthor::ClearObjects(TListBox* ltb)
{
    for (int i=0; i<ltb->Items->Count; i++)
    {
        TMObeject* tob = (TMObeject*)ltb->Items->Objects[i];
        delete tob;
    }
}

void_fastcall T FrmSelectNewAuthor::LbxNewAuthorsClick(TObject *Sender)
{// select an item in the listbox
    OKBtn->Enabled= true;
}

void_fastcall T FrmSelectNewAuthor::OKBtnClick(TObject *Sender)
{// we have chosen an item
    // as the listbox entries are userroles 0 or 3 we need to update their role
    // in users to reflect their new status
    // the entry is pass back to the calling process
    AnsiString Query;
    QueryGeneral->Close();
    QueryGeneral->SQL->Clear();
    TMObeject* tob = (TMObeject*)LbxNewAuthors->Item->Objectx[
    LbxNewAuthors->ItemIndex ];
    Query= "Update users set UserRole= " ;
    if (tob->GetUserRole() == 3) { // Editor becoming author as well
        Query= "4" ;
    } else { // only a user
        Query= "2" ;
    }
    Query+= " where user_id = " + IntToStr(tob->GetUser_id());
    QueryGeneral->SQL->Add( Query);
    try {
        QueryGeneral->ExcuteSQL();
        Successful=true;
    } catch (Exceptions e) {
        ShowMessage(e.Message);
    }
}
as
Ml

5ft

f^Hf

-—

C QJ

01 .* •*
tf)fitn G
0J JJ ra to o
TJ TJ fi
H •* £ QJ H
TJ M fd
G 01 >i M JJ
ra g 2H fi fd TJ ra
H 3 JH M fd H TJ H
H co tn 01 G ftrt H
ra to M X! M tO XJ MH
3 -HflJMH
. BraH JJ
O CO Q S rt
I
Id
Cu Cu XJ JG A A A
O TJ
fi O i 2 X J XJMl Mi Mi A XI
i JH AJ JH tJ> rji rji rji fji
: CT CT CT G G Ctiti
JJ JJ JJ AJ
C AJ
C AJ
G JJ
O M lui
01 C
01 C0J
G G
G G
i 01fll
01 0GJfll
e co p
l P HP HP HI HA HA HA -HP
ra p id •
AJJJAJJJJJAJAJJJ
ft tr
'-DQMincncocncococncn
= =
f d C C G C G G G C
J G O O O O O O O O
Q J C U O O U O O O O O C J
TJ TJ
3 3 C G G G G C G G G
H H M M M M M M M M M
OOQJQlQiaiaifllfllQJOJ
CGJJJJJJJJJJJJJJJJA
• H H X X X X X X X X X
-tt-ttaiQiaiaifliflifliQiaj

r>
c
A

QJ

ro %

CO
CO

M

CD

D
X
(1
I)
H

(
l
(f
)

A
UI

(fl1

MH

Tl
•H

n

0.
TJ
p
CTTJ JJ
0 01 QJ
M H CO
DJ -H
G ro H
M H Cu H
rsft H C)
D 0
•H JH £ A
•n 01 XJ CO
M CO P m
A

•*

J<

M

TJ
M
II
EH H •* 0
— 0
"v, ~-~ CD
H "s. TJ PJ"
M M
0 •v O 01
G O tfl
0 01 P
H tt —
JJftft
O P D
01 JJ JJ
M 01 fll
H CO CO
Q
TJ TJ
JJ H H
C O O
•H H > >
H

ra

0
C)

OJ
PC
u H
CD 0
o
P XJ

MH

1
H TJ
0 •A
0 0
XJ

> tf) >

M
(IJ
cn
>i
MH
•H
TJ
O

,-)

s
Hm

ra

es

f1
H
Cd
C)

srt
11

w
tt
S A

J->

CU

01
M
M
3
O

O M
Cu p
Id
I
I
EH II
QJ
< -. to
tfl
f/1
•^ Q
oi ai
MI fl)
P
A to EH X M Q)
JJ H Id 01 TJ 3
ra CO E H TJ M
H
01 M H E A
Tl
A
JJ 11
t H II
O
ra T J n <n H
TJ
01
tf)
m
tu
ft M >i 01 n
H
A
3 -H M M
(H
3 ai T J
•A
H
G CT 3 TJ
ra ai a ft l.
H
OX
rA + CD
ra C T H
rr,
0)fitias>i
CJ
A
3 F H fi A
to
O M Id (1)
MH [J JJ TJ 3 o
MH
H
CO W a u

>.

E MH
fi -H
MH Tl
TJ O

<

^

• X
* CT
01 -x
U
M M
3 QJ
O tf)
CO D
01 >.
A MH
H

°a
H

ra
ft
&

S ^
o

«

M

r>E v E+

+
E

ro

M

A + +
t >it

>1 H
0) TJ
3 M
HFi oi o •3 OJ
M
H — cn 3
TJ CO M
JJ OJraJJ
X M OJ II
QJ H M
3 OJ TJ
A tr to cu
i 01 P M
TJ
-H
II
n
•? i + oi
tfi O
X
CO u >i ro

§

&

ra

ra

ro -a

v lu

DJ
cu E
TJ MH 3 0
Id H a o

P

0)
3 II
ii a
cu

— fi

JJ TJ ra
X QJ Z
01 M AJ
EH H to
A 3 M
1 CTH
fllflltu
fi X t
as as
2 B II
CO o
M CJ >.
H — M
Cu fl)
TJ MH 3
W H O

o q tu
xirt-H
v > —-^

CD
TJ
G
01
CO
*
JJ

tf)

CD
•- JJ

A

5.S
UH MH

CU
cn
O
tn

0J
cn
O
tn

IH 0J
01 c
73 -H
C tJH

a a

a ax:

o. a

a A
a. a

XI XI A XI o,
a j3
CO rH
• CO CJ H o, • XI
•rl U a to •
A6 3
i
x: JJXI rH to
as o JJ
CJ
U 73 D i Q . U
JJ
CO cl to . ro 73
X
U
JJ
rH-rl >,
CJ
C/3 IS CO Uctl
^^*r^*^*r^'A
^- ^ S* -H rH ^-t t-l -4 -^
-I -A -\

a

a
cn o,
x:
A ai a
a n XI
Cl. .Q •
US ^
rj • tH cn

ouuooooximcg
v
o vu vu v v v v v v v v v v

tuaituaiCLiajfiitLiaituajruaj
73T37J737373T37373T373'073
3 3 3 3 3 3 3 3 3 3 3 3 3
--IrS-A^s^SM^^-l^-l^-t^t^t
O O O C J C J C J O O C J O C J C J C J

' >

c c c c c c c c c c c c c
HjatHtHt^at^^aCHtH;*^^

oi ra
JJ o
tO JJ
ra H JJ
X
•°as
H CM ro ** m UJ JJ 3 01
,-{ rA rA rA M fll
2
3 03 . H ^\
ai 01fllfllfllfllraG C
XJ XJ XJ XJ XJ XJ JH fllA
rorofororaco ttCQ
A A A A A A XS XS
* + -v + + Id Id
C
0
H H H H H H
01 01 01 01 01 01 JJ JJ A
A
XJ
XJ
XJ
XJ
XJ
H
H
X
J
^
-r, J^ I"
rararararaT J T J 3
73 3 JJ > p j j J A A Id Id CQ
EH H EH EH £H £H £-. £H H
» EH EH

M

...MO M...01
H O (fl

5 g' ra
ra

M
OJ
G
A
PQ

M
A
Id
H
XJ
p
+

**
* * C0 H

PO •H»
TJ
rS O
J
>iXJ
M ra
cn P
•K

•*

>iH
0J M QJ
XJ 01 XJ
3 ra
X A a P
EH H EH EH
A
A

ti
ra

A
H
C ?-.
H
M
QJ G
A M
Ml 3
OO CTi
CO H H
o
TJ TJ
Id id QJ 01
XJ rQ
A A p p
H H
TJ TJ
td Cd H H
M M 01 01
tn to XJ XJ
as
Se X A A
EH
FH E- EH

QJ
B
as
Z
>i

G
Ho
A

c
0
H -v
JJ QJ
3 H
JJ O
H tt
JJ M
CO 01
C (0
H p
XJ
X XJ
X
U U
* +

MH

CM en
H H
H H
01 01
XJ + XJ
A A
A JJ A X X HI A
•A * 0 o
TJ
CQ ffl
rA Id H o o H H
01 OJ
CD r X QJ
XJ cn XJ
XJ XI
as
o o
A X A U CJ A A

H TJ
S
a H
H
Id 0J Id CU

in

OJ

Qj

01 CO TJ
TJ * C
G
OJ
CD JJ CO
Ql CO U *
CO * fll

-*

•I-I

AJ

-A
O
0)
-r-i
XJ
O
EH

ti ti

raHro
ft
cn -^
£
H H H

H
H
MH

—.
M
01
TJ
G

JJ XJ o
O O OJ
01 E H -I-I
• r » — XJ
XJ JJ o
O -H H
EH x —
— Id .M
fllU
— rX
O JH H
Ho -X
H (J •H ro H
CJ H H P CJ
fll HU H C
tn
A CJ M ra A
tn
OJ
cn ro JJ O 3 CQ
CO
CD
TJ
X •A QS rA
u
73 TJ F CD ft Ql M G flJ
73 A as u P z DJfllU
G tt C
ft O 2 TJ
*-t 3 Ml TJ Ml Ml A TJ ra
•H
tn rt CQ X X Cd CJ
CO
to
M XJ
v
0J
| rA rA rA
OJ fu
H
H H •-i i-i rA
H
TJ TJ s
TJ
01 Id Id Id Id o o O O O
•f
JJ JJ JH JJ A
(n cn CO CO CO
r3 JH JJ A JJ
•H H •A •H MH 4H MH "JH MH
I 1
TJ TJ TJ TJ
1
H Cd Id US Cd
tu M H- rX rX
Xl m Cfl W CO TJ T J xi XD xi
•A H H H H
to
O 0 O O O
A X '•-•'
EH H IH H H
O
01
•n
XJ
O
EH
—
-V

a a

_naJ

* e tf)
ra

ro ra
ra ra ra

9
ro 5ra* * ro ro
A ra ro
ra
ro
H
* TJ
* **
Id
-X
(0

ii

• *

1

1 1

ro rara ra
ra
ra
ra
ra
ra
ra
ro
ro
X
EH

EH

EH

EH

EH E H

EH

EH

> >> > >


CommaRequired = true;
}
#
if (EdOtherInit->Text != Ur->OtherInit) {
    if (CommaRequired) Query += ";
    Query += "OtherInit = " + EdOtherInit->Text.Trim() + " ";
    CommaRequired = true;
#
if (EdSurname->Text != Ur->Surname) {
    if (CommaRequired) Query += ";
    Query += "Surname = " + EdSurname->Text.Trim() + " ";
    CommaRequired = true;
#
if (EdWebAddress->Text != Ur->WebAddress) {
    if (CommaRequired) Query += ";
    Query += "WebAddress = " + EdWebAddress->Text.Trim() + " ";
    CommaRequired = true;
#
if (EdAffiliation->Text != Ur->Affiliation) {
    if (CommaRequired) Query += ";
    Query += "Affiliation = " + EdAffiliation->Text.Trim() + " ";
    CommaRequired = true;
#
if (EdDisplayName->Text != Ur->DisplayName) {
    if (CommaRequired) Query += ";
    Query += "DisplayName = " + EdDisplayName->Text.Trim() + " ";
    CommaRequired = true;
#
if (CbxInstitution->ItemIndex != Ur->ItemIndex) {
    AnsiString Query = "Select Institution_ID from Institutions where SmallDesc = ";
    Query += CbxInstitution->Items->Strings[CbxInstitution->ItemIndex] + " ";
    Query += QuerySQL->Add("Institution_ID") + " ";
    CommaRequired = true;
}
if (CbxUserRole->ItemIndex != Ur->UserRole) {
    if (CommaRequired) Query += ";
    Query += "UserRole = " + IntToStr(CbxUserRole->ItemIndex) + " ";
    CommaRequired = true;
#
if (EdStartDate->Text != Ur->StartDate) {
    if (CommaRequired) Query += ";
    Query += "StartDate = " + EdStartDate->Text + " ";
    CommaRequired = true;
#
if (EdRenewalDate->Text != Ur->RenewalDate) {
    if (CommaRequired) Query += ";
    Query += "RenewalDate = " + EdRenewalDate->Text + " ";
    CommaRequired = true;
#
Query += where User_id = " + IntToStr(Ur->User_id) + " ";
Query += QuerySQL->Add("Institution_ID") + " ";
try {
    QueryModUser->SQL->Clear();
    QueryModUser->SQL->Add(AnsStrBool);  
    QueryModUser->SQL->Add("User_id") + " ";
    QueryModUser->SQL->Add(AnsStrBool);  
    QueryModUser->SQL->Add("User_Name") + " ";
    QueryModUser->SQL->Add(AnsStrBool);  
    QueryModUser->SQL->Add("User_Pwd") + " ";
    QueryModUser->SQL->Add(AnsStrBool);  
    QueryModUser->SQL->Add("OtherInit") + " ";
    QueryModUser->SQL->Add(AnsStrBool);  
    QueryModUser->SQL->Add("Surname") + " ";
    QueryModUser->SQL->Add(AnsStrBool);  
    QueryModUser->SQL->Add("WebAddress") + " ";
    QueryModUser->SQL->Add(AnsStrBool);  
    QueryModUser->SQL->Add("Affiliation") + " ";
    QueryModUser->SQL->Add(AnsStrBool);  
    QueryModUser->SQL->Add("UserRole") + " ";
    QueryModUser->SQL->Add(AnsStrBool);  
    QueryModUser->SQL->Add("StartDate") + " ";
    QueryModUser->SQL->Add(AnsStrBool);  
    QueryModUser->SQL->Add("RenewalDate") + " ";
    QueryModUser->SQL->Add(AnsStrBool);  
    QueryModUser->SQL->Add("SmallDesc") + " ";
    QueryModUser->SQL->Add(AnsStrBool);  
    QueryModUser->SQL->Add("Institution_ID") + " ";
    QueryModUser->SQL->Add(AnsStrBool);  
    QueryModUser->SQL->Execute();
}
lblError->Caption = EdEmailAddress->Text + " Update failed";
DBNotFailed = false;
BtnPrior->Enabled = false;
BtnNext->Enabled = false;
}
BtnUpdate->Enabled = false;

void _fastcall TDlgModifyUser::BtnNextClick(TObject *Sender) {
    if (BtnUpdate->Enabled) {
        if (MessageDlg("Update Record?", mConfirmation, 
            TMsgButtons() << mbYes << mbNo, 0) == mbYes) {
            BtnUpdateClick(Sender);
        }
    }
    if (BtnUpdate->Enabled) return; // failed
}
// go to next record if we can
lblError->Caption="";
Direction=2;
ModalResult=mrYes;
}

void _fastcall TDlgModifyUser::BtnPriorClick(TObject *Sender) {
    if (BtnUpdate->Enabled) {
        if (MessageDlg("Update Record?", mConfirmation, 
            TMsgButtons() << mbYes << mbNo, 0) == mbYes) {
            BtnUpdateClick(Sender);
        }
    }
    if (BtnUpdate->Enabled) return; // failed
}
// go to prior record if we can
lblError->Caption="";
Direction=-1;
ModalResult=mrYes;
}

void TDlgModifyUser::SetUpRecord() {
    SetUpInProgress=true;
    EdEmailAddress->Text = Ur->EmailAddress;
    EdPassword->Text = Ur->UserPassword;
    EdFirstName->Text = Ur->FirstName;
    EdOtherInit->Text = Ur->OtherInit;
    EdSurname->Text = Ur->Surname;
    EdWebAddress->Text = Ur->WebAddress;
    EdAffiliation->Text = Ur->Affiliation;
    EdDisplayName->Text = Ur->DisplayName;
    AnsiString Query = "Select SmallDesc from Institutions where Institution_ID = ";
    Query += IntToStr(Ur->InstitutionID) + " ";
    try {
        QueryModUser->SQL->Clear();
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("User_id") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("User_Name") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("User_Pwd") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("OtherInit") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("Surname") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("WebAddress") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("Affiliation") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("UserRole") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("StartDate") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("RenewalDate") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("SmallDesc") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("Institution_ID") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Execute();
    }
    lblError->Caption = EdEmailAddress->Text + " Updated successfully";
    catch (Exception &e) {
        ShowMessage(e.Message);
    }
    DBNotFailed = true;
    SetUpInProgress=false;
    SetUpInProgress=false;
    EdEmailAddress->Text = Ur->EmailAddress;
    EdPassword->Text = Ur->UserPassword;
    EdFirstName->Text = Ur->FirstName;
    EdOtherInit->Text = Ur->OtherInit;
    EdSurname->Text = Ur->Surname;
    EdWebAddress->Text = Ur->WebAddress;
    EdAffiliation->Text = Ur->Affiliation;
    EdDisplayName->Text = Ur->DisplayName;
    AnsiString Query = "Select SmallDesc from Institutions where Institution_ID = ";
    Query += IntToStr(Ur->InstitutionID) + " ";
    try {
        QueryModUser->SQL->Clear();
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("User_id") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("User_Name") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("User_Pwd") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("OtherInit") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("Surname") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("WebAddress") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("Affiliation") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("UserRole") + " ";
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("StartDate") + " ");
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("RenewalDate") + " ");
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("SmallDesc") + " ");
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Add("Institution_ID") + " ");
        QueryModUser->SQL->Add(AnsStrBool);  
        QueryModUser->SQL->Execute();
    }
    lblError->Caption = EdEmailAddress->Text + " Update failed";
    DBNotFailed = false;
    BtnPrior->Enabled = false;
    BtnNext->Enabled = false;
    }
    BtnUpdate->Enabled = false;

// TAppendix D - Module DLGUSERMODIFY


void_fastcall TDlgModifyUser::SetUpdateButton(TObject *Sender) {
    if (BtnUpdate->Enabled) {
        if (MessageBoxDlg("Update Record?", mtConfirmation, 
                         TMLogButtons() << mbYes << mbNo, 0) == mbYes) {
            BtnUpdateClick(Sender);
            if (BtnUpdate->Enabled) return; // failed
        }
    }
    ModalResult=-1;cancel;
}

void_fastcall TDlgModifyUser::SetUpdateButton() {
    BtnUpdate->Enabled = false; 
    if (EDEmailAddress->Text.Trim().Length() < 1) return;
    if ((EDEmailAddress->Text != Ur->EmailAddress)) ||
        (EDPassword->Text.Trim() != Ur->UserPassword) ||
        (EDFirst_Name->Text != Ur->First_Name) ||
        (EDOtherInit->Text != Ur->OtherInit) ||
        (EDSurname->Text != Ur->Surname) ||
        (EDWebAddress->Text != Ur->WebAddress) ||
        (EDAffiliation->Text != Ur->Affiliation) ||
        (EDDisplay_Name->Text != Ur->Display_Name) ||
        (CBUserRole->ItemIndex != Ur->UserRole) ||
        (EDStartDate->Text != Ur->StartDate.DateString()) ||
        (EDRenewalDate->Text != Ur->RenewalDate.DateString()) ) {
        BtnUpdate->Enabled = true; return;
    }
}
//..........................................................................
void_fastcall TDlgModifyUser::CBUserRoleExit(TObject *Sender) {
    SetUpdateButton();
}

//..........................................................................
void_fastcall TDlgModifyUser::EDRenewalDateExit(TObject *Sender) {
    TEdit* ted = dynamic_cast<TEdit*>(Sender);
    LblError->Caption = ted->Text + " Database failed";
    BtnUpdate->Enabled = false;
    BtnPrior->Enabled = false;
    BtnNext->Enabled = false;
    DbNotFailed = false;
    } 
CBxInstitution->ItemIndex = Ur->ItemIndex;
CBxUserRole->ItemIndex = Ur->UserRole;
EDStartDate->Text = Ur->StartDate.DateString();
EDRenewalDate->Text = Ur->RenewalDate.DateString();
BtnUpdate->Enabled = false;
// EDEmailAddress->SetFocus();
SetUpdateProgress = false;
} //..........................................................................
void TDlgModifyUser::SetUpdateProgress(bool b) {
    try {
        CBxInstitution->ItemIndex = Ur->ItemIndex;
        CBxUserRole->ItemIndex = Ur->UserRole;
        EDStartDate->Text = Ur->StartDate.DateString();
        EDRenewalDate->Text = Ur->RenewalDate.DateString();
       BtnUpdate->Enabled = false;
        // EDEmailAddress->SetFocus();
        SetUpdateProgress = false;
    } catch (Exception & e) {
        ShowMessage(e.Message);
        LblError->Caption = EDEmailAddress->Text + " Setup failed";
        BtnUpdate->Enabled = false;
        BtnPrior->Enabled = false;
        BtnNext->Enabled = false;
        DbNotFailed = false;
    } 
CBxInstitution->ItemIndex = Ur->ItemIndex;
CBxUserRole->ItemIndex = Ur->UserRole;
EDStartDate->Text = Ur->StartDate.DateString();
EDRenewalDate->Text = Ur->RenewalDate.DateString();
BtnUpdate->Enabled = false;
// EDEmailAddress->SetFocus();
SetUpdateProgress = false;
} //..........................................................................
void TDlgModifyUser::EDPasswordExit(TObject *Sender) {
    try {
        CBxInstitution->ItemIndex = Ur->ItemIndex;
        CBxUserRole->ItemIndex = Ur->UserRole;
        EDStartDate->Text = Ur->StartDate.DateString();
        EDRenewalDate->Text = Ur->RenewalDate.DateString();
        BtnUpdate->Enabled = false;
        // EDEmailAddress->SetFocus();
        SetUpdateProgress = false;
    } catch (Exception & e) {
        ShowMessage(e.Message);
        LblError->Caption = EDEmailAddress->Text + " Setup failed";
        BtnUpdate->Enabled = false;
        BtnPrior->Enabled = false;
        BtnNext->Enabled = false;
        DbNotFailed = false;
    } 
CBxInstitution->ItemIndex = Ur->ItemIndex;
CBxUserRole->ItemIndex = Ur->UserRole;
EDStartDate->Text = Ur->StartDate.DateString();
EDRenewalDate->Text = Ur->RenewalDate.DateString();
BtnUpdate->Enabled = false;
// EDEmailAddress->SetFocus();
SetUpdateProgress = false;
} //..........................................................................
APPENDIX D - MODULE DlgUserModify
MODULE NAME - DLGVWARTICLES

#ifndef DLGVWARTICLES
#define DLGVWARTICLES

#include <conio.h>
#include <Windows.h>
#include <SysUtil.h>
#include <Classes.h>
#include <Graphics.h>
#include <StdCtrls.h>
#include <Forms.h>
#include <Controls.h>
#include <Buttons.h>
#include <LstCtrls.h>
#include <StdCtrls.h>
#include < Dio.h>
#include <DBIds.h>
#include <DBTables.h>
#include <Grids.h>
#include <USBCtrls.h>

#endif

class TDlv FrmViewArticles: public TForm
{
    _published:
    Table *TBArticles;
    TDBTableSource *DBtArticles;
    TDBGrid *DBGridarticles;
    TPanel *Panel1;
    TDButton *BtnOk;
    TDButton *BtnModify;
    TDButton *BtnKeyWords;
    TDButton *BtnAuthors;
    TDButton *BtnRotate;
    TDButton *BtnPrint;
    TDButton *BtnDownloads;
    TDButton *BtnCloseBtn;
    TDNavigator *TDBNavigator;
    TQuery *QryDBArticles;
    TVirtualMenu *TMnuFilter;
    TMnuItem *ChangeFilter;
    TMnuItem *RemoveFilter;
    void_fastcall BtnDownLoadsClick(TObject *Sender);
    void_fastcall BtnModifyClick(TObject *Sender);
    void_fastcall BtnOKClick(TObject *Sender);
    void_fastcall BtnKeyWordsClick(TObject *Sender);
    void_fastcall BtnAuthorsClick(TObject *Sender);
    void_fastcall DBGridArticlesDeleteClick(TObject *Sender);
    void_fastcall DBGridArticlesColClick(TObject *Sender);
    void_fastcall DBFilterChangeFilterClick(TObject *Sender);
    void_fastcall DBFilterCurrentFilterClick(TObject *Sender);
    void_fastcall DBFilterChangeFilterRequested(TObject *Sender);
    private:
    void_fastcall ChangesortColumn();
    private:
    AnsiString ColumnFieldName;
    AnsiString FilterText;
    AnsiString FilterColumn;
    AnsiString SortColumn;
    static char *Baseqry;
    public:
    virtual_fastcall TDlv FrmViewArticles(TComponent *AOwner);
void __fastcall TDlgFrmViewArticles::DBGridArticlesColEnter(TObject *Sender)
    
    Column.FieldName = DBGridArticles->Columns->Items [DBGridArticles->SelectedIndex]->FieldName;

void __fastcall TDlgFrmViewArticles::RemoveFilter1Click(TObject *Sender)
    
    // Clear the filter text
    FilterText = "";
    FilterColumn = "";
    ChangeSortColumn();

// Appendix D - Module DLGVwArticles
# Module Name - DLGvWUsers

## DLGvWUsers.h

```c
#include <vcl.h>
#include "DialogCommon.h"
#include "DLGvWUsers.h"
#include "DlgAddUser.h"
#include "DlgUserModify.h"

// include <clnt.h>
//pragma hdrstop

#include <winsock2.h>
#include <ws2tcpip.h>

#pragma resource "*.dfm"
TdlgFrnViewUsers *TdlgFrnViewUsers;

fastcall TdlgFrnViewUsers::TdlgFrnViewUsers(TComponent * AOwner)
    : TForm(AOwner)
    

void __fastcall TdlgFrnViewUsers::BtnAddUserClick(TObject *Sender)
    { // add another user
        TdlgAddUser *Dlg = new TdlgAddUser();
        Dlg->Setup(QryUsers->DatabaseName);
        Dlg->ShowModal();
        delete Dlg;

        QryUsers->Last();
        TGrid->Refresh();
    }

fastcall TdlgFrnViewUsers::BtnModifyClick(TObject *Sender)
    { // modify a user
        UserRecord *CurrentUser;
        TdlgModifyUser *Dlg = new TdlgModifyUser();
        Dlg->Setup(QryUsers->DatabaseName);
        while (true)
            { Dlg->BtnNext->Enabled = true;
              Dlg->BtnPrior->Enabled = false;
              if (QryUsers->EOF)
                { DtblUsers->Prior();
                  Dlg->BtnNext->Enabled = false;
                }
              if (QryUsers->Bof)
                { DtblUsers->Next();
                  Dlg->BtnPrior->Enabled = false;
                }
        }
```

### Appendix D - Module DLGvWUsers
void_fastcall TDlg FrmViewUsers::ChangeSortColumn()  
{  
QryUsers->Close();  
QryUsers->SQL->Clear();  
QryUsers->SQL->Add(_BaseQry);  
if (!FilterText.Length() > 0)  
{  
QryUsers->SQL->Add("where " + FilterColumn + " like "+FilterText + "+'";  
}  
QryUsers->SQL->Add(" order by " + SortColumn);  
QryUsers->Open();  
}  

void_fastcall TDlg FrmViewUsers::ChangeFilter1Click(TObject *Sender)  
{  
// change the filter text  
AnsiString col= ColumnFieldName;  
if (col != FilterColumn)  
{  
FilterText = "";  
}  
AnsiString InputString=filterText;  
if (InputQuery("Set Filter Text for " + col, "Enter filter ", InputString))  
{  
if (InputString.Trim().Length() > 0)  
{  
FilterColumn=col;  
FilterText=InputString.Trim();  
ChangeSortColumn();  
}  
}  
}  

void_fastcall TDlg FrmViewUsers::RemoveFilter1Click(TObject *Sender)  
{  
// clear the filter text  
FilterText = "";  
FilterColumn=1;  
ChangeSortColumn();  
}  

void_fastcall TDlg FrmViewUsers::DMGrid1ColEnter(TObject *Sender)  
{  
ColumnFieldName = DMGrid1->Columns->Items [ DMGrid1->SelectedIndex]  
->FieldName;  
}  

void_fastcall TDlg FrmViewUsers::BtnDeleteClick(TObject *Sender)  
{  
// delete a user  
try {  
QryUsers->Close();  
DMGrid1->Refresh();  
}  
catch (Exceptions er) {  
ShowMessage(er.Message);  
}  
}  

void_fastcall TDlg FrmViewUsers::DMGrid1Delete(TObject *Sender)  
{  
DMGrid1->DeleteRow();  
}  

void_fastcall TDlg FrmViewUsers::DMGrid1Click(TObject *Sender)  
{  
// sort the column  
AnsiString col= Column->FieldName;  
AnsiString msg = "Change sort order to " + col + ";";  
if (MessageDlg(msg, mConfirmation, mErrorMsg) )  
{  
DMGrid1->Sort(col, mSortOrder);  
DMGrid1->SortColumn(col);  
}  
}  

char* TDlg FrmViewUsers::BaseQry = "select  
user_id, EmailAddress, UserPassword, FirstName, OtherInit, Surname, DisplayName, WebAddress,  
UserRole, Institution_ID, Affiliation, StartDate, RenewalDate, LastAccessed from Users";  

// DLMODULEDlgWVUSERS
#include "DlgVwVolumes.h"
#include "DlgAddArtToVolume.h"
#include "DlgNewVolume.h"

//----------------------------------------------------------------------------------
#pragma resource "*.dfm"

TFrwVolumes *FrmVwVolumes;

_fastcall TForm::TfrmVwVolumes(TComponent *AOwner)
  : TForm(AOwner)
  {
  }

_fastcall TForm::TfrwVolumes::TfrwVolumes()
  {
  ClearObjects(ListVolumes);
  }

void _fastcall TForm::ClearObjects(TListBox *lb){
  for (int i=0; i<lb->Items->Count; i++)
    TMvObject *obj = (TMvObject*) lb->Items->Objects[i];
    delete obj;
  }

void _fastcall TForm::SetUp(const AnsiString dbName) {
  DatabaseName = dbName;
  QueryGeneral->DatabaseName = dbName;
  SetUpListBox();
  }

void _fastcall TForm::ListVolumesClick(TObject *Sender)
  {
  TMvObject *obj = (TMvObject*) ListVolumes->Items->Objects[
    ListVolumes->Items->Count];
  FrmAddArtToVolume *v = new FrmAddArtToVolume(this);
  try {
    Dlg->Caption = L"ListVolumes->Items->Strings[ListView->Items->Index] + " Articles";
    Dlg->SetUp(DatabaseName, obj->VolumeID);
    Dlg->ShowModal();
  }
  catch (Exception e) {
    ShowMessage(e.Message);
  }
  delete Dlg;
  }

void _fastcall TForm::BtnNewClick(TObject *Sender)
  {
  // view articles on volume
  TMvObject *tvb = (TMvObject*) ListVolumes->Items->Objects[
    ListVolumes->Items->Count];
  FrmAddArtToVolume *v = new FrmAddArtToVolume(this);
  try {
    Dlg->Caption = L"ListVolumes->Items->Strings[ListView->Items->Index] + " Articles";
    Dlg->SetUp(DatabaseName, tvb->VolumeID);
    Dlg->ShowModal();
  }
  catch (Exception e) {
    ShowMessage(e.Message);
  }
  delete Dlg;
  }

void _fastcall TForm::BtnNewClick(TObject *Sender)
  {

APPENDIX D - MODULE DlgVwVolumes
```cpp
// add another volume
TfrmNewVolume *Dlg = new TfrmNewVolume(this);
TMOObject *tbv = new TMOObject;
if (lbxVolumes->items->count > 0) {
    *tbv = *(TMOObject*)lbxVolumes->items->objects[i];
lbxVolumes->items->count = 1;
}
tbv->issue_no++;
try {
    Dlg->SetUpDatabaseName(tbv->false);
    if (Dlg->ShowModal() == mrk) {
        SetUpListBox();
    }
    catch (Exceptions & e) {
        ShowMessage(e.Message);
    }
    delete tbv;
    delete Dlg;
}
void __fastcall TfrmVwVolumes::BtnChangeClick(TObject *Sender)
{
    // change volume
    if (lbxVolumes->items->index < 0) return;
    TfrmNewVolume *Dlg = new TfrmNewVolume(this);
    AnsiString caption = "Modify ";
    caption += lbxVolumes->items->Strings[lbxVolumes->items->index]
    Dlg->Caption = caption;
    TMOObject *tbv;
    tbv = (TMOObject*)lbxVolumes->items->objects[i];
lbxVolumes->items->index = i;
    try {
        Dlg->SetUpDatabaseName(tbv->true);
        if (Dlg->ShowModal() == mrk) {
            SetUpListBox();
        }
        catch (Exceptions & e) {
            ShowMessage(e.Message);
        }
        delete Dlg;
    }
    void __fastcall TfrmVwVolumes::SetUpListBox()
    {
        ClearObjects(lbxVolumes);
lbxVolumes->items->clear();
        // fill the list box
        QryGeneral->Close();
        QryGeneral->SQL->Clear();
        QryGeneral->SQL->Add("Select volume_id,VOLUME_NO,ISSUE_NO,ISSUEYear,ISSUEDate from VOLUMES");
        try {
            QryGeneral->Open();
            while (!QryGeneral->eof) {
                TMOObject *tbv = new TMOObject;
                tbv->volume_id = QryGeneral->fieldByName("Volume_id")->asinteger;
tbv->VOLUME_NO = QryGeneral->fieldByName("VOLUME_NO")->asinteger;
tbv->ISSUE_NO = QryGeneral->fieldByName("ISSUE_NO")->asinteger;
tbv->ISSUEYear = QryGeneral->fieldByName("ISSUEYear")->asinteger;
tbv->DateOfIssue = QryGeneral->fieldByName("ISSUEDate")->asdatetime;
        }
    }
    TMOObject *tbv = new TMOObject;
tbv->volume_id = QryGeneral->fieldByName("Volume_id")->asinteger;
tbv->VOLUME_NO = QryGeneral->fieldByName("VOLUME_NO")->asinteger;
tbv->ISSUE_NO = QryGeneral->fieldByName("ISSUE_NO")->asinteger;
tbv->ISSUEYear = QryGeneral->fieldByName("ISSUEYear")->asinteger;
tbv->DateOfIssue = QryGeneral->fieldByName("ISSUEDate")->asdatetime;
    // APPENDIX D - MODULE DLGVWVOLUMES
```
# MODULE NAME - JIPAMCOMMON

## JIPAMCOMMON.h

```c
#include "JIPAMCOMMON_H"

#define JIPAMCOMMON_H

struct UserRecord
{
    int user_id;
    AnsiString emailAddress;
    AnsiString userName;
    AnsiString firstName;
    AnsiString otherNames;
    AnsiString surname;
    AnsiString webAddress;
    AnsiString affiliation;
    AnsiString displayName;
    int institutionID;
    int itemIndex;
    int userId;
    TDateTime startDate;
    TDateTime endDate;
    int lastAccessed;
}

extern AnsiString EMAILPassword;
extern int lengthEmailaddress;
extern int lengthPassword;
extern int lengthFirstName;
extern int lengthOtherNames;
extern int lengthSurname;
extern int lengthWebAddress;
extern int lengthAffiliation;
extern int lengthKeyName;
extern int lengthNumCode;
extern int lengthMailDesc;
extern int lengthAddressLine1;
extern int lengthAddressLine2;
extern int lengthAddressLine3;
extern int lengthAddressLine4;
extern int lengthAddressLine5;
extern int lengthTelephone;
extern int lengthFaxNumber;

class ArticleObject : public TObject
{
    private:
        AnsiString articleId;
    public:
        ArticleObject() { moved=false; }
        ArticleObject(AnsiString aid) { moved=false; articleId = aid; }
        ArticleObject* (ArticleObject* t) { articleId = t->articleId; moved=false; }
        AnsiStrings& GetArticleId() { return articleId; }
    bool GetMoved() { return moved; }
        void SetArticleId(AnsiString aid) { articleId = aid; }
        void SetMoved() { moved=true; }
        ArticleObject* operator=(ArticleObject t) { articleId = t.articleId; return *this; }
}

class PrintTypeObject : public TObject
{
    private:
        int printType;
    public:
        PrintTypeObject() { printType=1; }
        PrintTypeObject(int pt) { printType = pt; }
        PrintTypeObject* (PrintTypeObject* t) { printType = t.printType; }
        int GetPrintType() { return printType; }
}

class TMObject : public TObject
{
    private:
        int user_id;
        int institution_id;
        int userRole;
        int year;
    public:
        TMObject(int uid, int lid, int yr, int yr=0) { user_id=uid; institution_id=lid; userRole=yr; year=yr; }
        TMObject(TMObject* t) { user_id=t->user_id; institution_id=t->institution_id; userRole=t->userRole; year=t->year; }
        TMObject() { user_id=0; institution_id=0; userRole=0; year=0; }
        int GetUserId() { return user_id; }
        int GetInstitutionId() { return institution_id; }
        int GetUserRole() { return userRole; }
        int GetYear() { return year; }
        void SetInstitution(int iid) { institution_id=iid; }
        TMObjects operator=(TMObjects t) { user_id=t.user_id; institution_id=t.institution_id; userRole=t.userRole; year=t.year; return this; }
}

class TMVolObject : public TObject
{
    public:
        int volume_id;
        int volume_no;
        int issue_no;
        int issueYear;
        TDateTime dateOfIssue;
    TMVolObject(int volume_id, int volume_no, int issue_no, int issueYear) { volume_id=volume_id; volume_no=volume_no; issue_no=issue_no; issueYear=issueYear; }
}

class TMArticleImageObj : public TObject
{
    public:
        short status; //1 = existing, 2 = new, 3 = deleted
        int ImageNumber;
        AnsiString sourceFile;
        AnsiString sourceFileType;
        AnsiString owner;
        AnsiString altTextChange;
        TMArticleImageObj() { status = 1; ImageNumber = 0; altTextChange = false; }
}

class TMEditorObj : public TObject
{
    public:
        int editor_id;
        TMObject() { editor_id=0; }
}

class TMFullArticleObj : public TObject
{
    public:
        AnsiString title;
        AnsiString receiveDate;
        AnsiString acceptedDate;
        int editorId;
        bool Available;
        bool Published;
}

APPENDIX D - MODULE JIPAMCOMMON
```
void __fastcall TfmMainForm::SetupDatabase( AnsiString DBName )
{
    TDatabase * tdb = DataModuleJipam->DBJipam;
    if ( tdb->Connected ) return;
    tdb->DatabaseName = DBName;
    TSession * ts = DataModuleJipam->SessionJipam;
    if ( !ts->Active ) ts->Open();
    int LoginTryCnt = 3;
    while (LoginTryCnt > 0) {
        try {
            tdb->Open();
            catch ( Exception & e ) {
                ShowMessage( e.Message );
            }
            if ( tdb->Connected ) return;
            LoginTryCnt--;
        }
        Close();
    }
}

void __fastcall TfmMainForm::BtnViewUsersClick(TObject *Sender)
{
    TDigFrnViewUsers * Dlg;
    try {
        Dlg = new TDigFrnViewUsers( this );
        catch ( Exception & e ) {
            ShowMessage( e.Message );
            return;
        }
        this->Hide();
        try {
            Dlg->Setup( FrmDefaultDatabase );
            Dlg->ShowModal();
            catch ( Exception & e ) {
                ShowMessage( e.Message );
            }
            this->Show();
            delete Dlg;
        }
    }
}

void __fastcall TfmMainForm::BtnWvArticlesClick(TObject *Sender)
{
    TDigFrnViewArticles * Dlg;
    try {
        Dlg = new TDigFrnViewArticles( this );
        catch ( Exception & e ) {
            ShowMessage( e.Message );
            return;
        }
        this->Hide();
        try {
            Dlg->Setup( FrmDefaultDatabase );
            Dlg->ShowModal();
            catch ( Exception & e ) {
                ShowMessage( e.Message );
            }
            this->Show();
            delete Dlg;
        }
    }

APPENDIX D - MODULE JIPAMMAIN
void __fastcall TFormMain::BtnInstitutionsClick(TObject *Sender)
{
    TFormInstitutions *Dlg;
    try {
        Dlg = new TFormInstitutions(this);
        Dlg->SetUpDatabase(PrmDefaultDatabase);
        Dlg->ShowModal();
    } catch (Exception &e) {
        ShowMessage(e.Message);
    }
    this->Hide();
    try {
        Dlg->Hide();
        Dlg->SetUpDatabase(PrmDefaultDatabase);
        Dlg->ShowModal();
    } catch (Exception &e) {
        ShowMessage(e.Message);
    }
    this->Show();
    delete Dlg;
}

void __fastcall TFormMain::BtnLeaveClick(TObject *Sender)
{
    Close();
}

void __fastcall TFormMain::BtnViewUsersClick(TObject *Sender)
{
    BtnViewUsers->Enabled=false;
    BtnViewArticels->Enabled=false;
    BtnInstitutions->Enabled=false;
    TFormViewUsers->Enabled=false;
    TFormInstitutions->Enabled=true;
    PrmDefaultDatabase->SetDatabaseName("...");
    TFormViewUsers->ShowModal();
    TFormInstitutions->Enabled=true;
    TFormInstitutions->Enabled=false;
    TFormViewUsers->Enabled=false;
    TFormInstitutions->Enabled=true;
    TFormViewUsers->Enabled=true;
    TFormInstitutions->Enabled=true;
    TFormViewUsers->Enabled=false;
    TFormInstitutions->Enabled=true;
}

APPENDIX D - MODULE JIPAMMAIN
Journal of Inequalities in Pure and Applied Mathematics

Administration Program Guide
Administration Program Guide

Trevor McGuckin
Phone 03 9743 5768

January 2001

School of Communications and Informatics
Faculty of Engineering and Science

© Trevor McGuckin, 2001
## Table of Contents

**CHAPTER 1 — INTRODUCTION**

JIPAM Web System .................................................. 1
Creating the JIPAM Database .................................... 2
MS Internet Information Server ................................. 2
ODBC Installation .................................................... 3
Admin. Program Installation ....................................... 6

**CHAPTER 2 — ADMIN. PROGRAM**

Starting the Admin. Program ...................................... 7

**CHAPTER 3 — SUBSCRIBERS**

Subscriber Data Maintenance ................................. 9
Subscriber Search ................................................... 10
Add a Subscriber ................................................... 11
Modify a Subscriber ............................................... 13
Delete a Subscriber ............................................... 14

**CHAPTER 4 — ARTICLES**

Add an Article ...................................................... 16
Add Images to an Abstract ...................................... 18
Modify an Article .................................................. 19
Add/Delete/Reorder Article Authors ......................... 20
Change a Subscriber Role to Author Role .................. 22
Change an Article Author's Institution ...................... 23

**CHAPTER 4 (CONT) — ARTICLES**

Add/Modify the Keyword of an Article ....................... 24
Add/Modify the Math Code of an Article ..................... 25
Add/Modify the PDF Files of an Article ...................... 27

**CHAPTER 5 — VOLUMES/ISSUES**

Volumes/Issue Maintenance ...................................... 29
Create a New Volume/Issue ...................................... 30
Add/Modify/Reorder of a Volume/Issue ....................... 31
Change Volume/Issue Details .................................... 33

**CHAPTER 6 — INSTITUTIONS**

Default Settings for the Institution Field .................. 35
Institution Data Maintenance ................................... 35
Add a New Institution ............................................ 36
Modify an Institution's Details ............................... 38
Delete an Institution ............................................. 39

**CHAPTER 7 — STATISTICS**

Tracking Users ...................................................... 40
Visitor Reports ..................................................... 41

**GLOSSARY** ......................................................... 44
Introduction

This guide contains detailed information as a reference source for the JIPAM Administrator to initially setup the JIPAM Web System and then to cover its ongoing maintenance. Several features have been included into this written guide to assist in administering the JIPAM web site effectively.

For instance, icons are shown in the left margin where appropriate to help navigate this guide.

JIPAM Web System

The JIPAM Web System (the JIPAM System) comprises the following three elements.

- A database component
- A web browser component
- An administration component

The JIPAM System is based on Microsoft SQL Server Version 7 database software and uses Microsoft Internet Information Server (IIS) as the web server. IIS uses Open Database Connectivity (ODBC) to retrieve information from the database. A JIPAM Administration Program (the administration component) has been developed to maintain the JIPAM System (See Chapter 2 onwards).

A JIPAM Installation CD-ROM has been provided (included as Appendix A) and full setup instructions for these three components follows.

Note

It is necessary for the Admin. Program to also use ODBC to connect to the database.
Creating the JIPAM Database

The JIPAM Database (called the database from now on) has been developed using Microsoft SQL Server Version 7 database software. It is the basis of the JIPAM System and is created using three SQL procedures followed by an NT Batch Process provided on the JIPAM Installation CD. The first two SQL procedures set up the database tables, the final SQL procedure loads real data into these tables while the NT batch file concludes loading the data by inserting the relevant PDF files and any associated image files.

The first SQL procedure assumes the presence of the JIPAM database. The database needs to be created by the SQL Server Database Administrator on the computer hosting the SQL database program.

1. Once the database exists, run the Query Analyzer Program provided with Microsoft SQL Server Version 7 and then login to the JIPAM database as administrator. Run the first SQL procedure file, JIPAM/SQL/SETUP.SQL from the JIPAM Installation CD using the Query Analyzer Program.

2. Run the second SQL procedure file, JIPAM/SQL/PROCEDURES.SQL from the JIPAM Installation CD using the Query Analyzer Program.

3. Run the third SQL file, JIPAM/SQL/DATALOAD/DATALOAD.SQL from the JIPAM Installation CD using the Query Analyzer Program to load the database with real data.

4. To finish the JIPAM database process, run the NT Batch file, JIPAM/SQL/DATALOAD/LOADPDF.CMD from the JIPAM Installation CD to load the database with the PDF files and any associated image files.

MS Internet Information Server

The MS Internet Information Server (IIS) administrator will need to create an application called JIPAM. The contents of the JIPAM/JIPAM directory are to be copied from the JIPAM Installation CD into the IIS JIPAM application directory. The copy process must also include all of subdirectories of JIPAM/JIPAM on the JIPAM Installation CD.

The IIS administrator will also need to create an ODBC system DSN called JIPAM (see the following section).
ODBC (Open Database Connectivity) Installation

The JIPAM administrator program (referred to as the admin program from now on) requires a data source based on the ODBC (Open Database Connectivity) driver (provided by Microsoft) to access the SQL Server 7 database. If the driver is not installed on the administrator’s computer, refer to the SQL Server 7 documentation for details about installing the driver.

➢ To create a data source named JIPAM

1. In Control Panel, click the ODBC Data Sources icon.

2. Click System DSN tab. Select SQL Server and then click Add.

3. Select SQL Server from the Name list and click Finish.

4. A wizard to create an ODBC data source is provided. Enter the data source name as JIPAM and the description as SQL Server Connection to JIPAM. Select the network address of the computer on which the JIPAM
database will reside (the appropriate SQL server) from the drop down list. Click **Next**.

5. The next dialogue box should be completed after consultation with the SQL Server Database Administrator. It provides two options regarding the logon process to the JIPAM database. Given that the database is also accessible via the web, the second option will probably be sufficient. Complete the screen and click **Next**.

6. Complete the dialogue box as shown below and click **Next**.
7. No changes are necessary, so again just click **Next**.

8. No changes are necessary in the final step to create a data source named **JIPAM** so click **Finish** in the dialogue box below.

9. On completion of the wizard, the specifications of the new ODBC data source are displayed. The specifications shown below are a test setup only and your configuration will be different.
10. Click **Test Data Source** to ascertain that the database can be accessed. Your details will be different to that shown below which is a test only. If your test is successful, click **OK** and the data source will be added to the System DSN table in the ODBC Manager. If unsuccessful, consult the SQL Database Administrator.

![SQL Server ODBC Data Source Test](image)

11. The creation of the ODBC interface is now complete.

**Admin. Program Installation**

The JIPAM Admin. Program allows the database to be maintained on an ongoing basis. Maintenance is undertaken based on the following four logical views of the database:

- Subscribers (online users (including authors and editors) who have registered to access the full text of articles published in JIPAM)
- Articles (articles published in all JIPAM volumes)
- Volumes (each issue of JIPAM is known as a volume)
- Institutions (subscribers provide institutional details of their institution.)

Run the file **JIPAM/INSTALL/SETUP.EXE** from the JIPAM Installation CD following the screen prompts to install the JIPAM Administration Program (called Admin. Program from now on) into the directory you require. A menu item called JIPAM Admin is added to the Programs menu as part of the Admin. Program installation process. It is accessed from the Start button on your desktop.

Ongoing maintenance of the JIPAM Web System can be undertaken after the Admin. Program has been successfully installed.
The Admin. Program

The Admin. Program is a management tool for the System Administrator to maintain the data in the JIPAM database. It is structured on four logical views of the data (for instance, the data associated with the subscribers, the articles, the volumes and the associated institutions) and options are provided for the administrator to view and/or modify the data.

Starting the Admin. Program

➢ To login to the Admin. Program

1. Click Start on your desktop. Click on Programs then click on the submenu item called JIPAM Admin.

2. Login to the Admin. Program as the System Administrator by entering your User Name and Password.

Note

To undertake System Administrator functions, the login user name must have read, write, modify and execute permissions for the JIPAM database. When the database was originally created, a login identify with user name JPADMN and password JPADMN was created for system administration functions.
3. The **Main Screen JIPAM Admin. Program** displays the options available to the System Administrator in terms of maintaining the database.

4. The first four buttons allow specific sections of the database to be viewed and/or updated while **Quit** is used to exit the JIPAM Admin. Program.

**Note**

Detailed documentation relating to maintenance activities associated with each of the four views of the database are provided in the following four chapters.
Subscribers

To gain access to the full text of articles published in JIPAM, it is necessary for users to be registered online as subscribers. This is a simple matter of providing relevant data such as an email address and login details that are stored within the JIPAM database. At this stage, subscriptions are free although that may change at some date in the future. The Admin. Program allows the JIPAM System Administrator access to subscriber details for maintenance purposes.

Subscriber Data Maintenance

In maintaining the subscriber-related data in the database, it is necessary to view the contents of the database.

Click View Subscribers on Main Screen JIPAM Admin. System. The View Subscribers screen provides easy access to the subscriber data.

The dialogue box is resizable to allow more or less detail to be shown in the table. Four navigation buttons are provided to allow the position of the selected subscriber’s record data to be changed in the database table (see above).

Maintenance of subscriber data is structured around the following three actions.
- Add a subscriber
- Modify a subscriber's data
- Delete a subscriber

**Subscriber Search**

Two support facilities have been incorporated into the View Subscribers screen to assist in locating a specific subscriber's record within the database. They can be used singly or in combination.

*Sort the subscriber table on any of the fields* by clicking on the required field in the Title Bar. A confirmation message is displayed which identifies the sort criteria. Click **Yes** to change the order of the table to your selection or **No** to leave the table unchanged. The subscribers table will initially open in **Surname** order.

![Confirm](image)

*Search the subscriber table using filter text* by right clicking on any column in the table although complex filters on more than one column are not currently supported. A pop up a menu appears with two options (1) Set filter and (2) Remove filter.

1. Selecting **Set filter** opens the Set Filter Text for Surname dialog box shown below which allows the filter text to be entered.

![Set Filter Text for Surname](image)

2. Click **OK** to apply the filter. The View Subscribers screen will then be displayed showing only the records that meet the filter criteria.

3. To remove the filter, right click again and select **Remove Filter**. (The **Cancel** button on the Set Filter Text for
Surname screen does not remove the filter.) Alternatively, close the View Subscribers screen and reopen it to return to the default setting with all subscribers records being shown in surname order (i.e. with no filters applied).

Note

To list all subscribers who have a surname containing the text 'jones', right click on the surname column in the View Subscribers screen and enter the text 'jones' in the Set Filter Text for Surname entry box. The View Subscribers screen will list only those subscriber records that contain the text 'jones' anywhere in the surname field.

Add a Subscriber

Users can be manually registered as new subscribers in instances where the online registration process is unsuccessful or they are unable to use the online process.

➢ To add a subscriber

1. Click View Subscribers on Main Screen JIPAM Admin. System.
2. Click Add on the View Subscribers screen.
3. The Add Subscriber dialog box is displayed.

4. Fill in the required data fields (detailed explanations of the data fields are provided below).

- Email Address is the email address of the user who is to be registered as a subscriber. This attribute identifies who the subscriber is to be and therefore must be unique within the database.
• **Password** is the subscriber’s password (minimum length is four characters).

• **First Name, Other Inits and Surname** fields are self-explanatory. They can be left blank.

• **Web Address** is provided for those authors/editors who have their own web site and who would like users to know the URL. The field may be left blank.

• **Display Name** is the name that subscribers wish to have displayed on any screens produced in the web environment. The field is only applicable to authors and editors.

• **Affiliation** is the name that subscribers enter to identify their institution. Its primary purpose is to provide information to the System Administrator so that they can allocate the correct institution to authors and editors. The field is potentially important to authors and editors.

• **Institution** can only be selected from the displayed list. When users originally register as subscribers, the default entry in their **Institution** field is **None**. At the moment, the **Institution** field is only maintained (that is, changed from **None**) for authors and editors, not for all subscribers. Thus, the administrator needs to manually change the default entry (None) for all authors and editors based on the affiliation data they originally entered with their subscription information. If the institution required is not in the list, see Chapter 6 for details on adding a new institution into the database. On the **Add Subscribers** screen, the ‘Affiliation’ edit box is locked and changes to reflect the contents of the institution field.

• **Role** is selected from the **Role** drop down combination box. There are currently defined roles of **General, Author, Editor, Both and Admin**. All subscribers are assigned the **General** tag when they register online. The **Author, Editor and Both** roles are expected to be used more heavily when authors and editors can control the submission and acceptance of articles.

5. Click **Add** when all necessary data has been entered and this adds the new subscriber’s details into the database. The addition will be successful provided the email address is unique. An error message will be displayed if the email address (the unique identifier) already exists.

6. Click **Close** to leave the subscribers table without adding any new subscribers to the database.
Modify a Subscriber

Although subscribers are able to modify most of their own details online, two data fields can only be changed by the System Administrator. These are the subscriber’s role and the institution to which they are currently assigned. Rights to change this data have been restricted to the administrator to provide higher levels of security for the JIPAM Web System.

- **Role.** This field is used in a number of SQL procedures that determine who are JIPAM authors and editors. It is also used to enable the Display Name field for authors and editors when they want to change their login details from the web. The Admin role enables a subscriber (who is also the administrator) to access the Web Reports screen. Access to the Web Reports screen and other administration functions is restricted for security purposes.

- **Institution.** This field is only accessible to the administrator. Refer to Chapter 6 for more information.

In addition, if a subscriber cannot remember their password (and they do not use the automatic email-back facility provided in the web login screen), the System Administrator has access to their password on the Modify Subscriber screen.

➢ **To modify a subscriber**

1. Click **View Subscribers** on **Main Screen JIPAM Admin. System.** Select the required subscriber from the list.

2. Click **Modify** on the **View Subscribers** screen.

3. The **Modify Subscriber** dialog box is displayed complete with the selected subscriber’s data.
4. Fill in the required **data fields** (detailed explanations of the data fields are provided below).

**Note**

All comments related to the contents of the fields described in the previous section on adding new subscribers also apply to modifying subscriber data.

- **Renewal Date** is twelve months from the subscription date. If the field needs to be changed, the date must be a valid date otherwise an error message will be displayed and the cursor will remain in the edit box.

- **Start Date** is the date when the subscriber initially subscribed to the JIPAM Web System.

5. Click **Update** to save the modified data. At this point, the subscriber’s record in the database is changed. However, due to system and program limitations, the **Modify Subscribers** screen must be closed before you will see the modified record.

6. Click **Close** when finished.

**Delete a Subscriber**

It is expected that it would rarely be necessary to manually delete a subscriber as subscriptions automatically lapse after a period of twelve months when the subscriber data is automatically deleted. However, subscriber data can be manually deleted.

➢ **To delete a subscriber**

1. Click **View Subscribers** on **Main Screen JIPAM Admin. System**.

2. **Select** the Subscriber on the **View Subscribers** screen.

3. Click **Delete**. (There will be no confirmation request).

4. If the subscriber cannot be deleted, an error message box (similar to below) will be displayed and the record will not be deleted.
The Admin. Program provides the administrator with the facility to manage the articles published in JIPAM. Articles can be added to the database as soon as the draft article is received from the author. These drafts are then immediately available online (as new papers). When they are accepted for publication, their status changes and they are included in the contents of the current volume/issue.

Articles comprise two components: (1) the article details which includes the article title, author/s, keyword/s, maths code/s and the abstract (which may contain image files if formulas are discussed in the abstract) and (2) the full text of the article.

Click View Articles on Main Screen JIPAM Admin. System. The View Articles screen provides easy access to the articles.

The dialogue box displays all the articles in the database in tabular form and is resizable to allow more or less detail to be shown in the table. You can also adjust the width of individual columns using the cursor in the title bar of the table. However, your column size changes are not saved and the dialog box will revert to the default column sizes (see above) the next time you select View Articles.

Four navigation buttons are also provided to allow the position of the selected article to be changed in the database table (as shown above).
Adding and maintaining articles is structured around the following six actions.

- Add an article
- Modify an article
- Add/Modify an article's author
- Add/Modify an article’s keywords
- Add/Modify an article’s maths code
- Add/Modify an article’s PDF files

Add an Article

When articles are first added to the database, they are flagged as **Unpublished Articles**. After they have been published in a volume/issue, their status is changed to **Published Articles** (See Chapter 5 for more information). There are several steps to adding an article.

➢ To add an article

1. Click **View Articles** on **Main Screen JIPAM Admin. System**.

2. Click **Add** on the **View Articles** screen.

3. The **Add an article** dialog box is displayed. It is resizable to allow the abstract text to be entered into the abstract memo box.

4. Fill in the required **data fields** (detailed explanations of the data fields are provided below).
• **Article Id (Identifier)** is pre-filled from the database with the next article id sequence. The value can be changed but it must be of the form '999_99' where 999 represents the article number and 99 represents the year the article was received.

• **Article Title** is where the text of the new title is entered. It is necessary to enter at least one non-space, printable character.

• **Receive Date** is the date the article draft was received. A valid date must be entered into this field; otherwise, an error message will be displayed.

• **Accept Date** is the date the article was accepted (approved) for publication. This field may be left blank but if a date is entered it must be valid otherwise an error message will be displayed.

• **Editor** contains a drop down list of the journal's editors. The editor accepting the article is chosen from this list. New editors can be added to the list by changing the user role as described in the 'Modifying a Subscriber' section in the previous chapter.

• **Available** provides two options to determine if the article being added to the database is to be available for web-based activities such as searching, or author article listings. Click Y (yes) if the draft article is to be made available online. The default for a new article is set to N (no). This flag is provided to allow the administrator to enter all the article details before it is made available on the web.

• **Abstract** is where the text of the abstract is entered. Only text can be added in this field that means that HTML code can also be entered to control the display of the article on the web. This is necessary to incorporate functions (as image files) into the abstract. Right clicking in the abstract area will bring up the image menu. See the following section for more details on how to include images in the abstract.

5. Click **Add** when all necessary data has been entered (Add will only be enabled when both the **Article Title** and the **Receive Date** have been completed correctly. This adds the new article's details into the database and closes the dialog box.

---

**Note**

If the article details entered cannot be written to the database, an error message will be displayed and the dialog box will not close.
6. Click Close to leave without adding any new articles to the database.

**Add Images to an Abstract**

Some abstracts may need to contain images because mathematical formula may also be referenced in the abstract text. As the abstract field in the database is a text field, two different methods can be used to include image files into the abstract.

Hard code the `<IMG>` HTML tags in the abstract field and point the SRC attribute (source) to a hard coded file name.

Use a marker scheme developed for the Admin Program that stores the image file in the database and references the image in the abstract. Using the marker scheme, an image is identified in the abstract by the code sequence `<###imagenn#alt-text##>`. The `nn` sequence identifies the image number and the `alt-text` sequence is the alternate text that will appear in the HTML response. In the web process, the ASP page code examines the abstract text looking for the markers and replaces the marker text with the appropriate HTML code to display the correct image.

➢ To include images into an abstract using the above marker scheme

1. Right click in the abstract area in either the Add An Article or Modify An Article screen to display a menu with the following two options.
   - **Insert Image** that will be enabled if the cursor position is not inside an existing image marker in the abstract.
   - **Change Image** that will be enabled when the cursor is inside an existing image marker.

2. Click either Insert Image or Change Image to display the Select an Image dialog box.

3. **Enter** the appropriate data (more detailed information follows) to either add or change image files.
The Images Available list box displays a list of the images that have been previously been assigned to the currently selected abstract. Click Add New Image to include additional entries.

- The file name and path shown below the Original Image Source File Name label records the location from which the selected image file was originally sourced when it was added to the abstract. The file is read into the database when the abstract is saved.

- The Alt text contains the alternate text for the image. There is a limit of 255 characters.

4. Click Select to write the currently selected image in the images available list box into the abstract using the marker scheme. If the menu selection was Change Image the existing marker in the abstract is removed and the marker for the selected image is inserted into the abstract.

5. Click Close to exit the Select the Image dialog box. Changes made while the dialog box was open remain in effect until the abstract is saved.

➢ To remove an image from an abstract

An image in an abstract can also be removed by deleting the marker text from within the text entered in the Abstract field on the Add an article dialog box.

Modify an Article

All of the article details originally added into the database (See the previous Add an article section) could be modified using this option.

➢ To modify the details in an article

1. Click View Articles on Main Screen JIPAM Admin. System. Select the required article from the list.

2. Click Modify on the View Articles screen to display the Modify Article dialog box that shows the selected article's data.

3. Modify the data fields as required.
Note

Comments regarding the field contents as described in the previous section on adding new articles also apply to modifying article data.

- The Available option box is enabled only if the article has not been published thus allowing an article to be made available in the Web environment before it is published.

- The Published option box is a read-only field that indicates whether this article currently appears in a JIPAM volume/issue. It is a system flag that changes automatically from N (no) to Y (yes) when the article is added to the table of contents of the volume/issue in the process of being publishing.

4. Click Update to save the modified data. At this point, the article record in the database is changed.

5. Click Close when finished.

Add/Delete/Reorder Article Authors

As part of the process to add a new article, it is necessary to assign authors to it. It is also possible that author names may have to be deleted from a specific article or the order of authors may need to be revised.
To add, delete or reorder authors of an article

1. Click **View Articles** on **Main Screen - JIPAM Admin. System**. Select the required article from the list.

2. Click **Authors** in the **View Articles** screen to display the **Change Author of an Article** dialog box.

3. The **Change Author of an Article** dialog box contains two list boxes: (1) authors assigned and (2) authors available.

   - **Authors Assigned** box on the left of the screen shows the authors already assigned to the article. This list box also shows the order of the authors, as they appear when the article is displayed.

   - **Authors Available** box lists all the subscribers who are currently classified as authors and who are not assigned to the current article. This list box is sorted on author surname order.

4. Click the relevant **Hand** buttons to make the changes to the author/s assigned to the selected article as required.

5. Click **Update** to save the changes.

6. Click **Close** to leave the dialog box without saving any of the changes.

**Note**

**Select Other.** It could be possible that the author of the selected article may not appear in the **Authors Available** list. However, it is likely the author is already entered into the database as a subscriber (that is, their currently assigned role is General). In this case, click **Select**
Other and then refer to the instructions in the next section, *Changing a Subscriber Role to an Author Role*.

**Change Instit (Institution).** This button allows the institution of the currently assigned author to be quickly changed (bypassing the View Users screen. Refer to the section titled *Change an Article Author's Institution* later in this chapter.

---

**Change a Subscriber Role to an Author Role**

In assigning an author to a specific article, a situation could arise where the article's author is not yet flagged as an author (i.e. their name will not appear in the *Authors Available* list box. Thus, their role needs to be changed from subscriber to author.

➢ To enter a new author (by changing their role from subscriber to author)

1. Click **View Articles** on *Main Screen - JIPAM Admin. System*. Select the required article from the list.

2. Click **Authors** in the *View Articles* screen to display the *Change Author of an Article* dialog box.

3. Click **Select Other** to display the *Select New Author* dialog box which lists all subscribers not already classified as authors. A filter capability has been provided to streamline the process of locating the relevant subscriber's record in the database. It is not necessary to enter the full surname, just enter part of the subscriber's surname.

4. In the Filter Text box, enter the text you want to filter on. After at least one character has been entered, the **Get** button is enabled.

5. Click **Get** to display the subscriber surnames that match the filter text.

6. Click the required **Name** from the list to enable the **Select** button.

7. Click **Select** to write the selected subscriber into the *Authors Assigned* list box on the *Change Author of an Article* dialog box. At this point, the
subscriber's record in the database is changed (e.g. their role is changed from subscriber to author) and the Add New Author dialog box is automatically closed.

8. Click Close to leave the Add New Author dialog box without saving any changes.

**Change an Article Author's Institution**

After an author's name appears in the Authors Assigned list box associated with a specific article, their current institution details are automatically added to the article details. It is possible that the database may not contain their correct institution details (e.g. they may have changed institutions after they subscribed without updating their subscriber details).

In regard to the author's institution details, the institution assigned to an article/author combination is maintained separately from the author/institution combination. This recognises that authors may change institutions during their academic life but once an article is published, the author/institution combination should not be changed to reflect such moves between different institutions.

To change the institution details of an author assigned to an article

1. Click View Articles on Main Screen - JIPAM Admin. System. Select the required article from the list.
2. Click Authors in the View Articles screen to display the Change Author of an Article dialog box.
3. In the Authors Assigned list box, select the Specific Author to be changed.
4. Click Change Institution button to display the Change Institution dialog box (the selected author’s name will also appear in the title bar). A list of all the institutions currently recorded in the database is displayed with the author's existing institution highlighted.

![Change Institution Dialog Box](image)
5. Select the **new (and more recent and correct) institution** from the list which then enables the **Update** button.

**Note**

If the new institution is not included in the institution list that appears in the **Change Institution** dialog box, click **add Instit.** to open the **Add Institution** dialog box.

For more details on adding a new institution, refer to chapter 6.

6. Select the check box **Update Institution in Users** to enable the selected institution to be recorded as the current institution in the subscribers record within the database when the **Update** button is clicked.

7. Click **Update**. If no problems exist, the screen will close and the **Change Author of an Article** dialog box becomes the active window.

8. Click **Update** on the **Change Author of an Article** dialog box to save the new institution details to the database.

9. Click **Close** at any stage to leave the **Change Institution** dialog box or the **Change Author of an Article** dialog box without making any changes to the database.

**Add/Modify the Keywords of an Article**

Keywords need to be entered into the database as part of the process to add a new article. Existing keywords may also need to be modified or deleted.

➢ **To add or modify an article's keywords**

1. Click **View Articles** on **Main Screen JIPAM Admin. System**. Select the required article that you want to add or modify the associated keywords.

2. Click **Keywords** on the **View Articles** screen.

3. The **Keywords for <title of article selected>** dialog box is displayed with a **Keywords** list box that shows the keywords currently applicable to the selected article. There is also a **New Keyword** edit box that allows keywords to be added or modified in the dialog box.
4. To enter a new keyword, click Enter in the New Keyword edit box. The Add button is enabled after at least one character has been entered.

5. To change an existing keyword, select it in the Keywords list box and then click Change to move it into the New Keyword edit box where it can be modified.

6. Click Add to move the keyword entered (either a new or a changed keyword) in the New Keyword edit box into the Keywords list box. Add also clears the New Keyword edit box.

**Note**

If a mistake is made in moving an entry from the Keywords list box into the New Keyword edit box, clicking Add will move it back into the Keywords list box.

7. Click Delete to remove the keyword selected in the Keywords list box.

8. Click Update when all necessary changes to the Keywords list box have been done. This adds the changes to the database and closes the dialog box. (The Update button will only be enabled if changes have been made to the entries in the Keywords list box.)

9. Click Close to leave the Keywords for <title of article selected> dialog box without saving any changes.

**Add/Modify the Math Code of an Article**

Math codes need to be entered into the database as part of the process to add a new article. Existing math codes may also need to be modified or deleted.
➢ To add or modify an article’s math codes

1. Click View Articles on Main Screen JIPAM Admin. System. Select the required article that you want to add or modify the associated math codes.

2. Click Math Codes on the View Articles screen.

3. The Math Codes for <title of article selected> dialog box is displayed. It contains a Math Codes list box that displays the math codes currently applicable to the selected article. There is also a New math code edit box that allows new math codes to be added or existing maths codes to be modified.

4. To enter a new math code, click in the New math code edit box. The Add button is enabled after at least one character has been entered.

5. To change an existing math code, select it in the Math Codes list box and then click Change to move it into the New math code edit box where it can be modified.

6. Click Add to move the math code entered in the New math code edit box (which could be either a new or a changed math code) into the Math Codes list box. Add also clears the New math code edit box.

Note

If an entry is moved from the Math Codes list box into the New Math Code edit box in error, clicking Add will move it back into the Math Codes list box.

7. Click Delete to remove the math code selected in the Math Codes list box.

8. Click Update when all necessary changes to the Math Codes list box have been done. This adds the changes to the database and closes the dialog.
box. (The **Update** button will only be enabled if changes have been made to the entries in the **Math Codes** list box.)

9. Click **Close** to leave the **Math Codes for <title of article selected>** dialog box without saving any changes.

### Add/Modify the PDF Files for an Article

PDF files of the article full text need to be entered into the database as part of the process to add a new article. Once a PDF file has been written to the database, it can only be updated by deleting the existing entry and the adding in the new PDF file.

In Phase 1, there are only two types of PDF files supported, e.g. PDF screen and PDF printer. However, future enhancements to increase the number of file types are possible by adding additional entries in the PrintFileTypes table in the database.

Since this is not expected to occur very often, a facility has not been provided in Phase 1. However, it could be achieved by a simple insert statement at some future date.

➢ To add or modify an article's PDF files

1. Click **View Articles** on **Main Screen JIPAM Admin. System.** Select the required article that you want to add or modify the associated keywords.

2. Click **Download** on the **View Articles** screen.

3. The **Article Downloads** dialog box shows two list boxes: **Used** and **Still Available.** If any PDF files are currently associated with the selected article, the type is displayed in the **Used** list box (e.g. it could be either the PDF screen or PDF printer version of the full text PDF file or both). The **Still Available** list box shows the types of file that are still able to be associated with the selected article.

4. To add a PDF file to the selected article, select the file type from the **Still Available** list and click **Add.** This opens a file search dialog box.
5. Select the file to be inserted into the database by navigating through the file structure and click Open. The file selection dialog box will close and you will be returned to the Article Downloads dialog box.

6. A progress indicator bar showing the percentage of the transfer process completed appears at the bottom of the Article Downloads dialog box.

7. When the transfer is 100% complete, the progress indicator bar disappears and the type of PDF file type that has just been transferred moves from the Still Available list into the Used list.

8. To delete PDF files from the selected article, select the file type from the Used list and click Delete which moves it into the Still Available list.

9. Click Close to leave the dialog box.
Volumes/Issues

The Admin. Program provides the administrator with the facility to manage the article publication process. New volume/issues can be created and published on the JIPAM web site. Approved articles can be added to the Table of Contents of the new volume/issue. Additional options are also provided to allow the contents of each volume/issue of JIPAM to be modified.

Note

Each JIPAM volume contains all the issues published in one calendar year. Currently, it is planned to publish two issues of JIPAM each year. In this guide, the terminology used to identify a specific issue of the journal is volume/issue.

Volume/Issue Maintenance

To maintain data related to the contents of specific issues of the journal, it is first necessary to view the volumes table in the database.

Click View Volumes/Issues on Main Screen JIPAM Admin. System. The View Volumes/Issues dialog box is displayed with a list box showing all of the volumes currently in the database. Select the relevant issue of the journal from the list.

Maintenance of volume/issue data is structured around the following three actions.
Add (create) a new volume/issue displays a dialog box to enter the attributes of the new volume/issue being created.

Add (modify/reorder) articles is used to add unpublished articles to the currently selected volume/issue. It also allows the publication order (the order that the articles will appear in the table of contents) within the currently selected volume/issue to be changed. Articles can also be moved between the published and unpublished article lists.

Modify a volume/issue changes the attributes of the volume/issue currently selected.

Create a New Volume/Issue
New volumes/issues will need to be created on an ongoing basis.

➢ To create a new volume/issue

1. Click the View Volumes button on Main Screen - JIPAM Admin. System. Select the required volume/issue from the list.

2. Click New on the JIPAM Volumes/Issues screen to display the dialog box in which the new volume/issue details are entered.

3. Fill in the required data fields (detailed explanations of the data fields are provided below).

   - **Volume No.** represents the publication year and is numbered consecutively from Year 2000 (Volume No.1). The field is preset to accept only numeric digits to a length of between one and three characters but the consecutive sequence is not enforced.

   - **Issue No.** are numbered consecutively and identify the issues published in a specific year (a volume). The field is preset to accept only numeric digits to a length of between one and three characters. Again, the consecutive sequence is not enforced.
• **Issue Year** identifies the year the volume represents. The field is preset to accept only numeric digits to a length of four characters and all four digits are required so you must enter the full year. The relationship with the Volume No. is also not enforced by the database.

• **Issue Date** is self-explanatory. The field is preset to accept only valid dates. Once the field is selected for input (has focus), no other field or button on the form can be selected until a valid date has been entered in this field. At this stage, the date of issue does not appear in any reports or on any display screens and is really for information only.

4. Click the **Update** button (which will become available if any of the above fields have been modified) to record the details entered in the database. If the addition of the new volume/issue details is successful, the dialog box will close.

5. Click the **Close** button to close the dialog box without making any changes to the database.

---

**Add/Modify/Reorder Articles in a Volume/Issue**

When approved, unpublished articles are published or added to the table of contents of the next volume/issue. The order of articles in the selected volume/issue’s table of contents can be changed. In addition, there may be certain circumstances where it is necessary to change articles from being published to unpublished in a specific volume/issue.

**Note**

If there is a problem with any of the fields, the update will fail and an error message will be displayed.

---

The assumption is that once an article has moved from being ‘unpublished’ to ‘published’, it should remain ‘published’ (that is, within the table of contents of that volume/issue). However, in the case of an error being detected, an option has been included to allow articles to be added, removed or changed in any volume/issue.
➢ To add, modify, reorder articles in a volume/issue

1. Click View Volumes/Issues on Main Screen - JIPAM Admin. System. Select the required volume/issue from the list.

2. Click Articles in the View Volumes/Issues screen to display a resizable dialog box containing two list boxes: (1) articles published in the volume/issue selected and (2) articles currently unpublished. When first displayed, the Published Articles list box contains the article titles that are currently published to the volume/issue shown in the title bar. The Unpublished Articles list box shows articles that are currently unpublished and which do not yet appear within the table of contents of any volume/issue.

3. To add articles to the current volume/issue, select the title of the article from the Articles Unpublished list and click the UpHand button. This moves the article to the Articles Published list where it is flagged in the database as being published.

Note

There are two hand buttons with different functions on the left side of each list box. Their function is determined by which list they are next to.

❐ Articles Published List. The UpHand button and the DownHand buttons control the order of the articles in this list. Click the UpHand button to move the currently selected article up one place and click the DownHand button to move the currently selected article moved down one place in the Articles Published list box.
Articles Unpublished List. The UpHand button and the DownHand button control the movement of articles from one list box to the other. By clicking the UpHand button, the currently selected article in the Articles Unpublished list is appended to the Articles Published list box (becoming part of the table of contents of the current volume/issue of JIPAM). If an error is made and the article should not have been published, it is possible to move a published article back into the Articles Unpublished list by clicking the DownHand button which flags the article as unpublished in the database. However, care must be taken to ensure that the online version of JIPAM remains consistent with the printed version in terms of the table of contents of each volume/issue. Note that moving articles from one list to another changes the order of the articles in both lists.

4. Click the Update button to save the changes you have made to the database and to close the dialog box. All changes are effected as a database transaction. If the transaction fails for whatever reason, an error message will be displayed and no changes will be made to the database.

5. Click Close to leave the dialog box without saving any of the changes.

Change a Volume/Issue Details

Once a volume/issue has been created and issued, there should be no need to change any of these details. This option would only be used if an error were to be made when the original volume/issue was created.

➢ To change a volume/issue details

1. Click View Volumes on Main Screen - JIPAM Admin. System. Select the required volume/issue from the list.

2. Click Modify on the JIPAM Volumes/Issues screen to display the dialog box in which the new volume/issue details are entered.
3. **Enter** the data required in the input screen (this screen is identical to the **Add new Volume/Issue** screen discussed previously).

4. Click **Update** to modify the database with the amended details. If the volume/issue update is successful, the dialog box will close.

   ![Note]
   
   **Note**
   
   If the volume/issue update is not successful, an error message will be displayed and the dialog box will remain open.

5. Click **Close** to close the dialog box without making any changes to the database.
Institutions

Data is recorded in the database about the current institutions with whom JIPAM's subscribers (including authors and editors) are associated because there are many benefits in knowing the institutional background of the journal's readership base. The following two levels of institutional data are recorded.

- The Affiliation field will generally be completed online when users register as new subscribers. They can then modify the affiliation information online at any stage in the future.
- The Institution field can only be completed by the administrator using the Admin. System program.

This two level data structure have been implemented for the following reasons: (1) to address the lack of security associated with web based information, (2) to ensure a consistent approach to the display of the institution details associated with authors and editors and (3) to provide a way of maintaining current institutional information for authors or editors who move from one institution to another.

Default Settings for the Institution Field

When the JIPAM database is originally created, an institution with None in the Name field is automatically added to the database as the default entry in the institution table. This default institution cannot be deleted or modified in the institution table of the database.

The default (None) is automatically entered into the record of all new subscribers. For those subscribers who are also authors and/or editors, the System Administrator needs to manually change None to the appropriate institution from the drop down list box. Refer to the Modify a Subscriber section in Chapter 3 Subscribers for more information about how to do this.

Institution Data Maintenance

To maintain the institutional data recorded, it is first necessary to view the institutions table in the database.
Click View Institutions on Main Screen JIPAM Admin. System. The View Institutions dialog box is displayed with a list box showing all of the institutions currently in the database and their associated address details in tabular form.

The dialogue box is resizeable to allow more or less detail to be shown in the table. Four navigation buttons are provided to allow the position of the selected subscriber’s record data to be changed in the database table (as detailed above).

Maintenance of data on institutions is structured around the following three actions.

- **Add** a new institution displays a dialog box for the entry of the institution’s details.
- **Modify** allows the details of the institution that is currently selected to be changed.
- **Delete** removes the currently selected institution from the database.

**Add a New Institution**

A new institution can only be added into the database if it is unique. Thus, it is important that the current institutions listed in the database are very thoroughly checked by the administrator to ensure the institution does not appear under a slightly altered name BEFORE a new institution is added.

One of the reasons for a two level approach to institutional data was to ensure that there is consistency in how the way that institutional data is displayed.

➢ To add a new institution

1. Click View Institutions on Main Screen JIPAM Admin. System.
2. Click Add on the View Institutions screen.
3. The **Add Institution** dialog box is displayed.

![Adding an institution dialog box](image)

4. **Enter** the data required. The **Institution Name** is self-explanatory and is the only field that must be completed in this dialog box. Once the cursor is in the field, at least one character must be entered before any other field or button can be selected because the minimum size of this field is one character. The maximum sizes for all fields are shown below.

- Institution Name: 60 characters
- Address 1: 255 characters
- Address 2: 255 characters
- Address 3: 255 characters
- Address 4: 255 characters
- Address 5: 255 characters
- Telephone: 60 characters
- Facsimile: 60 characters

**Note**

All the above fields will accept any printable character and once any of the fields has been modified, the **Update** button becomes enabled. Any leading or trailing spaces in the text entered in any of the fields are discarded before they are written to the database.

5. Click **Update** to write the new institution details to the database. If no problems exist, the screen will close.

**Error Message**

If there is a problem with the **Institution Name** field, a message will be displayed.
6. Click Close to leave the Institutions table without adding any new institutions to the database.

Modify an Institution Details

➢ To modify an institution’s details

1. Click View Institutions on Main Screen JIPAM Admin. System.
2. Select the institution you wish to modify from the View Institutions screen.
3. Click Change on the View Institutions screen. The Modify Institution dialog box is displayed complete with the selected institution’s data.

4. Modify the data as necessary. All of the fields (including the Institution Name) can be changed using this dialog box. Refer to the information provided in the previous section (Add a New Institution) for more detail about these fields.

Note

All the fields will accept any printable character and once any of the fields has been modified, the Update button becomes enabled. Any leading or trailing spaces in the text entered in any of the fields are discarded before they are written to the database.

5. Click Update to write the modified institution details to the database. If no problems exist, the screen will close.
If there is a problem with the Institution Name field, a message will be displayed.

6. Click Close to leave the Institutions table without adding any new institutions to the database.

Delete an Institution

Any institution other than the default institution (that is, the entry with None in the Institution Name field) may be deleted provided the institution is not referenced anywhere else in the database.

➢ To delete an institution

1. Click View Institutions on Main Screen JIPAM Admin. System.
2. Select the institution you wish to delete from the View Institutions screen.
3. Click Delete on the View Institutions screen. There is no request to confirm the deletion.
4. If the institution cannot be deleted, a message box similar to that shown below will be displayed.
Statistics

User traffic to the JIPAM web site is tracked. The first time in a session that a user accesses the site, statistical data is gathered from the HTML request header. The data recorded includes the browser type being used; the primary language set in that browser as well as some basic referral information.

User Tracking

Every time a user accesses the JIPAM web site, a new session object is created by the web server provided they do not already have a current session object. As part of the creation of the session object, a record is written to the Sessions database table. The code for writing the record is contained in the session’s on_start procedure that resides in the global.asa file in the root JIPAM directory.

The record written to the database contains information extracted from the users’ request header and it is raw data. The raw data needs to be summarised into other tables before it can be analysed and reported on. The following two database procedures are provided to support this summarisation process.

SummariseSession.sql which summarises the data

DeleteOldRecords.sql removes the summarised records from the Sessions table as well as removing all records over 26 weeks old from the other statistical tables in the database.

As part of the original database setup procedure (See Chapter 1), an SQL job called Weekly JIPAM Session Update2 is created which then runs these two procedures once a week. However, the job will only run if the SQL Server Agent is active in the database (refer to your database administrator).

Note

If, for whatever reason, the procedure is not run for a week or more, the code within the SQL procedures will automatically take care of summarising the weeks that have been missed. This means that the summarisation process could be run every four weeks instead of every week, if that is preferable.
Visitor Reports

Different visitor reports can be created as an aid to analysing site usage. The nine reports included in Phase 1 of the JIPAM Web System are as listed below.

- Breakdown of Browser Types (chart in IE4 or higher)
- Summary of Browser Types
- Summary of Browser Types including 'via' details
- Breakdown of User Operating Systems (chart in IE4 or higher)
- Breakdown of User Agent Languages (chart in IE4 or higher)
- Summary of User Agent Languages
- Listing of Referrer Click-throughs
- Weekly Summary of Referrer Click-throughs
- Listing of Referrer Target URLs

➢ To Run a Visitor Report

First, it is necessary to login to the JIPAM web site as administrator. The REPORTS menu button appears on the bottom of the menu sidebar. Click this button to display the Visitor Report Menu.

Click the REPORTS button from the JIPAM sidebar menu
Two separate time frames can be chosen for the visitor's reports. The first specifies an individual week while the second reports on a range of weeks with a starting week and an ending week. The 'For this Period' checkbox determines which report is run. If the checkbox is selected, the report covers the range of weeks entered into the two input boxes. However, if the checkbox is not selected, the report generated covers the week specified in the 'From week' input boxes.

To Run a Visitor Report Covering a Range of Weeks
1. Click on the For this period checkbox to enable the From week and To week input boxes. When the pairs of boxes initially appear, they contain the current week and year.
2. In the From week input area, enter the week that starts the period you wish to report on into the first input box and the associated year in the second input box.
3. In the To week input area, enter the week that ends the period you wish to report on into the first input box and the associated year in the second input box. Again, there are no validity checks undertaken on the values entered into these two input boxes.
4. Click the required Report Name.
5. Click Run Query to display the required report on the screen.

To Run a Visitor Report Covering a Specific Week
1. Check the default value of the From week input boxes. If the specific week that you want to report on is different to the week currently shown, select the 'For this period' checkbox to enable the 'From week' input boxes.
2. Enter the starting week in the first input box and the starting year in the second input box of the From week input area. No error checking is attempted for valid values in these fields.
3. Deselect the For this period checkbox.
4. Click the required Report Name.
5. Click Run Query to display the required report on the screen.
Note

It is possible to view the SQL query that is constructed to run the required Visitor’s Report. To do so, click the ‘Show SQL Query’ checkbox. You should note that it is not possible to view the SQL query for all the report types particularly the graphical reports as there are many queries constructed and run to support the generation of the report.
# Glossary

<table>
<thead>
<tr>
<th><strong>Accepting Editor’s Name</strong></th>
<th>The name of the JIPAM editor who accepts an article draft and ensures that the article is ready for publication.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Affiliation</strong></td>
<td>The name of the establishment with which a subscriber is associated. This name assists the administrator when they associate an institution with an author or an editor.</td>
</tr>
<tr>
<td><strong>Annual Date</strong></td>
<td>Registration as a subscriber to JIPAM is for a defined period (12 calendar months from the date of registration). The annual date is the last day of the registration period.</td>
</tr>
<tr>
<td><strong>Article</strong></td>
<td>Journal article submitted in draft form which is reviewed and either (1) rejected or (2) accepted with revisions (known as unpublished articles) or (3) approved for publication in the next volume/issue of JIPAM with no changes (known as published articles).</td>
</tr>
<tr>
<td><strong>Article Abstract</strong></td>
<td>An abstract (a summary of the specific article) is included at the start of every article published in JIPAM.</td>
</tr>
<tr>
<td><strong>Article Author Details</strong></td>
<td>The name and contact details (a hyperlink to email and personal web site addresses) of the author/s is included on the first page of every article published in JIPAM.</td>
</tr>
<tr>
<td><strong>Article Author Institution’s Details</strong></td>
<td>The name and contact details (a hyperlink to email and web site addresses) of the institutions of each author/s is included on the first page of every JIPAM article.</td>
</tr>
<tr>
<td><strong>Article Details</strong></td>
<td>Provides the following details for each article:</td>
</tr>
<tr>
<td></td>
<td>- Article Title</td>
</tr>
<tr>
<td></td>
<td>- Article Author Details</td>
</tr>
<tr>
<td></td>
<td>- Author’s Institutions</td>
</tr>
<tr>
<td></td>
<td>- Date Article Draft Received</td>
</tr>
<tr>
<td></td>
<td>- Data Article Draft Accepted for Publication</td>
</tr>
<tr>
<td></td>
<td>- Accepting Editor’s Name</td>
</tr>
<tr>
<td></td>
<td>- Abstract</td>
</tr>
<tr>
<td></td>
<td>- Keyword/s</td>
</tr>
<tr>
<td></td>
<td>- Classification Code/s</td>
</tr>
<tr>
<td><strong>ARTICLE ID</strong></td>
<td>The unique character sequence used to identify an article with the form '999_99' (999 represents the article number and 99 represents the year the article was received).</td>
</tr>
<tr>
<td><strong>ARTICLE TITLE</strong></td>
<td>The article title is included on the first page of every article published in JIPAM and is assigned to the article by the authors.</td>
</tr>
<tr>
<td><strong>ARTICLES PUBLISHED</strong></td>
<td>Journal articles which have been approved and published.</td>
</tr>
<tr>
<td><strong>AVAILABLE</strong></td>
<td>The flag that determines whether details of an article in the JIPAM database is accessible from the web.</td>
</tr>
<tr>
<td><strong>CLASSIFICATION CODE</strong></td>
<td>Individual article authors provide a list of mathematical classification codes to help users/subscribers search JIPAM volumes for specific articles.</td>
</tr>
<tr>
<td><strong>CURRENT SESSION</strong></td>
<td>A session becomes current for a subscriber from the moment they access their first JIPAM web page. The session ends either when their web browser is closed down or 20 minutes has expired since they accessed a web page from JIPAM.</td>
</tr>
<tr>
<td><strong>DATE ARTICLE DRAFT RECEIVED</strong></td>
<td>The date that an article is received in draft form from the author.</td>
</tr>
<tr>
<td><strong>DATE ARTICLE DRAFT ACCEPTED FOR PUBLICATION</strong></td>
<td>The date that a JIPAM editor determines that an article draft is ready for publication.</td>
</tr>
<tr>
<td><strong>DISPLAY NAME</strong></td>
<td>The name an author or an editor would like associated with their articles when those articles are displayed on the web.</td>
</tr>
<tr>
<td><strong>GENERAL USER</strong></td>
<td>Any member of the WWW community whose browser is directed towards the JIPAM web site (URL xxxx). Users have access to the Table of Contents for all currently published volumes of the JIPAM complete with article abstracts. However, users are unable to access the full text of articles until they register as subscribers.</td>
</tr>
<tr>
<td><strong>INSTITUTION</strong></td>
<td>The formal name of the establishment an author or editor is associated with when submitting or editing an article for publication. Only the administrator can create institutions (See affiliation).</td>
</tr>
<tr>
<td><strong>JOURNALS UNPUBLISHED</strong></td>
<td>Journal articles that have been accepted but which are not yet approved for publication. Unpublished articles can be viewed from the NEW PAPERS button on the web site.</td>
</tr>
<tr>
<td><strong>KEYWORD</strong></td>
<td>Individual article authors provide a list of keywords for users/subscribers to search JIPAM for specific articles.</td>
</tr>
<tr>
<td><strong>LOG IN</strong></td>
<td>To gain access to the full text of articles published in JIPAM, it is necessary to be registered as a subscriber and to be logged in.</td>
</tr>
<tr>
<td><strong>PASSWORD</strong></td>
<td>A subscriber's secret word for advising the system that they are indeed the subscriber who is logging in.</td>
</tr>
<tr>
<td><strong>PUBLISHED</strong></td>
<td>The internal JIPAM database flag that indicates that an article has been associated with a JIPAM volume/issue.</td>
</tr>
<tr>
<td><strong>RENEWAL DATE</strong></td>
<td>The date on which a subscriber's annual subscription expires. In phase 1, this date is automatically extended for the subscription period.</td>
</tr>
<tr>
<td><strong>ROLE</strong></td>
<td>The actions a subscriber can perform in the JIPAM system. All subscribers are assigned the General tag when they register online. The Author role is assigned to a subscriber who has submitted an article. The Editor role is assigned to a subscriber who is available to edit articles, while the Both tag is assigned to subscribers who are both an author and an editor.</td>
</tr>
<tr>
<td><strong>START DATE</strong></td>
<td>The date a user first registered as a subscriber with the JIPAM system.</td>
</tr>
<tr>
<td><strong>SUBSCRIBER</strong></td>
<td>A user of JIPAM who has completed the subscription form and had their details stored in the JIPAM database. A subscriber is allowed access to the full text of all articles published in JIPAM. This access allows download of each article file in PDF format.</td>
</tr>
<tr>
<td><strong>SUBSCRIPTION</strong></td>
<td>Online process to register users so that they can access and download the full text of all JIPAM articles. Currently, subscription and thus access to full text articles is free but charges may be levered at some stage in the future.</td>
</tr>
<tr>
<td><strong>SUBSCRIPTION PERIOD</strong></td>
<td>Registration as a subscriber of JIPAM is for a fixed period of 12 calendar months from the initial subscription date.</td>
</tr>
<tr>
<td><strong>WEB ADDRESS</strong></td>
<td>The URL address of a subscriber's home page.</td>
</tr>
<tr>
<td><strong>VOLUME</strong></td>
<td>A JIPAM volume contains all the issues published in one calendar year. Currently, it is planned to publish two issues of JIPAM each year.</td>
</tr>
</tbody>
</table>