An Online Project Allocation System for the University of Portsmouth

by

Javi Ruiz
(Supervised by Dr. Jim Briggs)

2006

ABSTRACT

A new computing system called SUMS (Student and Unit Management System) is being implemented in the faculty of Technology. PUMS is the Project Unit Management System which manages the information related to student's project. Project ideas submission, supervisors, moderators, projects allocations as well as tracking student progress are some tasks carried out by this system. A new technology is being used in order to replace the old system by the new one. SUMS has five distinct sub-systems that need to be fully compatible between them. One of these is the Project Allocation Sub-system which manage the allocation of student projects.

The main aim of this project is to develop a computing system fully compatible with SUMS, which allows students to choose projects ideas for their dissertation topic. Also, the application will help staff members to allocate these projects to students according to the student's options and staff's criteria. A wide range of reports will allow staff members to have some useful information about the project allocation process

ACKNOWLEDGEMENTS

To my supervisor and my client Dr. Jim Briggs for his support, patience guidance, and advice in all the steps of my project development, and without whom I would have not been able to carry out this project.

I would like to thanks my family for his support during the time I spent in England doing this project, my friend Josh for his amazing support in the difficult moments, and all my classmates.

Table of Contents

1. Introduction.....6

1.1 The need of a new system.....6
1.2 SUMS components.....6
1.3 Project Allocation Process.....7
1.4 Project Aims & Objectives.....8
1.5 Organization of this Report.....8

2. Java EE 5 Platform.....10

2.1 Introduction.....10
2.2 Java EE Main Components.....10
2.2.1 Servlets.....10
2.2.2 Java Server Pages.....12
2.2.3 Java Beans.....14
2.3 The Model-View-Controller (MVC) Architecture.....14
2.3.1 Implementing MVC in Java Web Applications.....15
2.4 Web-based Applications Frameworks.....17
2.5 Apache Struts Framework.....18
2.5.1 Implementing MVC in Struts.....18
2.5.2 ActionServlet.....19
2.5.3 ActionMapping.....20
2.5.4 ActionForward.....21
2.5.5 ActionForm.....21
2.5.6 ActionClass.....21
2.5.7 Struts Control Flow.....22
2.6 Object Persistent Frameworks.....23
2.6.1 Hibernate.....24
2.6.2 Hibernate Architecture.....24
2.6.3 The persistence Lifecycle.....25
2.6.4 HQL(Hibernate query Language).....26
2.7 Summary.....26

3. Analysis.....27

3.1 Introduction.....27
3.2 Project Planning.....27
3.3 System Requirements.....28
3.3.1 System Actors.....28
3.3.2 Use Cases Diagrams.....29
3.3.3 Functional Requirements.....30
3.3.4 Requirements Prioritisation.....32
3.4 Review of SUMS.....32
3.5 Summary.....33

4. Design.....34

4.1 Introduction.....34
4.2 Presentation Layer Design.....34
4.2.1 Web Architecture Map.....34
4.2.2 Web Wireframes.....36

4.3 Persistence Layer Design....37
4.3.1 Data Objects....37
4.3.2 Database Tables....39
4.4 Business Layer Design....43
4.4.1 Front Controller Pattern....43
4.4.2 Business Delegate Pattern....44
4.6 Summary....45

5. Implementation....46

5.1 Introduction....46
5.2 Development Tools....46
5.2.1 Hardware....46
5.2.2 Software....46
5.2.2.1 MySQL server 4.1....47
5.2.2.2 MySQL Command Line Client....47
5.2.2.3 NetBeans 5.0....47
5.2.2.4 Macromedia Dreamweaver MX....47
5.2.2.5 Adobe Photoshop CS....48
5.2.3 Problems found on development tools....48
5.3 Presentation Layer implementation....48
5.4 Data Layer implementation....50
5.5 Business Layer implementation....52
5.6 Implementing OPAS functions....55
5.6.1 Creating Project Ideas....55
5.6.2 Project Ideas Choices....57
5.7 Summary....59

6. Testing & Evaluation....60

6.1 Introduction....60
6.2 Debugging OPAS....60
6.3 Functionality Testing....60
6.3.1 Project Ideas....60
6.3.2 External Contacts....62
6.3.3 Students....63
6.3.4 Supervisors....64
6.3.5 Unit Coordinators....66
6.4 Evaluation....66
6.4.1 Project Ideas....66
6.4.2 External Contacts....66
6.4.3 Students....66
6.4.4 Supervisors....67
6.5 Summary....67

7. Conclusion....68

7.1 Project Planning....68
7.2 Designing and Implementing OPAS....68
7.3 Learning Achievements....69
7.4 Project Results....69

References....71

Appendix A- Project Specification.....73
Appendix B- User Requirements.....79
Appendix C- Gantt Chart.....91
Appendix D- Wireframes.....93
Appendix E- OPAS Presentation.....100

Chapter 1 – Introduction

This chapter will discuss:

- The problem background to be solved.
- The aims and objectives of this project.
- The general overview of this report.

1.1 The need of a new system

All the information related to project students in the School of Computing of the university of Portsmouth is carried out by an application called PUMS (Project Unit Management System). This application manages the project ideas submission, student registration as well as tracking student projects and progress. All the post-graduates and final year undergraduate students have to undertake a project in order to complete the course. Students must register in PUMS at the beginning of the academic year. Then, they select a project idea and project coordinators can then allocate these projects.

Dr Jim Briggs, who is a Principal Lecturer in the School of Computing, developed PUMS using CGI (Common Gategay Interface) technology and PERL scripting language. Besides, it also uses Ascii files in order to store the information generated by the application. No DBMS (Database Management System) is being used for manage this information. The old technology needs to be updated by an alternative technology. As it is commonly known, in a CGI technology, a new CGI process is started for each HTTP request. It means that the code of the program is loaded each time a new HTTP request is made. Therefore, when a lot of user connections are made, a lot of server resources are wasted, affecting the application performance. The disadvantages of CGI technology as well as the no use of a DBMS makes difficult the use of the application in this environment.

Briggs decided to use J2EE/Java architecture for developing the new web-based application. Hence, SUMS (Student Unit Management System) was born. In addition, the use of a web application framework makes this application more robust, more scalable, more flexible and easier to be maintained. All the benefits of this technology will be discussed in the following chapter.

1.2 SUMS Components

There are five distinct sub-systems which are part of SUMS. Only the Project Marking sub-system is in use at this moment. The others are in other stages of the implementation. The following figure represents the sub-systems related to the SUMS.

Figure 1.1 - Sums Components.

Each sub-system must be developed according to SUMS requirements as they share a common database. This is a very important requirement as the whole system must be developed using the technology required. Furthermore, splitting the system into five different sub-systems improves the system maintenance as every change made in a sub-system does not affect the performance of the others. Some of these sub-systems are still being developed by other postgraduate students as a final projects.

1.3 Project Allocation Process

The SUMS sub-system that this project is concerned to is OPAS (On-line Project Allocation Sub-system), which controls the project allocation process. This report describes the design and implementation of it. The different stages of this process and users who interact with it are illustrated in the following figure.

Figure 1.2 - OPAS process.

The first stage of this process is the Project Ideas Submission (1). Supervisors, Students, Project Coordinators as well as external contacts can submit their project ideas. Once the user has logged into the system, a set of options are available in order to insert, edit, delete and amend the ideas. Every project idea need to be approved by a staff member as students can only select approved ideas (2). If any project idea is not approved, it will be withdrawn from the system and students will not be able to select it.

Students are allowed to choose a maximum number of project ideas. This number depends on the student's course. The list of submitted project choices can be arranged in order to indicate the ideas preference. The selected idea that is on the top of the list has more preference than the others and usually it is the most wanted project choice.

In the other hand, supervisors submit their preferred students (3). The list of preferred students defines the supervisor's most desired students. Every student can be associated to one of the project ideas submitted by the supervisor. Therefore, unit coordinators use this information in order to allocate projects to students.

The final stage in the project allocation process is the decision of which project is allocated to each student (4). This task is carried out by the unit coordinator. The information provided by the students (project choices) as well as the information provided by supervisors (preferred students), will help unit coordinators to allocate the projects to the students. Then, every student receive a notification (usually by email) about the project that has been assigned.

1.4 Project Aims & Objectives

The main aim of this project is to develop an On-line Project Allocation System (OPAS). This system must be fully compatible with SUMS as it needs to be integrated properly with the whole system (Project Specification - Appendix A). Also, OPAS needs to contain more functionality than the system being used at this moment. Therefore, looking for a way to improve the system is another aim of this project. The user requirements have to be sort it out and documented (User Requirements - Appendix B).

This project should accomplish a set of objectives in order to be completed successfully. The objectives defined can be summarised as follows:

- Gathering client requirements. This information should be retrieved by means of client interviews and a review of the old system. Project specification and user requirements need to be documented.
- Familiarising to SUMS architecture.
- Researching the technologies that are being used for developing SUMS.
- Designing the application and producing the required documents such as UML diagrams, story boards, data dictionary, etc.
- Implementing and testing the application. A testing plan should be defined in order to carry out the application testing in a more effective way. According to the testing results, the required system changes must be carried out.
- Producing technical documents such as Javadocs and user guide if appropriate.
- Delivering the application to the client.

1.5 Organisation of this Report

This report is organised in eight main chapters. The following list summarises the content of every chapter of this report.

- Chapter 2 gives more details about the technologies that are being used in SUMS development.

- Chapter 3 talks in more detail about the problem to be solved and the system description.
- Chapter 4 describes the design phase of this project. Also, the additional database tables that need to be implemented as well as the user interfaces are included.
- Chapter 5 describes the implementation phase of this project. Also, it includes the main problems encountered while implementing the prototype and how they were intended to be solved.
- Chapter 6 describes the planing and execution of the application testing.
- Chapter 7 includes the final conclusion about this project. Besides, the author of this report gives a personal reflection on about how this project was carried out.

CHAPTER 2 – Java EE 5 PLATFORM

This chapter will discuss:

- The main features and components of version 5 of the Java Platform, Enterprise Edition (Java EE) as well as main advantages of using this platform for building web-based applications.
- The benefits of using application frameworks such as Apache Struts.
- The benefits of using Object/Relational mapping framework such as Hibernate.

2.1 Introduction

Java EE (formerly referred as J2EE) is a platform that is used for developing large Web-based applications. With the new version 5, applications can be easier to be developed as many new features has been included. The main aim of this platform is to add convenience, improve performance, and reduce development time [Stearns & Chinnici (2006)]. Another positive aspect is that this platform is greatly supported by the open source community. Therefore, a lot of libraries are available, giving more functionality to the applications. This platform makes use of a wide range of technologies such as Java Servlets, JSP(Java Server Pages), and Java Beans, which help to develop web-based applications in a more effective way. Furthermore, the use of XML increases the capabilities of these applications as it provides an standard language for configuring and exchanging data between them.

People have built on top of Java. There are a lot of libraries that facilitate the reuse of code. Also, there are libraries that implement application frameworks. Some examples of this are Apache Struts and Hibernate which are described with more details in the following sections.

2.2 Java EE Main Components

2.2.1 Servlets

Servlets are programs written in Java that run in the server-side of the application. Their main task is to process the data received from a user or another program and to return the required information received directly from a database. This is the most common way to retrieved data although it can also be obtained from several sources such as Web Services, RMI (Remote Method invocation), another servlet or directly generated within the servlet. There are several protocols that can be used for communicating with the Web browser and the application running on the server. The most widely used protocol for sending information over the Internet is HTTP (Hypertext Transfer Protocol). HTTP servlets handles the incoming HTTP requests made by clients. Figure 2.1 represents the servlet's role in a Web-page application environment.

Figure 2.1 - The servlet's role. Adapted from Hall & Brown(2004).

These stages can be summarised as follows:

1. The Web user makes an HTTP request that is received by the HTTP server (Web Server)
2. The HTTP Server redirects the data to the Servlet Container, also referred to as the servlet engine.
3. The servlet engine decides what servlet to run according to the deployment descriptor file (web.xml). It checks if the requested path matches the action URI (Uniform Resource Identifier) defined in this file.
    - 3.1. If the servlet has not been created yet, the servlet's *init* method is invoked.Then, the servlet's *service* method is invoked and passes an instance of ServletRequest and ServletResponse object.
    - 3.2. Depending on the request method for sending the data (Get or Post), the doPost or doGet method is called.
    - 3.3. Then, the appropriate methods are called and the data is retrieved from the required source.
    - 3.4. If the view is generated in the servlet, the data is displayed usually using print statements. Otherwise, the application control is transferred to another servlet or JSP page if appropriate.

4. Servlet engine sends the HTTP response to the HTTP server.
5. HTTP server sends the output to the client's Web browser and the results are displayed.

As well as the data sent by the user, normally from a Web form, servlets must read the implicit HTTP request data (HTTP request headers) indirectly sent by the browser. This kind of data contains information about cookies, host information, type of data returned, etc. Furthermore, as well as the explicit data sent by the servlets as a result of the user action, clients must receive HTTP response data (HTTP response headers). This implicit data determines the type of document being sent, if a password is required and another useful information for displaying the content of HTML page. An example of a basic servlet structure is illustrated in the following figure.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class servletName extends HttpServlet
{
   public void doGet(HttpServletRequest request, HttpServletResponse  response)
       throws ServletException, IOException
   {
    response.setContentType("text/html");

    PrintWriter out=response.getWriter();

     out.print("<html>");
       .
       .
       .
     out.print("</html>");

    out.close();
  }
}
```

Imported Classes

Incoming data

Outgoing information

Type of content Output

Creating a new text-output Stream

Print statements with the HTML tags and content of the Web page

Close the stream

Figure 2.2 – Basic Servlet Structure.

2.2.2 Java Server Pages

A Java Server Page (JSP) is like a HTML file. The difference is that a JSP page may contain Java code embedded (scriptlet). Figure 2.3 illustrates a scriptlet that in used for retrieving data from a scoped attribute in a JSP page. Almost everything that can be done in a servlet may be done using JSP pages. However, servlets are not good for tasks related to presentation. They are widely used for tasks oriented towards processing as it is difficult to design and maintain html files generated in servlets using Java code. As Bakharia (2002) explains JSP pages have several advantages. An object-oriented programming language together with a wide range of APIs can be used, giving a lot of functionality to them. Furthermore, it simplifies the web development as the presentation is separated from the business logic. Broadly speaking, separating the business from the presentation logic provides a better application maintenance, allows the use of web development tools such as Macromedia Dreamweaver and makes possible to split up the application work into different development teams.

```
<html>
 <body>
  <table>
  <%

  ArrayList person=new ArrayList();
  person=(ArrayList)session.getAttribute("bean");

    for (Iterator it=person.iterator();it.hasNext();)
    {

      personBean person=(personBean)it.next();

      String name       = person.getName();
      String surname = person.getSurname();
      String address   = person.getAddress();
      String city          = person.getCity();
      String phone     = person.getPhone();

      out.print("<tr>\n");
      out.print("<td height='75' valign='middle'>" + name + "</td>\n");
      out.print("<td height='75' valign='middle'>" + surname + "</td>\n");
      out.print("<td height='75' valign='middle'>" + address + "</td>\n");
      out.print("<td height='75' valign='middle'>" + city + "</td>\n");
      out.print("<td height='75' valign='middle'>" + phone + "</td>\n");
      out.print("</tr>\n");

    }

  %>
  </table>
 </body>
</html>
```

Session scope bean is retrieved

Iterating over the data structure

Data is stored in variables

Data is displayed using print statements

Figure 2.3 – Iterating over a data structure using scriptlets.

However, It is possible to build JSP pages without using Java Code and having the same functionality (Figure 2.4). JSTL (JavaServer Pages Standard Tag Library) eradicates the use of scriptlets making easier to access the request and response objects as well as the scoped attributes [Geary, 2003].

Action that iterates over the data structure

Session scope bean

Variable name used in the iteration

```
<table width="800" border="0" align="center" cellpadding="0" cellspacing="0">
  <c:forEach items='${sessionScope.bean}' var='person'>
      <tr>
       <td height="75" valign="middle"><c:out value='${person.name}'/></td>
       <td height="75" valign="middle"><c:out value='${person.surname}'/></td>
       <td height="150" valign="middle"><c:out value='${person.address}'/></td>
       <td height="75" valign="middle"><c:out value='${person.city}'/></td>
       <td height="50" valign="middle">><c:out value='${person.phone}'/></td>
      </tr>
  </c:forEach>
</table>
```

Action for displaying the information

Figure 2.4 – Iterating over a data structure using JSTL.

Furthermore, a set of standard tags are available in JSP pages. These tags have the same functionality as using scriptlets. It is also important to say that when a JSP page is requested, the JSP engine translates it into a servlet. After that, the servlet is compiled and only a new servlet is generated again when the JSP is updated. It conforms the servlet life cycle.

2.2.3Java Beans

A Java Beans is a Java classe that enables the data encapsulation in the application. A Java Bean must have a constructor without arguments. Furthermore, no public attributes must be declared and values in the class must be accessed only using accessor (getXXX) and mutator (setXXX) methods (Figure 2.5). Therefore, values declared in a class are only retrieved and changed by these methods so that these values are protected from external access.

Figure 2.5 – Basic Java bean structure.

According to Herbert (2002), from a developer's perspective, Java Beans provides several advantages. The use of auxiliary software can help to configure a Bean. Properties and methods declared in the Bean can be easily controlled by the application builder. Hence, Beans can be populated with the required information and stored in a scoped variable in order to be shared in other servlets or JSP pages. After that, that information can be displayed in a JSP page using the expression language defined in the JSTL. An expression languages makes easier to access data stored in a bean. Figure 2.4 illustrates the use of an expression language to retrieve data from a bean.

2.3 The Model-View-Controller (MVC) Architecture

The MVC is a software architecture that separates an application into three distinct components as illustrated in the following figure.

Figure 2.6 - MVC architecture components [Husted et al (2003)].

The information used by the application is represented by the model. The view renders the model and represents this information into a suitable way so that user can interact with it. Finally, the controller responds to the user actions, process the information and select the appropriate view. The main advantage of implementing this architecture is that changes made in one component does not affect the others. As a result of this, the application development and maintenance are carried out in a more effective way. As Walnes et al. (2004) explains, the use of MVC has several benefits. The team work can be divided up and every developers group can focus their work in a specific task. The business logic is easier to understand and the problems can be solved faster. Developers can choose the best option to design, develop, and testing the user interfaces without taking into account the business rules. Hence, a web programmer without knowledge about Java programming can develop the user interfaces effectively.

2.3.1 Implementing MVC in Java Web Applications

MVC architecture works well in web-based applications. In Java EE platform, the model is typically represented by a collection of Java Beans. The view is represented by a set of JSP pages and the controller is represented by a set of servlets. The required steps for implementing a MVC architecture in this platform are illustrated in the following figure.

Figure 2.7 – Implementing MVC [adapted from Hall & Brown (2004)].

Firstly, the Beans which represents the results to be displayed are defined (1). A user, for instance, wants to make a web searching for displaying a list of results. When an HTTP request is made, the request parameters are retrieved by the servlet which process this information and retrieve the required data from the database or any other source (2). After the data is retrieved, the Beans are populated (data is stored in the bean)(3). The state can be stored in a request, session or servlet-context object(4).

- Request object: the Bean is only accessible in the destination.
- Session object: the Bean is accessible in all servlets and JSP pages for the current user.
- Servlet-Context object: the Bean is accessible to any servlet or JSP page of the application for any user.

Then, the appropriate JSP page is selected and the control is transferred to this page using the redirect or forwarding method (5). When the JSP receives the Bean, extracts the data from the Bean using JSP tags, JSTL or scriptlets. Figure 2.8 gives a general vision about this process and the elements involved.

Figure 2.8 – MVC Model 2 [Vivek et al. 2005].

There are two ways to implement a MVC architecture. Figure 2.8 represents the model 2 where a servlet is used as a controller for processing the information. At the beginning of web development it was a tendency to embed the HTML pages within the application logic using CGI (Common Gateway Interface) or servlets. However, using PHP, JSP, and other scripting languages, developers embedded business logic within the display [Vivek et al. 2005]. This is referred to the Model 1 architecture illustrated in Figure 2.9.

Figure 2.9 – MVC Model 1 [Vivek et al. 2005].

In this model, the information is processed within the JSP pages. Therefore, the controller as well as the view is represented by JSP pages. Although Model 1 is less complex than Model 2, it has several disadvantages as the readability of the code if affected and the business methods are more difficult to be understood. Furthermore, the application maintenance and testing can be much more difficult to be carried out. Hence, this model is not suitable for large web-based applications.

2.4 Web-based Application Frameworks

A web framework is a software tool that is used for developing web applications. As Web-based applications become more complex, frameworks play an important role in order to help developers to design, develop, and integrate the applications in a more effective way. In consequence, the application maintenance is easier to carry out. Code libraries as well as programming language are also provided, improving the functionality of these applications. Furthermore, making easier the software development, developers can focus their work in meeting the user requirements. Hence, goals can be achieved on time and the quality of the application can be greatly improved.

According to Husted et al. (2003), web application frameworks make use of common components that makes easier the design and development of the application (Figure 2.10). The configuration file provides implementation details for configuring the framework. The central controller (also referred as a Front Controller) provides a centralised place to manage HTTP requests. The presentation system controls the user-interfaces.

Figure 2.10 – Frameworks Main Components [Husted et al. (2003)].

Nevertheless, as Whali et al. (2003) explains, frameworks introduce a set of limitations. They are not flexible. Frameworks architecture can not be easily changed by the provider as they can broke other existing codes. The compatibility is one of the most important points to take into account when choosing a framework. Furthermore, the learning process to manage the framework can be long. As a result, their adoption in a team can affect the deadlines for releasing the final applications.

There are a wide range of Web frameworks available. Depending on the applications requirements and the technology being used some frameworks are more appropriate than others. The most widely used in Java are the open-source frameworks. They provide high functionality with little cost as they have the support of the open-source community. This has a lot of benefits as useful documentation can be found on the Internet, helping developers to understand and use them properly.

2.5 Apache Struts Framework

Struts is one of the most used open source framework for building Java Web applications based on the MVC(Model 2) architecture. It provides three key components[Apache Struts (2006)].

- Front Controller: Dispatches requests to an action handler.
- Location Handler: Transfers the control to the appropriate resource.
- Tag-library: Very useful for creating form-based applications.

Struts makes use of servlets, JSPs, and Java classes in order to implement the Model-View-Controller architecture. In addition, it provides a wide range of utility classes that help to improve the application functionality.

2.5.1 Implementing MVC in Struts

Struts applications contain several objects that help to implement MVC architecture(Figure 2.11).

Figure 2.11 – MVC implementation in Struts [Gulzar (2003)].

These objects are the following [Husted et al.(2003)].

- ActionForward class: View selection.
- ActionForm class: It represents the data for the state change.
- ActionMapping class: The state change event.
- ActionServlet class: It is a part of the controller object that receives the user actions and state changes. It also controls the selection view.
- Action class: It is a part of the controller object that executes the state change or query interacting with the model and send to the actionServlet the next view to be selected.

In order to configure the application, Struts provides some configuration files for helping developers to build the application.

- struts-config.xml file is an XML file that contains the configuration of all the Struts components.
- applicationResources.properties file contains all the messages and labels of the application. Hence, this information is much easier to be managed allowing the application to be internationalized.

Also, in order to manage the information contained in the Java Beans, Struts provide a JSP tag extensions. These are very useful for managing the information retrieved from the Java Beans in the presentation layer. Having taken a look at the general aspects of the Struts framework, the following sections give a more detailed view about the main Struts components

2.5.2 ActionServlet

The ActionServlet provides a mechanism for handling all client requests. Its main objective is to coordinate the activities carried out in the application. It works behind the scenes, building the other Struts components together [Husted et al. (2003)]. There is only one ActionServlet for each Struts application and normally each application uses the same ActionServlet. This object is configured in the deployment descriptor file (Figure 2.12) that maps all the URI with the .do extension to this servlet. All the incoming requests are mapped to this object using this configuration. After that, the requests are delegated to the standard RequestProcessor which provides the appropriate behaviour for each method.

```
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
   <param-name>config</param-name>
   <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
</servlet>
<servlet-mapping>
   <servlet-name>action</servlet-name>
   <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

Short name associated to the ActionServlet class

ActionServlet receives any requests that matches the pattern *.do

Figure 2.12 – ActionServlet configuration (Web.xml).

2.5.3 ActionMapping

Every action or business operation needs to be defined in the Struts configuration file (Figure 2.13). A specific URI (Uniform Resource Identifier) is associated with each Action in the application. The ActionServlet checks if the requested path matches any Action URI in the Struts configuration file. After it is found, the ActionMapping gives information about the behaviour of that action and the elements such as ActionForm, ActionClass that interacts with it. Each Action has its own behaviour and it is configured for an specific task.

```
<action-mappings>
  <action
    input="/index.jsp"
    name="logonForm"
    path="/submitLogonForm"
    scope="request"
  type="general.actions.SubmitLogonForm"
    validate="true">
  <forward
    name="success"
    path="/Index.do"
    redirect="true"/>
  </action>
   .
   .
   .

</action-mappings>
```

Form to which control is returned when a validate error is found
Logical name for the form bean
The URI path to select this mapping
Scope type where the form bean is stored
If the validate method in the form bean is called
Logical name given to this ActionForward
URI for this ActionForward
Control is redirected instead of forwarding

Figure 2.13 – Action Mapping configuration (struts-config.xml).

2.5.4 ActionForward

When an Action is carried out, the application control needs to be sent to another place. The Struts application must know where to go after an action is finished. Furthermore, any place in the application must be defined by an ActionForward object. This object is defined in the Struts configuration file (Figure 2.13) and provides the name of the path (JSP page, another Action, etc) that matches the logical name given to that ActionForward.

2.5.5 ActionForm

This object is also known as a form Bean. This object is simply a Java Bean with the getter and setter methods. When a HTML form is submitted and the ActionServlet finds the Action URI in the Struts configuration file that matches with the requested path, it checks if there is any formBean associated to that Action. An ActionForm represents the HTML form data that has been submitted. This object is automatically populated with the data from the HTML form using the setter methods.

The value of the *validate* property in the ActionMapping is also checked. If it is set to true, the ActionSevlet through the request processor calls the validate method in the ActionForm. If the data is not valid, the control is sent to the path defined in the *input* property of the ActionMapping. The ActionForm provides a easy way to handle the data from a HTML form, to validate this data and to redisplay the form with the submitted data if appropriate.

2.5.6 Action Class

After the ActionServlet finds the Action URI that matches the requested path, validates the form data and everything that needs to be checked, it reads the *type* property in the ActionMapping. This value contains the name of the Action object that needs to be used for the requested action. According to Husted et al. (2003), the responsibilities of the Action objects are the following. Validating preconditions;Calling business methods if appropriate;Detecting any processing error;Routing control to the appropriate view.

Although all the code related to business logic can be implemented in the Action objects, this way to establish the rules for processing the data is not the most appropriate. Helper methods declared in other classes are very useful as they provide a better way to understand and design the business layer. An example of an Action class basic structure is illustrated in the following figure.

```
import org.apache.struts.action.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class ActionClassName extends Action
{
   public ActionForward execute(ActionMapping mapping, ActionForm form,
     HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException
   {
      FormBeanName formBean=(FormBeanName) form;
      .
      .
      .
      .
      return mapping.findForward("forwardName");

   }
}
```

Imported Classes

ActionMapping used to invoke this action

The ActionForm specified by the mapping

Casting for accessing to the ActionForm properties

Business Logic

A Forward is selected

Figure 2.14 – Basic ActionClass structure.

2.5.7 Struts Control Flow

The sequence of events when a HTTP request is made in the Struts framework is illustrated in the following figure.

Figure 2.15 – Struts Control Flow [Adapted from Polgar et al. (2005)].

These events and how the Struts components interact in the system can be summarised as follows:

1. A HTTP request is made by the client.
2. The request is sent to the ActionServlet.
3. ActionServlet checks if the requested path matches the Action URI (Uniform Resource Identifier) pattern defined in the Struts configuration file.

4. If the mapping for that path specifies a form Bean (ActionForm), ActionServlet checks if it is exists and the ActionForm object is populated.
5. If the attribute validate is set to true in the mapping, the validate method in the ActionForm is called (This method is used in order to validate the data sent by the user).
6. If the validation is not correct, the control is returned to the path specified in the input property in the mapping.
7. If the validation is correct, the execute method in the Action class is called.
8. The required information is retrieved and the bean is populated.
9. The bean is stored in the request,session or application scope.
10. Action returns an ActionForward to the ActionServlet.
11. This actionForward can be a JSP page or another action.
12. The JSP extracts the data from the Bean and the information is displayed using JSTL, JSP standard tags or scriptlets.
13. A HttpResponse is sent to the client

2.6 Object Persistent Frameworks

Persistent data is vital for the application development as the information generated by users must be preserved and also be organized in a database. A persistence layer that is responsible for data storage as well as the data retrieval is a basis in a layered architecture(Figure 2.16).

Figure 2.16 – Persistence Layer [Bauer & King (2005)].

RDBMS(Relational Database Management System) is the most common type of database in use at this moment and everybody is familiarised with them. As applications becomes more complex, it is necessary to adopt object-oriented techniques in order to develop these applications in a more efficient way. Developing software quickly and correctly is a main goal at a time for any organisation. Object-Oriented programming languages give developers a powerful tool for reusing software components. Therefore, software-development groups can be more productive using these techniques rather than using traditional structured programming languages and applications can be easier to understand and modify. This and the support of the open-source community, makes Java one of the most widely used object-oriented programming language at this moment [Deitel & Deitel (2005)].

Objects need to be stored in the database and later retrieved preserving their properties and relationships. These objects need to be persistent. The perfect solution is to use an OODBMS(Object-Oriented Database Management System) as the information is represented in the form of objects. Although, object databases has a growth period in 2004 due to the release of open-source object databases, these are still uncommon and difficult to manage [Wikipedia (2006)]. Despite of RDBMS are unable to represent these objects in the proper way, they are used instead.

Object Persistent frameworks provides a solution for the use of RDBMS and objects. They contain libraries which provide a ORM (object/relational mapping) mechanism that transforms data from objects in an Object-Oriented application to the tables in a relational database and vice versa. According to Jasnowski (n.d), the benefits of using a Object Persistence Framework are the following:

- Caching of retrieved objects.
- Separation of SQL from application logic.
- Separation of the data layer from application.
- Simplifies access to complex databases structures.

In the other hand, like any application framework, the learning curve can be long affecting negatively in the application development. Having taken a look into the general aspects of the persistence frameworks, the next section describes in a more detail a persistence framework such as Hibernate.

2.6.1 Hibernate

Hibernate is at this moment the most widely used open-source persistence framework for building Java web applications. It provides a complete technical documentation as well as a wide range of libraries and utilities such as HQL (Hibernate Query Language) for object/relational mapping.

2.6.2 Hibernate Architecture

There are several Hibernate interfaces that are used for separate the business logic from persistence logic as illustrated in the following figure.

Figure 2.17 – Hibernate Architecture [Hibernate (2006)].

The most important Hibernate interfaces are the following:

- The session interface is the primary interface in a Hibernate application. It is responsible of the persistence management. When an object state is changed, the session interface provides the required operations for retrieving and storing the objects. In a Hibernate application session objects must be created and destroyed all the time. This is why these objects are lightweight in comparison to other Hibernate objects.
- The sessionFactory interface is responsible for creating new Session instances. There is a sessionFactory for every database in a Hibernate application.
- The transaction interface is highly associated with the session interface. A session may carried out multiple queries at the same time. When all of them are required to be committed or rolled backed is necessary the use of transactions. This interface provides the required methods for declaring the boundaries of database transactions.
- The query interface is responsible for performing queries against the database. It provides a wide range of methods for helping developers to run queries using native SQL (Standard Query Language) dialect or HQL (Hibernate query Language).

2.6.3 The Persistence Lifecycle

The object lifecycle of a hibernate application is illustrated in the following figure.

Figure 2.18 – Hibernate object lifecycle [Bauer & King (2005)].

The three possible states of an Hibernate object are the following:

- Transient : When objects are first instantiated using the *new* method, they are not associated to the database yet. These objects are called transient objects. They act as a normal object without any connection to any database row. Therefore,

any change that is made in these objects does not affect the information stored in the database.

- Persistent : Any new object can be persistent via the execution of the *save* method. Persistent objects are associated directly with the persistence manager (session interface) that communicates directly with the database. Any change in the state of a persistent object is synchronised with the database. However, this state is saved in memory until the end of the transaction. When a transaction is committed all these changes are propagated to the database. Calling the *delete* method, the object is removed from the database and becomes a transient object.
- Detached : After executing a transaction, the persistence instance are still associated with the persistence manager (session interface). When the *close* method is invoked, this association is removed and these objects are not longer synchronised with the database.

2.6.4 HQL (Hibernate Query Language)

Although, native SQL queries can be executed in a Hibernate application, a powerful query language called HQL is also available in order to manage the data retrieval. As well as HQL, there are two more ways to express queries in Hibernate (Figure 2.19). One is using the Criteria API (Application Program Interface) and other is using SQL that maps results set to objects automatically. Using these two methods database queries can be carried out in a more effective way as it was designed for being used in a object-oriented environment. Therefore, notions such as polymorphism and inheritance can be applied to these queries. However, HQL is preferred when complex queries need to be created [Bauer & King(2005)].

Using HQL

```
session.createQuery("from Category c where c.name like 'Laptop%'");
```

Using Criteria API

```
session.createCriteria(Category.class)
              .add(Expression.like("name","Laptop%"));
```

Using direct SQL with automatic mapping

```
session.createSQLQuery(
       "select {c.*} from CATEGORY {c} where NAME like 'Laptop%'",
       "c",
       Category.class);
```

Figure 2.19 – Ways to express queries in Hibernate[Bauer & King(2005)].

2.7 Summary

This chapter has explained the general aspects of Java EE 5 platform and the benefits of using frameworks for building web-based applications. Furthermore, a deep look into Apache Struts and Hibernate, has provided a more detailed view about the features of these frameworks. Having defined the technology that is being used for building SUMS, the next chapter includes a detailed description of the system.

Chapter 3 – Analysis

This chapter will discuss:

- An analysis of the problem to be solved.
- A detailed description of the system including UML diagrams and the requirements specification.

3.1 Introduction

System analysis is necessary in order to document and establish the system requirements prior to start coding the application. A well requirement documentation is necessary in order to understand the system from the client's point of view. Also, UML (Unified Modeling Language) provides a set of methods for organising the system design in a more effective way. Therefore, the system developed can be much easier to be understood. To implement object-oriented systems with high quality, it is important to use these methods in order to obtain a good product. Lots of benefits can be gained using Object Oriented Design in Software Development. Reusable modules, managing complexity, good maintenance, are some examples of these benefits. UML will help developers to understand how the system works and all the important elements related to it. These methods have been adopted in order to produce the required documentation for system design.

3.2 Project Planning

In the first stages of the project, a Gantt chart diagram was created (Appendix C) in order to plan the project stages with a lot of detail. Deadlines for each phase and sub-phases were established in the diagram. The waterfall model is most common strategy for software development (Figure 3.1).

Figure 3.1 - The waterfall model (Coopeland, 2003).

Every development phase should be completed in order to go to the next phase. This way to divide the project into several stages may be very successful for planning the application properly. Nevertheless, as Woojong & Barnes (2004) suggest, the prototyping method for developing web-based applications is more suitable as web-based applications are continuously being adapted to their environment (Figure 3.2).

Figure 3.2 - Evolutionary Prototyping Method (Barnes & Woojong, 2004)

The initial stages of the project planning iterates for a better understanding of the user requirements. The prototype is developed according to the user feedback and only when the user and the client agree with the prototype, the real application starts being developed. In some cases, the prototype may be completed and released as the final product. This is also known as evolutionary prototype. This method has a lot of benefits as it increases the comprehension about that the client wants, helping to minimize the misunderstandings about user requirements. Furthermore it provides a more flexible project planning as several tasks can be carried out at the same time.

3.3System Requirements

In the early stages of the system development, the user requirements were documented according to the standards specified by the IEEE (Institute of Electrical and Electronics Engineers, Inc.) [ANSI/IEEE Std 830-1998] that are included in the Appendix B. This formal specification presents the good practice for specifying the system to be developed. All the system requirements were gathered from interviews with Dr. Jim Briggs and reviews of the PUMS(Project Unit Management System). A description of the Online Project Allocation Process is included in the chapter 1, section 3. It defines the required steps needed for allocate projects to students.

3.3.1 System Actors

There are four actors who interact with the system .

- Clients: People who are interested in submitting project ideas and are not related to the university have to register into the system. Then, they will submit, edit, delete, amend project ideas and change their contact details. They usually represent companies which need a specific application. The rest of users will play the same role as the clients.

- Students: Students who are enrolled in a postgraduate or undergraduate final course in the University need to select a number of projects. This project

selection is made from the list of project ideas submitted by all users who interact with the system.

- Supervisors: They oversee the student's project. They are responsible for ensuring that students carry out their project development in the proper way. They will track the student process and will assess the project report.

- Unit Coordinators: After students submit their project choices, Unit Coordinators have the required information in order to allocate projects to students. They are responsible for allocating the projects according to the project allocation criteria.

Both, Coordinator and Supervisor are represented by the Staff Member actor. They share the properties described in the Staff Member actor. Another generalisation that could be included is illustrated in Figure 3.3. All the system actors can be represented by an actor called person. Having this actor, most of the functions that are shared by all the system actors such as project ideas searching and project ideas submission can be grouped in this actor.

Figure 3.3 – System actors.

3.3.2 Use Cases Diagrams

It is important to know how the system works from the user's point of view. The users need to be involved in the application design in order to develop the system according to their requirements. UML's case diagrams provide useful way to document the system functionality and how the distinct users interact with the system. The following figure

illustrates the Use Case Diagram for the project allocation process. The actor person has not been represented in Figure 3.4 because it would be difficult to include all the actors in the diagram. The system functions that are shared by all the actors are supposed to be represented by the person actor.

Figure 3.4 – Project Allocation Use Case Diagram.

3.3.3 Functional Requirements

After having defined the distinct actors (Figure 3.3) who take part in the project allocation process and the main functions that the system should provide to them(Figure 3.4), the next list contains a summary of the system requirements explained with a more detail and categorised by the system actors.

1. Person
   - Search project ideas: Every actor should be able to search for any project idea stored in the system. The searching can be carried out according to a wide range

of criteria such as project idea status code, title, client who submitted the idea, etc.

- Submit project ideas: System users can submit as many project ideas as they want. Furthermore, the system should provide a mechanism for editing, deleting, and amending the ideas already submitted.
- View project ideas: Users should be able to view a list of their project ideas. So, they can manage them in order to make the required changes in their information.

2. Client
   - Create new account: Before using the system, external clients should create a new account with relevant information such as name, address, mail, username, password, etc.
   - Change user details: After creating a new client account, clients should be able to amend their personal and security information.

3. Student
   - Submit project choices: Students should be able to select a number of project choices. These choices will be taken into account in order to allocate the projects to students.
   - Amend project choices: After the project choices are submitted, students should be able to view the project ideas which were selected and rearrange the preference order. They also should be able to submit another project choices if necessary. Only the newer list of options will be taken into account when the project allocation is carried out by the unit coordinator.

4. Staff Member
   - View students assigned to staff member: Both, supervisors and unit coordinators should be able to view a list of their students.
   - Search for students: Information about students can be retrieved. The system should provide a list of students according the searching criteria chosen. Furthermore, a list of project choices for each students should be provided.

5. Supervisor
   - Preferred students selection: Supervisors should be able to submit a list of their preferred students. Also, these students can be linked to the supervisor's project ideas.
   - Supervisor's project ideas chosen by students: Supervisors should be able to view a list of students who selected their project ideas.

6. Unit Coordinator
   - Change project ideas status: After a project idea is submitted, it needs to be approved before being available to be selected by any student. Staff members should be able to change the status in order to approve, withdraw, reject or project ideas
   - Students without project choices: The system should be able to provide a list of students who did not submit their project choices. A notification mechanism should be used for sending emails to these students.
   - Project allocation decision: After students submit their project choices and supervisors submit their preferred students, unit coordinators should be able to

allocate projects to students. In the event of a project being unable to be allocated because it was already allocated to other student, the system should notify this to the unit coordinator.

3.3.4 Requirements Prioritisation

OPAS should include all the functions described in section 3.3.3. Some of them can not be implemented until others are fully developed. For instance, the function submit project choices carried out by students can not be implemented until the function submit project ideas is finished.
In order to provide the system with a minimal functionality is necessary to develop some basic functions first. The following list provides a list of basic system requirements:

- Project Ideas Submission carried out by all the system actors.
- Project choices carried out by the student actor.
- Preferred student selection carried out by the supervisor actor.
- Project allocation carried out by the unit coordinator actor.

After these functions are completed, the rest of system functions can be developed.

3.4 Review of SUMS

In order to meet the system requirements is necessary to carry out a review of the old system. In the early stages of the system development, Briggs provided several files containing the data structure (Java classes) of Sums. The only sub-system that is in use at the moment is the Project Marking Sub-system that was built using Apache Struts but without using Hibernate. The use of Apache Struts and Hibernate is a requirement for the development of SUMS. Therefore, the Project Marking Sub-system need to be redesigned using Hibernate. This task is carried out by Briggs. The SUMS database tables are illustrated in Figure 3.5 [Powell (2005)].

Figure 3.5 – Sums database tables relationships [Powell(2005)].

OPAS needs to be designed taken into account the existing SUMS data structure so that it will be integrated into SUMS properly. Therefore, some data objects that are represent by Java classes and are necessary for the development of OPAS such as person, student, cohort_coordinators etc., can be taken from SUMS. Additional data objects are necessary in order to provide the required functionality to OPAS. The data objects that have been identified are the following:

- Project Choice : the project ideas that are chosen by students.
- Preferred Students: preferred students chosen by supervisors
- Project Ideas: although there is a data object that is used for representing the contact project ideas, it need to be redesigned in order to provide to every system user a way to submit project ideas.
- Contacts: SUMS uses a data object called contacts in order to register the clients. This data object need to be defined as a association of the person data object.

These data objects required for OPAS will be added to SUMS structure.

3.5 Summary

This chapter has described the actors and how they interact with the system. Having defined the functions that the system should provide in a detailed way, the next chapter focuses in the design phase of the application explaining how the different parts of the system were developed.

Chapter 4 – Design

This chapter will discuss:

- Presentation layer design and the methods chosen for improving the application usability.
- Persistence layer design with the new data objects and database tables that were created.
- The business layer design and the controller components of the application.

4.1 Introduction

The use of the MVC or layered architecture, makes that the application design is split up into distinct parts. Each part is independent of the others and any change that is made in one part does not affect the others. As discussed in chapter 2, the use of frameworks such Struts and Hibernate has a lot of benefits as the application development can be organized in a more effective way. The design of each component that is part of the application is described with a lot of detail in this chapter.

4.2 Presentation Layer Design

The user interface on a web site plays an important role in order to enhance the relationship between the user and the organization. In this context, usability principles have become one of the most important criteria to be taken into account when designing a Web site. Although OPAS is not aimed to be a typical commercial web page where users purchase items, it was recommended to use some usability principles in order to making easier to use the application. How the web page is structured plays an important role. Page layouts techniques, enhance the readability of web applications. Therefore, Gestalt principles can be used in order to improve the structure coherence of the web page [Brinck, et al (2002)]. These principles describes techniques that improves the perceptual organization of the web page. Distributing the information properly in the page makes easier the interaction between the user and the system.

Navigability is another important factor to be taken into account. Users should know where they are and how to get to another section easily. Hypertext links are identified clearly and multi-page for long result are available in order to organize the results properly. Search interfaces are easily identified, having the results properly displayed in the page. The OPAS web pages were designed for an screen resolution of 800 x 600. Furthermore, the use of variable pixel-width for designing elements was avoided as this can provoke some problems for displaying the data properly.

4.2.1 Web Architecture Map

A web architecture map (Figure 4.1) represents how the navigation paths are designed so that users can follow them through the site. It also provides a general view about the distinct sections available for each user in the application and how they are connected.

Figure 4.1 – Web Architecture Map.

4.2.2 Web Wireframes

Web wireframes provides a visual guide about the placement of fundamental elements in the interface design. They gives a reference about the basic structure every web page. Figure 4.2 illustrates the OPAS home page. All the pages in OPAS should be designed according to the layout distribution illustrated in this wireframe. The rest of wireframes have been included in Appendix D.

Figure 4.2 – OPAS Home Page Wireframe.

4.3 Persistence Layer Design

The persistence layer is responsible for saving to and retrieving from the database the data that is used in the application. Using Hibernate, when the persistence object methods such as save, update, etc., are executed, the persistence layer will automatically perform an action in the database. In order to represent the data, Java classes can be designed first and then be mapped into database tables. Also, database tables can be created first and then mapped into Java classes. Briggs in the early stages of the application development provided me several Java classes that represent the structure of the data that is used in SUMS. Having this, the required data objects for representing OPAS data structure were created and then mapped to database tables. In this section the new data objects that were created as well as the database tables are described.

4.3.1 Data Objects

Data objects that are used in SUMS must be preserved. Therefore, the new required Java classes are added to this structure so that OPAS will be integrated into SUMS properly. Having into account the system requirements and the project allocation process, that have been already defined, the new required classes that are used in OPAS are illustrated in the Figure 4.3.

Figure 4.3 – OPAS Class Diagram.

In order to manage contact's project ideas in SUMS, *Contacts* and *Ideas2* classes were created. Although these data objects were available in SUMS original design, they need to be revised. Originally, *Contacts* class was associated to *Ideas2* class in a one to many relationship. No other relationships were defined in *ideas2 class*. In the new design, every system actor can submit project ideas. Therefore, contact class was defined as a association of the super class person and *Ideas2* class is associated to *Person* class in a one to one relationship.

ProjectChoice and *ProjectChoiceOptions classes* were created in order to manage the information related to the student's project choices. *PreferredStudents* class was created in order to manage the information about the supervisor's preferred students. This class is also associated with *Student and Ideas2* classes as every student can be associated with any supervisor's project idea.

The rest of the classes illustrated in the class diagram such as *Person*, *Student*, *MarkerCapacity*, *FinalProject* are taken from SUMS data structure. Although no significant changes must be done in the structure of these classes, the required references to the new OPAS Java classes must be included.

4.3.2 OPAS Database Tables

SUMS database design is illustrated in Figure 3.5 in chapter 3. After adding the required data objects that are used in OPAS and mapping them to database tables using Hibernate, the original database structure is modified. Figure 4.4 illustrates all SUMS database tables including those related to OPAS and their relationships between them.

Figure 4.4 – New SUMS database design [Adapted from Powell (2005)].

The following figures represent the data dictionary for the revised and new tables.

Table Name	*CONTACTS*	
Field	**Type**	**Description**
orgid	bigint (autoincrement)	Primary Key
personId	bigint	Foreign Key to PERSON(personId). Identifies the person.
orgaddr	varchar	Postal address
orgdoes	varchar	Outline of what the organizations does.
orgemail	varchar	Email address
orgname	varchar	Organization name
orgpostcode	varchar	Mail postcode
orgtel	varchar	Telephone number

Figure 4.5 – Contacts table.

Table Name	*IDEAS2*	
Field	**Type**	**Description**
projid	bigint (autoincrement)	Primary Key
personId	bigint	Foreign Key to PERSON (personID). Identifies the person
projapproved	varchar	Project Idea Status
projdate	date	Submission date
projdeliver	text	Anticipated Deliverables
projaims	text	Aims and Objectives
projquest	text	Academic question to be answered
projstudent	varchar	Student for whom project has been devised
projtitle	varchar	Project Title

Figure 4.6 – Ideas2 table.

Table Name	*PROJECT_CHOICE*	
Field	**Type**	**Description**
choiceId	bigint(autoincrement)	Primary Key
studentId	bigint	Foreign Key to STUDENT(studentId). Identifies the student.
choiceDate	date	Project choice submission date

Figure 4.7 – Project_Choice table.

Table Name	*PROJECT_CHOICE_OPTIONS*	
Field	**Type**	**Description**
pcoId	bigint(autoincrement)	Primary Key
choiceId	bigint	Foreign Key to PROJECT_CHOICE(choiceId). Identifies the Project Choice.
projId	bigint	Foreign Key to IDEAS2(projid). Identifies the project idea related to this choice
preferenceOrder	smallint	The preference order for that project chosen.

Figure 4.8 – Project_Choice_Options.

Table Name	*PREFERRED_STUDENTS*	
Field	**Type**	**Description**
preferredId	bigint(autoincrement)	Primary Key
studentId	bigint	Foreign Key to STUDENT(studentId).Identifies the student
projid	bigint	Foreign Key to IDEAS2(projid). Identifies the project idea
markerCapacityId	bigint	Foreign Key to MARKER_CAPACITY (markerCapacityId). Identifies the supervisor

Figure 4.9 – Preferred_Students table.

Having defined the data dictionary, next figure illustrates the OPAS Entity-Relationship diagram in a more detailed way.

Figure 4.10 – OPAS Entity-Relationship Diagram.

4.4 Business Layer Design

In order to design the business layer of this application, two patters were adopted;the front controller and the business delegate pattern. This section shows how these patterns were applied for developing OPAS and what are the benefits of their implementation.

4.4.1 Front Controller Pattern

As discussed in chapter 2, In Java2 EE platform, MVC model 2 approach uses servlets in order to process the application requests. Therefore, it avoids the data processing within the view. Using the Front Controller pattern (Figure 4.11), the requests processing as well as the view selection are centralized in a controller component. Having this configuration, some mechanism such as authentication, error management, delegate business processing, etc., can be controlled in a more effective way as the amount of code is highly reduce, avoiding the problems of having duplicate code in the application. Besides, multiples views can be available for a single request. Therefore, the required view will be selected according to the results obtained from the data processed. The main benefit of this configuration is that the application maintenance will be easier to be carried out and the code will be easier to be understood.

Figure 4.11 – Front Controller Pattern [Sun Developer Network(2006-a)].

The number of requests handlers are not limited to one. In an application, depending on the user's action a specific handler will be invoked. As explained in chapter 2, using Struts, ActionClass objects are used in order to process the requests. In OPAS, a set of ActionClass objects were defined in order to process the request data sent by the user. For every user's action an ActionClass object is available for processing the information and selecting the appropriate view.

However, although the business methods can be entirely implemented in the ActionClass object, this is not appropriate as the application become difficult to maintain. As Husted, et al. (2003) explains, the problems including the business methods within the ActionClass objects are similar to those discovered when using the MVC Model 1 architecture. The use of the Business Delegate Pattern solves this problem.

4.4.2 Business Delegate Pattern

The use of the Business Delegate Pattern (Figure 4.12) allows to delegate all the responsibility of the business logic to another class that contain the methods for retrieving the data from the required source. ActionClass objects need to contain the less amount of Java code, including all the business methods within the Business Delegate classes.

Figure 4.12 – Business Delegate Pattern [Sun Developer Network(2006-b)].

When these methods need to be used, a new instance of the Delegate class will be created. After that, the required method will be invoked and the information will be retrieved. OPAS makes use of DAO(Data Access Objects) in order to encapsulate the data access. These objects contain the required methods for accessing to the database, retrieve the data and return it in the proper way.

4.5 Summary

This chapter has described the system design phase. How the distinct application layers were designed as well as the methods chosen for that have been explained in a detailed way. The next chapter focuses on the system implementation, explaining how it was carried out and the main problems encountered.

Chapter 5 – Implementation

This chapter will discuss:

- Development tools that were used.
- How the presentation, data, and business layer implementation were carried out.
- Problems encountered while implementing the application.

5.1 Introduction

The system implementation phase was split into three parts. Firstly, the required user interfaces were implemented using a HTML editor. Secondly, OPAS data structure represented by Hibernate beans was defined. Therefore, mapping these objects to database tables, the database structure was created. Finally, the user functions and the business logic were implemented. However, some parts of the application had to be redesigned while implementing others in order to integrate the system properly. This chapter focuses in those relevant aspects of the system implementation. Problems found while implementing the system as well as the decisions taken in order to solve them are also described.

5.2 Development Tools

The hardware and software development tools that were used for developing the application are covered in this section.

5.2.1 Hardware

The system development did not required the use of specific hardware. Although the application will be run in servers which are allocated in the university, the application was developed using a normal laptop. The computer has the following specifications (Figure 5.1):

Model	Compaq Presario V4160 EA
CPU	Pentium M Centrino – 1,86 GHz
Memory	1024 MB DDR2
Hard disk	80 GB
Screen	Widescreen 15,4"
Graphics Processor	Mobile Intel(R) 915GM/GMS Express Chipset Family

5.2.2 Software

The following sections describes the software development tools that were used for implementing the system.

5.2.2.1 MySQL Server 4.1

Oracle Server 10i is the DBMS(Database Management System) that is used in SUMS. However, using Hibernate, OPAS could be developed using a different DBMS as Hibernate can be configured easily in order to be adapted to a different environment. The only parameter that must be changed is the SQL dialect that Hibernates uses to communicate with the database. MySQL Server is the most popular open source database. It is very flexible, powerful, and very easy to manage. Furthermore, It is being used by large enterprises companies as they can save a lot of money having the same quality and technical support than proprietary databases. [MySQL (2006)].
Although the most recent stable release is MySQL 5.0, the upgrade to this version was not taken into account as this version provides enough features for being used to develop OPAS.

5.2.2.2 MySQL Command Line Client

In order to manage MySQL database server, an additional software was required. MySQL administrator provides a friendly interface that makes very easy to create and maintain the databases. Another tools such as the command line client uses a MS-DOS console. This way to manage the database server is much more difficult as users need to type commands in order to execute actions. This way is more suitable for advanced users. I have experience using MySQL and this is the way I have always been managing MySQL server. So, MySQL Command Line Client was the chosen tool to manage the database server.

5.2.2.3 NetBeans 5.0

There are several Java IDE (Integrated Development Environment) that are used for developing Java web applications. Jbuilder (Borland), Eclipse(IBM), and NetBeans(Suns Microsystems) are some examples. NetBeans is an open source Java IDE that gives support for web frameworks such as Struts. For using hibernate, the required modules were downloaded using the update center. Although version 5.5 is available, it is a Beta release and might have some problems while developing the application as this is not a final version. Therefore, version 5.0 was chosen as it is more stable.

5.2.2.4 Macromedia Dreamweaver MX

Macromedia Dreamweaver was used in order to design the user interfaces. Web pages can be designed without using an HTML editor. Using plain text files, experienced web developers can design the required web pages for a web-based application. NetBeans provides a list of HTML elements that can be added to the HTML and JSP pages. However, if the user interface design is very complex, the use of an HTML editor is more convenient. Without using an HTML editor, the presentation layer design in OPAS had been much more difficult to be carried out.

5.2.2.5 Adobe Photoshop CSS

As it is commonly known, Adobe Photoshop is a software for image-editing. Although OPAS user interfaces do not contain pictures or graphics, some design elements such as space-delimiters and borders were created with Photoshop. Space-delimiters help to keep the original web pages structure when the dynamic content is displayed.

5.2.3 Problems Found on Development Tools

The lack of experience in the use of NetBeans, specially the use of Struts and Hibernate caused delays at the start of the system development. More time was required in order to get use to managing these tools. No problems were found in the other tools due to the experience in using them.

5.3 Presentation Layer Implementation

OPAS has several web pages. Depending on the user logged in, the options available and the content displayed is different. Wireframes for every user have been included in Appendix D . They were useful for helping to develop the required user interfaces in a more effective way.

Using a HTML editor such Macromedia Dreamweaver, the web pages development was carried out easily. Firstly, using HTML, the layout of every web page was created. Then, the code was copied to the JSP pages that were edited using NetBeans. The distinct elements taking part in the layout are imported using the include directive as illustrated in Figure 5.1.

```
<tr>
    <td height="10" valign="top"><img src="design/topes/1x10b.gif" width="1" height="10"/>
    </td>
</tr>
<tr>
    <td height="89" valign="top">**<%@include file="sections/generalOptions.html"%>**
    </td>
</tr>
<tr>
    <td height="10" valign="top"><img src="design/topes/1x10b.gif" width="1" height="10"/>
    </td>
</tr>
<tr>
    <td height="77" valign="top">**<%@include file="sections/ideasSearch.html"%>**
    </td>
</tr>
```

Figure 5.1- The include Directive.

Therefore, the content of the file is imported rather than the output (jsp:include Directive). As a result, only one servlet is created. The content is first inserted into the page and then the page is translated into a servlet. This way to design web pages that have a lot of elements in common improves maintainability of the user interface.

At the early stages of the web pages implementation, the use of templates were taken into account. Macromedia Dreamweaver allows the use of templates in order to speed up the development and make easier the maintenance of web pages. Therefore, pages that have a common layout structure can be designed using a template. When a change is made in the template, web pages that were designed using that template are automatically updated. However, instead of using templates and having several web pages with the same layout, only one index page (index.jsp) was created for all users. Depending on the user logged in, the information displayed is different. Using the include Directive as well as JSTL for testing the user type, the options available for each user are different. Next figure illustrates which page is inserted according to the search engine used (a scoped attribute using JSTL is checked).

```
<td width="580" rowspan="8" valign="top">

  <c:choose>
    <c:when test='${not empty sessionScope.displayIS}'>
        <c:if test='${sessionScope.displayIS=="ideas"}'>
         <%@include file="sections/displayProjectIdeas.jsp"%>
        </c:if>
        <c:if test='${sessionScope.displayIS=="students"}'>
         <%@include file="sections/displayStudents.jsp"%>
        </c:if>
    </c:when>
  </c:choose>

</td>
```

Figure 5.2 – JSTL Conditional Statements.

Dynamic content can be added to web pages using JSTL tag library. However, Struts framework provides a Struts tag library which can be used for managing the information to be displayed in web pages. JSTL should be used instead of Struts tags as it contains a useful set of tags, many of which overlap with the Struts tags [Husted, et al (2003)]. Therefore, JSTL makes obsolete the use of Struts logic and bean taglibs. Therefore, JSTL was used instead Struts taglibs. However, JSTL is not a substitute for the Struts HTML taglib as it does not provide this functionality. Figure 5.3 shows the code for displaying a list of project ideas when users search for project ideas.

Checking if the scoped attribute exists and contains any value

Iterate over the scoped attribute

```
<c:if test='${not empty sessionScope.searchIdeasBean}'>
     <c:forEach items='${sessionScope.searchIdeasBean}' var='idea'>
         <tr>
          <td height="10" colspan="7" valign="top"></td>
         </tr>
         <tr>
          <td height="20" colspan="2" valign="top" class="resultsLink">
<html:link action="displayIdeaInfo" paramId="id" paramName="idea" paramProperty="projid">
             <c:out value='${idea.projtitle}'/>
          </html:link>
          </td>
          <td width="5" valign="top"></td>
          <td width="259" valign="top"  class="normalText">
            <c:out value='${idea.contacts.orgname}'/>
          </td>
          <td width="70" align="left" valign="top" class="normalText">
            <c:out value='${idea.projapproved}'/>
          </td>
          <td width="63" align="right" valign="top" class="normalText">
            <c.out value='${idea.projdate}'/>
          </td>
         </tr>
     </c:forEach>
</c:if>
```

Link with a parameter. Its value is the value of the given property

Printing the value of the given property

Figure 5.3 – Scoped Attributes & JSTL.

5.4 Data Layer Implementation

Most of the data objects that are used in OPAS are available in SUMS. Briggs provided me several data objects (Java Beans) that represent the current database structure of SUMS. As commented in chapter 4, Hibernate beans are first created (Figure 5.4) and then are mapped into database tables, creating the database structure. Each property in a Hibernate bean represents a field in the database table. Furthermore, foreign keys are created according to the relationship between Java classes that are defined in every Hibernate bean. Although columns types in database tables can be defined for each property, Hibernate creates the appropriate type for the field according to the data type defined in the Java class if the column type is not defined.

```
/**
 *
 *
 * @hibernate.class
 *     table="CONTACTS"
 *
 */
public class Contacts {

   // <editor-fold defaultstate="collapsed" desc=" PrimaryKey:   long
orgid ">

   private long orgid;
/**
  *   @hibernate.id
  *     generator-class="increment"
  *     column="orgid"
  *     type="long"
  *   @hibernate.column
  *     name="orgid"
  *     sql-type="bigint"
  *     not-null="true"
  */
   public long getOrgid () {
      return orgid;
   }
   public void setOrgid (long orgid) {
      this.orgid = orgid;
   }
   //</editor-fold>
   .
   .
   . The rest of the attributes##########
   .
   .
   // <editor-fold defaultstate="collapsed" desc=" 1-N
            Relation to Collection /*pojo.Ideas2*/ ideas2s ">

   private Set ideas2s;

   /**
    * @hibernate.set
    *     role="ideas2s"
    * @hibernate.collection-key
    *     column="orgid"
    * @hibernate.collection-one-to-many
    *     class="pojo.Ideas2"
    */
   public Set getIdeas2s() {
       return this.ideas2s;
   }

   public void setIdeas2s(Set ideas2s) {
       this.ideas2s = ideas2s;
   }
   // </editor-fold>
```

Name and type of the Attribute

Defined as Primary Key

Information used in order to generate the database fields

Getter and Setter methods

A one-to-many relationship with another object is defined

Information used in order to generate relations between objects

Setter and Getter methods that make reference to the object

Figure 5.4 – Hibernate Bean (Class Contacts)

SUMS is being developed using Oracle database. However, OPAS was developed and tested using MySQL server. Some properties defined in SUMS Java classes use the

Clob(Character Large Object) data type. Using MySQL there were problems for mapping these classes to database tables. Therefore, the use of Clob data type was avoided. String data type was used instead.

At the beginning, some problems were encountered when trying to map Java classes to database tables. MySQL 4.1 use MyISAM storage engine by default. It does not support transactions and relationships between tables could not be defined. InnoDB storage engine was used instead. Furthermore, utf8 is the default character for every table created in the database. Using this character there were some problems when mapping Java classes as the primary keys as well as indexes could not be created properly in the database. Changing the default character to latin1 in the MySQL configuration file (my.ini) solved the problem as this configuration supports key lengths longer than 1000 bytes.

As commented in chapter 2 (section 2.5.5), one of the main components of the Struts framework is the ActionForm object. This object is used in order to represent the HTML form data that was submitted by users. Therefore, several ActionForm objects were defined in OPAS in order to deal with this data. This object structure is similar to Java beans structure as it contains attributes and the setter and getter methods. However, the problem arises when there are a lot data to manage or the number of parameters varies. When students make the project selection, a list of projects is displayed. Then, they select the required projects using checkboxes and finally submit the form. The number of projects selected may vary. In order to manage this data, an array of strings had to be created (Figure 5.5) in the ActionForm.

```
package general.forms;

import org.apache.struts.action.*;

public class ProjectChoicesForm1 extends ActionForm
{
   protected String[] selectedItems={};

   public void setSelectedItems(String[] selectedItems)
   {
      this.selectedItems=selectedItems;
   }
   public String[] getSelectedItems()
   {
      return this.selectedItems;
   }
}
```

Figure 5.5 – ActionForm example.

5.5Business Layer Implementation

A set of Action classes were created in order to process the information. The main function of these objects is to prepare the data in order to be displayed in the presentation layer. Every operation that the application can undertake is defined in the ActionMapping located in the Struts configuration file. When the user performs an action, the requested path is checked in order to know if it matches any Action URI in

the Struts configuration file. All these objects represent the front controller pattern (chapter 4, section 4.1). Furthermore, it is not appropriate to include all the required Java code for processing the information in these objects. The use of helper classes makes easier the understanding of the business logic as the readability of the code is greatly improved. These classes represent the business delegate pattern (chapter 4, section 4.2).

OPAS business layer was developed using the front controller and business delegate pattern. Figure 5.6 represents the structure of a typical Struts Action showing how the helper methods are managed. Every Action class in OPAS was created using this structure.

```
package general.actions;        Package containing this Action class

import org.apache.struts.action.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;                                   Required Classes
import org.hibernate.HibernateException;
import process.*;
import pojo.*;

public class EditIdeaDetails extends Action         Class extended from Action class
{
   public ActionForward execute(ActionMapping mapping, ActionForm form,
       HttpServletRequest request, HttpServletResponse response)
              throws IOException, ServletException
   {
      The value of the parameter/s (if exists) is retrieved
    long projid=Long.parseLong(request.getParameter("id"));

    try
     {

       Ideas2DAO ideas2DAO=new Ideas2DAO();         A new instance of the helper class is created

       idea=ideas2DAO.findIdeasByProjid(projid);      The required method is invoked and the
                                                      information is retrieved
       request.setAttribute("ideaBean", idea);      A request scoped attribute is created

       return mapping.findForward("success");       Return success
     }
     catch(HibernateException exception)
     {
        return mapping.findForward("error");
     }
     catch(Exception exception)                     Managing Exceptions
     {
        return mapping.findForward("error");
     }
     finally
     {

     }
   }
}
```

Figure 5.6 – Action class example

Figure 5.7 shows the code for the helper method that is used in the Struts Action object illustrated above.

```
public Ideas2 findIdeasByProjid(long projid)
   {
      A new session instance is obtained

      Session session=process.HibernateUtil.currentSession();

       Using a criteria query

      Ideas2 idea=(Ideas2)session.createCriteria(Ideas2.class)
                   An association with another object is specified

                   .setFetchMode("contacts", FetchMode.JOIN)
                   Expressing a constraint on the property projid

                   .add(Restrictions.eq("projid",projid))
                   Ordering query result. The uniqueResult method executes
                   the query and returns an object as a result

                   .addOrder(Order.desc("projid")).uniqueResult();

      return idea;    Return the object with the required data
   }
```

Figure 5.7 – Helper method.

The Action class and the helper method shown above were used in OPAS in order to edit project ideas details. When users want to retrieve information about project ideas, the Action class in Figure 5.6 is invoked. Then, it uses the helper method in order to find the information related to that project idea. As commented in Chapter 2 (section 6.6), Hibernate provides three ways to express queries. In OPAS, all the queries were created using the criteria API.

5.6 Implementing OPAS functions

This section describes some functions that were implemented in OPAS. Although OPAS contains more functionality, only those that had some problems for being developed are described.

5.6.1 Creating Project Ideas

Every system user may create new project ideas. Next figure illustrates the user interface that is used for managing the user's project ideas.

Figure 5.8 – Managing Project Ideas.

Two sections are available. The section at the top is used for editing and deleting project ideas already submitted. The section at the bottom allows users to submit or amend project ideas. When users submit the form, the Action class *submitIdeaForm* is invoked. Then, the data from the *ideaForm* class (ActionForm class) and using the appropriate method from the *ideas2ADO* class (helper class), inserts the data into the database. Next Figure shows how the helper methods in *ideas2ADO* inserts a new project idea into the database.

```
public void saveIdeas2(Ideas2 ideas2)
    {
        A new instance of the session object is created from the session factory
        Session session = process.HibernateUtil.currentSession();
        Hibernate transaction is initialized
        Transaction tx = session.beginTransaction();
        Making the session object persistent
        session.save(ideas2);
        The transaction is executed
        tx.commit();

    }
```

Figure 5.9 – Making an object persistent.

The Hyperlinks edit and delete that are displayed in Figure 5.8 allows users to edit and delete project ideas. These Hyperlinks are created using the Struts HTML tag *<html:link>* as shown in the next Figure.

```
<c:if test='${not empty sessionScope.contactIdeasBean}'>
<c:forEach items='${sessionScope.contactIdeasBean}' var='ideas'>
<tr>
   .                    Iterating over a session scoped attribute
   .
  <td align="center" valign="middle" class="normalText">
    <html:link action="/editIdeaDetails" paramId="id" paramName="ideas"
                                                 paramProperty="projid">
       <bean:message key="general.title.edit"/>
    </html:link>        Name of the request parameter        Name of the variable of this iteration
  </td>
  <td align="center" valign="middle" class="normalText">
    <html:link action="/deleteIdea" paramId="id" paramName="ideas"
                                                 paramProperty="projid">
       <bean:message key="general.title.delete"/>
    </html:link>
  </td>                 Logical name of the action
                                                 The name of the bean property that dinamically is
                                                 added to this hyperlink
   .
   .
  </tr>
</c:forEach>
</c:if>
```

Figure 5.10 – Creating Hyperlinks using Struts HTML tags.

When users press delete, the Action class *deleteIdea* is invoked. Then, the method *deleteIdea* in the helper class *Ideas2ADO* is used in order to delete the idea.

5.6.2 Project Ideas Choices

Students can select project ideas. First, a list of project ideas available is displayed (Figure 5.11). After submitting the project choices, a new page is displayed (Figure 5.12). This page contains a list of project ideas choices already submitted. Students can also rearrange their options according to their preference order.

Figure 5.11 – Project Ideas Available.

Figure 5.12 – Rearrange Project Ideas

When students rearrange their project choices list, the Action class *RearrangeProjectChoices* is invoked. Students only can insert numbers in the textfields. The maximum number to be inserted does not have to be more than the total of project ideas. Also, two textfields cannot contain the same information. ActionForm classes provide a validate method in order to check the input data. However, due to problems for redisplaying the form, the input data was checked in the Action class. A helper method was used for this task (Figure 5.13).

```
public static String validateNumbers(String[] preferenceOrder)
  {
    String text="OK";
    for(int i=0;i<preferenceOrder.length;i++)
     {
       If there is a non valid character an numberFormat exception is
       thrown. It is caught in the Action class
      int number=Integer.parseInt(preferenceOrder[i]);
       Checking if there is a number higher than the maximum allowed
      if (number>preferenceOrder.length)
      {
       text="projChoicesForm.errors.maxNumber";
       break;
      }

      int total=0;
    for(int z=0;z<preferenceOrder.length;z++)
      {
       Checking if there is two equal numbers
       if(number==Integer.parseInt(preferenceOrder[z]))
        total+=1;
      }
      if (total>=2)
      {
        text="projChoicesForm.errors.equalNumber";
        break;
      }
     }
    return text;
  }
```

Figure 5.13 – Checking project choices form data.

5.7 Summary

This chapter has described how OPAS was implemented. The implementation was split up into three distinct parts. Therefore, the business, data, and presentation layer implementation was covered in a detailed way. Furthermore, this chapter has focused in the main problems found while implementing the system and how they were solved. Next chapter describes the system testing and the main functions that were implemented in OPAS.

Chapter 6 – Testing and Evaluation

This chapter will discuss:

- A testing on the main functions implemented in OPAS.
- Evaluation of the results obtained in the system testing.

6.1 Introduction

System testing is necessary to ensure if user requirements have been met. Also, errors can be identified and those functions that were badly implemented can be redesigned. There are a wide range of methods that developers use to undertake software testing. Those that required knowledge of the internal structure of the system such as White Box tests are usually carried out by software developers. However, others that not require knowledge of how the system was developed such as Black Box tests can be undertaken by both developers and system users. A test planning was used in OPAS to evaluate the system functionality, checking if the output data was generated properly according to each user action.

6.2 Debugging OPAS

While developing OPAS, some debugging techniques were adopted in order to ensure that the data was properly generated. NetBeans 5.0 provides a debugger tool that is very useful for debugging applications. Breakpoints can be added into the code in order to discover failures. Also, the values of every variable can be displayed. Therefore, checking these values, it was possible to find out if the programming methodology used in OPAS was the appropriate to process the data. This way to test the application code helped to develop a web application in a more effective way.

6.3 Functionality Testing

This section provide a set of test cases to test if the output data is generated properly according to user actions. Also, it indicates the level of functionality that was achieved in OPAS according to the user requirements established in Appendix B.

6.3.1 Project Ideas

Project Ideas Form	*Input*	*Output*	*Result*
Create new project Ideas	All the required information is provided	A new Project idea is created	**OK**
	One or more textfields are left empty	An error message is displayed for every textfield that was left empty. The form is populated with the information provided and the project idea is not created.	**OK**
Edit project ideas	Select my project ideas in the options menu	A list of previous project ideas for that user is displayed displaying the project idea name, status code and the submission date	**OK**
	The user selects a project idea by clicking in the Hyperlink provided in the list of project ideas	The project idea selected is edited and all the information is displayed in the project idea form	**OK**
Delete project ideas	The user clicks the delete Hyperlink provided in the list of project ideas submitted by that user	The project idea is deleted and a new list of project ideas is displayed without containing the project idea that was deleted	**OK**
Amend project ideas	The user edits a project idea already created, change some of the information, and submit the new information	A new project idea is created. The project idea information is not updated.	**FAIL**
Search Project Ideas	***Input***	***Output***	***Result***
	The user searches for project ideas without using the searching criteria	A list of project ideas is displayed. The name, the person or organization that submitted that idea, and the status code provided for every project idea	**OK**
	The user searches for project ideas using the searching criteria (status code)	A list of project ideas is displayed.	**OK**
	The user searches for project ideas using the	A list of project ideas is displayed. However,	**FAIL**

	searching interface(project ideas submitted in the last xx days)	this list does not correspond to the criteria chosen.	
	The user selects a project idea by clicking on the Hyperlink provided in the list of project ideas	A detailed information about that idea and the person who submitted it is displayed.	**OK**

6.3.2 External Contacts

External Contact Form	***Input***	***Output***	***Result***
Create new external contact	All the required information is provided	A new external contact is created	**OK**
	One ore more text fields are left empty	An error message is displayed for every textfield that was left empty. The form is populated with the information provided and the new external contact is not created.	**OK**
	The username was previously selected by other external contact	An error message is displayed and the new external contact is not created	**OK**
	The name was previously selected by other user	An error message is displayed and the new external contact is not created	**OK**
	The email address is provided without using the required format	The information is submitted and the new external contact is created	**FAIL**
	Password information need to be inserted twice. Both textfields contain different passwords	An error message is displayed indicating that the password have to be the same	**OK**
Edit external contact details	The external contact selects edit my details in the options menu	All the information related to that user is displayed properly	**OK**
Amend external contact details	After the external contact details is edited, the user changes some information and submits the form	External contact details are amended properly	**OK**

External Contact Login Interface	*Input*	*Output*	*Result*
Login into the system	Name and/or username are left in blank	An error message is displayed indicating that the textfield is empty	**OK**
	Name and/or username are incorrects	An error message is displayed indicating that the information provided is wrong	**OK**

6.3.3 Students

Project Choice Form	*Input*	*Output*	*Result*
	Select the “project available” option in the options menu	A list of projects available is displayed. Project idea name, user or organization that submitted that idea, status code, and submission date are the information provided in the list	**OK**
Submit project choices	Select projects ideas using the checkboxes available in the form and submit the information	The information is submitted and a new page is displayed (Project Choice Form 2) containing the list of options for that project choice. Also a list of previous project choices is displayed	**OK**
	The student does not select any project idea and submit the form	The information is submitted although no projects have been selected	**FAIL**
Project Choice Form 2	*Input*	*Output*	*Result*
	Select the option “your project choices” available in the options menu	A new page is displayed with the list of projects choices.	**OK**
	Select the option “your project choices” available in the options menu	If one project has been allocated to that student that information is not displayed in this form	**FAIL**
Rearrange options	Rearrange the options by indicating the preference number	The list of options for that project choice is rearranged according	**OK**

		to the preference numbers indicated by students	
	Insert a non-numerical character in the textfield	An error message is displayed indicating that the textfield contains a wrong format character	**OK**
	Insert two or more equal numbers	An error message is displayed indicating that two or more numbers are the same	**OK**
	Insert a number higher than the maximum number allowed (The maximum number is the number of options in that project choice)	An error message is displayed indicating that the maximum value possible is xx	**OK**
Delete a project option	Delete an option by clicking the Hyperlink "delete" available for each project idea in the list	The project idea is deleted from the list. The page is redisplayed and the new list of options is displayed without the deleted project idea	**OK**
Retrieve project idea information	Retrieve information about project ideas by clicking on the Hyperlink available in the list of options	A list of project ideas is displayed. The name, the person or organization that submitted that idea, and the status code provided for every project idea	**OK**
Edit project choices	Select a project choice already submitted from the list of project choices	The list of options for that project choice is displayed	**OK**

6.3.4 Supervisors

Search Students	***Input***	***Output***	***Result***
	Search students without using any searching criteria	A list of students is displayed. Also checkboxes are available in order to select preferred students	**OK**
	Search students without using any searching criteria	A list of students is displayed. Relevant information about the	**FAIL**

		name, course, and year is not indicated properly	
	Search students using searching criteria (Course)	The drop-list was not populated with information about the courses available. Students can not be searched using this criteria	**FAIL**
	Retrieve student information by clicking in the Hyperlink (Hemis number) available for each student in the list	Information about that student is displayed. Hemis number and project choices are also included. However, name, surname, and the course are not displayed	**FAIL**
Preferred Students Form	***Input***	***Output***	***Result***
Submit preferred students	After a list of students is displayed using the students searching interface, supervisors select their preferred students by using the checkboxes available. Then, the form is submitted	The students are added to list of supervisor's preferred students. A new page is displayed (preferred students form 2) containing the list of preferred students.	**OK**
	A student who was previously added to the list of preferred students is selected	The student is repeated in the list of preferred students	**FAIL**
Preferred Students Form2	***Input***	***Output***	***Result***
Associate preferred students to supervisor's project ideas	After select “my preferred students” in the options menu, a list of preferred students is displayed. Supervisors associate one of their project ideas to one of their preferred student	Drop-list are not populated with information about supervisor's project ideas. Supervisors can not associate preferred students to their project ideas	**FAIL**
Delete preferred students	Delete a student from the list of preferred students by clicking on the “delete ” Hyperlink provided for each student in the list	The student is deleted from the list of preferred students	**OK**

6.3.5 Unit Coordinators

A formal evaluation of unit coordinators functions can not be undertaken as these functions have not been implemented in OPAS.

6.4 Evaluation

This section provides an evaluation of the results obtained in the testing phase.

6.4.1 Project Ideas

Project ideas can be created. No errors were found while testing this function. Furthermore, the edition and deletion of project ideas were carried out without having any problems. However, project ideas details can not be updated. This function was not implemented in OPAS. When deleting a project idea a confirmation message should be displayed in order to authorise the deletion. Therefore, an accidental deleting would be avoided.

When users use the searching interface, a list of project ideas is displayed. The status code for every project idea is displayed using only one character. This way to display this information is not appropriate as nobody knows what is the meaning of those characters (A=Approved, P=Pending Approved, etc..). An additional information about the meaning should be provided in the web page using a key label. A better solution would be to display the status code for each project idea without using the abbreviate way.

6.4.2 External Contacts

External contacts can register in OPAS by providing personal and security information. Although no errors were found while testing this section, a validation mechanism should be implemented in order to check the data inserted. Therefore, email should be included using the right format and telephone number should not contain alphabetical characters. Furthermore, passwords should have provided using a minimal number of characters.

6.4.3 Students

When displaying student project choices ,a list of previous choices is also displayed. Students can edit that information and rearrange the project ideas indicating a preference order. However, unit coordinator will only take into account the last list of projects submitted. Therefore, it has no sense to rearrange a list that is not going to be taken into account unless the most recent list was deleted. Only the most recent list should be allowed to be rearranged. Another solution should be to leave the most recent list and to delete the former list. Although this solution is the most logical, it would not allow students to know information about former project choices.

6.4.4 Supervisors

Supervisors use the student searching interface in order to submit preferred students. Then, another web page is displayed containing the list of preferred students and the possibility to associate each student to any of the supervisor's project ideas. Although this function has not been implemented in OPAS, a future development identified would be to associate one student with one or more supervisor's project ideas, instead of selecting only one project idea for each student.

6.5 Summary

This chapter has focused on testing the main functions provided by OPAS. Although not all the required user functions have been implemented, OPAS provides a good level of functionality. According to the testing results, a system evaluation was carried out providing useful recommendations in order to improve the system functionality in the required areas. Next chapter includes a final conclusion about this project and a personal reflection on how the work was carried out.

Chapter 7 – Conclusion

This chapter will discuss:

- The achievements that were obtained in developing OPAS
- A constructive review about my work and how it was carried out

7.1 Project Planning

Before starting OPAS development, a plan needed to be defined properly in order to stablish the required stages to develop OPAS. A Gantt chart diagram was designed using the Waterfall model. This strategy to develop web applications is not very flexible as every stage needs to be completed before starting the next one. The OPAS development required a more flexible plan so that several tasks could be carried out simultaneously. Therefore, an alternative strategy was adopted. The use of evolutionary prototyping helped to plan the OPAS development in a more effective way as the initial stages of the project planning iterates for a better understanding of the user requirements. Using this strategy a prototype is first developed and using user feedback, the required changes are carried out until the real application is released. This would help to integrate OPAS in SUMS in a more effective way. However, the main constraints of doing the project was the deadlines. Several factors such as the programming learning curve using Struts and Hibernate contributed to delay the starting of OPAS development. Therefore, the applications could not be finished on time.

7.2 Designing and Implementing OPAS

As discussed in chapter 5, OPAS design was split up into three parts. The presentation layer design took most of the time as several web pages needed to be developed. Then, the data and business layer were designed. Several Wireframes were created in order to implement the presentation layers in a more effective way. Also, use case diagrams, class diagrams, and entity-relationship diagrams were produced for a better understanding of the data layer implementation. Being familiar with the SUMS architecture was a requirement in order to develop OPAS properly. This requirement was not fully achieved as many methods that SUMS used to process the data as well as to manage errors were not used in OPAS. Hence, the integration of OPAS in SUMS will be much more difficult to carry out.

Before starting to implement OPAS, it was necessary to find out more about of the technology being used. Although my lack of experience using Apache Struts and Hibernate delayed the start of the application development, contributed to develop the application much faster as most of the problems found while implementing the application were already solved before.

Instead of spending most of the time designing and implementing the presentation layer, it would have been better to have focused the work on trying to design and implement the front controller and business delegate pattern in a more effective way. It has more importance the business layer rather than the presentation layer as when OPAS will be integrated in SUMS the OPAS user interface will be different.

While implementing OPAS, the model had to be redesigned. Some unnecessary classes such as supervisor class were created. SUMS provides a class called Marker_Capacity that manages the information related to each supervisor. The redesigned was carried out after most of the functions were developed. Therefore, most of the actions classes, helper methods and relationship between classes were revised spending a lot of time doing this task.

The most important function that OPAS must provide is to allow unit coordinators to allocate projects to students. It is the last step in the project allocating process as explained in chapter 1. The system implementation was started developing the project ideas submission system and external contacts functions. Then, student and supervisors functions were implemented. However, unit coordinator functions were not implemented. External contacts functions implementation should have been developed at the final stage of OPAS implementation. Therefore, the process could be completed as external contacts functions have much less importance than the rest of user functions. The rest of the user functions were completely implemented. Project ideas can be submitted. Students can choice their projects, and supervisors can submit their preferred students. These functions provide OPAS a basic functionality.

OPAS testing was carried out taking into account the user point of view. Functional requirements were tested using the black box technique in order to know if the system behaviour was as required according to the user action. The main OPAS functions implemented were revised by the client. However, a few meetings with the client were arranged in order to test the application functionality. More user feedbacks could have avoided misunderstandings about the user requirements and could have improved the application quality.

7.3 Learning Achievements

This project enhanced the developer's knowledge on the following list of topics:

- Developing web applications using J2EE platform and the MVC Model 2 architecture.
- Using web based application frameworks such as Apache Struts and Hibernate.
- Using of the JSP standard tag library contributed to learn how to design web application layers without using scriptlets.
- Using debugging tools that NetBeans provides has contributed to get familiar with this kind of tools and to use them for testing web applications while being developed.

Broadly speaking, this project has greatly contributed to learn how to apply object oriented techniques to develop Java web applications using the Model-View-Controller architecture.

7.4 Project Results

The aim of this project was to develop an online project allocating system fully compatible with SUMS as it needs to be integrated properly with the rest of sub-systems. Although the application does not contain all the user requirements established in Appendix B, it provides a good level of functionality, allowing users to

execute a wide range of actions. The main functions implemented in OPAS can be integrated in SUMS without having a lot of problems as OPAS was developed according to the required technology. OPAS can greatly contribute to SUMS development. Furthermore, some parts of OPAS such as the presentation layer can be a reference for developing the rest of SUMS sub-systems as it was designed and developed using advance web usability principles improving the interaction between the system and users.

Although OPAS was developed for the School of Computing, it could be applied to the rest of departments in the University of Portsmouth in order to manage the project allocation process for every student in the university. The new system would manage more information and would handle more user requests. Therefore, more powerful servers will be needed to run SUMS. However, no many changes would be carried out in OPAS in order to be adapted to the new environment.

REFERENCES

- *Apache Struts* (2006). Retrieved August 15, 2006, from Apache Software Fundation Web site: http://struts.apache.org/
- Bakharia, A. (2002). *Java Servlet Pages Fast & Easy Development*. Prima Publishing.
- Bauer, C., King, G. (2005). *Hibernate in Action*. Manning Publications.
- Brinck, T., Gergle, D., Word, S.D. (2002). *Designing web sites that work, Usability for the web*. San Francisco: Morgan Kaufmann Publishers.
- Copeland, L. (2003). *A Practitioner's Guide to SoftwareTest Design*. Artech House.
- Deitel, H.M., Deitel, P.J. (2005). *Java, How to Program, Sixth Edition*. Prentice Hall.
- Geary, D.M. (2003). *Core JSTL, Mastering the JSP Standard Tag Library*. Prentice Hall.
- Gulzar, N. (2003). *Practical J2EE Application Architecture*. McGraw-Hill Osborne.
- Hall, M., Brown, L. (2004). *Core Servlets and JavaServer Pages, Volume 1. Core Technologies*, 2nd Edition. Santa Clara: Prentice Hall.
- Herbert, S. (2002). *Java2: The Complete Reference, Fifth Edition*. McGraw-Hill Professional.
- Hibernate (2006). *Hibernate Reference Documentation Version 3.1.3.* Retrieved August 19, 2006, from Jboss Website: http://www.hibernate.org
- Husted, T. Dumoulin, C., Franciscus, G., Winterfeldt, D. (2003). *Struts in Action, Building Web Applications with the Leading Java framework.* Mannning Publications Co.
- IEEE/ANSI. (1998). *IEEE Std 830-1998, Recommended Practice for Software Requirements Specifications*.
- Jasnowski, M. (n.d). *Presistence Frameworks*. Retrieved August 19, 2006, from JDJ Internet Magazine Web site: http://java.sys-con.com/read/36150.htm
- MySQL. (2006). *Top Ten Reasons to Use MySQL.* Retrieved August 27, 2006, from MySQL Web page: http://www.mysql.com/why-mysql/toptenreasons.html

- Polgar, J., Bram, R.M., Polgar, T. (2005). *Building and Managing Enterprise-Wide Portals*. Idea Group Publishing.

- Powell, S.J. (2005). *An online Project Marking system for the University of Portsmouth*. School of Computing, University of Portsmouth.

- Stearns, J., Chinnici, R. (2006). *Update: An Introduction to the Java EE 5 Platform*. Retrieved August 18, 2006, from Sun Developer Network Web site: http://java.sun.com/developer/technicalArticles/J2EE/intro_ee5/

- Sun Developer Network (2006-a). Core J2EE patterns – Front Controller. Retrieved August 25, 2006, from Sun Microsystems Web site: http://java.sun.com/blueprints/corej2eepatterns/Patterns/FrontController.html

- Sun Developer Network (2006-b). Core J2EE patterns – Business Delegate. Retrieved August 25, 2006, from Sun Microsystems Web site: http://java.sun.com/blueprints/corej2eepatterns/Patterns/BusinessDelegate.html

- Vivek, C., Li, S., Jones, R., Eaves, J., Bell, J.T. (2005). *Beginning Java Server Pages*. John Wiley & Sons.

- Wahli, U., Norguet, J., Andersen, J., Hargrove, N., Meser, M. (2003). *WebSphere Version 5. Application Development Handbook*. IBM Redbooks.

- Walnes, J., Abrahamian, A., Cannon-Brookes, M., Lightbody, P. (2004). *Java Open Source Programming: With Xdoclet, Junit, WebWork, Hibernate*. John Wiley & Sons.

- Wikipedia (2006). *Object Database*. Retrieved August 19, 2006, from Wikipedia Web site: http://en.wikipedia.org/wiki/Object_database

- Woojong, S., Barnes S.J. (2004). *Web Enginnering: Principles and Techniques.* Idea Group Publishing.

APPENDIX A

Project Specification

University of Portsmouth

Computing and Multimedia Postgraduate Programme

MSc in Internet Systems Development

Project Specification

Javi Ruiz

Project Specification

1 Basic details

Student name:	Javi Ruiz
Draft project title:	Project Allocation Sub-system
Course and year:	Msc Internet Systems Development
Client organisation:	University of Portsmouth (Faculty of Technology)
Client contact name:	Dr. Jim Briggs
Project supervisor:	Dr. Jim Briggs

2 Outline of the project environment

A new computing system called SUMS (Student and Unit Management System – Son of Pums) is being implemented in the faculty of Technology. PUMS is the Project Unit Management System which manages the information related to student's project. Project ideas submission, supervisors, moderators, projects allocations as well as tracking student progress are some tasks carried out by this system. A new technology is being used in order to replace the old system by the new one. SUMS has five distinct sub-systems that need to be fully compatible between them. Some of them have to be entirely developed . One of these is the Project Allocation Sub-system which manage the allocation of student projects. Hence, this sub-system need to be designed and developed in order to be fully compatible with SUMS.

3 The problem to be solved

The computing system (PUMS) which is currently being used in the Faculty of Technology in order to manage the project allocations to students was developed using PERL scripting language. Also it uses ASCII files in order to store the information generated by this system. No DBMS is being used in order to manage this data.

3.1 Aims

The main aim of this project is to develop a computing system fully compatible with SUMS, which allows students to choose projects ideas for their dissertation topic. Also, the application will help staff members to allocate these projects to students according to the student’s options and staff's criteria. A wide range of reports will allow staff members to have some useful information about the project allocation process.

3.2 Objectives

The main objectives of this project can be summarized as follows:

- Establishing client requirements.
- Producing project requirements documents.
- Establish technical background.
- Designing the application and produce the required documents. Architecture diagrams, UML diagrams, story boards.
- Implementing and testing the application.
- Carrying out the user evaluation.
- Producing technical documents (Java docs, data dictionary).
- Performing integration testing.
- Delivering the web application.

3.3 Constraints

In order to integrate this application into SUMS properly, the project allocation sub-system will be developed using J2EE technology. Also the use of Apache Struts is requested to be used in order to develop the application using a MVC architecture. Hibernate is also required for building the application. The use of both technologies is compulsory in order to be the Project Allocation Sub-system fully compatible with SUMS. The application will be developed using MySQL as a DBMS. However, the system will have to be integrated into SUMS, which uses Oracle Database.

4 Breakdown of tasks

It is necessary to carry out an analysis of the current system (PUMS) in order to understand its architecture and how it was developed. After that, it is important to identify those areas that can be improved using the new system. The understanding of the business logic will help to develop the application in a more effective way. Also, an analysis of the new system (SUMS) will be necessary in order to adapt the new application into the new system without any problem.
For those skills not yet acquired such as the use of Apache Struts and Hibernate further work will be undertaken in order to know how to integrate these technologies all together using J2EE. The required database tables for the application will be designed in order to provide an efficient way to organize the data. Also, the design of new user interfaces would improve the application performance.
The application needs to be developed taking into account that it has to be fully compatible with SUMS.

5 Project deliverables

After finishing the project, a web application will be ready for allocating projects to students. Some useful documents such as Java docs and data dictionary will be produced in order to help other developers if any changes need to be done. A user guide will be also produced with the required instructions for the correct use of the application.

6 Requirements

The project requirements will be obtained from interviews with Dr. Jim Briggs. Besides, the understanding of the project allocation process will be very useful in order to establish the basic conditions to develop the application. After that, these requirements will be documented according to the standard ANSI/IEEE Std 830-1998, which specifies the requirements of software to be developed. The preliminary project requirements are summarised as follows:

6.1 Main features of the application

- Submission of project ideas choices carried out by students.
- Project allocations carried out by staff members.
- Sending email notifications to students and staff members with relevant information about the project allocation process.
- Providing a wide range of reports to staff members with useful information about the project allocation process.
- Providing a facility for advising staff members of those students who have not chosen their project choices before the deadline.
- Establishing different levels of access to the system.

6.2 Assumptions and dependencies

- Password authentication will be necessary in order to access into the system.
- Errors occurred in the application will be logged.
- Any changes made by staff members will be logged in order control the modifications of the data.

7 Legal, ethical, professional, social issues

No legal, ethical, or social issues will impose any constraint on the project development. Nevertheless, there is a professional issue that must be taken into account. This program is indented to be used by the university staff, not to be only a prototype. Therefore, it is necessary to develop the application according the user requirements and with the required security components in order to avoid the misuse of the application. Furthermore, the application will be tested by users. The methods that are going to be used will not have any legal/ethical/professional/social implication that could affect the testing process.

8 Facilities and resources

An old version of this system is being used in the University. Nowadays, the University has all the necessary computing equipment to run this application properly. Hence, no further resources will be necessary.
In order to develop the application, according to the technology being used in SUMS, is necessary to acquire some software. NetBeans 5.0 will be used for developing the Web application. Macromedia Dreamweaver will be necessary for designing and building the user interfaces. Furthermore, MySQL 5.0 will be required for building and managing the database.

9 Project plan

See Gantt chart that has been attached to the project specification document. This is a preliminary diagram that can be changed according to the system requirements.

10 Project mode

Registration mode	**Full Time**	Part Time
Project mode	**Full Time**	Part Time
Planned submission deadline	September 2006	

11. Signatures

	Signature:	Date:
Student	Javi Ruiz	
Client	Jim Briggs	
Project supervisor	Jim Briggs	

APPENDIX B

User Requirements

1. Introduction

1.1 Purpose

The main purpose of this SRS document is to give a detailed overview of the software product with its main goals and parameters. In this document, user interfaces, hardware and user requirements as well as assumptions and dependencies of the application are described. Furthermore, it defines the functionality of the application from a client's point of view. The client of this software product is Dr. Jim Briggs from the School of Computing at the University of Portsmouth. All the requirements presented in this document have been established according to the client's criteria and the logical thinking of the developer.

1.2 Scope

A new computing system called SUMS is being implemented in the faculty of Technology. PUMS manages the information related to student's project. Project ideas submission, supervisors, moderators, projects allocations as well as tracking student progress are some tasks carried out by this system. A new technology is being used in order to replace the old system by the new one. SUMS has five distinct sub-systems that need to be fully compatible between them. Some of them have to be entirely developed. One of these is OPAS, which manage the allocation of student projects. Hence, this sub-system need to be designed and developed in order to be fully compatible with SUMS.

The computing system (PUMS) which is currently being used in the Faculty of Technology in order to manage the project allocations to students was developed using PERL scripting language. Also it uses ASCII files in order to store the information generated by this system. No DBMS is being used in order to manage this data.

The main aim of this project is to develop a computing system fully compatible with SUMS, which allows students to choose projects ideas for their dissertation topic. Also, clients should be able to submit these project ideas. The application will help staff members to allocate these projects to students according to the student’s options and staff's criteria. A wide range of reports will allow staff members to have some useful information about the project allocation process.

1.3 Definitions, Acronyms, and Abbreviations

ANSI – American National Standards Institute
ASCII – American Standard Code for Information Interchange
DBMS – Database Management System
HTML - Hypertext Markup Language
HTTP - Hypertext Transfer Protocol
IEEE – the Institute of Electrical and electronic Engineers
JSP - Java Server Pages
OPAS – Online Project Allocation System
PERL – Practical Extraction and Report Language
PUMS – Project Unit Management System

SRS – Software Requirements Specification
SUMS – Student Unit Management System

1.4 References

- IEEE/ANSI. (1998). IEEE Std 830-1998, Recommended Practice for Software Requirements Specifications.

1.5 Overview of the Document

This document has been organized in the following sections:
Section 1 : General overview about the software requirement specification document with references and acronyms that are used.
Section 2 : A view of the product according to the client's point of view. How users interact with the application
Section 3 : A more detailed view of the product according to the developer's point of view by discussing the input and output data as well as technical information of the product.

2. Overall Description
2.1 Product Perspective

2.1.1 System Interfaces

As commented before, the software product will be working in conjunction with other sub-systems integrated to SUMS. The following diagram (figure 1) represents the sub-systems existing in SUMS.

Figure 1. SUMS Sub-systems

It has five distinct sub-systems, which are being developing by other students. Registration Sub-System is partially developed and not in use. Project Marking Sub-System is fully developed and in use. The others are being developed at the moment by other students. Hence, it is necessary to develop the project allocation sub-system according to the technology used in SUMS. J2EE/Java using Struts and Hibernate will be compulsory in order to integrate the project allocation sub-system into SUMS properly. Furthermore, the DBMS that is used in SUMS is Oracle. Although the application is going to be developed using MySQL, it will have to be running properly using Oracle database. So, it is necessary to investigate how to integrate this sub-system into SUMS in a proper way.

2.1.2 User Interfaces

The project allocation sub-system is a web-page application. So, the user interfaces will be developed following basic usability principles and techniques oriented to facilitate the navigation among the pages. A good layout design will be used for distributing the information in a proper way. So, the user will be able to identify the sections easily as well as the different options available. The user interface will be developed for being used in screens with a resolution of 1024x 768 pixels. Nevertheless, the user will be capable of running the application using smaller screen resolution such as 640x480 pixels, without having any problem.

It ill be necessary to implement a good error management system. When an error occurs, the page will be redirected to an error page, displaying relevant information about what caused the defective functioning. Besides, when the input data is not correct, the user will be able to identify easily the wrong data. This will be possible, displaying error messages next to the form elements.

The system will provide a wide range of reports. Although, these reports will be available using the web browser, users will have the chance of producing them in .pdf format. Hence, users will be able to save or send them by email.

2.1.3 Hardware Interfaces

Like all the web-based applications, in order to access to the system, a Internet connection will be required. The hardware requirements for running web-based applications are minimal. Modern computers and all those currently being used in the University of Portsmouth have the required hardware for running a web browser.

The University has powerful servers that are used for managing the application and providing services to the clients. This system will be hosted in these servers.

2.1.4 Software Interfaces

The only requirement needed for the clients Operating System is the ability to run a web browser which supports the HTML version 4.0 or higher. The HTML version 3.2 can be supported but it can have some problems when CSS styles are used.
In order to manage the user requests in the server-side of the application, Apache Web Server 2.0 will be required. Furthermore, Apache Tomcat 5.5 will be needed for implementing the servlets and the JSP specifications. Oracle version 10 will be needed for managing the application data.

2.1.5 Communication Interfaces

HTTP protocol will be used for sending the information between the server and the client. For those communications that require security, HTTP(s) protocol will be used for encrypting the data.

2.1.6 Memory Constraints

OPAS will not require more memory requirements in order to be run.

2.1.7 Operations

User Authentication will be placed in Apache Tomcat Server. At the moment, the user authentication in SUMS is carried out using a database. However, plain files are used for authenticating staff members when accessing to the system. The OPAS does not have to manage authentications in order to access to the system. User interfaces as well as authentications methods will not be developed in this part of the system. Security and backup policies in SUMS are carried out by the Department of Technology in the University of Portsmouth.

2.1.8 Site Adaptation Requirements

It is not required to do any modification in the servers that are used in the University of Portsmouth for running the web application.

2.2 Overview of Functional Requirements

The overview of the functional requirements is the following

- Selection of project ideas choices carried out by students.
- Project allocations carried out by staff members.
- Sending email notifications to students and staff members with relevant information about the project allocation process.
- Allowing supervisors to select preferred students for their projects.
- Providing a wide range of reports to staff members with useful information about the
  project allocation process.
- Providing a facility for advising staff members of those students who have not chosen their project choices before the deadline.
- Establishing different levels of access to the system.

As discussed with the client in the last meeting, this system will also manage the project ideas submission. The following list summarize the functional requirements for this task.

- Managing information about the clients such as personal and security information.
- Creating a new user.
- Authentication system in order to allow clients to access to their account and submit project ideas.
- Enable clients to submit, delete, and amend project ideas.
- List of project ideas submitted by the client who starts a session.
- Searching for project ideas using a wide range of criteria.
- Staff member can approve the project ideas already submitted.
- After project ideas are approved by staff members, the system will send an email to the appropriate clients in order to notify them about this.

2.3 User characteristics

There are three types of users who will access to the system.

2.3.1 Students

Students at Portsmouth University will be interested in accessing to the system in order to submit their project choices. These users are undergraduates and postgraduates students. They are fully to be competent in using web applications. Although it is not required to have experience in using web based applications students from the Faculty of Technology are easier to get used to using web applications than those who do not have a technological background. This system is going to be used for students from the Faculty of Technology. No special training will be required in order to teach them how to use the system.

2.3.2 Staff Members

This kind of users are supervisors and unit coordinators who need to access to the system in order to get information about the project allocation process. They have experience in using web based applications. A system is being currently used in the University of Portsmouth. Although the user interfaces and functions can be different in the new system, no special training will be required.

2.3.3 Clients

Any person who is interested in having a specific computing system for being implemented in a company will be able to submit project ideas. Usually, this kind of users does not have any experience in using web based applications. Although, they will not need special training, user interfaces for submitting project ideas should be designed friendly in order to facilitate the understanding of the application as much as possible.

2.4 Constraints

In order to integrate this application into SUMS properly, the project allocation sub-system will be developed using J2EE technology. Also the use of Apache Struts is requested to be used in order to develop the application using a MVC architecture. Hibernate is also required for building the application. The use of both technologies is compulsory in order to be the Project Allocation Sub-system fully compatible with SUMS as commented in the system interfaces section.

The system will be finished in September 2006 as it is part of the Master's final project of the student who is developing this system.

2.5 Assumptions and Dependencies

There are some dependencies and assumptions that have to be taken into account in order to develop the system properly. Usernames and passwords will be necessary for accessing to the system. All the users will be authenticated as valid users if they want to use the services provided by this system. Furthermore, the errors occurred in the

system will be logged as this help to discover and debug them in a more effective way. The modifications of the data made by staff members will be also logged in order to track all the changes made in the system.

2.6 Apportioning of Requirements

The application design need to be understand by other developers in order to be integrated to SUMS properly.

3 Specific Requirements

This section describes the system functional requirements with a high level of detail. The system requirements are split according to the group users existing in the system as described in the user characteristics section.

3.1 Students

3.1.1 Searching for Project Ideas (Also for staff members and clients)

Purpose	In order to select project ideas, student should be able to search for project ideas based on several parameters.
Inputs	The project ideas searching can be carried out according the following criteria; Project Idea title, Organisation Name, Objectives, Contact Name, date of submission, Project Idea Status. Retrieval of information about the project ideas is based on the project idea's ID. Only one textfield will be required to insert the keyword.
Processing	The information based in the criteria chosen is retrieved from the database
Outputs	A list of project ideas showing the title, organisation name, status, and submission data should be displayed

3.1.1 Submitting Project Choices

Purpose	Students should be able to submit their project choices from a list of Project Ideas.
Inputs	Undergraduate Students should be able to submit a maximum of seven project choices. Postgraduate students should be able to submit a maximum of three project choices (This number can be changed by the administrator at any time). After students search for project ideas, the project choices submission can be carried out. Only approved project ideas should be allowed to be chosen. The use of check boxes will facilitate this task to the students
Processing	After the student submit their choices, the system should check if the number of options are correct. When the user complete the project submission properly, the system should record the data into the database. An email will be sent to the student in order to confirm the choices selection.
Outputs	If the options are not correct, a message indicating the problem should be displayed. After the student submit the choices properly, a message confirming the choices should be displayed. Furthermore, a list of the project ideas submitted should be displayed as well.

3.1.1 Amending Project Choices (Also for unit coordinator)

Purpose	Students should be able to amend the project choices already selected.
Inputs	If no project has been allocated to the student yet, students could amend their project choices . A list of the projects already chosen should be displayed in the screen. After this, the user should be able to select any project idea and amend it using checkboxes.
Processing	The system should check if the project idea chosen has been allocated to another student. This information should be retrieved from the database.
Outputs	If the process is correct, a message should be displayed confirming the data updating. Furthermore, a list of project choices with the information amended should be displayed as well.

In addition to this requirements, students should be able to has the same functionality as the clients. Hence, they should be able to submit project ideas as well as amend or delete them.

3.2 Supervisors

3.2.1 Searching for Students (Also Unit Coordinator)

Purpose	In order to select preferred students and view students' project choices, supervisors should be able to search for students.
Inputs	The student searching is carried out according the name, hemis number, unit or course. Also, the searching should be carried out according to the project choices as well as the project allocated.
Processing	The information is retrieved from the database.
Outputs	A list of students showing their name and their course name is displayed.

3.2.2 Viewing all Students

Purpose	Supervisors should be able to see a list of all students associated with them.
Inputs	Retrieval of the information is carried out by selecting a link on the menu. The supervisor's ID is used for retrieving this information.
Processing	The information is retrieving from the database by using the supervisor's ID
Outputs	A list of students with relevant information about them is displayed

3.2.3 Viewing those students who have been chosen the supervisor's project idea

Purpose	Supervisors should be able to see a list of students who selected their project ideas for the project choices
Inputs	Retrieval of the information is carried out by selecting a link on the menu. The supervisor's ID is used for retrieving this information.
Processing	The information is retrieving from the database by using the supervisor's ID
Outputs	A list of students is displayed. Furthermore, supervisors can add any of these students to their list of preferred students.

3.2.4 Selecting Preferred Students

Purpose	Supervisors should be able to select their preferred students for their projects.
Inputs	After a list of students is displayed according to the criteria chosen, a student can be selected and added to one or more of the supervisor projects. A maximum of three students should be allowed to be selected.
Processing	After the student selection is done, the system record the data into the database.
Outputs	A list of the projects with the preferred students is displayed.

3.2.5 Viewing preferred students

Purpose	After selecting the preferred students, supervisors should be able to see a list of their preferred students. Besides, they should be able to amend this list by adding more students or deleting them.
Inputs	Retrieval of the information is carried out by selecting a link on the menu. The supervisor's ID is used for retrieving this information.
Processing	The information is retrieving from the database by using the supervisor's ID
Outputs	A list of preferred students is displayed. Furthermore, a button is available for each student in order to delete it from the list

3.2.6 Viewing Project Choices

Purpose	Supervisors should be able to see the students' project choices.
Inputs	Supervisors can search for any student according to the criteria used in search for students function. Retrieval of information about the project choices is based on the students' ID.
Processing	The system will retrieve this information from the database.
Outputs	A list of students with their project options is displayed. Also, the user has the chance of creating a .pdf document with this information.

Besides these requirements, supervisors should be able to play the same role as the clients. Hence, they will be able to submit project ideas as well as amend and delete them when appropriate.

3.3 Unit Coordinator

The unit coordinator is responsible of allocation projects to Students. He has also the chance to view the project ideas already chosen by the students.

3.3.1 Allocating Projects to Students

Purpose	Unit coordinators should be able to allocate project to students according to the project choices carried out by the students.
Inputs	The projects are allocated to students, using the project ID. The project can be chosen between the list of project choices provided by each student. Radio buttons is used for making easier the student project allocation. Furthermore, the supervisor should be able to allocate a project not selected from the list provided by the student. A textbox is used for typing the code and name of the project.
Processing	The project is recorded into the database using the project ID.
Outputs	A message confirming that the project allocation has been carried out properly is displayed.

3.3.2 Viewing Students without Project Choices

Purpose	Unit coordinators should be able to know the students who have not submitted the project choices yet.
Inputs	A html link is available in order to display a list of those students without project selection.
Processing	This information is retrieved from the database
Outputs	A list of students is available in .pdf format as well as in a table in the web page.

3.3.3 Viewing Project Choices according to student preferences

Purpose	Unit coordinators should be able to view those projects ideas which have been selected by too many users.
Inputs	A html link is available in order to display the list of project ideas.
Processing	This information is retrieved from the database.
Outputs	A list of project ideas with the student names who have been chosen them is displayed. This list is ordered by the number of students who have chosen the project idea. Furthermore, this list will be available in .pdf format as well as in a web page.

3.3.4 Changing Project Idea Status

Purpose	After the project ideas are submitted by the clients, staff members should be able to change the project idea status in order to approve, allocate, amend, or withdraw the project idea.
Inputs	Staff member can search for any project idea. Using a drop-list, the project idea status can be changed according to the criteria of the staff member. A project idea can not be withdrawn if it was already allocated to any student.
Processing	After the project idea status is selected and the user press submit, the system record the data into the database.
Outputs	A message conforming the change is displayed.

3.4 Clients

3.4.1 Creating a New User

Purpose	Clients should be able to access to the system creating a new user if it is the first time the client accesses to the system.
Inputs	The client must provide the following information in order to create an account; name, address, postcode, contact name, outline of what the organisation does, telephone, email address, username, and password (re-enter password as well).
Processing	The system checks if the required information has been provided. Besides, the information provided is also checked in order to know if it matches with the required format. Finally, the system checks the name and username in order to know if this information match with other information provided by other user.
Outputs	After submitting the user details properly, the system displays a message confirming the creation of a new user.

3.4.2 Amending Personal Information

Purpose	Clients can amend their personal details as well as the security information required to access to the system (username and password).
Inputs	Clients must type the new information where appropriate.
Processing	The information is checked in the same way as creating a new user.
Outputs	After submitting the new information, a message confirming the data updating is displayed.

3.4.3 Submitting Project Ideas

Purpose	Clients should be able to submit their project ideas.
Inputs	The clients must provide the following information; title, aims and objectives, academic question to be answered, anticipated deliverables.
Processing	The system checks if the user provides the required information with the correct format.
Outputs	After submitting the project idea, a message confirming the project idea submission is displayed.

3.4.4 Amending Project Ideas

Purpose	After the project idea is submitted, the client should be able to update any information related to the project idea chosen.
Inputs	The user can search their project ideas. A list of project ideas is displayed. After the required project idea is edited, the user can change any information related to that project idea. The project idea details can not be changed if that project has been allocated to any student.
Processing	The information provided is checked in order to know if it has been submitted properly.

Purpose	After the project idea is submitted, the client should be able to update any information related to the project idea chosen.
Outputs	A message confirming the data updating is displayed.

3.4.5 Logging to the System (Also all members)

Purpose	In order to submit project ideas, users should log to the system.
Inputs	Username and password are required using two textfields in the main page of the application.
Processing	After the user press submit the information, the system checks if the username and password match with any user already registered to the system.
Outputs	If the username or password is incorrect, a message informing the user of this is displayed. In the other hand, if the username and password are correct, the user is logged to the system and a set of different options available are displayed.

4 Change History

Date	***Version***	***Details***
05-06-2006	1.0	First version of software requirements specification.
15-06-2006	1.1	The inclusion of project ideas submission system.
28-06-2006	1.2	Changes in supervisor requirements; Listing historical students and students who chose the supervisor's project idea. Changes in the unit coordinator requirements; Changing the student's project choices. Supervisors and students have the same functions as the clients

APPENDIX C

Gantt Chart

APPENDIX D

Wireframes

1. External Contacts

1.1 Homepage

1.2 New External Contact

1.3 Your Project Ideas

2. Students

2.1 Homepage

2.2 Select Projects

OPAS – Online Project Allocation Sub-System

Make Your Choices | Back to the Index

Project Code | Project Title | Organisation

Objectives

Project Code | Project Title | Organisation

Objectives

Project Code | Project Title | Organisation

Objectives

Submit Project Choices

2.3 Project choices

OPAS – Online Project Allocation Sub-System

Your Project Choices | Back to the Menu

Project Allocated: Project Code and Name

1 Project Code | Project Title | Organisation | 1 | Delete

Objectives

2 Project Code | Project Title | Organisation | 2 | Delete

Objectives

3 Project Code | Project Title | Organisation | 3 | Delete

Objectives

4 Project Code | Project Title | Organisation | 4 | Delete

Objectives

Update Changes

3. Supervisors

3.1 My Students

OPAS – Online Project Allocation Sub-System

Options

Welcome Supervisor xxx

Your Students

Your Preferred Students

Your Project Ideas

Logout

Search Project Idea

Keyword

Status Proj. Status

Last

Search

Search Student

Keyword

Course Sel. Course

Search

My Students

Student Code	Student Name	Course
Student Code	Student Name	Course
Student Code	Student Name	Course

3.2 Submit Preferred Students

OPAS – Online Project Allocation Sub-System

Options

Welcome Supervisor xxx

My Students

Your Preferred Students

Your Project Ideas

Logout

Search Project Idea

Keyword

Status Proj. Status

Last

Search

Search Student

Keyword

Course Sel. Course

Search

Students Searching

Student ID	Name & Surname	Course	
Student ID	Name & Surname	Course	
Student ID	Name & Surname	Course	
Student ID	Name & Surname	Course	

Submit Preferred Students

3.3 Preferred Students

OPAS – Online Project Allocation Sub-System

Preferred Students | Back to the Main Menu

Student Code | Student Name | Course | Choose a Project... | Delete

Project Choices: Project Code Project Name
Project Choices: Project Code Project Name
Project Choices: Project Code Project Name

Student Code | Student Name | Course | Choose a Project... | Delete

Project Choices: Project Code Project Name
Project Choices: Project Code Project Name
Project Choices: Project Code Project Name

Student Code | Student Name | Course | Choose a Project... | Delete

Project Choices: Project Code Project Name
Project Choices: Project Code Project Name
Project Choices: Project Code Project Name

Submit Changes

4. Unit Coordinators

4.1 Allocate Projects

OPAS – Online Project Allocation Sub-System

Allocate Project | Back to my Students

Student Number :
Name :
Course :

List of Project Choices

1 Project Code Project Title Organisation Supervisor

2 Project Code Project Title Organisation Supervisor

3 Project Code Project Title Organisation Supervisor

Aditional Project

Project ID: Project Name :

Submit

4.2 My Students

OPAS – Online Project Allocation Sub-System

My Students | Back to the Main Menu

Student Code	Student Name	Course	Edit Choices
Project Choices:	Project Code	Project Name	Allocate Project
Project Choices:	Project Code	Project Name	
Project Choices:	Project Code	Project Name	

Student Code	Student Name	Course	Edit Choices
Project Choices:	Project Code	Project Name	Allocate Project
Project Choices:	Project Code	Project Name	
Project Choices:	Project Code	Project Name	

Student Code	Student Name	Course	Edit Choices
Project Choices:	Project Code	Project Name	Allocate Project
Project Choices:	Project Code	Project Name	
Project Choices:	Project Code	Project Name	

4. Students without Project Choices

OPAS – Online Project Allocation Sub-System

Students Without Project Choices | Back to the Main Menu

Student Code	Student Name	Course
Student Code	Student Name	Course
Student Code	Student Name	Course
Student Code	Student Name	Course
Student Code	Student Name	Course
Student Code	Student Name	Course

Send Them an email

APPENDIX E

OPAS Presentation

OPAS

Online Project Allocating System

Javi Ruiz

1

Summary

- The client
- The problem to be solved
- Project allocation process
- Technologies implemented
- Objectives achieved

2

The client

- Dr. Jim Briggs
- Principal Lecturer in the School of Computing (University of Portsmouth)
- PUMS/SUMS system developer

3

The problem to be solved

- PUMS (Project Unit Management System)
  - Manages information related to student's projects
  - Developed using Perl scripting language
  - Makes use of ASCII files
  - Technology needs to be replaced
  - A new system is being developed (SUMS – Student Unit Management System)

4

The problem to be solved

- **SUMS architecture**

5

Project allocation process

- **Required steps for allocating projects**

6

Technologies

- **JavaEE 5 platform**
  - Formerly known as J2EE or Java 2 platform
  - Programming platform for developing multi-tier architecture Java applications
  - Support from the open source community
  - Main technologies
    - Servlets
    - Java Beans
    - JSP
    - XML
    - JDBC

7

Technologies

- **Web-based application frameworks**
  - Help to develop web applications in a more effective way
  - Application maintenance much easier to carried out
  - Provide a wide range of code libraries
  - Open source Frameworks
    - Apache Struts
    - Hibernate

8

Technologies

- **MySQL database**
  - The most popular open source database
  - Very easy to manage
  - Used by large enterprises
  - MySQL Tools
    - MySQL Migration Toolkit
    - MySQL Administrator
    - MySQL Query Browser

9

Objectives achieved

- **Functions implemented in OPAS**
  - User login interface
  - Search project ideas
  - Create new external contact
  - Create project ideas
  - Choose projects
  - **Indicate Preferred Students**

22

Online Project Allocation System - Index Page - Mozilla Firefox
Archivo Editar Ver Ir Marcadores Herramientas Ayuda
http://localhost:8084/ProjectApplication/
Ir
Haz click aquí! Hotmail gratuito Personalizar vínculos Windows Media Windows
Online Project Allocation System
User Login
Username:
Password:
Remember
New user
Submit
Options
Contact Us
Search Project Ideas
Keyword:
Status:
Any Status
Last:
1
Days
Submit
Terminado

Online Project Allocation System - Index Page - Mozilla Firefox
Archivo Editar Ver Ir Marcadores Herramientas Ayuda
http://localhost:8084/ProjectApplication/searchIdeas.do;jsessionid=8EBD3778FEDD
Ir
Haz click aquí! Hotmail gratuito Personalizar vínculos Windows Media Windows
Online Project Allocation System
User Login
Username:
Password:
Remember
New user
Submit
Options
Contact Us
Search Project Ideas
Keyword:
Status:
Pending Appr
Last:
1
Days
Submit
Project Ideas Results
Idea 1 Javi Ruiz P
Idea 2 Javi Ruiz P
Idea 3 Javi Ruiz P
Idea jim 1 Jim P
Idea jim 2 Jim P
Idea jim 3 Jim P
http://localhost:8084/ProjectApplication/displayIdeaInfo.do?id=1

OPAS - Project Idea Information - Mozilla Firefox
Archivo Editar Ver Ir Marcadores Herramientas Ayuda
http://localhost:8084/ProjectApplication/displayIdeaInfo.do?id=1
Haz click aquí! Hotmail gratuito Personalizar vínculos Windows Media Windows
Online Project Allocation System
Go back
The Idea
Project Title: Idea 1
Aims and Objectives: Aims idea 1
Academix Question: question idea1
Anticipated Deliverables: deliverables idea1
Student: student idea1
Status: P
Date Submitted: date
The Organization
Contact Name: Javi Ruiz
Postal Address: Maluquer 4 A 2 - 46007
Outline of what the Organization does: Nothing Important
Contact Name: Javi
Telephone Number: 963805435
email Address: javichu7@gmail.com
Terminado

OPAS - Contact Form - Mozilla Firefox
Archivo Editar Ver Ir Marcadores Herramientas Ayuda
http://localhost:8084/ProjectApplication/editContactDetails.do
Haz click aquí! Hotmail gratuito Personalizar vínculos Windows Media Windows
Online Project Allocation System
Go back
PROJECT IDEA SUBMISSION FORM
Name: Javi Ruiz
Address: Maluquer 4 A 2
Postcode: 46007
Contact: Javi
Outline of what your organisation does: Nothing Important
Telephone: 963805435
Email Address: javi@gmail.com
Username: javi
Password: ****
Re-enter Password:
Submit the Information
Reset
Terminado

OPAS - Project Idea Form - Mozilla Firefox
Archivo Editar Ver Ir Marcadores Herramientas Ayuda
http://localhost:8084/ProjectApplication/searchContactIdeas.do
Ir
Haz click aquí! Hotmail gratuito Personalizar vínculos Windows Media Windows
Online Project Allocation System
Go back
YOUR PROJECT IDEAS
Idea 1 P 2006-09-01 00:00:00.0 Edit Delete
Idea 2 P 2006-09-01 00:00:00.0 Edit Delete
Idea 3 P 2006-09-01 00:00:00.0 Edit Delete
SUBMIT YOUR PROJECT IDEA
Idea Title:
Aims and Objectives:
Academic question to be andwered (blank if unkown):
Anticipated Deliverables:
Terminado

OPAS - Projects Available - Mozilla Firefox
Archivo Editar Ver Ir Marcadores Herramientas Ayuda
http://localhost:8084/ProjectApplication/searchProjectsAvailable.do;jsessionid=19
Ir
Haz click aquí! Hotmail gratuito Personalizar vínculos Windows Media Windows
Online Project Allocation System
Go back
Project Ideas Available
Idea 1 Javi Ruiz P 2006-09-01 00:00:00.0
Idea 2 Javi Ruiz P 2006-09-01 00:00:00.0
Idea 3 Javi Ruiz P 2006-09-01 00:00:00.0
Idea jim 1 Jim P 2006-09-01 00:00:00.0
Idea jim 2 Jim P 2006-09-01 00:00:00.0
Idea jim 3 Jim P 2006-09-01 00:00:00.0
Submit your Choices
Terminado

OPAS - My Project Choices - Mozilla Firefox
Archivo Editar Ver Ir Marcadores Herramientas Ayuda
http://localhost:8084/ProjectApplication/editProjectChoices.do;jsessionid=1237BCAE
Ir
Haz click aquí! Hotmail gratuito Personalizar vínculos Windows Media Windows
Online Project Allocation System
Go back
Your Project Choices
Allocated Project:
No Project has been allocated yet
Make a new Project Choice
1 Idea 1 Javi Ruiz 1 Delete
2 Idea 2 Javi Ruiz 2 Delete
3 Idea 3 Javi Ruiz 3 Delete
Rearrange your Options
Your previous Project Choices
2006-09-01 21:40:23.0
2006-09-01 21:40:16.0
2006-09-01 21:40:08.0
Terminado

Online Project Allocation System - Index Page - Mozilla Firefox
Archivo Editar Ver Ir Marcadores Herramientas Ayuda
http://localhost:8084/ProjectApplication/Index.do
Ir
Haz click aquí! Hotmail gratuito Personalizar vínculos Windows Media Windows
Online Project Allocation System
Options
Welcome null!
Your Students
Your Preferred Students
Your Project Ideas
Logout
Options
Contact Us
Search Project Ideas
Keyword:
Status: Any Status
Last: 1 Days
Submit
Search Students
Keyword:
Course: Any
Submit
Students Results
111111 Name & Surname Msc Internet Systems Dev. Final Year
222222 Name & Surname Msc Internet Systems Dev. Final Year
333333 Name & Surname Msc Internet Systems Dev. Final Year
444444 Name & Surname Msc Internet Systems Dev. Final Year
555555 Name & Surname Msc Internet Systems Dev. Final Year
Submit your Preferred Students
Terminado

Summary

- Problem background
- SUMS architecture
- Project Allocation Process
- Technologies
  - JavaEE 5
  - Web based frameworks (Struts & Hibernate)
  - MySQL Server
- Functions implemented

27