WXES 3182 PROJEK ILMIAH TAHAP AKHIR II

XML WEB SERVICES

FCSIT XML Web Services

BY : WONG KOK CHUAN WEK000119

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Computer Science, Faculty of Computer Science and Information Technology, University of Malaya.

SUPERVISOR : MR ANG TAN FONG

MODERATOR : MR MOHD. NIZAM HJ. AYUB

PARTNER : MR LEE SAN KUAN WEK000026

ABSTRACT

This document describes all the development processes that have been gone through from the very early phase of feasibility study till the end of maintenance phase. First of all this document explains on the overview of the web services, why we need web services and so on. Then it states the objectives of why there are needs for web services to be implemented in FCSIT, the scope of the project, and the project planning.

Next, this document lists out all the literature reviews that I have done such as the ways or techniques for information gathering, the similar systems, the architecture models, the frameworks, the languages, the platforms, the servers such as web servers and database server and the tools that can be used. In this following section, the methodology that is used to implement the system is described in details explaining the overview, the features and why I have chosen to use it.

In the analysis section of this document, the functional requirements are elaborated in a very comprehensive way and coupled up with the UML diagrams. The non-functional requirements explain the constraints that need to be handled to produce the ideal systems. The subsequent section describes the design phase of the system. In the system design, I have broke the system into small independent services so that they are easier to be focused on, developed, tested, debug, maintained, and enhanced.

Next is the implementation phase which covers a brief description on the development environment used, explanations on a few algorithms used and the coding techniques and patterns which I followed. Then, the following chapter explains about the testing phase which I have done. These involved development of test plan, the unit testing and the integration testing.

After that, a lengthy chapter which is devoted for discussion about the project outcomes. Several important events, problems faced and solutions used, pros and cons of the system developed and future enhancements and improvements that can be made are included in this discussion.

ACKNOWLEGDEMENTS

I would like to express my deepest gratitude to my supervisor, Mr Ang Tan Fong for accepting me to be under his supervision and entrusting me to develop the Web Services for the faculty. Also I would like to thank him for his guidance, considerations and always kept us informed on the latest updates in this thesis and for his efforts in providing us with the best facilities during this project development.

I would like to thank my very best moderator, Mr Nizam Hj Ayub for being very thoughtful and considerate. I am very glad that he understood my difficulties and limitations that I faced during the project development as the scope was quite large and the technologies used were very new to all of us.

At first, I thought Mr Ling Teck Chaw was really strict to us but latter discovered it's the other way round. He is very demanding in term of our project but very helpful and always suggesting us with the better ideas on how to implement this project. So, I would like to thank him for all his constructive comments.

Well, I would like to thank my project partner, Mr Lee San Kuan for all his cooperation and help he gave to me. Throughout the project, we have been exchanged ideas and findings on how to design and develop the system. Without him, I think the system will not be as complete as we had proposed it to be.

I would like to thank my family so much for understanding me throughout the development process. They always give me their full support, morally and materially, which makes me feel so much comfortable while doing the project.

Well, I would like to thanks my "bunch of friends", though I am not going to list all of their names as there are too many of them. They have provided us with some valuable ideas and jokes which really release your tension.Without all their help, I am very sure that this project will not be as successful as it is planned and smoothly done. So, I am really grateful to all of them and I would like to say thanks again.

TABLE OF CONTENTS

ABSTRACT ii

ACKNOWLEGDEMENTS iii

TABLE OF CONTENTS iv

LIST OF TABLES ix

LIST OF FIGURES x

1 INTRODUCTION 1

1.1 Project Overview 1

1.1.1 The Roadmap towards Web Services 1

1.1.2 So what's a Web Service? 2

1.1.3 The FCSIT XML Web Services 3

1.1.4 The Faculty's Current ALMS System 5

1.2 Problem Statements and Project Motivation 7

1.3 Project Visions and Objectives 9

1.4 Project Schedule 11

1.5 Project Constraints 14

1.6 Project Scope 16

1.7 Expected Outcome 17

2 LITERACY REVIEW 18

2.1 Information Gathering Techniques 18

2.1.1 The Internet and World Wide Web 18

2.1.2 Reference Books 18

2.1.3 Discussion with the User 18

2.1.4 System Documentation 18

2.1.5 Trial on the Existing System 19

2.1.6 Brainstorming 19
2.2 Similar Systems 20
2.2.1 Microsoft .NET Passport (www.passport.com) 20
2.2.2 Google Web APIs Services (www.google.com/apis) 22
2.2.3 The Faculty's Current ALMS (http://lms.fsktm.um.edu.my) 23
2.2.4 The softTIME attendance system (www.webattendance.com) 24
2.3 Development Technologies 27
2.3.1 The Architecture 27
2.3.2 The Frameworks 31
2.3.3 The Implementation Languages 33
2.3.4 The Operating Systems 37
2.3.5 The Web Servers 39
2.3.6 The Database Management System 41
2.3.7 The Development Tools 44
3 METHODOLOGY 48
3.1 The Object-oriented Methodology (OOM) 48
3.1.1 The Overview 48
3.1.2 The Phases 48
3.1.3 Why OOM? 48
3.1.4 The UML 49
3.2 The Phase Development Software Life Cycle 50
3.2.1 The Overview 50
3.2.2 Why this Approach? 51
4 SYSTEM ANALYSIS 52
4.1 Design Requirements 52

4.1.1 Functional Requirements 52
4.1.2 Non-functional Requirements 63
4.2 System Requirements 65
4.2.1 Hardware Specifications 65
4.2.2 System Platform 66
4.2.3 Web Server 66
4.2.4 Database Server 67
4.2.5 Framework 67
4.2.6 Languages 68
4.2.7 Tools 68
5 SYSTEM DESIGN 69
5.1 Architectural Design 69
5.1.1 The Overview 69
5.1.2 The FCSITPortal 71
5.1.4 The License Management Web Service 72
5.1.3 The Authentication Web Service 73
5.1.5 The Authorization Web Service 74
5.1.4 The HR Manager Web Service 75
5.1.6 The Activity Web Service 76
5.2 Detailed Design 77
5.2.1 The FCSITPortal 77
5.2.3 The License Management Web Service 79
5.2.2 The Authentication Web Service 80
5.2.4 The Authorization Web Service 83
5.2.5 The Activity Web Services 86

5.3 Database Design 89
5.4 Interface Design 92
5.4.1 Graphical User Interface (GUI) Design 92
5.4.2 Application Programming Interface (API) Design 93
6 SYSTEM IMPLEMENTATION AND DEVELOPMENT 104
6.1 Program Structuring 104
6.2 Header Block 109
6.3 Inline Comment 111
6.4 Self Explained Code 112
7 SYSTEM TESTING 114
8 DISCUSSION 117
8.1 Project Outcome 117
8.2 Problems Faced 118
8.2.1 Time Limitation 118
8.2.2 Inexperience 118
8.2.3 Requirement Doubts and Design Ambiguities 119
8.2.4 Programming Bugs and Runtime Errors 119
8.3 Solutions Formulated and Applied 121
8.3.1 Follow Schedule Closely 121
8.3.2 Refers to Various Resources 121
8.3.3 Utilization of the Proper Tools and Techniques 122
8.4 Product Strength 123
8.4.1 Modularity 123
8.4.2 User friendly 123
8.4.3 Interoperability 124

8.4.4 Reusability 124
8.4.5 Security 125
8.5 Product Limitations 126
8.5.1 License Manager Web Service 126
8.5.2 Ticket Manager Web Service 126
8.5.3 Role Manager Web Service 127
8.5.4 HR Manager Web Service 127
8.5.5 Activity Manager Web Service 128
8.5.6 FCSITPortal web application 128
8.6 Future Enhancements 129
8.6.1 License Manager Web Service 129
8.6.2 Ticket Manager Web Service 129
8.6.3 Role Manager Web Service 130
8.6.4 HR Manager Web Service 130
8.6.5 Activity Manager Web Service 130
8.6.6 FCSITPortal web application 130
8.7 Conclusions 131
GLOSSARIES 132
REFERENCES 134
BIBLIOGRAPHY 135

LIST OF TABLES

Table 4-1 Estimated hardware requirements for server 65

Table 4-2 Estimated hardware requirements for client 65

LIST OF FIGURES

Figure 1-1 The Evolution of Internet and World Wide Web 3

Figure 1-2 The visionary FSCIT XML Web Services 4

Figure 1-3 The Web Services that are covered in our WXES3182 5

Figure 2-1 The homepage for Microsoft .NET Passport 20

Figure 2-2 The homepage for Google Web APIs 22

Figure 2-3 The login page for FCSIT ALMS 24

Figure 2-4 The time clock GUI in softTime system 25

Figure 2-5 The comparisons of classical, two-tier, three-tier and n-tier architecture. 27

Figure 2-6 The service-oriented view of Web Service architecture 29

Figure 2-7 The n-tier like Web Services architecture 30

Figure 3-1 The Phased Development Software Process Model 50

Figure 4-1 Use Case diagrams of the FCSITPortal 52

Figure 5-1 The overview architecture of FCSIT Web Services 70

Figure 5-2 The layered architecture of a common web service 71

Figure 5-3 Component diagram showing the modules made up the FCSIT Integrator 71

Figure 5-4 The architectural design of the License Management Web Service 72

Figure 5-5 The architectural design of the Authentication Web Service 73

Figure 5-6 The architectural design of the Authorization Web Service 74

Figure 5-7 The architectural design of the HR Manager Web Service 75

Figure 5-8 The architectural design of the Activity Web Service 76

Figure 5-9 Sequence diagram showing request of web services by the FCSITPortal 77

Figure 5-10 Collaboration diagram showing request of web services by the FCSIT Integrator 78

Figure 5-11 StateChart diagram showing several state of the license using the License Management Web Service 79

Figure 5-12 Class diagram of the Authentication Web Service 80

Figure 5-13 Activity diagram showing user logon using the Authentication Web Service 81

Figure 5-14 Sequence diagram showing the details process when user logon 82

Figure 5-15 Collaboration diagram of the logon process using the Authentication Web Service 83

Figure 5-16 Activity diagram showing how user obtains privileges using the Authorization Web Service 84

Figure 5-17 Sequence diagram showing the details in time frame on how user gets his or her access rights from the Authentication Web Service 85

Figure 5-18 Collaboration diagram of the Authorization Web Service 86

Figure 5-19 Activity diagram showing the retrieving activities using the Activity Web Service 87

Figure 5-20 Sequence diagrams showing hot the Activity Web Services retrieve the activities 88

Figure 5-21 Collaboration diagram of Activity Web Service 89

Figure 5-22 Partitioning of the FCSIT Database to provide abstraction 90

Figure 5-23 Logical view of the FCSIT Web Services database design. Each web service manages its own database. 91

Figure 5-24 The FCSITPortal default page 92

Figure 6-1 The use of components and classes in a complex system such as the FCSITPortal. 104

Figure 7-1 The three stages of testing planned 114

Figure 7-2 The bottom-up testing approach 115

Figure 7-3 The easier and harder part when testing object-oriented systems [Graham1996a] 116

1 INTRODUCTION

1.1 Project Overview

1.1.1 The Roadmap towards Web Services

Our daily life has changed so much since the advent of the Internet and the World Wide Web. It changes the way we communicate and we work. It makes information available conveniently almost anytime and anywhere In fact, the Internet has already become a significant part of our life without us realizing the aspects it covers, from our work till our entertainment.

On seeing the potential of the Internet, business organizations move to increase the marketing scope and also refine their operations through the use of Internet. Companies set up sites to provide all kinds of services to attract customer ranging from online banking, e commerce, free email, online storage, music portals and many more. These services typically use transport protocols like TCP/IP in combination with some specific application protocols.

As more and more individuals get connected, companies were force to upgrade their network bandwidths to support the ever heavier traffics. However, as demands for services were so high, the centralized server architecture can no longer meets the demand anymore. The scalability of the architecture has come to its limit.

So, the approach is to decentralize those servers by using the distributed computing architecture. Components based software was introduced using technologies like DCOM, CORBA, and EJB. However, these approaches require too much agreement among business systems and thus coupling between various components in a system is too tight to be effective for low-overhead services. Also, they only can be applied to the intranet where these applications operate behind the firewalls.

As the result, the current trend in software implementation is moving away from these tightly coupled monolithic systems and towards systems which are more loosely coupled, dynamically bound components and Internet friendly. Systems built

with these principles are more likely to dominate the next generation of software systems, with flexibility being the overriding characteristic of their success.

To enable this to happen, the industry has come out with an Internet standard based architecture which believes that applications will be based on compositions of services discovered and marshaled dynamically at runtime or better known as just-in-time integration. Services integration becomes the innovation of the next generation of software development taking advantage of the Internet and object oriented paradigm and leveraging new technologies, such as XML to save operations cost and time.

Web Services drive this transformation by promising a fundamental change in the way applications are developed and deployed. The Web Services architecture describes principles for creating dynamic, loosely coupled systems based on services, but with no single implementation technique.

1.1.2 So what's a Web Service?

Probably you already heard about it or probably not. In its simplest form a Web Service is an executable component that can be invoked from machine to machine via a common set of data formats and protocols over the Internet or other networks. It is loosely coupled, platform and language independent. Well, the exact definition for a Web Services is not yet standardize as different vendors defines it with their own jargons but still refers to this same thing.

First of all, don't mistaken this term as services that are offered in the normal websites such as Hotmail, MP3.com, e-Bay, amazon.com or so on. They are indeed offering services like e-mail, mp3s, online purchase or so on, but to be precise they are known as Web-based services, not Web Services.

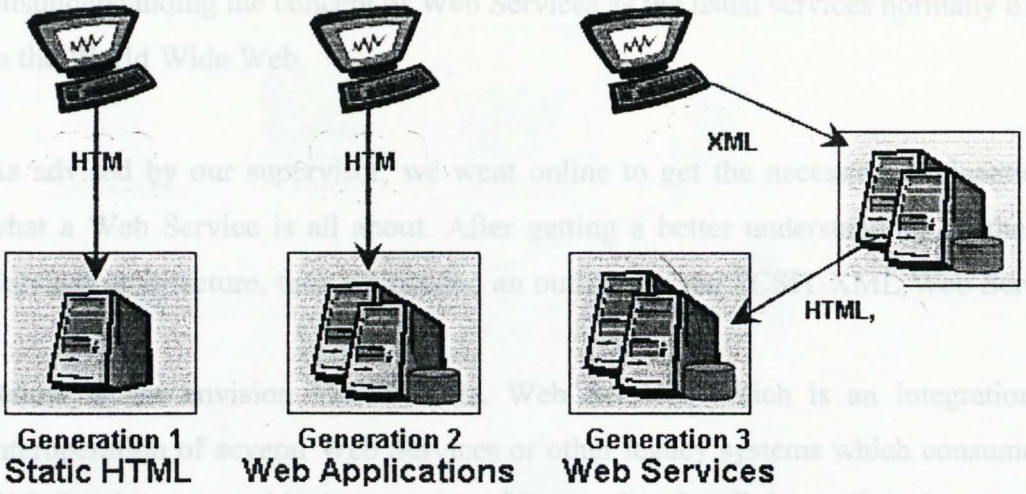


Figure 1-1 The Evolution of Internet and World Wide Web

Let me explain it this way. When people want some information, they look for certain websites that offer some sort of services to get the information they needed. However, when applications want information, they look for certain Web Services and get that information by requesting it from another computer. People receive Web pages in the form of Hypertext Markup Language (HTML). Computers receive information from Web services in the form of SOAP.

Let's say you want to restrict your websites only to users who sign up with you. So, most probably you will create a webpage that contains the login service for your users to sign in before they can access your sites.

Now, let's say I want to use your login service, (instead of creating a new one), but I wanted to implement the login service (not your login page along with your company logo, of course!) directly in my login page to authenticate my website's users. The question is how can my program invoke the method that resides in your machine in order to use it? The answer is very simple; implement it as a Web Service.

1.1.3 The FCSIT XML Web Services

I and my partner, Mr. Lee San Kuan got this idea or thesis title from our supervisor, Mr. Ang Tan Fong. When I got to know this title at the very first, I admitted that I

misunderstanding the concept of Web Services as the usual services normally offered in the World Wide Web.

As advised by our supervisor, we went online to get the necessary explanation on what a Web Service is all about. After getting a better understanding on the Web Services architecture, then we created an outline for the FCSIT XML Web Services.

Below is the envision FSCIT XML Web Services which is an integration and interoperation of several Web Services or other legacy systems which consumes the Web Services to provide the necessary functionality for all the staffs and students.

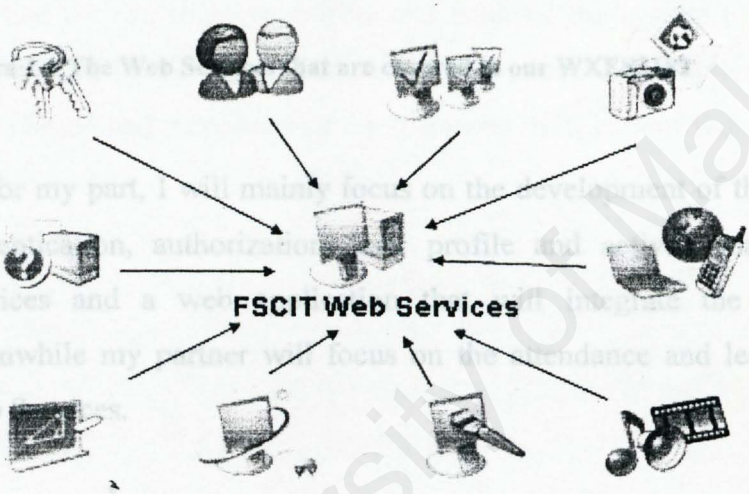


Figure 1-2 The visionary FSCIT XML Web Services

Of course, this is a visionary system and I am not sure if it is able to be fulfilled. I and my partner were not able to develop the full system by our own as this visionary system is expected to be very large and complex. Each of the Web Services is yet another full system on its own. So, in our thesis we will just develop parts of the whole FSCIT XML Web Services. Below is the XML Web Services that we will cover and build in our WXES3182 project.

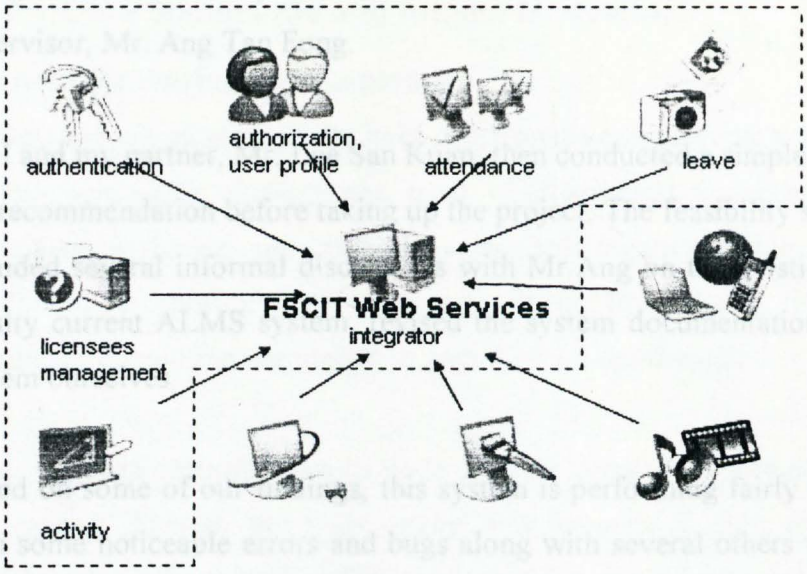


Figure 1-3 The Web Services that are covered in our WXES3182

As for my part, I will mainly focus on the development of the license management, authentication, authorization, user profile and activity management XML Web Services and a web application that will integrate the XML Web Services. Meanwhile my partner will focus on the attendance and leave management XML Web Services.

And here, I hope that the rest of the visionary FSCIT Web Services will be built by other students as their WXES3182 project. The areas where the FCSIT Web Services that needs further development are incorporating new Web Services into the system such as HR WS, Payroll WS, Inventory WS or so on which you may find useful and beneficial to the staffs or students of FSCIT. Also, other students may find that they can further enhance the Web Services that I and my partner have created to improve the overall functionality, interoperability, performance, modularity or remove flaws and errors.

1.1.4 The Faculty's Current ALMS System

As to why we redevelop the attendance and leave management system as part of our WXES3182 project is fulfill the requirement of system integration and to

demonstrate the interoperable feature of XML Web Services as specified by our supervisor, Mr. Ang Tan Fong.

So, I and my partner, Mr. Lee San Kuan, then conducted a simple feasibility study on the recommendation before taking up the project. The feasibility study conducted has included several informal discussions with Mr.Ang on the existing problems of our faculty current ALMS system, revised the system documentations and tries out the system ourselves.

Based on some of our findings, this system is performing fairly well right now, but with some noticeable errors and bugs along with several others weaknesses that we find that we can improve further and enhance the system to a higher level. So, we then decided to take up the task to redevelop the ALMS system to remove correct the errors found and reengineered the system as independent Web Services.

1.2 Problem Statements and Project Motivation

The need for sharing and cooperation

Sharing and cooperation in the inter departments environment within an organization is now achievable through the used of distributed computing. However, inter organizations sharing and cooperation are still obstructed due to the limitation of distributed computing where remote object invocations implemented in the distributed computing are unable to pass through the firewall.

Thus, the organizations have to trade off their security by opening their firewall for this reason. Though there are many new security feature of network security, including the advent VPN, many organizations are still refused to do so in fear of malicious attack and other security issues.

Constraints in software development

Business entities have to be fast and cost effective to ensure their survival in the ever fast changing environment today. So, lifecycle of the software development is inevitably affected by these changes too. Time and resources constraints are the major issues in software development while fulfilling all the business requirements is a must.

Difficulty in system maintenance

While the developed systems are getting larger and more complex and thus the systems are prone to more errors. So, keeping the software easy to debug and maintain are few other very important aspects. When the business operation gets larger, the system has to be able to scale itself to coup with the greater demands is another added feature.

The limitation of distributed computing architecture

Enables data and also objects sharing but in a limited coverage as unable to pass through the firewall that protects the organization resource. However, it uses the monolithic system architecture design that consists of tightly coupled modules which makes the system not very much scalable and open. While no standardized format of implementations and are tight to one of the proprietary technologies available such as

EJB from Sun Microsystems or DCOM from Microsoft or CORBA from OMG, the interoperability issues arise between systems of different implementations.

The current ALMS system contains errors or bugs in its functionalities

Over the years of usage, the users of the system have discovered some bugs, errors and inconsistency in some of the functionalities the system provides. So, it is time in which a need for redevelopment of the system is much required and anticipated.

The obsolete, irrelevant and unused functionalities

The current ALMS system has been developed for almost more than a year and a few functionalities have proven to be obsolete. Also, as the system is used in this period of usage, many of the irrelevant and redundant functionalities had been discovered and can be removed.

The design of the current ALMS system and its database

The design of the current ALMS system is not open and interoperable with other system. It is design to function as a standalone entity not considering the reusable components and sharing of data with other systems. As the system is directly manipulates the main central database of the faculty, it has to provide some means of functionalities that can be used by other system to access the database instead of causing each of the system maintains their own copies of database which resulting the problem of redundancy and inconsistency.

The issue of user friendliness interface design

Overall the current ALMS system is quite simple and straightforward in nature. However, due to the lacking of documentation and help functionality and also some aspects poor interface design such as deep linking and unrecoverable actions and inadequate user status and information, the need for redesign the interface is need to achieve the better clarity, ease of use and lastly but not least important, more attractive and interactive graphical interface design.

1.3 Project Visions and Objectives

To develop a system that is scalable and easily enhance

The system is implemented as an integration independent web services. As mentioned earlier, web services are loosely coupled and thus making them very scalable. If there is a need for new functionalities, new services can be invoked from other web services without affecting the current system. If there happens to be any web services are felt obsolete, they can be redeveloped and then be deployed so that the other systems can then discover the new services and make reference to these new services and then invoke them from their system.

To develop a system that is reusable

The system design has support reusability as to avoid the redundancy of data and components that provide the same functionality. It enables the reusability of functionalities that are deployed by other web services. One only has to discover the functionalities, makes a reference to the services and then invokes the services through the network or the Internet. As the result, the developers can avoid developing modules that provides the same functionalities all over again and thus reduce development and testing time.

To develop a system that is interoperable and open

The services provided by the system will be able to be invoked and used in other systems independent to their deployment platform or programming language. So, this promotes the interoperability of heterogeneous systems as long as they conform to the communications protocols such as UDDI for discovery of services, WSDL for describing the interface and SOAP for invocating services and data exchanges. This feature of openness has benefited especially the development of large system as nowadays each of the modules can be implemented independently by different vendors and technologies.

To develop a system that is easy to debug and maintain

The system will be consisted of independent and loosely coupled system if seen as high level while every service designed will consist of classes and objects. By combining the web services design and object oriented methodology, debugging the

system will be much easier. The same applies to the maintenance of the system which can be done independently without affecting other systems.

To developed a system with single sign on feature without compromising the system security

This system is hoped to ease user from hassle of multiple logins and remembering different password when using services provide by different services by featuring the single sign on advantage. Although user just have to sign in once to several services provided by different systems, the security issues are not compromised. The security issues will be still up hold by using user authentication such as form login, use of cookies, and expiration of user's session. Next, the security issues will be tackled by using the role based authorization where the user is assigned to his or her privileges based on his or her role.

To redevelop the current ALMS system

As stated in the problem statements, we found out several weaknesses that we think we can refine and improve over the existing system. So, in redeveloping the system, we will redesign the programming logic to enhance the functionalities, to remove the errors found and converting all the functionalities into so called web methods as to realize the web services architecture. Then, we will also redesign the database to remove redundancy, ensure consistency and to improve clarity through normalized database design. However, we will retain all the useful and necessary functionalities while remove those that are not.

To develop a system that is user friendly

No matter how good the system performs, the overall rating to the system is based by the graphical user interface of the system. The graphical user interface of this system will be based on common web interface design but particularly focusing on the aspect of ease of use, simple organization and structuring, self described, well documented with user help, interactive and lastly but not least important, the attractive user interface. The other aspect that is taken into serious consideration is to deceive the user from knowing the actual implementation such that the invocation of individuals services are made transparent.

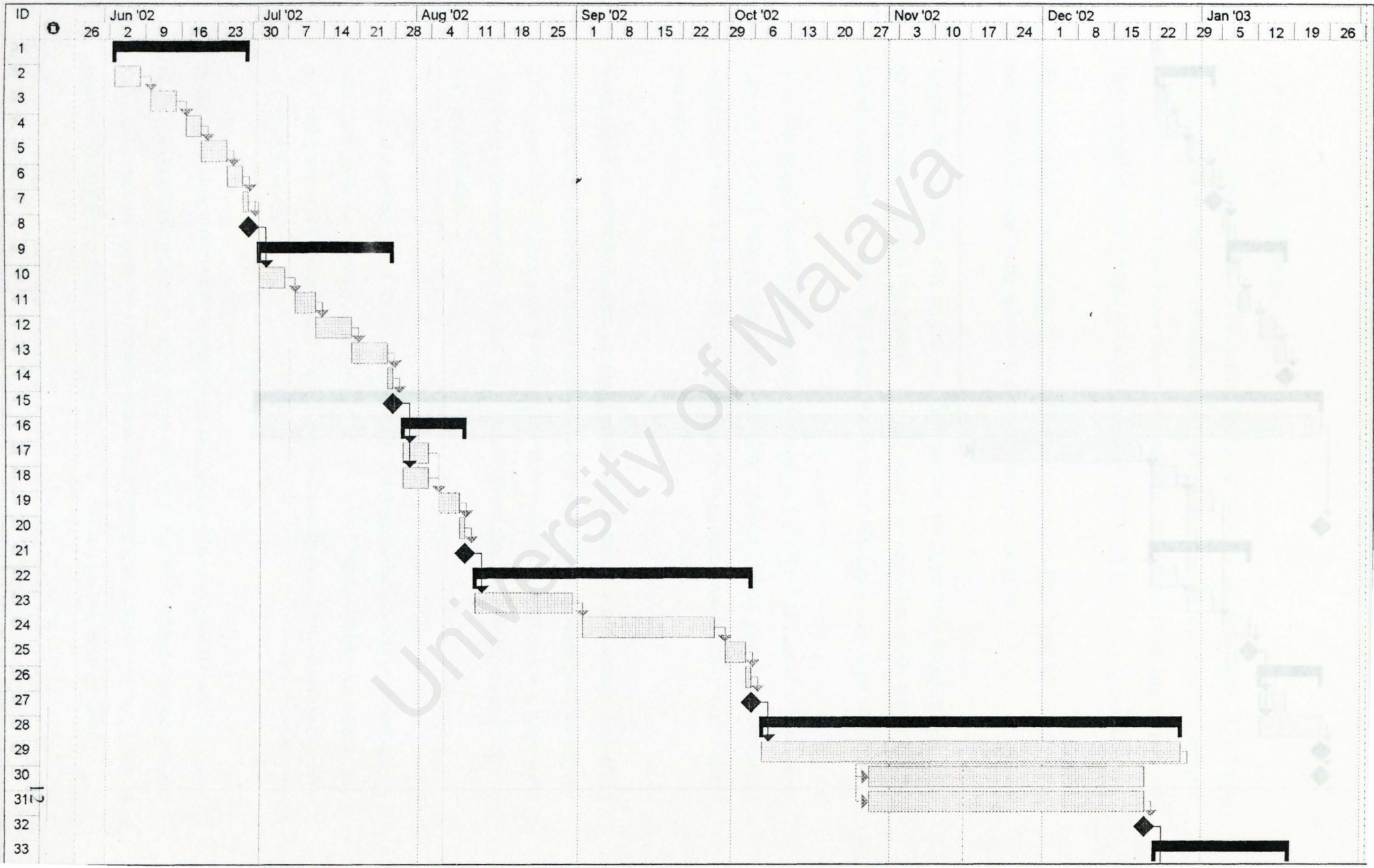
1.4 Project Schedule

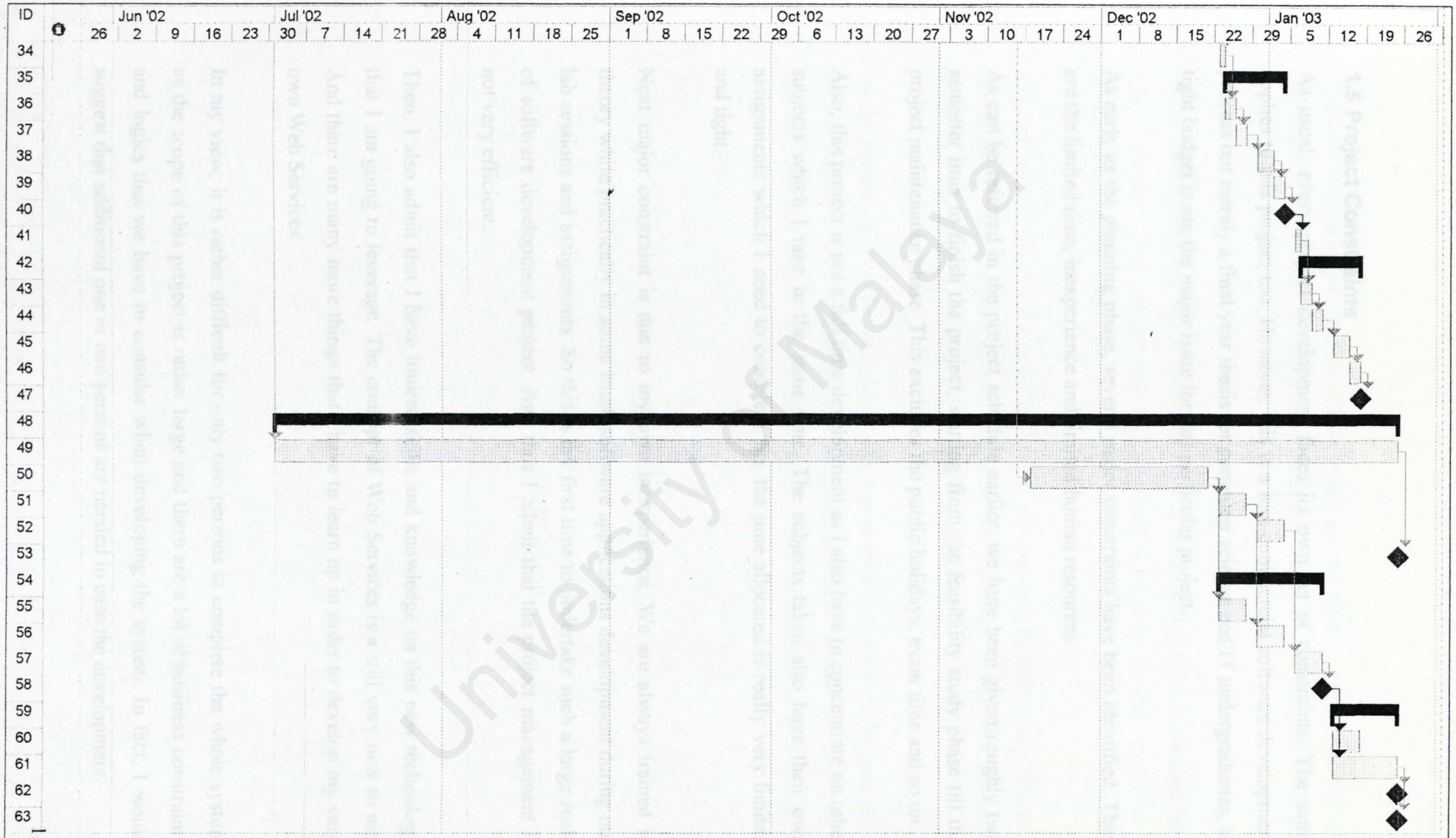
In order to accomplish the project in time, a project schedule has been formulated to act as guidance as we step through the development process. The schedule was drawn using Microsoft Project 2000 using the provided software development template.

The schedule is rather tight in its nature. As development process of this project is not full time as I also have to cope with other courses as well in the same time, so once in a while, the start time of some activities will be delayed or the time for some activities will be shortened accordingly, based on the situation which I face.

So, when I was formulating this schedule, a lot of considerations have been taken into account. I always reserved extra time for each of the activities scheduled and the following activities will start as soon as possible after the previous activity has accomplished.

As a conclusion, I will always try my best to follow the schedule as close as possible. And thus, I hope that the project can be accomplished smoothly as it is planned in this project schedule drawn.





1.5 Project Constraints

As usual, every project development faces its own set of constraints. The same applies to this project too. However, this is a non-commercial software development process but merely a final year thesis for computer science and IT undergraduates, so tight budget is not the major issue for this particular project.

As early as the planning phase, several major constraints have been identified. They are the limited time, inexperience and limited human resources.

As can be referred in the project schedule earlier, we have been given roughly two semester time to finish the project, starting from the feasibility study phase till the project maintenance phase. This excludes the public holidays, exam time and so on.

Also, this project is not a full time development as I also have to concentrate on other subjects which I take in the same time. The subjects taken also have their own assignments which I need to complete. So, the time allocated is really very limited and tight.

Next major constraint is due to my own inexperience. We are always trained in theory while practically in some small software applications development during our lab sessions and assignments. So this is my first time to undertake such a large scale of software development project. And thus I admit that the project management is not very efficient.

Then, I also admit that I have limited skills and knowledge on this new technology that I am going to leverage. The concept of Web Services is a still very new to me. And there are many more things that I have to learn up in order to develop my very own Web Services.

In my view, it is rather difficult for only two persons to complete the whole system as the scope of this project is rather large and there are a lot of business constraints and logics that we have to consider when developing the system. In fact, I would suggest that additional one or two persons are needed to ease the development.

Apart from these major constraints mentioned, several minor constraints have been identified as well. These are the limited reference books, budget and some technical constraints.

When developing the project, I was unable to get the reference books needed from the UM library. They are too new and with limited amounts. So they are either in order process or have been borrowed out.

Limited budget which I am referring here is my personal budget to buy reference books and to upgrade my personal computer. The reference books sold are rather expensive. So, I have to limit myself to only few references.

As this technology is rather new, so greater system requirements is needed in order to run the software. So, I have to upgrade the specifications of my personal computer to the recommended requirements stated in order to run and develop the Web Services.

1.6 Project Scope

The Web Services can be implemented over various kinds of domain and usages. However, the scope of the project will not be covering all the aspects the real enterprise systems. However, I will try to cover as much as possible into this project but basically will provide the necessary functionalities first.

The target user will mainly consist of the staffs of Faculty of Computer Science and Information Technology.

The main functionalities of the FCSIT XML Web Services that will be built by me include the licensee management, user authentication, user authorization, user profiles and activity management.

The attendance and leave management XML Web Services will be developed by my partner and will preserve the all the necessary features and functionalities that the current ALMS system is providing while enhancing some extra functionalities which are lacked and removing the unnecessary ones.

Also, I will be building a web application that acts as the client to the FCSIT XML Web Services by consuming and integrating all the XML Web Services that me and my partner have built. This web application will have the user friendly, attractive GUI and scalable features, providing a comprehensive documentation and help files for the end users, developers and licensees.

1.7 Expected Outcome

The expected and planned outcome of this project will consist of 7 XML Web Services and a Web Application. The XML Web Services are License Management Web Service, Authentication Web Service, Authorization Web Service, User Profile Web Service, Activity Management Web Service, Attendance Management Web Service, Leave Management Web Service and an Integrator which is a web application.

Each of these XML Web Services is independent but interrelated. The License Management Web Service is used to protect the rest of the XML Web Services from unauthorized usage.

As their name implies, the others XML Web Services provides the necessary functions to authenticate and authorize user, user data management, activity outlook, attendance and leave management.

Then, a web application that will integrate the XML Web Services together as a whole and complete system is needed to deceives the user for the actual implementation of the system while provides the user with a user-friendly interfaces, expandable features and a whole new GUI design.

A full documentation of this system is included describing the architecture of the system, how the system is built, how to setup and maintain the system, how to develop new enhancements and consumes the XML Web Services developed.

2 LITERACY REVIEW

2.1 Information Gathering Techniques

2.1.1 The Internet and World Wide Web

Most of the information is obtained from the Internet and the World Wide Web in various kinds of forms such as articles, white papers, product documentations, web casts and e-books. From the Internet and World Wide Web, I managed to obtain a lot of information, new ideas, latest technologies and various kinds of implementation techniques and tools which I can used to develop my project.

2.1.2 Reference Books

I have also been using several reference books and e-books for more in depth and comprehensive explanation. These reference books and e-books that I referred are bought from the local bookstores, or obtained from my supervisor, my friends or the UM main library.

2.1.3 Discussion with the User

To understand on the user requirements and the problems arise in the current system, I and my partner had several informal discussions with the system users especially my supervisor Mr. Ang Tan Fong and the system administrator, Ms Azlin. From these discussions, we then derived the pros and cons of the system and user expectations on our system.

2.1.4 System Documentation

To obtain the technical and further information about the system, I have referred to the documentation of similar systems which I obtained from the faculty document room. From there I can find out the architecture and design of the similar systems.

2.1.5 Trial on the Existing System

I and my partner are lucky to be able to try on the existing system itself. So, from with the experience of using the system, we can then see and judge the system from the user's perspective.

2.1.6 Brainstorming

After getting all the information needed, then it is the time for brainstorming. The brainstorming session are sometimes done individually, with my project partner and supervisor and also with other friends and other lecturers. Brainstorming sessions are very much importance in the sense of formulating the system architecture and design, implementation techniques and solve the ambiguities that I faced.

2.2 Similar Systems

2.2.1 Microsoft .NET Passport (www.passport.com)

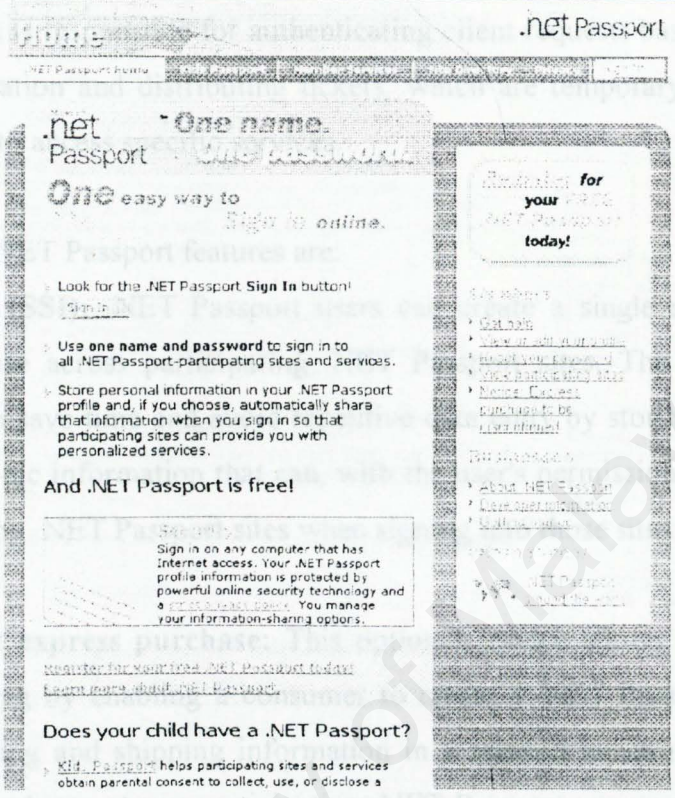


Figure 2-1 The homepage for Microsoft .NET Passport

Right now, there are over 165 million .NET Passport accounts. Microsoft .NET Passport, launched in 1999, is a suite of Web-based services that makes using the Internet and purchasing online easier and faster. It allows customers "single sign in" convenience, meaning they have to put their user name and password in only one time to access all of the sites that use .NET Passport.

.NET Passport helps provide a high-quality online experience for a large user base and uses powerful encryption technologies—such as Secure Sockets Layer (SSL) and the Triple Data Encryption Standard (3DES) algorithm—for data protection. Privacy is a key priority as well, and all participating sites sign a contract in which they agree to post and follow a privacy policy that adheres to industry-accepted guidelines.

The latest version of .NET Passport will be implementing the Kerberos-distributed security protocol for the .NET My Services web services. Kerberos is a proven industry standard security protocol that is used by the Microsoft Windows 2000 and Microsoft Windows XP operating systems for user authentication. In the simplest sense, Kerberos is responsible for authenticating client requests based on centralized security information and distributing tickets, which are temporary encryption keys that clients use to access specific services.

Currently, the .NET Passport features are:

Single sign in (SSI): .NET Passport users can create a single sign-in name and password for use across participating .NET Passport sites. The SSI service also enables users to save time and avoid repetitive data entry by storing a limited set of basic demographic information that can, with the user's permission, be easily shared with participating .NET Passport sites when signing into those sites.

.NET Passport express purchase: This optional service allows for easy and fast online purchasing by enabling a consumer to create a .NET Passport "wallet" that stores their billing and shipping information in a secured location. Consumers can make online purchases at any participating .NET Passport express purchase sites by signing in to their wallet and, with a single click, sending their purchase information to the merchant instantly—eliminating the need to retype it.

Kids Passport: Microsoft Kids Passport is a feature of the .NET Passport SSI service and is an example of Microsoft's ongoing initiative to provide children with a positive, safe online experience. The Kids Passport service is designed to provide participating Kids Passport sites with assistance in complying with the Children's Online Privacy Protection Act (COPPA) and provides parents or guardians a way to control and manage what .NET Passport profile information their children can share at .NET Passport–participating sites.

2.2.2 Google Web APIs Services (www.google.com/apis)

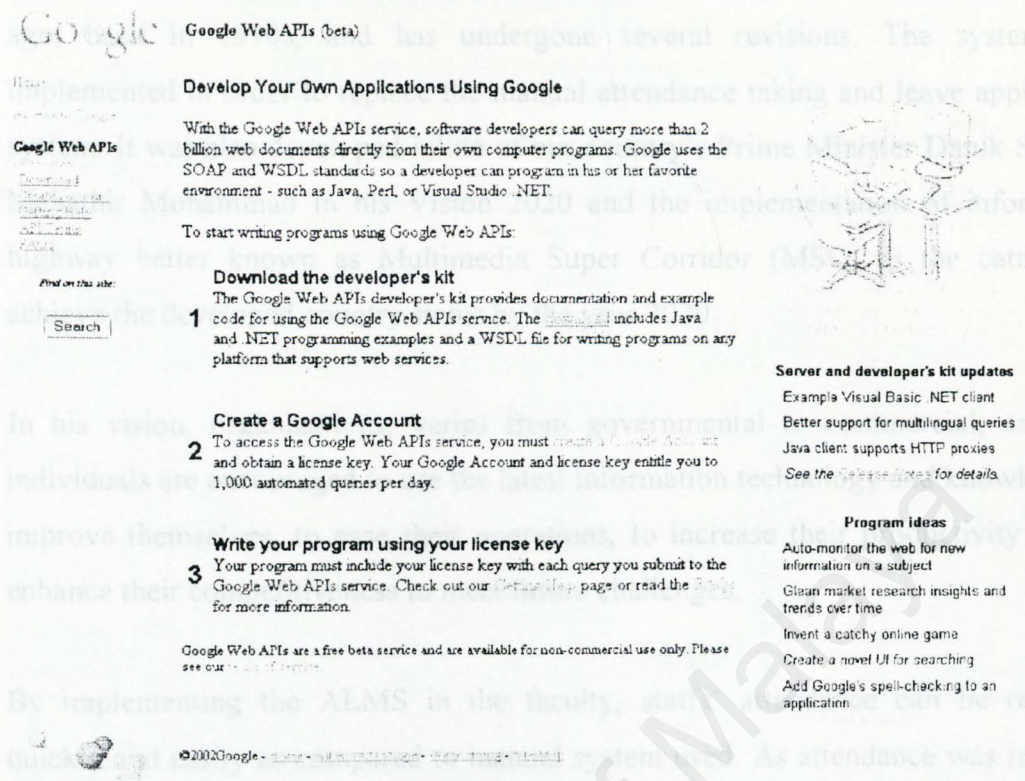


Figure 2-2 The homepage for Google Web APIs

Google Web APIs are implemented as a web service. Currently, it is still in the beta release. The service supports several SOAP methods; these are described in an accompanying WSDL file that can be imported. Alternately, you can use the custom Java library to call the service.

The Google Web APIs service enables developers to easily find and manipulate information on the web. The Google Web APIs service gives you query access to Google's web search, enabling you to develop software that accesses billions of web documents that are constantly refreshed.

Developers can issue search requests to Google's index of more than 2 billion web pages and receive results as structured data, access information in the Google cache, and check the spelling of words. Google Web APIs support the same search syntax as the Google.com site.

2.2.3 The Faculty's Current ALMS (http://lms.fsktm.um.edu.my)

The first version ALMS system has been implemented in our faculty several years ago, back in 1990s, and has undergone several revisions. The system was implemented in order to replace the manual attendance taking and leave application system. It was also developed inline of our country's Prime Minister Datuk Seri Dr. Mahathir Mohammad in his Vision 2020 and the implementation of information highway better known as Multimedia Super Corridor (MSC) as the catalyst to achieve the developed country status by the year 2020.

In his vision, organizations, varies from governmental to commercial, and also individuals are encouraged to use the latest information technology and knowledge to improve themselves, to ease their operations, to increase their productivity and to enhance their competitiveness to meet future challenges.

By implementing the ALMS in the faculty, staffs' attendance can be recorded quickly and easily as compared to manual system used. As attendance was recorded electronically, administrative staff no longer needs to organize the paper attendance and thus increased departmental productivity and improve cost effectiveness. Besides that, it was environmental friendlier as it promoted paperless environment.

However, the first release of the ALMS was a standalone windows based application with a complimentary web based system. The academic staffs only could take their attendance on a specific station with the system installed while non-academic staffs could either do so or accessed to the web based system to take their attendance. Another major problem was the database of the system which is not optimized and full of redundancy

The leave management system was then a separate entity, another standalone system by itself. While the two systems were unable to share on the same data, redundancy happened where replication of similar information occurred. Another major problem was inability to synchronize data and relate interdependent functionality such as when you apply leave using the leave management system, the attendance system must not penalize you for not coming for work.

Figure 2-3 The login page for FCSIT ALMS

As the number of staffs in our faculty had increased dramatically, the system proved to be insufficient and inefficient to meet the demands. So, a better release of the ALMS system had been introduced. This system is currently used in our faculty. Some of the noted features of the system are single sign on to both systems (AMS and LMS), centralized database design, improved integrations, fully web based, and provides sufficient functionally to manage the attendance taking and the leave application processes.

2.2.4 The softTIME attendance system (www.webattendance.com)

The SoftTIME attendance management system from Software Technique Inc. allows you to track employee absences with the simple click of the mouse. It offers a multitude of features that incorporate over 20 useful reports that show who is absent and why. This product is unique and easy to use because it utilizes object-oriented technology that allows the user to drag and drop employees' absences onto the calendar.

Just think of the time and money that is wasted documenting each absence manually. It is so simple to use that you don't need to open the manual. It will import your existing data from Excel, FoxPro or ASCII files and is password protected for security. It offers unlimited accrual and carryover days and hours, allowing you an easy way to set up and manage vacation time, sick days, leaves of absence, and other absence categories.

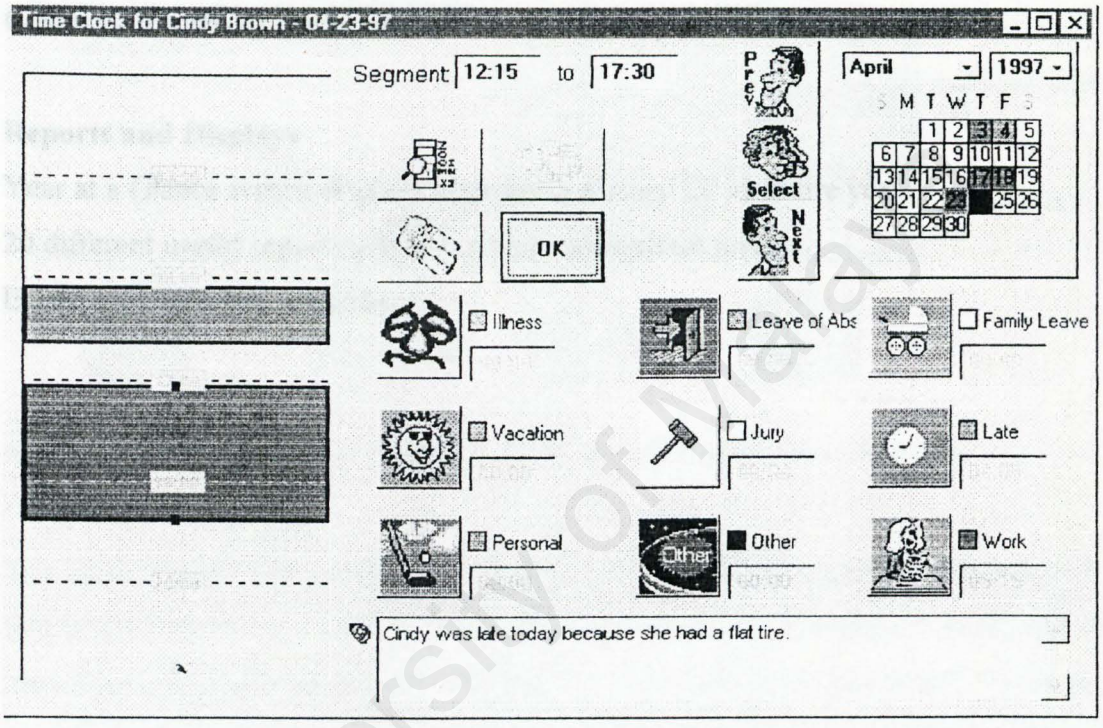


Figure 2-4 The time clock GUI in softTime system

Some of the noted features are:

Colorful and easy to use

To record an absence, simply drag an absence icon and drop it onto the calendar

Record and track

Track attendance on calendar, anniversary, or fiscal year separately for each employee

Record absences as they occur at any time during a 24 hour day in increments as small as 5 minutes

Carry Over and Accrue

Carry over employee's unused vacation, illness, and personal time

Accrue vacation, illness, and personal time based on time worked

Automatically totals employee absences

Flexible

Set up shifts by department or individual

Create your own absence categories and customize them with your own icons

Create your own company holidays

Reports and Displays

Year at a Glance screen displays attendance history for an entire year

20 different useful reports which can be previewed on screen

Easily spot vacation conflicts

2.3 Development Technologies

2.3.1 The Architecture

2.3.1.1 The n-tier Architecture

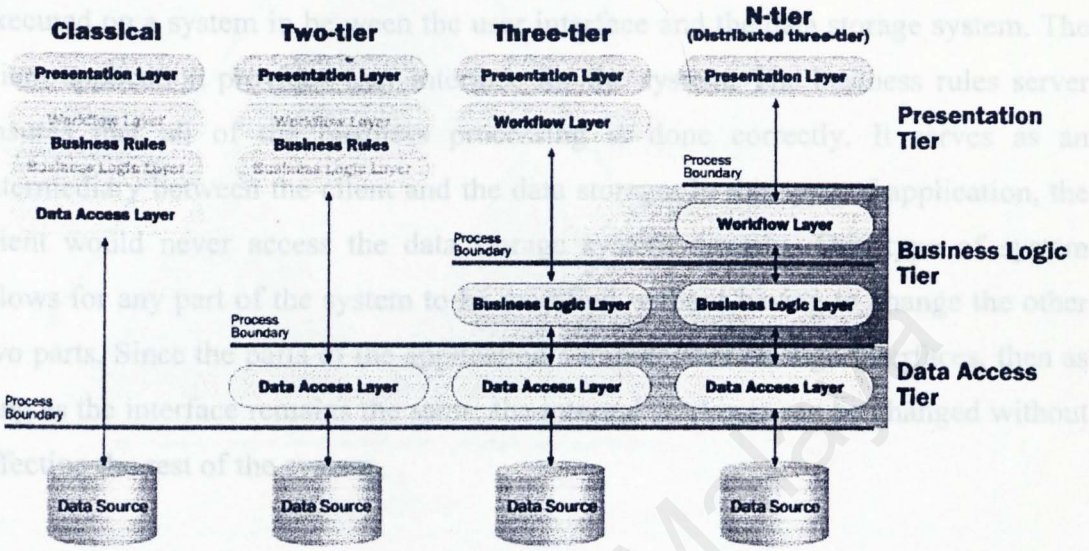


Figure 2-5 The comparisons of classical, two-tier, three-tier and n-tier architecture

The classic application or better known as the monolithic application consists of a single application layer that supports the user interface, the business rules, and the manipulation of the data all in one. The data itself could be physically stored in a remote location, but the logic for accessing it is part of the application. The user interface is an integral part of the application. The business rules, such as how to paginate and hyphenate, are also part of the application. The file access routines, to manipulate the data of the document, are also part of the application. Even if there are multiple DLLs that handle the different functionality, it is still a monolithic application.

In a two-tier application, the business rules and user interface remain as part of the client application. The data retrieval and manipulation is performed by another separate application, usually found on a physically separate system. In another type of two-tier application, the business rules are executed on the data storage system. This is the case in applications that are using stored procedures to manipulate the database. A stored procedure is a database function that is stored at the database

server. It can be executed one of two ways. A client application can explicitly call a stored procedure, which would then be run on the server. A trigger can also execute a stored procedure, which is the occurrence of a specific event in the data.

With three-tier applications, the business rules are removed from the client and are executed on a system in between the user interface and the data storage system. The client application provides user interface for the system. The business rules server ensures that all of the business processing is done correctly. It serves as an intermediary between the client and the data storage. In this type of application, the client would never access the data storage system directly. This type of system allows for any part of the system to be modified without having to change the other two parts. Since the parts of the application communicate through interfaces, then as long as the interface remains the same, the internal workings can be changed without affecting the rest of the system.

2.3.1.2 The Web Services Architecture

On the high level view, the Web Services architecture describes the principles behind the next generation of system architecture, presenting a logical evolution from object-oriented systems to service-oriented systems. Web Services systems promote significant decoupling and dynamic binding of components: All components in a system are services, in that they encapsulate behavior and publish a messaging API to other collaborating components on the network. Services are marshaled by applications using service discovery for dynamic binding of collaborations.

Both approaches have been proven in dealing with the complexity of large systems. As in object-oriented systems, some of the fundamental concepts in Web Services are encapsulation, message passing, dynamic binding, and service description and querying. Fundamental to Web Services, then, is the notion that everything is a service, publishing an API for use by other services on the network and encapsulating implementation details.

The Web Services architecture then requires three fundamental operations: publish, find, and bind. A service provider creates a web service and its service definition and

then publishes the service to a service broker. Service requesters find required services using a service broker and bind to them.

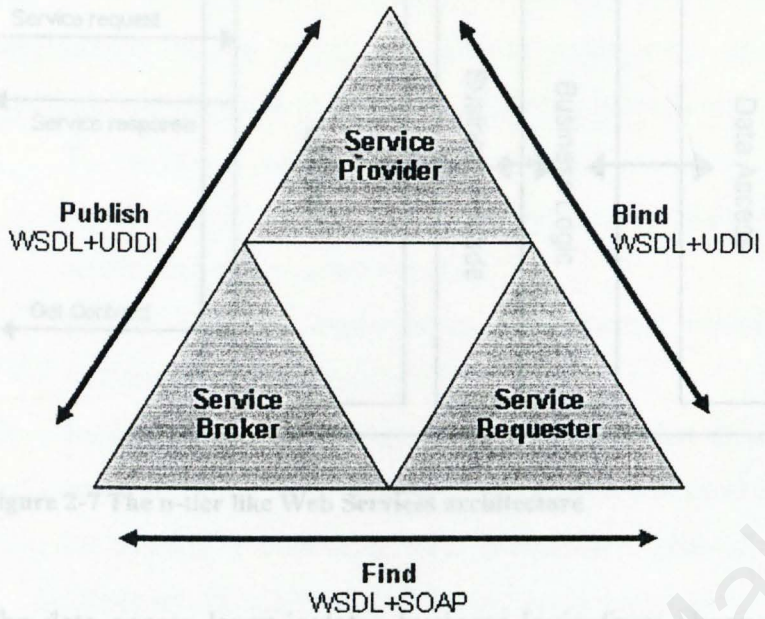


Figure 2-6 The service-oriented view of Web Service architecture

The Web Services architecture provides several benefits, including:

- Promoting interoperability by minimizing the requirements for shared understanding
- Enabling just-in-time integration
- Reducing complexity by encapsulation
- Enabling interoperability of legacy applications

On the low level view, the Web Service architecture is more or less similar to n-tier architecture. However, the Web Service architecture is divided into five logical layers. Furthest from the client is the data layer, which stores information required by the Web Service. Above the data layer is the data access layer, which presents a logical view of the physical data to the business layer.

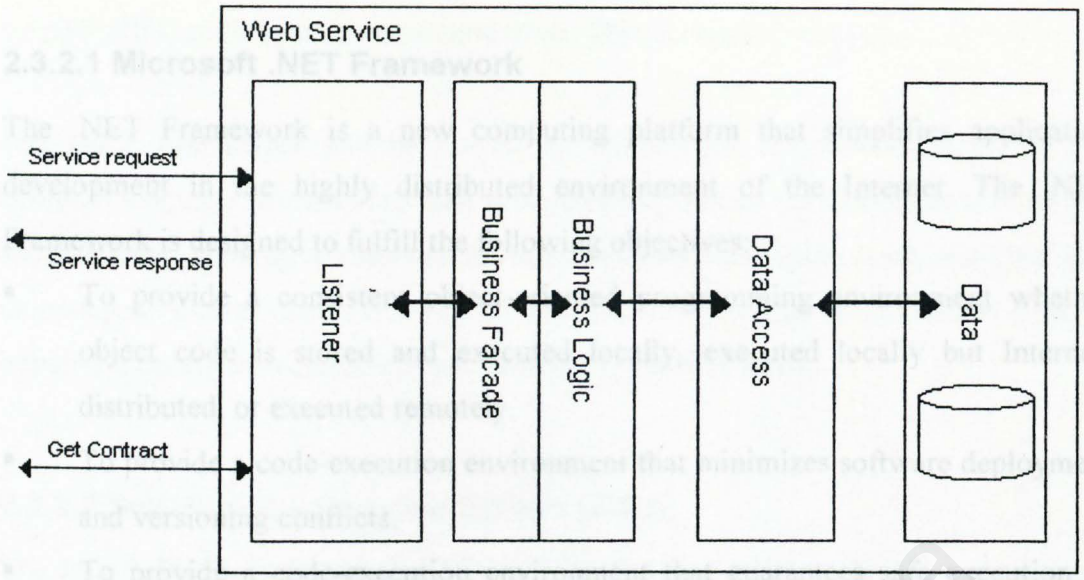


Figure 2-7 The n-tier like Web Services architecture

The data access layer isolates business logic from changes to the underlying data stores and ensures the integrity of the data. The business layer implements the business logic of the Web Service. It is often subdivided into two parts: the business facade and the business logic.

The business facade provides a simple interface that maps directly to operations exposed by the Web Service. The business facade uses services provided by the business logic layer. In a simple Web Service, all the business logic might be implemented by the business facade, which would interact directly with the data access layer.

Client applications interact with the Web Service listener. The listener is responsible for receiving incoming messages containing requests for service, parsing the messages, and dispatching the request to the appropriate method on the business facade. If the service returns a response, the listener is also responsible for packaging the response from the business facade into a message and sending that back to the client. The listener also handles requests for contracts and other documents about the Web Service.

2.3.2 The Frameworks

2.3.2.1 Microsoft .NET Framework

The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet. The .NET Framework is designed to fulfill the following objectives:

- To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that guarantees safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
- To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

The .NET Framework has two main components: the common language runtime and the .NET Framework class library.

The common language runtime is the foundation of the .NET Framework. You can think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and remoting, while also enforcing strict type safety and other forms of code accuracy that ensure security and robustness. In fact, the concept of code management is a fundamental principle of the runtime. Code that targets the runtime is known as managed code, while code that does not target the runtime is known as unmanaged code.

The class library, the other main component of the .NET Framework, is a comprehensive, object-oriented collection of reusable types that you can use to develop applications ranging from traditional command-line or graphical user

interface (GUI) applications to applications based on the latest innovations provided by ASP.NET, such as Web Forms and XML Web services.

The .NET Framework can be hosted by unmanaged components that load the common language runtime into their processes and initiate the execution of managed code, thereby creating a software environment that can exploit both managed and unmanaged features. The .NET Framework not only provides several runtime hosts, but also supports the development of third-party runtime hosts.

2.3.2.2 Sun Java 2 Enterprise Edition (J2EE)

The Java 2 Platform, Enterprise Edition (J2EE) defines the standard for developing multitier enterprise applications. J2EE simplifies enterprise applications by basing them on standardized, modular components, by providing a complete set of services to those components, and by handling many details of application behavior automatically, without complex programming.

The Java 2 Platform, Enterprise Edition, takes advantage of many features of the Java 2 Platform, Standard Edition, such as "Write Once, Run Anywhere" portability, JDBC™ API for database access, CORBA technology for interaction with existing enterprise resources, and a security model that protects data even in internet applications.

Building on this base, Java 2 Enterprise Edition adds full support for Enterprise JavaBeans components, Java Servlets API, Java Server Pages and XML technology. The J2EE standard includes complete specifications and compliance tests to ensure portability of applications across the wide range of existing enterprise systems capable of supporting J2EE.

The J2EE application model divides enterprise applications into three fundamental parts: components, containers, and connectors. Components are the key focus of application developers, while system vendors implement containers and connectors to conceal complexity and promote portability.

Containers intercede between clients and components, providing services transparently to both, including transaction support and resource pooling. Container mediation allows many component behaviors to be specified at deployment time, rather than in program code.

Connectors sit beneath the J2EE platform, defining a portable service API to plug into existing enterprise vendor offerings. Connectors promote flexibility by enabling a variety of implementations of specific services.

2.3.3 The Implementation Languages

2.3.3.1 XML

XML is the shorthand for Extensible Markup Language, and is an acronym of Extensible Markup Language. XML was conceived as a means of regaining the power and flexibility of SGML without most of its complexity. Although a restricted form of SGML, XML nonetheless preserves most of SGML's power and richness, and yet still retains all of SGML's commonly used features. While retaining these beneficial features, XML removes many of the more complex features of SGML that make the authoring and design of suitable software both difficult and costly.

Development of XML started in 1996 and has been a W3C Recommendation since February 1998, which may make you suspect that this is rather immature technology. In fact, the technology isn't very new. Before XML, there was SGML developed in the early '80s, an ISO standard since 1986, and widely used for large documentation projects. The development of HTML started in 1990. The designers of XML simply took the best parts of SGML, guided by the experience with HTML, and produced something that is no less powerful than SGML, and vastly more regular and simple to use. Some evolutions, however, are hard to distinguish from revolutions... And it must be said that while SGML is mostly used for technical documentation and much less for other kinds of data, with XML it is exactly the opposite.

Structured data includes things like spreadsheets, address books, configuration parameters, financial transactions, and technical drawings. XML is a set of rules (you

may also think of them as guidelines or conventions) for designing text formats that let you structure your data. XML is not a programming language, and you don't have to be a programmer to use it or learn it. XML makes it easy for a computer to generate data, read data, and ensure that the data structure is unambiguous. XML avoids common pitfalls in language design: it is extensible, platform-independent, and it supports internationalization and localization. XML is fully Unicode-compliant.

XML allows you to define a new document format by combining and reusing other formats. Since two formats developed independently may have elements or attributes with the same name, care must be taken when combining those formats (does "<p>" mean "paragraph" from this format or "person" from that one?). To eliminate name confusion when combining formats, XML provides a namespace mechanism. XSL and RDF are good examples of XML-based formats that use namespaces. XML Schema is designed to mirror this support for modularity at the level of defining XML document structures, by making it easy to combine two schemas to produce a third which covers a merged document structure.

2.3.3.2 ASP.NET

ASP.NET is a unified Web development platform that provides the services necessary for you to build enterprise-class Web applications. While ASP.NET is largely syntax compatible with Active Server Pages (ASP), it provides a new programming model and infrastructure that allow you to create a powerful new class of applications.

ASP.NET is more than the next version of Active Server Pages (ASP); it is a unified Web development platform that provides the services necessary for developers to build enterprise-class Web applications. While ASP.NET is largely syntax compatible with ASP, it also provides a new programming model and infrastructure for more secure, scalable, and stable applications. You can feel free to augment your existing ASP applications by incrementally adding ASP.NET functionality to them.

ASP.NET is a compiled, .NET-based environment; you can author applications in any .NET compatible language, including Visual Basic .NET, C#, and JScript .NET. Additionally, the entire .NET Framework is available to any ASP.NET application. Developers can easily access the benefits of these technologies, which include the managed common language runtime environment, type safety, inheritance, and so on.

ASP.NET has been designed to work seamlessly with WYSIWYG HTML editors and other programming tools, including Microsoft Visual Studio .NET. Not only does this make Web development easier, but it also provides all the benefits that these tools have to offer, including a GUI that developers can use to drop server controls onto a Web page and fully integrated debugging support.

2.3.3.3 JSP

Java Server Pages (JSP) technology allows web developers and designers to rapidly develop and easily maintain, information-rich, dynamic web pages that leverage existing business systems. As part of the Java family, JSP technology enables rapid development of web-based applications that are platform independent. Java Server Pages technology separates the user interface from content generation enabling designers to change the overall page layout without altering the underlying dynamic content.

Java Server Pages technology uses XML-like tags and scriptlets written in the Java programming language to encapsulate the logic that generates the content for the page. Additionally, the application logic can reside in server-based resources (such as JavaBeans component architecture) that the page accesses with these tags and scriptlets. Any and all formatting (HTML or XML) tags are passed directly back to the response page. By separating the page logic from its design and display and supporting a reusable component-based design, JSP technology makes it faster and easier than ever to build web-based applications.

Java Server Pages technology is an extension of the Java Servlet technology. Servlets are platform-independent, 100% pure Java server-side modules that fit seamlessly

into a web server framework and can be used to extend the capabilities of a web server with minimal overhead, maintenance, and support. Unlike other scripting languages, servlets involve no platform-specific consideration or modifications; they are Java application components that are downloaded, on demand, to the part of the system that needs them. Together, JSP technology and servlets provide an attractive alternative to other types of dynamic web scripting/programming that offers platform independence, enhanced performance, and separation of logic from display, ease of administration, extensibility into the enterprise and most importantly, ease of use.

2.3.3.4 PHP

PHP (recursive acronym for "PHP: Hypertext Preprocessor") is a widely-used Open Source general-purpose scripting language that is especially suited for Web development and can be embedded into HTML. Its syntax draws upon C, Java, and Perl, and is easy to learn. The main goal of the language is to allow web developers to write dynamically generated web pages quickly, but you can do much more with PHP. PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.

PHP started as a quick Perl hack written by Rasmus Lerdorf in late 1994. Over the next two to three years, it evolved into what we today know as PHP/FI 2.0. PHP/FI started to get a lot of users, but things didn't start flying until Zeev Suraski and Andi Gutmans suddenly came along with a new parser in the summer of 1997, leading to PHP 3.0. PHP 3.0 defined the syntax and semantics used in both versions 3 and 4.

PHP has undergone major architecture changes since version 3. First of all, the language parser itself has become a self-contained component called The Zend Engine. Secondly, PHP function modules, now called PHP extensions, are also basically self-contained. Thirdly, there is a Web server abstraction layer called SAPI that greatly simplifies the task of adding native support for new Web servers. SAPI also has the advantage of increasing PHP 4 stability, in addition to supporting multi-threaded Web servers.

2.3.4 The Operating Systems

2.3.4.1 Microsoft Windows 2000 Server

Microsoft Windows 2000 Server is built on the NT technology that has proven to be very stable. It is designed to meet the needs for businesses of all sizes, from small, centralized organizations to the largest distributed enterprise. Windows 2000 Server integrates standards-based directory, Web applications, network, file and print services with powerful end-to-end management and reliability to provide the best foundation for integrating your business with the Internet.

Internet-ready

Windows 2000 Server will enable organizations to readily pursue Internet-based solutions and opportunities. With comprehensive Web, security and communication technologies built-in, and the scalability and performance to handle the demands of Internet traffic, Windows 2000 Server delivers a unique, Internet-enabled platform on which to take advantage of the Business Internet.

Windows 2000 Server comes with the built-in Web services of Internet Information Services 5.0 (IIS), and support for the Internet development language XML. Integrated communications services, such as Virtual Private Networking and Remote Access Services, coupled with comprehensive security options, enable you to securely connect mobile employees, branch offices, partners and customers to your network.

More Reliable

Windows 2000 Server will enable organizations to minimize network interruptions to end-users. With system architecture improvements for higher server uptime, fault tolerant and redundant systems for increased availability, and online configuration and maintenance capabilities, Windows 2000 Server delivers you the confidence that your servers will be up and running, and your organization will be open for business.

High System Uptime with significant improvements to the core operating system, such as improved memory management, reduces unplanned downtime. New file protection capabilities prevent new software installations from replacing essential

system files and causing failures. Server and Network Availability are increased with the use of redundant directory, networking and file services.

Easier to Use and Manage

Windows 2000 Server will increase efficiencies and productivity across your organization. With improvements that make the system easier to deploy, manage and use, powerful centralized administration enabled by the Active Directory service, and a standards-based approach to interoperability with your existing systems, Windows 2000 Server will increase efficiencies of your IT staff, end users and systems.

Build and deploy servers faster with configuration wizards that set up system services, including Active Directory and DNS, and tools that deploy copies of that system image across multiple servers.

Broad Hardware Support

Windows 2000 Server supports the latest advances in networking and peripheral hardware, including USB devices, high bandwidth and directory-enabled networking devices, to ensure the platform you build today takes advantage of the latest technology advances and supports your future investments.

Interoperability with client systems, including previous Windows systems, Macintosh and UNIX; server systems including Novell NetWare and UNIX; directories including Microsoft Exchange, Novell NDS, and others; and network devices and peripherals, helps ensure Windows 2000 Server will work with systems you already have in place.

2.3.4.2 Red Hat Linux 7.3 Professional

Red Hat Linux 7.3 Professional is a versatile solution for small networks and small businesses. From personal production to basic Web serving, Red Hat Linux Professional contains everything needed for a stable and secure working environment. For help with basic configuration and maintenance, Red Hat support and Red Hat Network Basic Service are included.

Small Network Setup

System administrator can use Kickstart robot to install many systems at one time. Firewall setup is configured as early as in installation phase. User is allowed to choose between server, workstation, laptop or custom installation.

System Configuration

Linux is able to be deployed for heterogeneous networks make it very ideal for enterprise installation. Now, Red Hat Linux 7.3 is delivered with a complete set of graphical tools in control panel for easier management and configuration.

Flexible Server Options

This version of Linux comes together with the small network servers included: web, mail, file & print services. Linux servers are able to work in multiple-OS networks.

Multi-use Versatility

In this version, the productivity tool such as StarOffice 5.2 office suite is provided. Better graphical user interfaces such as the point and click desktops GNOME 1.4 and KDE 3.0. Lastly, the journaling file system used in Linux provides reliable data access and management.

Security & Management

User has the full control over the system security by turning on or off services as needed. The Red Hat Linux 7.3 comes with simplified system maintenance and immediate notification feature for latest security updates from Red Hat Network.

2.3.5 The Web Servers

2.3.5.1 Microsoft IIS 5.0

Internet Information Services 5.0 has many new features to help Web administrators to create scalable, flexible Web applications. This version, which comes as part of the Windows 2000 Server operating system, consists of three major services which

are the World Wide Web (www) server, File Transfer Protocol server and the Simple Mail Transfer Protocol server.

The advantage of IIS can be explained in term of its four major features below:

- IIS 5.0 comes with several industry standard security features such as the digest authentication, secure connection through SSL 3.0 and TSL 1.0, server gated cryptography with strong 128-bit encryption, IP and Internet domain restriction, Kerberos v5, certificate storage, Fortezza, and a security wizard that enables the administrator to easily configure the IIS based on its policy.
- Administration in IIS 5.0 has been improved with support for remote administration, terminal services, accounting and throttling processes, easy start, stop and restart IIS and lastly customizable error messages.
- IIS 5.0 conforms to the latest Internet standard such as HTTP 1.1, HTTP compression, Web Distributed Authoring and Versioning (WebDAV), and multiple sites one IP address feature.
- Programmability in IIS 5.0 has been tremendously improved with the support of ASP 3.0, application protection and ADSI 2.0.

2.3.5.2 Apache Server 1.3

The Apache httpd server is a powerful, flexible and HTTP/1.1 compliant web server. It is highly configurable and extensible with third-party modules which makes it very flexible. The Apache httpd server can be customized by writing 'modules' using the Apache module API.

It comes with full source code and unrestrictive license. It is able to run on Windows NT/9x, Netware 5.x and above, OS/2, and most versions of Unix, as well as several other operating systems makes it deployable in various existing environment.

Below are the extra features of Apache httpd Server:

- DBM databases for authentication allows you to easily set up password-protected pages with enormous numbers of authorized users, without bogging down the server.

- Customized responses to errors and problems allows you to set up files, or even CGI scripts, which are returned by the server in response to errors and problems, e.g. setup a script to intercept errors and perform on-the-fly diagnostics for both users and yourself.

- Unlimited flexible URL rewriting and aliasing may be declared in the config files. In addition, a powerful rewriting engine can be used to solve most URL manipulation problems.

- Virtual Hosts is a much requested feature, sometimes known as multi-homed servers. This allows the server to distinguish between requests made to different IP addresses or names (mapped to the same machine).

- You can configure Apache to generate logs in the format that you want. In addition, on most Unix architectures, Apache can send log files to a pipe, allowing for log rotation, hit filtering, real-time splitting of multiple vhosts into separate logs, and asynchronous DNS resolving on the fly.

2.3.6 The Database Management System

2.3.6.1 Microsoft SQL Server 2000

As the most recent major release of SQL Server, SQL Server 2000 builds upon the modern, extensible foundation of SQL Server 7.0, a critical release in Microsoft's database lineup and one in which much of the SQL Server product was both re-architected and rewritten. Microsoft SQL Server 2000 is the complete database and analysis solution for rapidly delivering the next generation of scalable Web applications.

SQL Server 2000 includes rich support for XML and HTTP; performance and availability features to partition load and ensure uptime; and advanced management and tuning functionality to automate routine tasks and lower total cost of ownership. Additionally, SQL Server 2000 takes full advantage of Windows 2000, including support for the Active Directory™ service, and up to 32 processors and 64 GB of RAM.

Besides providing the necessary enterprise "abilities" for data management and analysis, SQL Server 2000 helps deliver *agility*. Agility is a characteristic of organizations that can rapidly adapt to changing environments for competitive advantage. By going beyond simple data storage/retrieval and offering true business intelligence functionality, SQL Server 2000 allows business to understand their data and act decisively on analysis results.

It includes features and technologies that make it:

Fully Web-Enabled

SQL Server 2000 provides extensive database programming capabilities built on Web standards. Rich XML and Internet standard support give you the ability to store and retrieve data in XML format easily with built-in stored procedures. You can also use XML updategrams to insert, update and delete data easily.

Highly Scalable and Reliable

Achieve unparalleled scalability and reliability with SQL Server 2000. With scale up and scale out capabilities, SQL Server meets the needs of demanding ecommerce and enterprise applications.

Deliver Fastest Time-to-Market

SQL Server has long been considered the fastest way to build, deploy and manage e-commerce, line of business, and data warehousing solutions. Research studies by independent firms have demonstrated not only that SQL Server is easier to use than its primary competitor, but also that it demonstrates significantly lower total cost of ownership.

2.3.6.2 MySQL

MySQL Server was originally developed to handle large databases much faster than existing solutions and has been successfully used in highly demanding production environments for several years. Though under constant development, MySQL Server today offers a rich and useful set of functions. Its connectivity, speed, and security make MySQL Server highly suited for accessing databases on the Internet. MySQL is now a full-fledged database server with a robust feature, optimized for speed, reliability, flexible and high load support.

The noted technologies and features of MySQL are:

The Scalable Architecture

The MySQL Database Software is a client/server system that consists of a multithreaded SQL server that supports different backend, several different client programs and libraries, administrative tools, and a wide range of programming interfaces (APIs).

MySQL Server is fully multi-threaded using kernel threads. This means it can easily use multiple CPUs if available and uses a multi-threaded library which you can link into your applications to get a smaller, faster, easier-to-manage product.

Portability

MySQL is written in C and C++ language and has been tested for a broad range of different compilers on various platforms. APIs support for various languages such C, C++, Eiffel, Java, Perl, PHP, Python, and Tcl.

Performance

To ensure the performance, techniques such as SQL functions implemented through a highly optimized class library, very fast B-tree disk tables with index compression, very fast thread-based memory allocation system, very fast joins using an optimized one-sweep multi-join and in-memory hash tables which are used as temporary tables are applied in MySQL.

Security

A privilege and password systems that are very flexible and secure, and allows host-based verification. Passwords are secure because all password traffic is encrypted when you connect to a server.

Connectivity

Clients may connect to the MySQL server using TCP/IP Sockets, Unix Sockets (Unix), or Named Pipes (NT). ODBC (Open-DataBase-Connectivity) support for Win32 systems.

2.3.7 The Development Tools

2.3.7.1 Microsoft Visual Studio .NET Professional

Visual Studio .NET is the comprehensive tool for building next-generation applications for Microsoft Windows and the Web. With rapid design, development, and deployment support for XML Web services, high-performance data-driven solutions, and server-side visual designers, Visual Studio .NET delivers superior functionality for streamlining business processes, enabling new business opportunities.

Visual Studio .NET also dramatically increases developer productivity, enabling developers to build solutions for the broadest range of clients, including applications for the Web, Windows, and thin-client devices. Finally, the single, shared Visual Studio .NET integrated development environment (IDE) and a choice of programming languages—including Microsoft Visual Basic, Microsoft Visual C++, and Microsoft Visual C#—allow developers to build powerful applications quickly.

The three most notable features of Microsoft Visual Studio .NET are:

- **Build the next-generation Internet.** Developers can employ XML web services and built-in Microsoft ADO.NET tools to build high-performance, data-driven applications that target a variety of platforms.
- **Develop powerful applications quickly.** With an integrated development environment (IDE) for all languages, developers can take advantage of a

common toolbox, debugger, and task window, greatly reducing the developer learning curve.

- **Create solutions that span any device and integrate with any platform.** Visual Studio .NET Professional gives developers the tools for integrating solutions across operating systems and languages.

2.3.7.2 Sun One 4.0 Community Edition

Sun One 4.0, Community Edition is an entry-level product that offers an extensible, cross-platform, integrated development environment for creating simple Java technology-based desktop applications. The product is available for download at no cost and installs in a few minutes. It forms the foundation of the Sun One development environments and includes the user interface, editor, compiler, debugger, and GUI development tools for integrated visual design.

Through a comprehensive set of wizards, utilities, and templates, Sun One software accelerates development time and increases productivity. It offers the advantage of time-saving functions, including dynamic source code completion within the editor and a multithreaded debugger. These highly integrated tools, including an editor, debugger, and graphical design environment, form the foundation for the Java technology.

These tools leverage the tremendous ease of use of the Java language. In addition, the rapid application development capabilities provide a complete solution to the development of a broad range of applications:

- Distributed transaction middleware
- Execution of components in a managed server environment
- Distributed systems administration
- Integration of existing, legacy-based applications with customer-focused, business-to-business e-commerce initiatives

2.3.7.3 Microsoft Office XP Professional

Microsoft Office is a very complete productivity suite. It contains applications such a word processing, spreadsheet, presentation, graphical editor, database management, e-mail and much more.

This integration provides ample functionalities for many general purposes while providing other advance features such as smart tags, task panes, integrated e-mail, document recovery, and send for review which makes it easier to use, increase your productivity and enabling collaboration with others.

2.3.7.4 Microsoft Internet Explorer 6.0

Microsoft Internet Explorer 6.0 is the latest version of web browser from Microsoft. This browser is tightly integrated in Microsoft Windows operating systems. Microsoft Internet Explorer is now the most widely used web browser.

The Internet Explorer 6.0 comes with many features such as simple and familiar interface, security and privacy features, content control support, connection wizard, and accessibility for less fortunate groups of users.

Also technically, Microsoft Internet Explorer 6.0 supports and conforms to various Internet standards such as XHTML, XML, DOM, CSS, SSL 3.0, TSL 1.0, FTP and many others. It is also support new features by using the latest add-ins or plug-ins such as XML parser, Java Virtual Machine, Shockwave and others.

2.3.7.5 Adobe Photoshop 6.0

Adobe Photoshop 6.0 is a professional standard graphical package that various kinds of graphical authoring and editing functionalities. It contains a lot of built in filters that enables graphic manipulation. Adobe Photoshop supports various graphic formats and animation too. Adobe Photoshop can also be used to optimize graphic for web used.

2.3.7.6 Rational Rose Enterprise 2000

Rational Rose 2000 Enterprise Edition is a very good example of integrated CASE tools as its covers various stages and functionalities needed in software development lifecycle. Rational Rose 2000 provides a wide range of flexibility, supporting multiple methodologies and the creation of custom attributes and reports. Its publishing features enabled us to communicate with the many people involved in the development of the architecture and its use throughout the organization.

Rational Rose 2000 offers several noted key features: multi-user repository for creating, storing, reviewing and sharing information about relationships, events and objects; multiple methodology support that integrates UML, system engineering and reverse engineering; extensibility, or ability to customize business rules to the client's needs and transparently merge custom attributes into the standard repository; Web and Office-Suite reporting; and several enterprise standard framework support, for various languages, and from high-level conceptual models to logical, object or data models.

During the system modeling and design, visual modeling is a key technology for the successful deployment of a software solution. Using products based on standards such as UML and XML, with the automation of code generation from the modeled business processes, will ensure the timely and rapid development and deployment required by businesses today. This cycle will also ensure that the solution deployed matches the true business processes.

Developers is also provided with vast numbers of built in components, data types, stereotypes and notations to be used on the fly which eases and quicken the design and modeling process. Also, Rational Rose 2000 is flexible enough to support custom or user defined components, data types or stereotypes without constraining user to only the predefined repository objects.

3 METHODOLOGY

A methodology is a collection of methods, procedures, techniques, tools and paradigms for solving a class of problem.

3.1 The Object-oriented Methodology (OOM)

3.1.1 The Overview

Object-oriented methodology is an approach to software design based on objects and classes. The abstraction concept forms the basis of object-orientation. An object is an abstraction that contains data (attributes) and behaviours (methods). An object is an instance of a class. A class describes a set of objects with common attributes and behaviours.

3.1.2 The Phases

There are few variations in the object-oriented methodology such as the Booch Methdology from Booch, Object–oriented Analysis and Design (OOAD) from Coad-Yourdon, Object Modelling Technique (OMT) from Rumbaugh and so on. But roughly speaking, all these variations agree to the common phases such as:

The Requirement Elicitations Phase

The Analysis Phase

The Design Phase

The Implementation Phase

The Deployment Phase

The Maintenance Phase

3.1.3 Why OOM?

There several advantages of using the object-oriented methodology in software development such as:

- Modeling real world objects enables software units can be defined more naturally and provides high level of abstraction
- Offering better support for reusability and modularity by encapsulating data and behaviors together

- Provides information hiding by communication between object are by message passing across well defined interfaces and access privilege
- Separate interfaces from implementation enables changes on class's implementation but not affecting the clients as long as the interface remains the same

3.1.4 The UML

The Unified Modeling Languages is a set of standardize notations that resulted from the brainchild of Grady Booch, James Rambaugh and Ivar Jacobson. Prior to the UML, there was no clear leading modeling language. Users had to choose from among many similar modeling languages with minor differences in overall expressive power. Most of the modeling languages shared a set of commonly accepted concepts that are expressed slightly differently in various languages.

The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of the best engineering practices that have proven successful in the modeling of large and complex systems.

Developing a model for an industrial-strength software system prior to its construction or renovation is as essential as having a blueprint for large building. Good models are essential for communication among project teams and to assure architectural soundness. We build models of complex systems because we cannot comprehend any such system in its entirety. As the complexity of systems increase, so does the importance of good modeling techniques.

UML has been design for a wide range of modeling. Meanwhile, the system development focuses on three different models, which are:

The Functional Model which describes the functionality of the system from the user's point of view. Usually, it is represented in UML using the use case diagram.

The Object Model which describes the structure of a system in terms of the objects, attributes, associations and operations. Usually, it is described in UML using the class diagram.

The Dynamic Model describes the internal behaviour of a system. It is usually represented by sequence diagram, statechart diagram and activity diagram in UML.

3.2 The Phase Development Software Life Cycle

3.2.1 The Overview

The software process model or better known as the software life cycle model defines the set of activities that are found in the software development process, specifies the input and output of each activity and also defines the roles played in each of the activities.

There are many different kinds of software process models available and each of then has it commonalities, differences, advantages and disadvantageous over the others. Among the available software process model, I have chosen to use the Phase Development Model using both Incremental and Iteration Approaches.

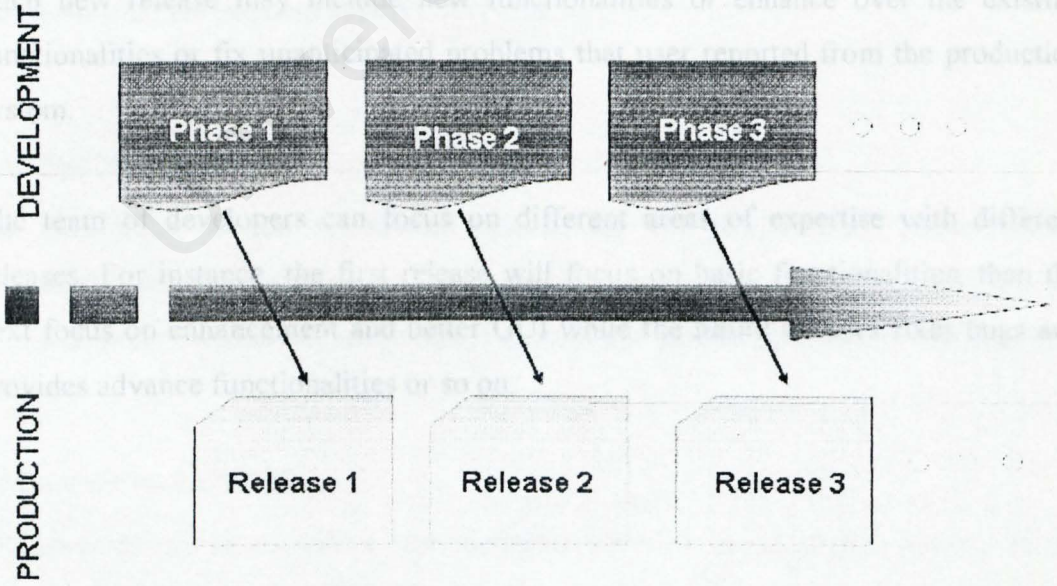


Figure 3-1 The Phased Development Software Process Model

3.2.2 Why this Approach?

There are several reasons behind on why this approach is used for this project:

Phased development is better suited for large and complex systems. It enables developers to focus on each smaller module of the system on each release without being distracted by other functionalities of the whole complex system offers. This enables system to be developed according to user's requirements.

Phased development reduces the cycle time of software development. The system is designed and delivered in small pieces, enabling users to have the basic functionalities while others are being developed.

Training can begin on an early release. The training process allows developers to observe and response to user's needs and working style by making the necessary modifications and enhancements for the next release.

Development and production are carried out parallel and simultaneously which means that customers can start using the software in production system while developers can enhance and build the next release in development system without interruption to the customer production.

Each new release may include new functionalities or enhance over the existing functionalities or fix unanticipated problems that user reported from the production system.

The team of developers can focus on different areas of expertise with different releases. For instance, the first release will focus on basic functionalities, then the next focus on enhancement and better GUI while the future releases fixes bugs and provides advance functionalities or so on.

4 SYSTEM ANALYSIS

4.1 Design Requirements

4.1.1 Functional Requirements

4.1.1.1 The FCSITPortal

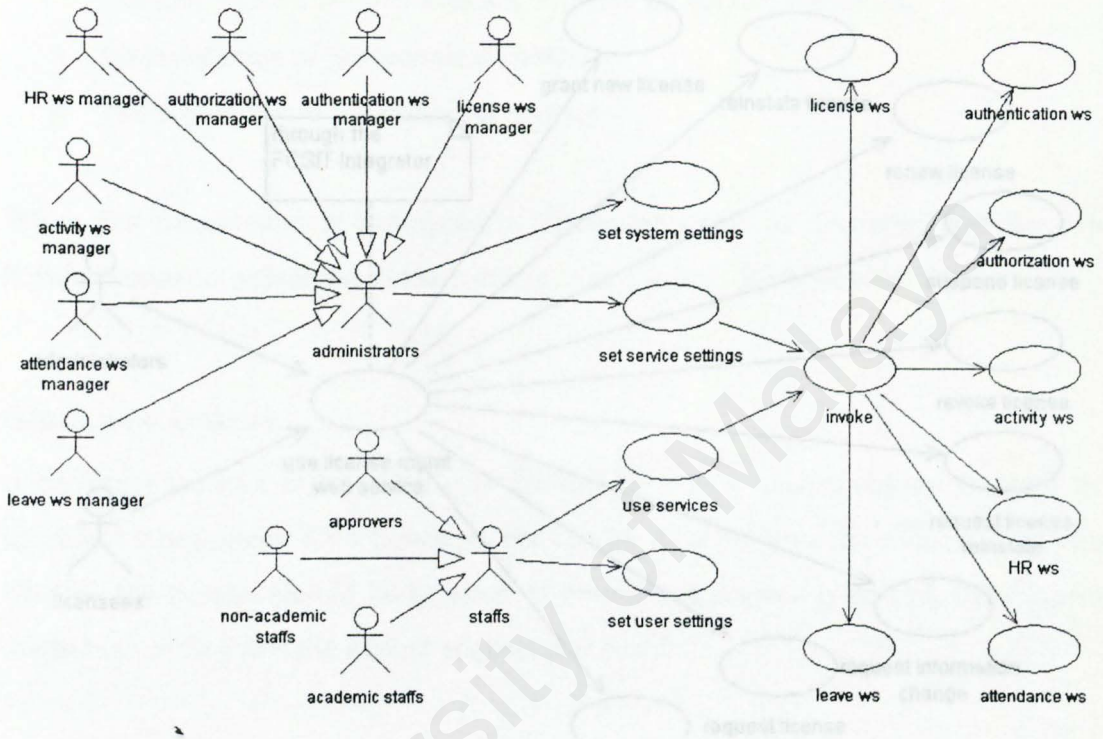


Figure 4-1 Use Case diagrams of the FCSITPortal

Set System Setting

The administrators can set several options for the FCSITPortal. These options include the site title settings, add or remove modules to enhance the FCSITPortal, add, edit or remove tab and set the page content and layout dynamically on runtime.

Set services settings

The services managers and/or administrators are able to set several options to the web services through this FCSITPortal. These include the setting of system security, activity management, attendance and leave settings.

Use the services

The users of the FCSIT Web Services will interact with the XML Web Services via the GUI provided by this FCSITPortal.

4.1.1.2 The License Management Web Service

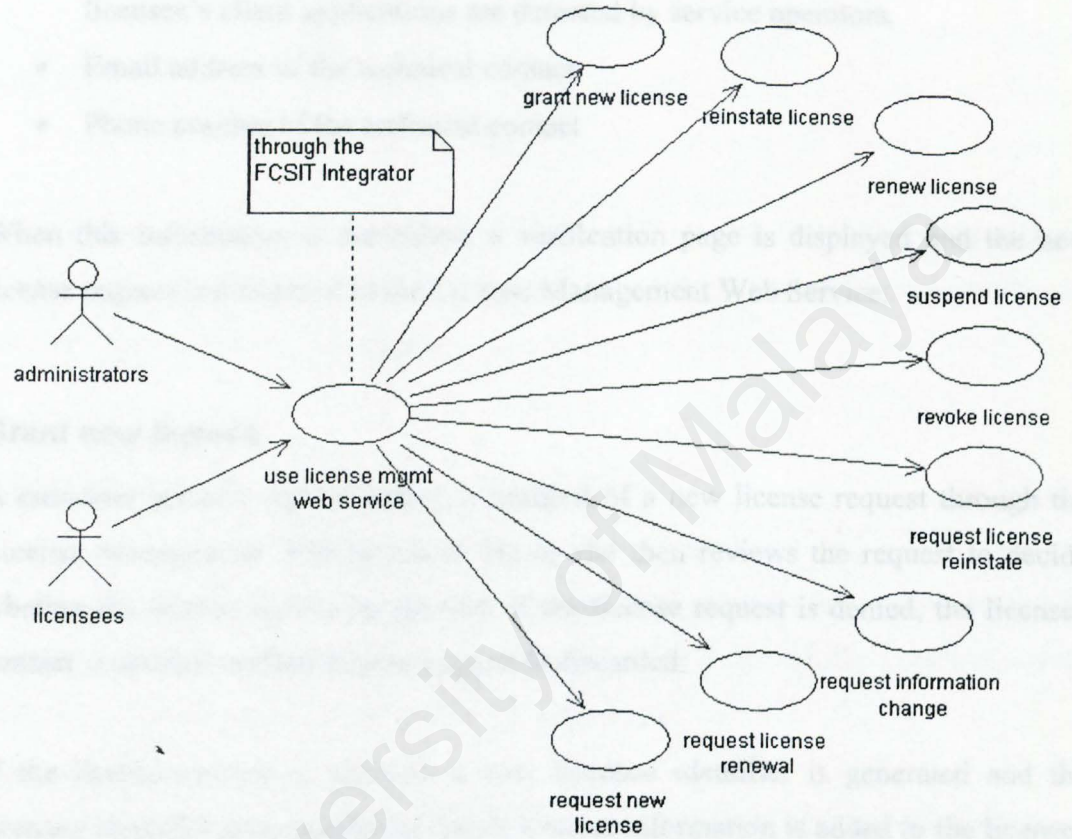


Figure 4-2 Use Case diagrams of the License Management Web Service

Request new license

A person who is interested in using the FCSIT Web Services visits the License Management Web Service web site. This web site includes one or more pages that describe the features of the web services, requirements for using the service, and licensing terms. The web site also includes a page that lets the person request a new license to the FCSIT Web Services. If the person selects this option, he or she is prompted to enter the following information *(Note: Subject to change)*:

- Company name

- Company address
- Name of licensee contact that should receive license information and renewal notifications
- Email address of the licensee contact
- Phone number of the licensee contact
- Name of technical contact that should be contacted if problems with the licensee's client applications are detected by service operators.
- Email address of the technical contact
- Phone number of the technical contact

When this information is submitted, a verification page is displayed and the new license request is submitted to the License Management Web Service.

Grant new license

A customer account representative is notified of a new license request through the License Management Web Services. He or she then reviews the request to decide whether the license should be granted. If the license request is denied, the licensee contact is notified and the license request is discarded.

If the license request is granted, a new licensee identifier is generated and the licensee identifier plus supplied licensee contact information is added to the licensee data store. An email message will be sent to the licensee contact containing the licensee identifier and information on how to access developer and operations documentation for the FCSIT Web Services.

Request change to licensee contact information

A licensee contact or technical contact visits the License Management Web Service web site. One of the options on the site is to update licensee contact information. If the contact selects this option, he or she is prompted to enter the licensee identifier. The web site retrieves the following information from the licensee data store and displays the information in a data entry form *(Note: Subject to change)*:

- Company name

- Company address
- Name of licensee contact that should receive license information and renewal notifications
- Email address of the licensee contact
- Phone number of the licensee contact
- Name of technical contact that should be contacted if problems with the licensee's client applications are detected by service operators.
- Email address of the technical contact
- Phone number of the technical contact

The contact may modify any of the fields. When the updated information is submitted, a verification page is displayed and the change request is submitted to the License Management Web Service.

Update licensee contact information

A customer account representative is notified of a change request through the License Management Web Service. He or she then reviews the request to decide whether it should be accepted. If the change request is denied, the current licensee contact is notified and the change request is discarded.

If the change request is granted, the contact information is updated in the licensee data store. An email message is sent to the original and new licensee contacts indicating what changes have been made to the contact information.

Request license renewal

If a licensee contact agrees to the renewal licensing term which causes the Favorites License Management Web Service to be notified of the renewal request. The licensee contact can simply visit the License Management Web Service web site. One of the options on the site is to renew a license. If the licensee contact selects this option, he or she is prompted to enter the licensee identifier.

The License Management Web Service web site verifies that the license is about to expire and displays the possible license terms. When the licensee contact selects a license term, a verification page is displayed and License Management Web Service is notified of the renewal request.

Renew license

When the License Management Web Service is notified of a renewal request, it updates the license expiration date, marks the license as not expired and not suspended, and clears out any renewal status information in the licensee data store. An email message is sent to the licensee contact verifying that the license has been renewed and what the new expiration date is.

Notify licensee contact

Once a day, the License Management Web Service searches the licensee data store for licenses that will expire soon. A notification message will be sent to the licensee contact indicating when their current license will expire.

Revoke license

Once a day, the License Management Web Service searches the licensee data store for licenses for which that has overdue. For each such license, License Management Web Service updates the licensee record to indicate the license is expired.

Suspend license

A service operator detects that a client application is using the FSCIT Web Services inappropriately. The service operator uses the License Management Web Service to mark the licensee record in the licensee data store as temporarily suspended. License Management Web Service sends a notification message to the technical contact indicating that their license has been suspended, why the license was suspended, and how to get the license reinstated.

Request license reinstated

A technical contact visits the License Management Web Service web site. One of the options on the site is to reinstate a suspended license. If the technical contact selects this option, he or she is prompted to enter the licensee identifier. The License Management Web Service web site verifies that the license is currently suspended. If not, it displays a page indicating the license is not suspended. Otherwise, the License Management Web Service is notified of the request to reinstate the license.

Reinstate license

When the License Management Web Service is notified of a reinstate license request, it marks the license as not expired in the licensee data store. A notification message is sent to the technical contact verifying that the license has been reinstated.

4.1.1.3 The Ticket Manager Web Service for User Authentication

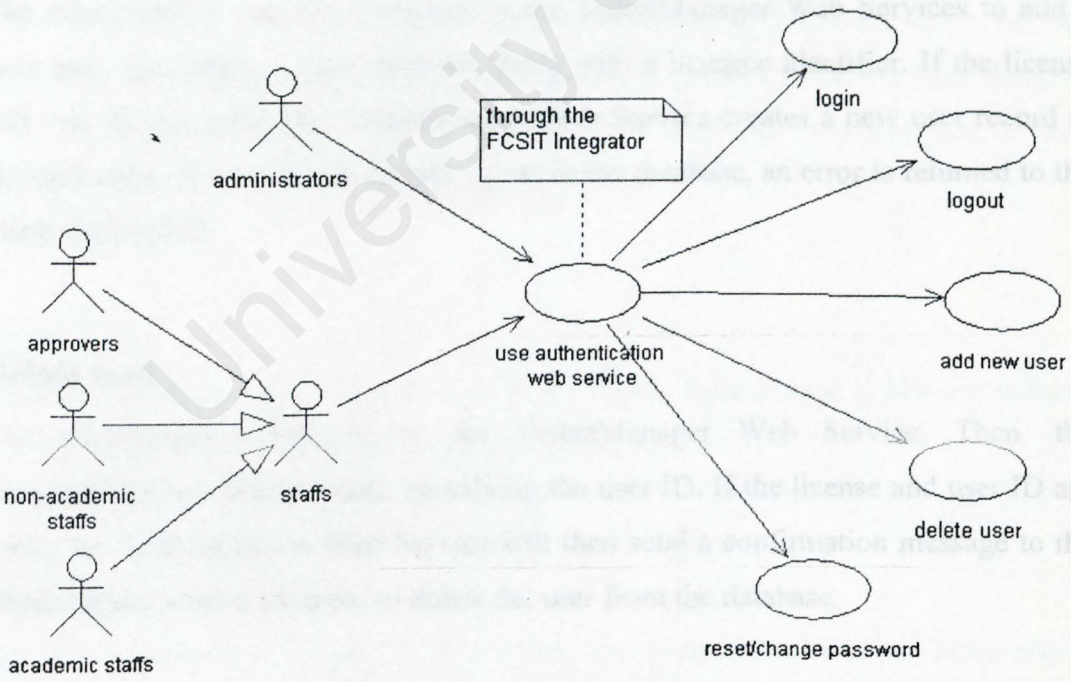


Figure 4-3 Use Case diagrams of the TicketManager Web Service

Login

The user of the FCSIT Web Services must supplies his or her user ID and password in the logon screen and click logon. This will make the TicketManager Web Service verifies his or her identity and gives the permission to use the FSCIT Web Services.

Logout

Logout means terminating the current session of the user. This prevents other unauthorized users from manipulating your information.

Change Password

The user can change his or her password only after he or she successfully login to the system. User will have to supply his or her user ID, the old password, the new password and verify the new password.

Add new user

The administrator submits a request to the TicketManager Web Services to add a new user, specifying a user identifier along with a licensee identifier. If the license and user ID are valid, the Authentication Web Service creates a new user record in the data store. If the user ID already exists in the database, an error is returned to the client application.

Delete user

The administrator logs on to the TicketManager Web Service. Then, the administrator can delete a user, specifying the user ID. If the license and user ID are valid, the Authentication Web Service will then send a confirmation message to the administrator before attempts to delete the user from the database.

4.1.1.4 The Role Manager Web Service for User Authorization

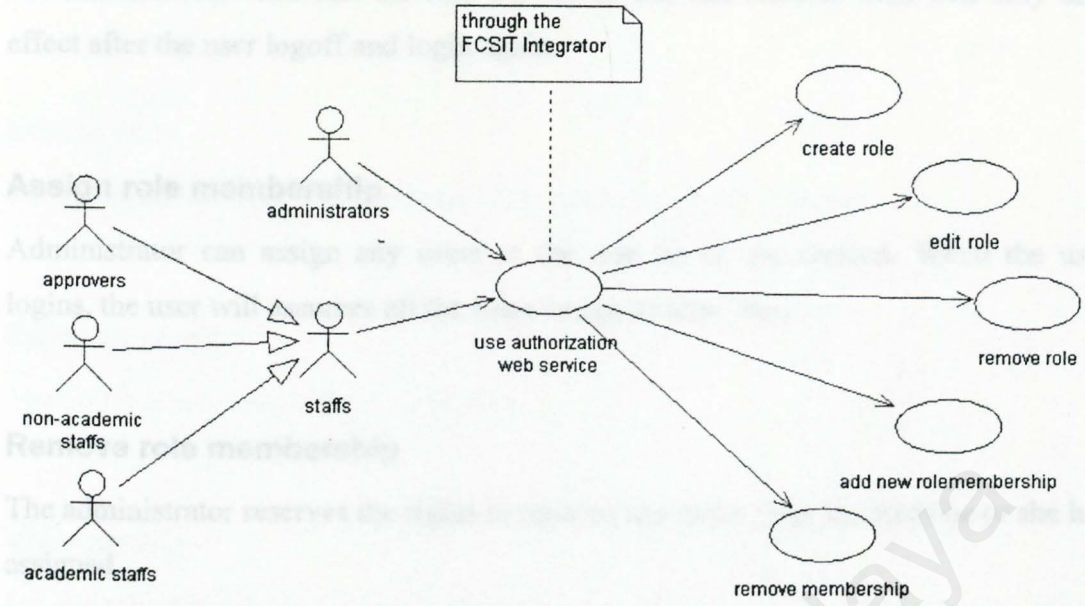


Figure 4-4 Use Case diagrams of the RoleManager Web Service

Verify role

When user logs on to the FSCITPortal, the RoleManager Web Service will verify the role of the user based on its user ID and the licensee key. Then, the RoleManager Web Services will retrieve its all its privileges that has been assigned to this particular user ID.

Create role

Only the administrator can create roles for the users. Role created is like a container that will map the role to the users assigned to this role.

Delete role

Only the administrator has the right to delete any of the roles he or she has created. Before system attempts to delete any valid role, a confirmation message will be notified to ensure the deletion.

Edit role

The administrator can edit the role that he or she has created. This will only take effect after the user logoff and login again.

Assign role membership

Administrator can assign any users to the role he or she created. When the user logins, the user will assumes all the roles assign to him / her.

Remove role membership

The administrator reserves the rights to remove any users from the roles he or she has assigned.

4.1.1.5 The HRManager Web Service

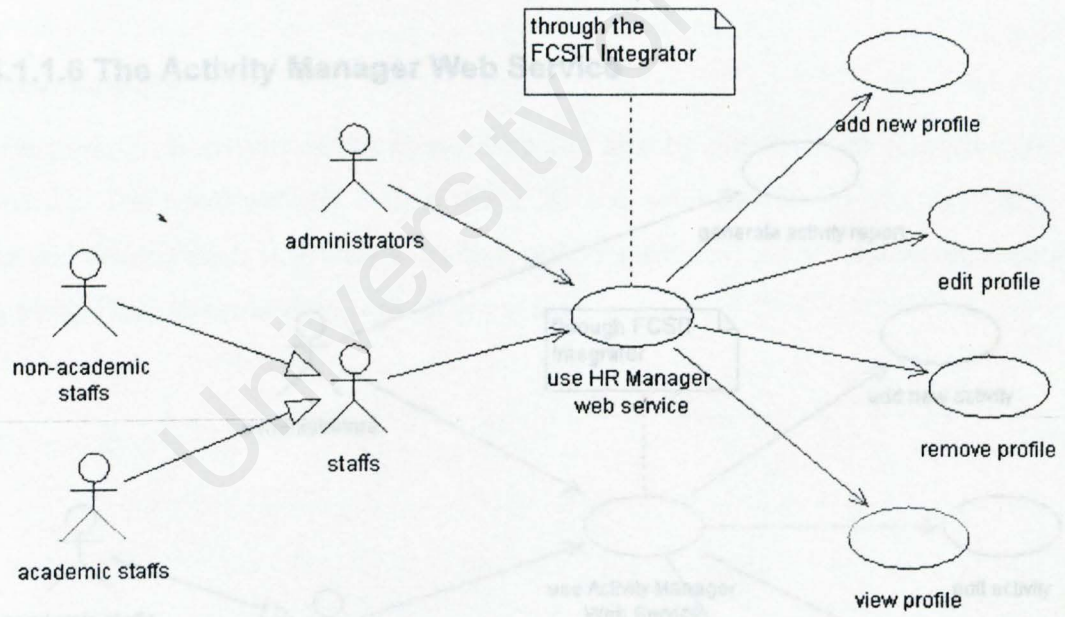


Figure 4-5 Use Case diagrams of the HRManager Web Service

Create new profile

The administrator can create a new profile for a new user.

Edit profile

The administrator can edit and make changes to the profile created.

Remove profile

The administrator can remove a profile.

View profile

Let the administrator and user to view a profile. User can view their personal details only but not others.

4.1.1.6 The Activity Manager Web Service

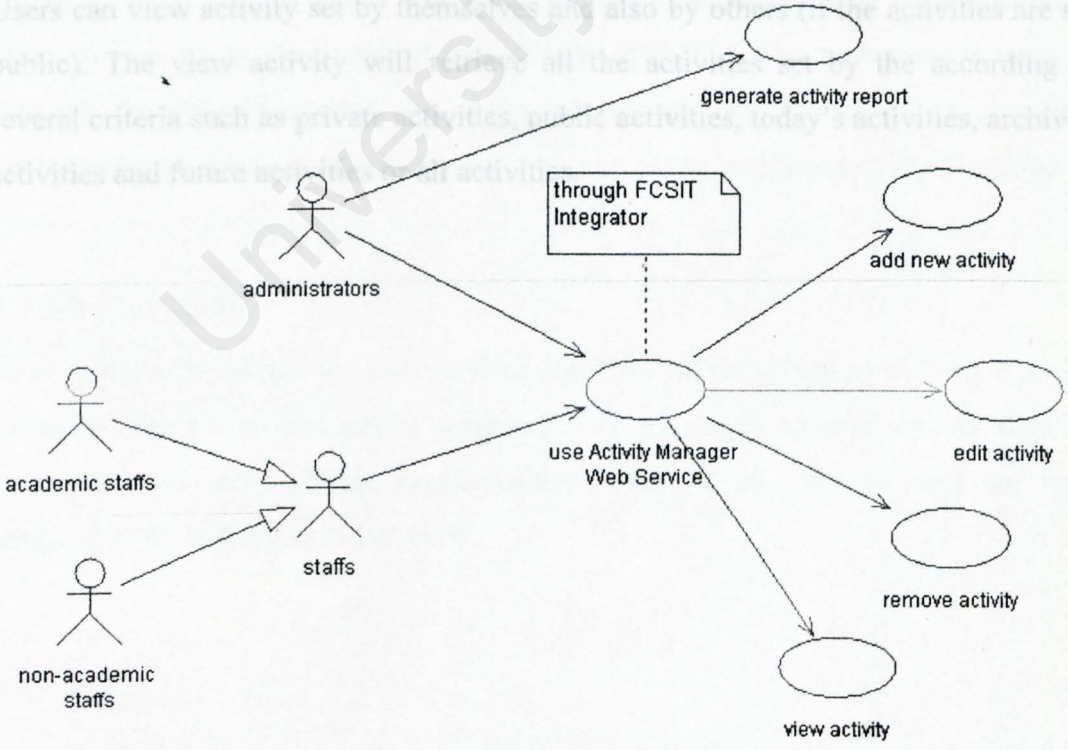


Figure 4-6 Use Case diagrams of the Activity Manager Web Service

Add activity

User can add new activity. Then user can set if the activity is personal and public. Personal activities will be view only by the owner of the activity while public activity will be able to be viewed by other users.

Remove activity

User can only remove any of its own activities. However, there is an exception where user is unable to remove his or her public activities after they have expired. However, this restriction is not applicable to his or her private activities.

Edit activity

User can only change any of its own activities. However, there is an exception where user is unable to remove his or her public activities after they have expired. However, this restriction is not applicable to his or her private activities.

View activity

Users can view activity set by themselves and also by others (if the activities are set public). The view activity will retrieve all the activities set by the according to several criteria such as private activities, public activities, today's activities, archived activities and future activities or all activities.

4.1.2 Non-functional Requirements

4.1.2.1 Interoperability

The major design requirement besides providing the basic functionalities is to enables the interoperability of these web services. They must be able to be used in system across different network, over different platforms and implementation language and vendor independent.

4.1.2.2 Reusability

The reusability of these web services is a must as one of the basic advantage of implementing web services. Other systems that wish to use the web services will just need to discover the services, get the description of the services and then invoke the services using the standard Internet protocol such as HTTP and XML.

4.1.2.3 Maintainability

Object oriented paradigm is used to design each web services so that it encapsulates the data and behaviours in within the web services thus providing information and implementation hiding. These web services communicate to other system through message passing in the standard data format of XML. This technique is used to enhance the maintainability and manageability by modification of a service's implementation does not affect the others as long as the interface remains the same.

4.1.2.4 Flexibility

To ensure the flexibility, the web services are designed to be high modularity but low cohesion. These web services is designed to be as simple as possible and each of them provides very specific functionalities. Each of the web services are also designed to be as generic as possible.

4.1.2.5 Reliability

Each layer will manipulate the data in a consistent manner and provides the access control to layer above it. By enabling access control, checking for consistency and exception handlings construction for necessary situations, the reliability of the system can be ensured.

4.1.2.6 Efficient

The web services and the integrator are designed to be performance wise by using the latest features of ASP .NET such as caching. Also, they are designed to pass as minimal data as possible, only the needed data are passed but not more. The GUI is designed using as less graphics as possible but still remain attractive by using the new features provided by DHTML and CSS.

4.1.2.7 Secure

The design of the web services also addresses several security issues such as impersonation, unauthorized access, data privacy and data integrity issues. Such compromises are tackled using the techniques of authentication, authorization and data encryption.

4.1.2.8 User-friendly

When designing the system, user friendliness, is always put in mind. Then interface of the system is designed as simple as possible, easy to access feature, and no deep linking into the system. Also the system will be well documented with the help system provided.

4.2 System Requirements

4.2.1 Hardware Specifications

Processor	Pentium III 550 Mhz or above (or equivalent x86 architecture processor)
RAM	128 MB or above
Hard disk	2 GB
Display	support VGA (recommended resolution 800x600)
Internet	connection using network card or modem 56K

Table 4-1 Estimated hardware requirements for server

The hardware specifications described above is just an estimation server machine. The hardware requirements should depend on the demand and server loads. Usually this can be observed from the response time, processor usage, memory usage, hard disk usage, paging frequency and so on.

Processor	Pentium II 200 Mhz or above (or equivalent x86 architecture processor)
RAM	64 MB or above
Hard disk	1 GB
Display	support VGA (recommended resolution 800x600)
Internet	connection using network card or modem 56K

Table 4-2 Estimated hardware requirements for client

The hardware specifications described above is just an estimation end user machine. The hardware requirements also depend on user's machine work load, memory size and the network speed.

4.2.2 System Platform

Windows 2000 Server is chosen to be the server platform because of several identified advantages and necessities to run the Web Services. The advantages of using Windows 2000 Server are that it comes along with the IIS 5.0 Web Server and the Internet Explorer 5.5, easy to setup, easy to configure, highly secure and a stable platform for serving Web Services.

The necessities of using Windows 2000 Server are it is the supported platform to installed Microsoft SQL Server 2000, Microsoft .NET framework, Microsoft Web Services Development Kit, Microsoft Visual Studio .NET and Microsoft Office 2000.

As for the client, the platform recommended is Windows 2000 Professional with Internet Explorer 5.0 and above and .NET framework and Microsoft Web Services Development Kit installed though other operating system such as Windows family, Linux or so on with the appropriate web browser such as Internet Explorer 5.0 and above, Netscape 6.0 and above or so on are estimated to be supported but not fully tested.

4.2.3 Web Server

The IIS 5.0 is chosen to publish the Web Services. This is because of several benefits and also the necessity that have been identified. The advantage of using the IIS 5.0 is that is already built in with the Windows 2000 Server, easy to configure, reliable high uptime, provide various security features and support the latest Internet standard such as HTTP 1.1.

The necessity that has been identified is the support for ASP .NET Web Services after the .NET framework installed. Other popular web servers that identified do not support ASP .NET. The additional feature that the IIS 5.0 offers is the SMTP service that I think it might be useful for the future enhancement of the Web Services to send email notifications.

4.2.4 Database Server

I have chosen Microsoft SQL Server 2000 as the database server for the Web Services. This is because of its high scalability, fast performance, security features, and the support for stored procedure.

Other features are the support for XML, easy to maintain, the availability and support of SQL database connection driver in .NET framework, well documented, easy to use interface and easy to maintain while offering all the necessary functionalities.

As a conclusion, by using the SQL Server 2000, database design, setup and configuration for the Web Services can be done in a short time without much hassle.

4.2.5 Framework

I have chosen to use Microsoft .NET framework for developing the FCSIT Web Services. This is because it supports various implementation languages, provides a lot of reusable classes, provide a stable and secure runtime, very well documented and its highly support for Web Services development.

The other identified framework is the J2EE. I decided not to choose J2EE because of the lack support for Web Services though it is a proven technology for developing enterprise applications. With the lack of support for the Web Services, the development is much more tedious and complicated as compared to .NET framework. To develop a Web Service using the J2EE requires download for some additional libraries plug-ins to parse the SOAP and WSDL document in the XML format.

Also, the documentation and samples on how to develop Web Services using J2EE is scarcely available and restricted as compared to .NET.

4.2.6 Languages

Various implementation languages are supported in the .NET framework. However, the language that I will use in developing the FCSIT Web Services is the ASP .NET. However, in the same time, I will be using C# .NET and VB .NET as the language that run in the code behind as ASP .NET supports languages like C#.NET, VB .NET and Jscript .NET.

ASP .NET offers several noted features that are very much important to the development of the web services such as the readily support for web services by providing several reusable classes, the common language runtime in which type safety, inheritance, language interoperability, and versioning is supported and compiled code and caching which means faster performance and better security.

4.2.7 Tools

The development tools that will be used for developing the FSCIT Web Services is the Microsoft Visual Studio .NET. This tools provides a complete and easy to use Integrated Development Environment (IDE) features that enables developers develop their software much easier, faster and more convenience way.

This tool offers several smart and advance features such as the IntelliSense that will predicts the code that is going to be written by the developers, provides automatic code generations for different kinds of projects, comprehensive help to assist the user, debugger for debugging code, connection to database server and built in web browser.

When developing Web Services using the Microsoft Visual Studio .NET, many of the complex coding is generated automatically by this tool without your intervention. To test the web services developed, this tools automatically generates the necessary interface for the testing purposes. If the developers are to consume a web service instead, the tools easily help the developers bind and generate the proxy that invoke the web services automatically.

5 SYSTEM DESIGN

5.1 Architectural Design

5.1.1 The Overview

A Web Service is programmable application logic accessible using standard Internet protocols. Web Services combine the best aspects of component-based development and the Web. Like components, Web Services represent black-box functionality that can be reused without worrying about how the service is implemented. Unlike current component technologies, Web Services are not accessed via object-model-specific protocols, such as the distributed Component Object Model (DCOM), Remote Method Invocation (RMI), or Internet Inter-ORB Protocol (IIOP).

Instead, Web Services are accessed via ubiquitous Web protocols and data formats, such as Hypertext Transfer Protocol (HTTP) and Extensible Markup Language (XML). Furthermore, a Web Service interface is defined strictly in terms of the messages the Web Service accepts and generates. Consumers of the Web Service can be implemented on any platform in any programming language, as long as they can create and consume the messages defined for the Web Service interface.

In a raw draft, the FCSIT Web Services architecture consists of a group of end users, in this instance the staffs and the students of FSCIT, a group licensee or the SOAP clients to the FCSIT Web Services, in this instance is represented by the FCSIT Integrator and a group of Web services, in this instance the FCSIT Web Services such as the Authentication, Authorization, License Management, Activity, Attendance and Leave Web Services.

The FCSIT Integrator provides the user interface, whereas the FCSIT Web Services are the programming interfaces that expose operations. The FCSIT Web Services make the underlying business logic functionality available. Each FCSIT Web Service uses its own database and uses the services from other FCSIT Web Services. A diagram of the high level design is shown in the following figure.

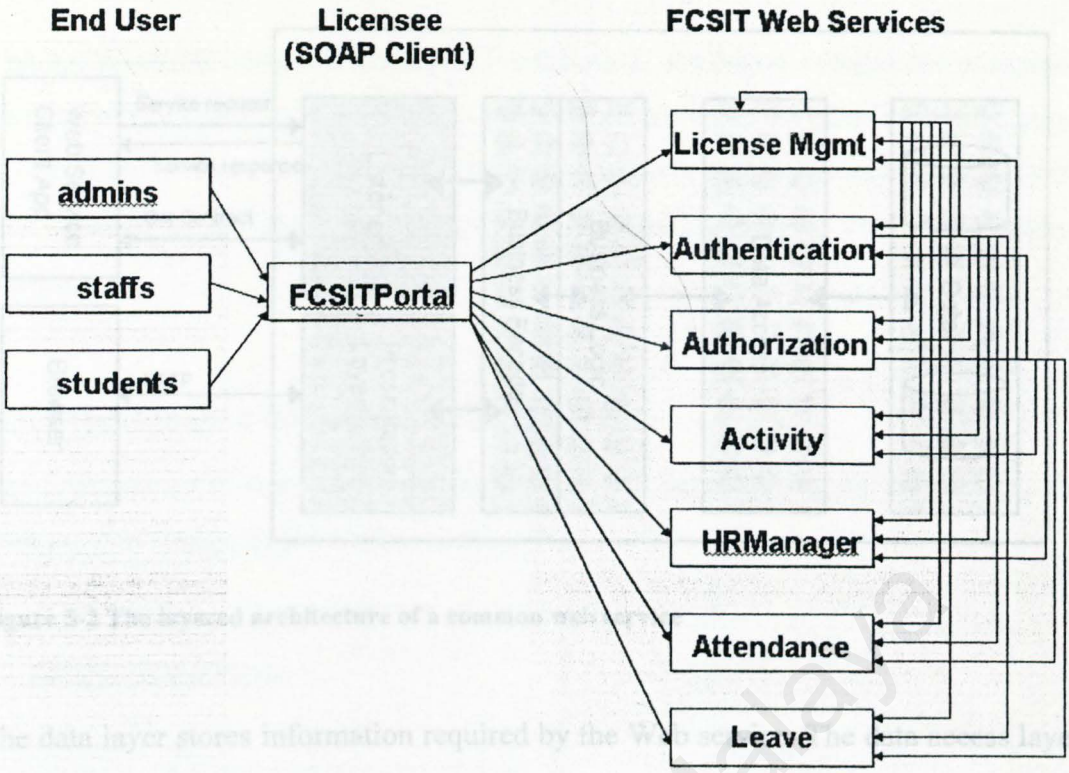


Figure 5-1 The overview architecture of FCSIT Web Services

The FCSIT Integrator is business specific—exclusively designed for the FCSIT to consume the FCSIT Web Services. However, this Integrator can also be easily modified for other business entities as well.

The FCSIT Web Services, however, are more generic; any business application designed to authenticate their user, authorize them, manage the license of their services, manage activities, manage attendance or manage leave can uses these Web Services offered.

Though in this case the database for the Web Services contains data for FCSIT staffs only, it could also be populated with data for the other business entities. *(Note: For further details see how the database is designed.)*

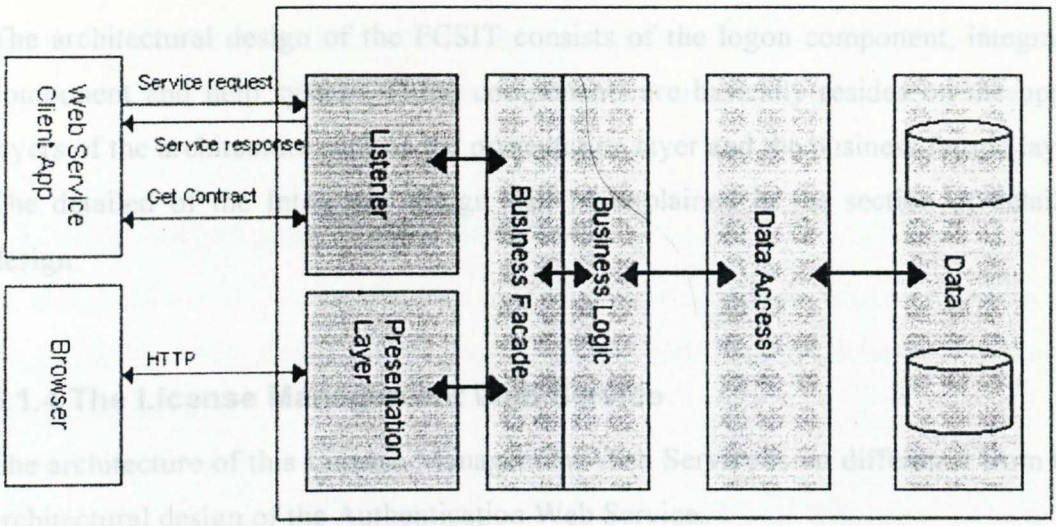


Figure 5-2 The layered architecture of a common web service

The data layer stores information required by the Web service. The data access layer provides logical views of the data and ensures data integrity. The business logic is represented in the business logic layer. The facade's interface maps directly to the operations exposed by the Web service. The SOAP listener is implemented where the Web service's SOAP protocol programming interface is used.

5.1.2 The FCSITPortal

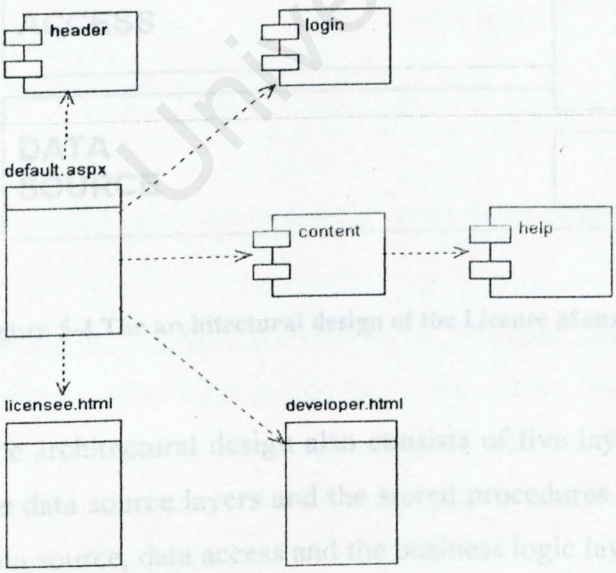


Figure 5-3 Component diagram showing the modules made up the FCSIT Integrator

The architectural design of the FCSIT consists of the logon component, integrator component and help system. These components are basically resides on the upper layers of the architecture such as the presentation layer and the business façade layer. The detailed of the Integrator design will be explained in the section of detailed design.

5.1.4 The License Management Web Service

The architecture of this License Management Web Service is no difference from the architectural design of the Authentication Web Service.

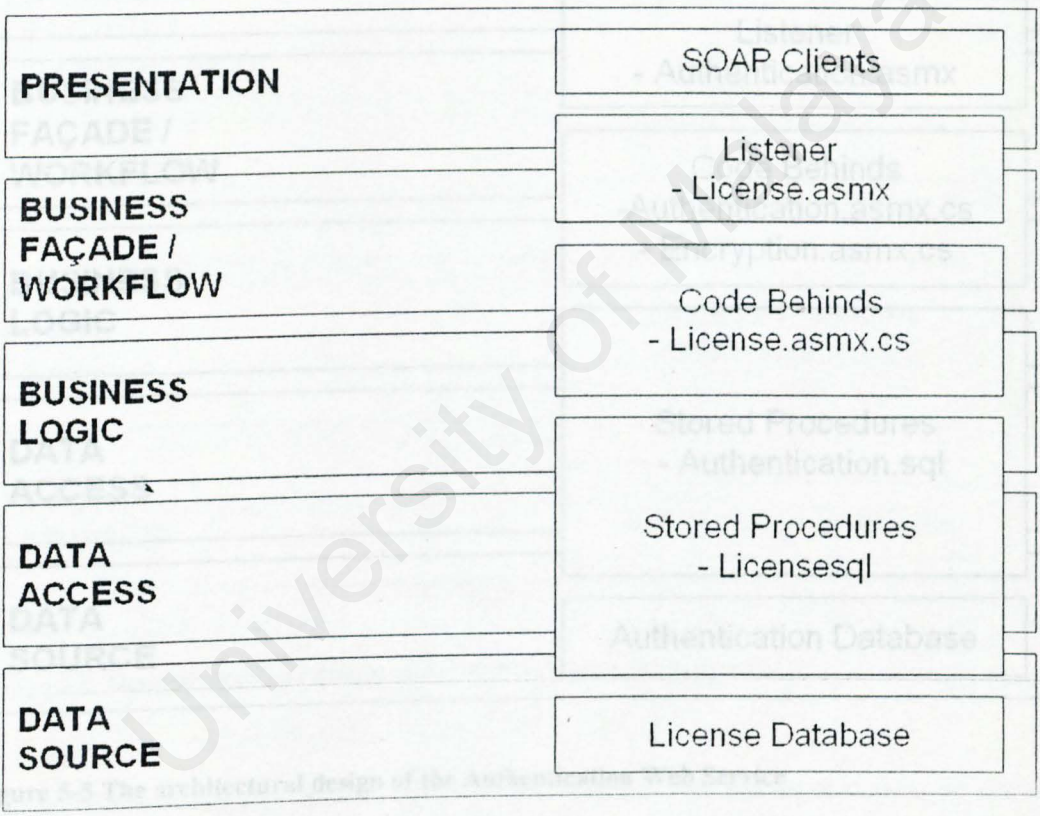


Figure 5-4 The architectural design of the License Management Web Service

The architectural design also consists of five layers. The license database resides in the data source layers and the stored procedures spanning three layers which are the data source, data access and the business logic layer.

The code behind will be spanning in two upper layers which are the business logic and the business façade. The listener also spanning two layers which are the business façade and the presentation. The SOAP client will reside on the presentation layer.

5.1.3 The Authentication Web Service

As discussed earlier, the architecture of a Web Service design consist of five layers. So the architectural design of the authentication Web Services follows this trait of design architecture.

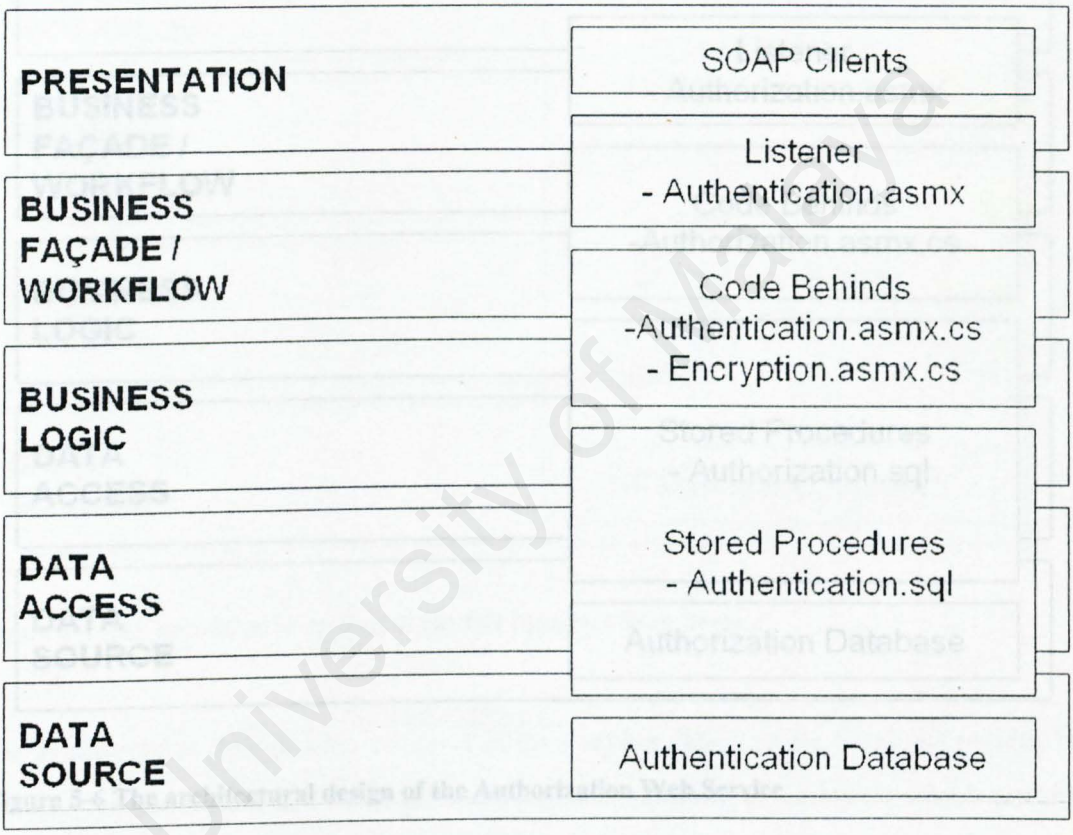


Figure 5-5 The architectural design of the Authentication Web Service

As can be seen in the above figure, the database of the authentication Web Service resides on the data source layer, the stored procedure here span over three layer which are the data source, data access and the business logic layer, actual code that provides the functionality resides on two layers which are the business logic and the business façade layer.

The listener, which are special applicable to all Web Services design will reside on the business façade and the presentation layer as together with the SOAP client.

5.1.5 The Authorization Web Service

This Authorization Web Service is also design to be based on the five layer architecture.

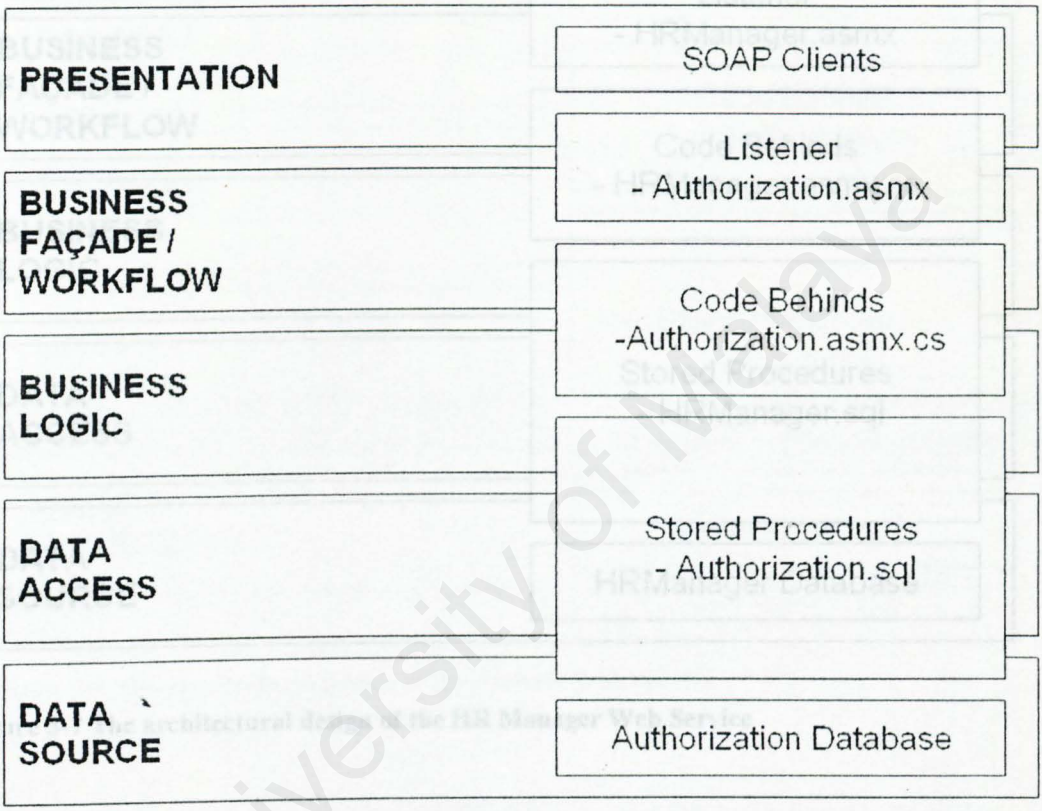


Figure 5-6 The architectural design of the Authorization Web Service

The architectural design also consists of five layers. The database resides in the data source layers and the stored procedures spanning three layers which are the data source, data access and the business logic layer.

The code behind will be spanning in two upper layers which are the business logic and the business façade. The listener also spanning two layers which are the business façade and the presentation. The SOAP client will reside on the presentation layer.

5.1.4 The HR Manager Web Service

The architecture of this HR Manager Web Service is no difference from the architectural design of other web services which I designed.

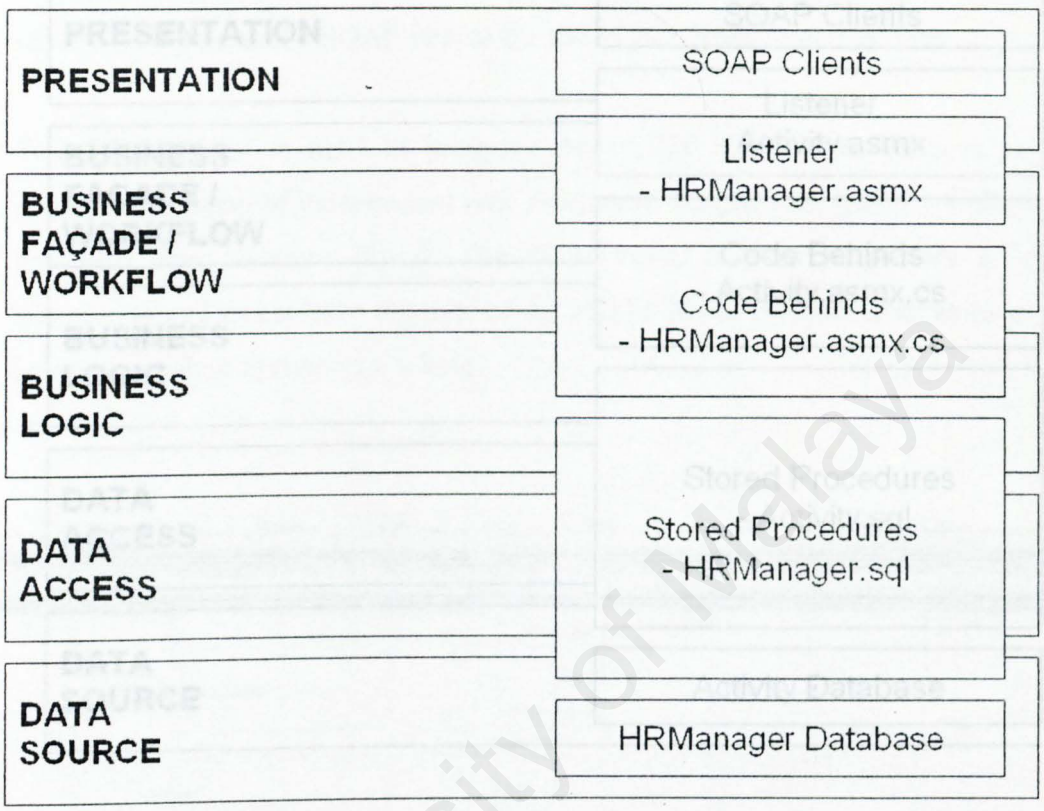


Figure 5-7 The architectural design of the HR Manager Web Service

The architectural design also consists of five layers. The profile database resides in the data source layers and the stored procedures spanning three layers which are the data source, data access and the business logic layer.

The code behind will be spanning in two upper layers which are the business logic and the business façade. The listener also spanning two layers which are the business façade and the presentation. The SOAP client will reside on the presentation layer.

5.1.6 The Activity Web Service

The Activity Web Service is also the same with other Web Services, building upon the five layered architectural design of a common Web Service.

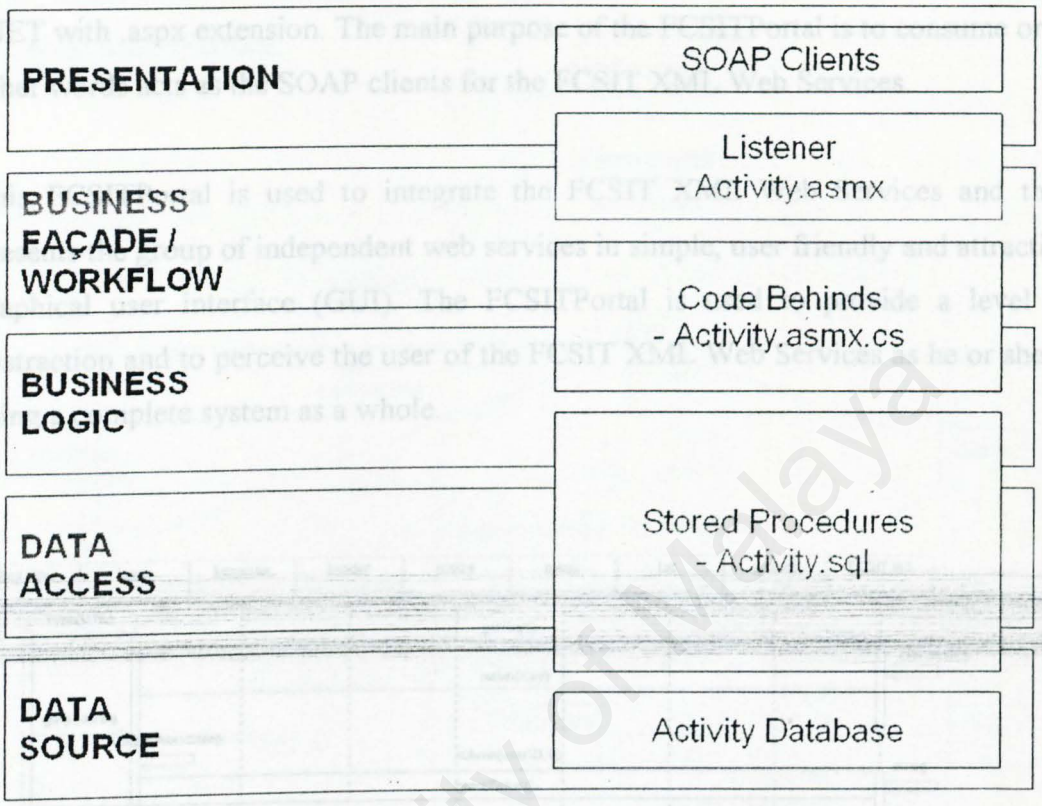


Figure 5-8 The architectural design of the Activity Web Service

The architectural design also consists of five layers. The database resides in the data source layers and the stored procedures spanning three layers which are the data source, data access and the business logic layer.

The code behind will be spanning in two upper layers which are the business logic and the business façade. The listener also spanning two layers which are the business façade and the presentation. The SOAP client will reside on the presentation layer.

5.2 Detailed Design

5.2.1 The FCSITPortal

The detailed design of the FCSITPortal is a web application written using the ASP .NET with .aspx extension. The main purpose of the FCSITPortal is to consume or in other words acts as the SOAP clients for the FCSIT XML Web Services.

This FCSITPortal is used to integrate the FCSIT XML Web Services and then presents the group of independent web services in simple, user friendly and attractive graphical user interface (GUI). The FCSITPortal is used to provide a level of abstraction and to perceive the user of the FCSIT XML Web Services as he or she is using a complete system as a whole.

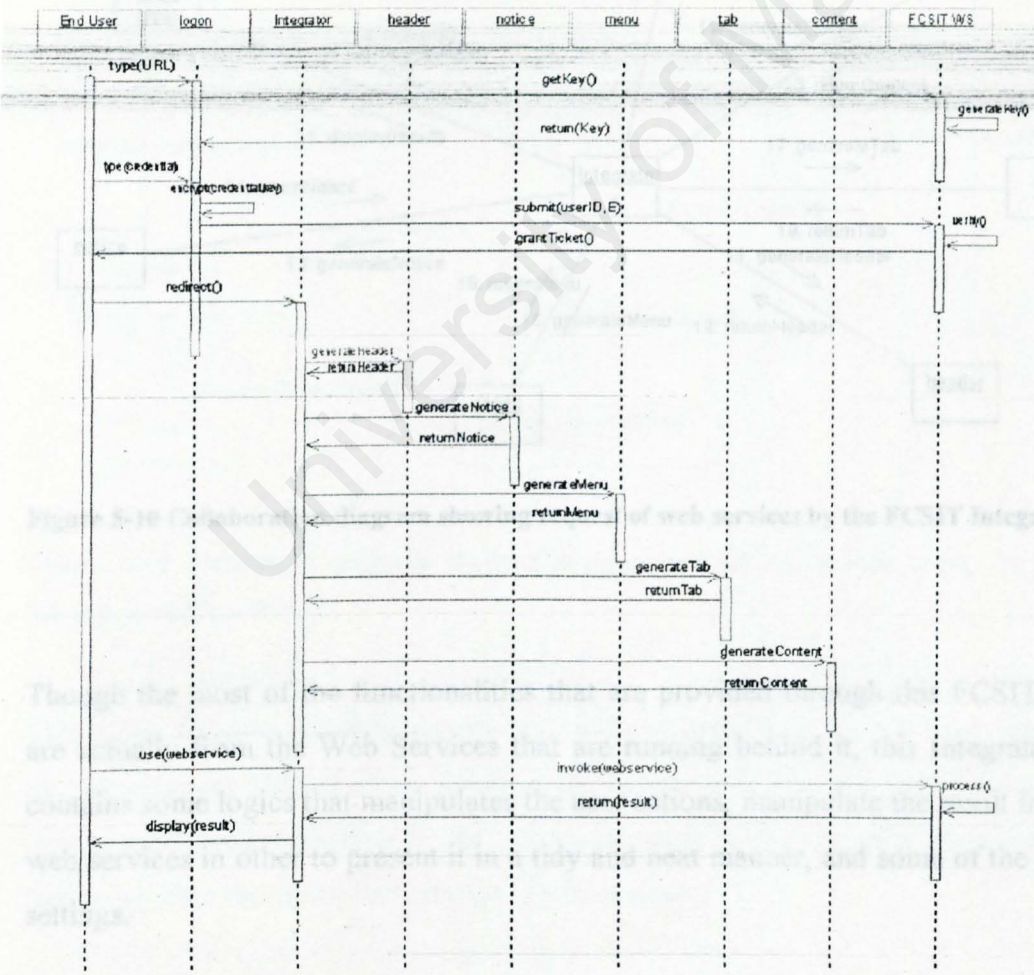


Figure 5-9 Sequence diagram showing request of web services by the FCSITPortal

When user does an action, the FCSITPortal will invoke the appropriate Web Services to provide the necessary result and then presents it back to the user. So, in the inner view, it is actually the web services that manipulate the user actions and return the result to the FCSITPortal that will then displays the result back to the user in a nice graphical user interface.

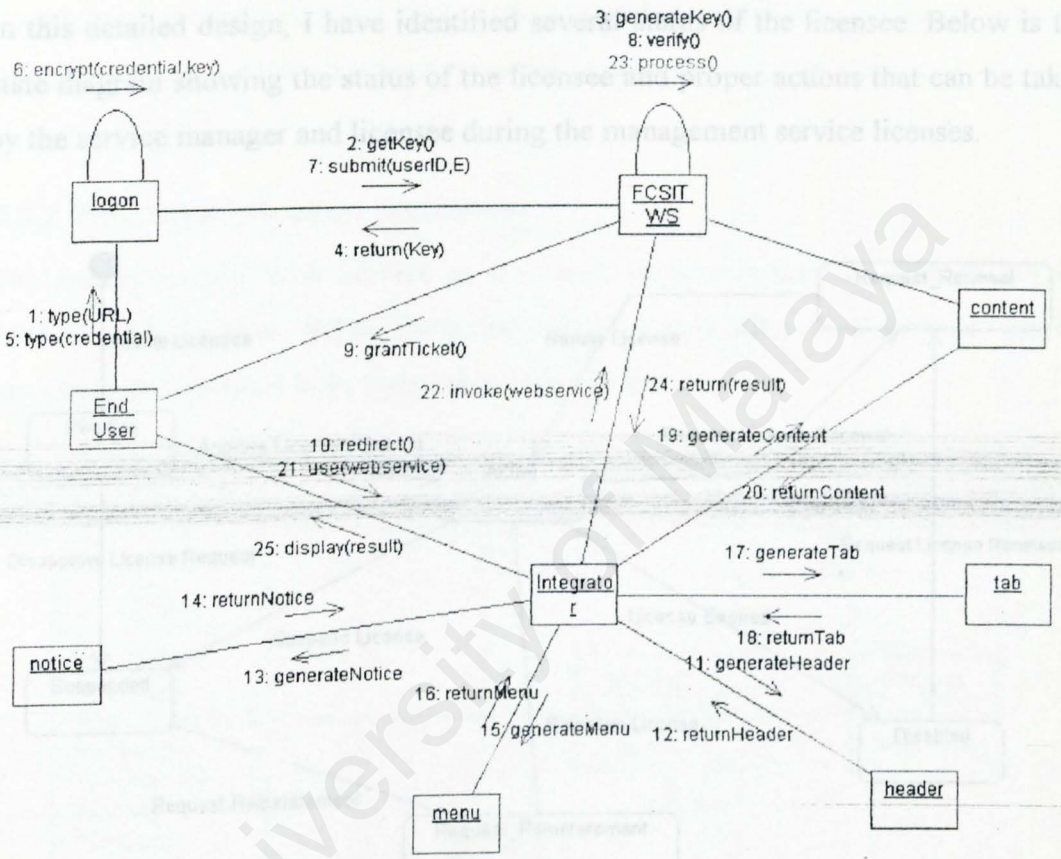


Figure 5-10 Collaboration diagram showing request of web services by the FCSIT Integrator

Though the most of the functionalities that are provided through this FCSIT Portal are actually from the Web Services that are running behind it, this Integrator does contains some logics that manipulates the user actions, manipulate the result from the web services in other to present it in a tidy and neat manner, and some of the system settings.

5.2.3 The License Management Web Service

The license management is particularly important to ensure the proper usage of the web services offered. By having the control over who can use the system and revoke the license for those who abuse the web services ensure that the web services remains in secure manner.

In this detailed design, I have identified several status of the licensee. Below is the state diagram showing the status of the licensee and proper actions that can be taken by the service manager and licensee during the management service licenses.

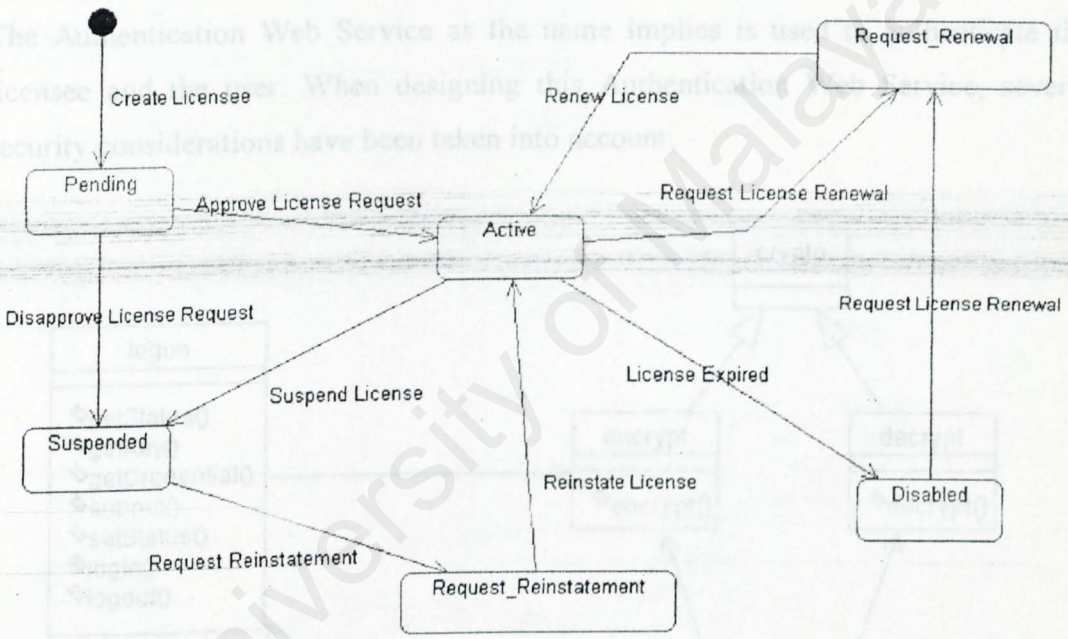


Figure 5-11 StateChart diagram showing several state of the license using the License Management Web Service

This state diagram is pretty simple and self explained. As can be seen, when the service manager created a new licensee, the license will start in a pending state. The license can be either approve by the service manager or disapprove it. If the license is approved, then the status will change to active state.

In the active state, the license will be check for its expiry date. Before the license is expired, the licensee can request for renewal of its license. If not the license will be disabled when it expires. In the active state, the service manager reserves the right to suspend any licensee if he or she feels that the web service has been abused.

Let us look back to the case where the license is disapproved by the service manager. In this case, the license will move to the suspended state. When a license is in the suspended state, the licensee can request for reinstatement.

5.2.2 The Authentication Web Service

The Authentication Web Service as the name implies is used to authenticate the licensee and the user. When designing this Authentication Web Service, several security considerations have been taken into account.

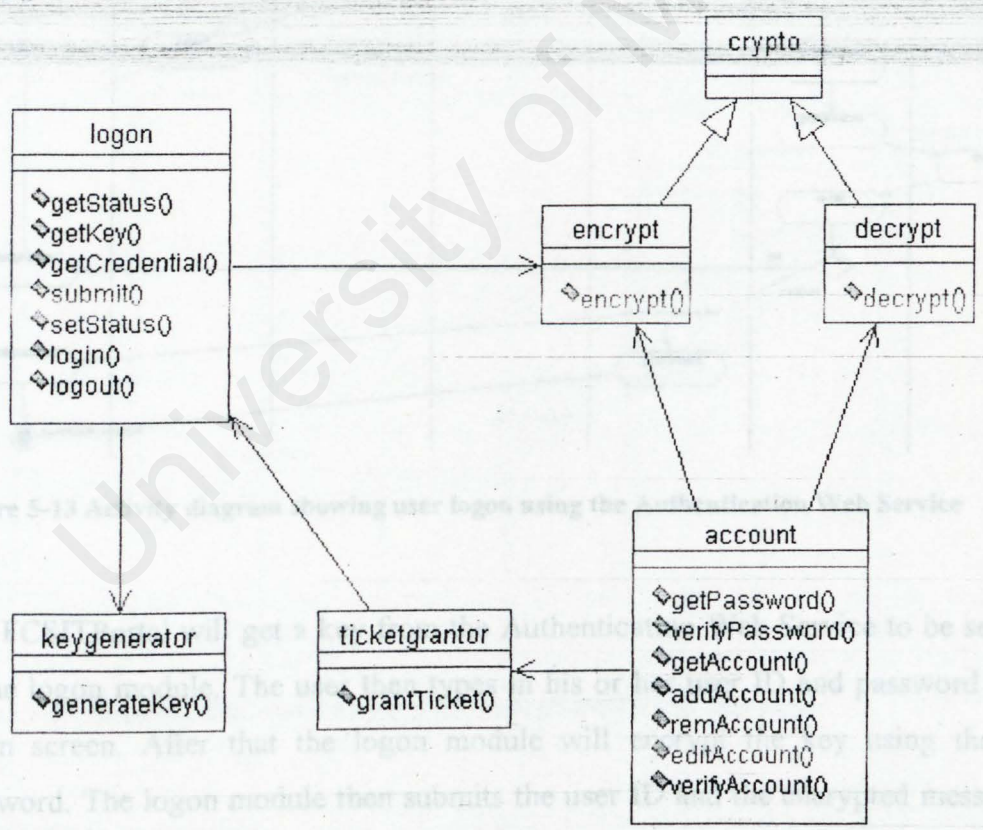


Figure 5-12 Class diagram of the Authentication Web Service

First of all, it must be able to identify the identity of the licensee. Then, it will identify the identity of the user. However, when user sends his or her credential such as password over the network to be authenticated by the server, this poses a serious threat.

So, to ensure that the user's credentials are protected, the technique used to authenticate a user has been designed to be as follows. First of all, when user tries to use the FCSIT Integrator, the Integrator will check if the user has been authenticated. If, not then the user must logon first.

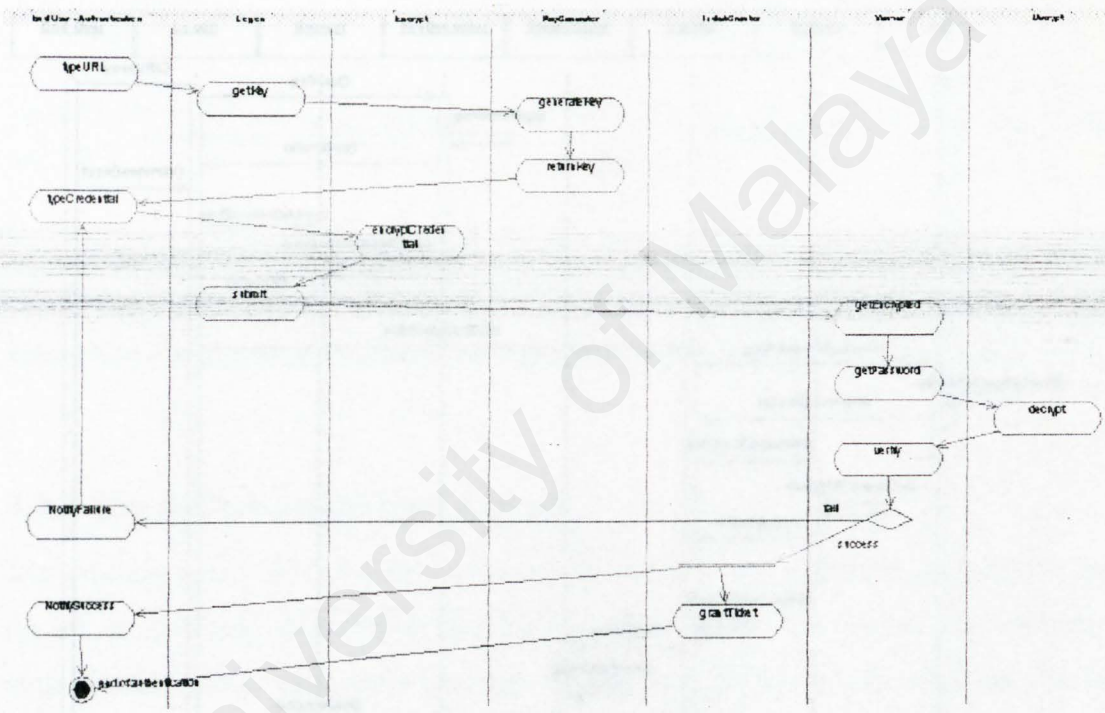


Figure 5-13 Activity diagram showing user logon using the Authentication Web Service

The FCSITPortal will get a key from the Authentication Web Service to be send on to the logon module. The user then types in his or her user ID and password to the logon screen. After that the logon module will encrypt the key using the user password. The logon module then submits the user ID and the encrypted message to the Authentication Web Service.

By using this technique, the user password will not be sent over the network and there is no way of finding out the password even though the packet is sniffed out by

the attackers. Secondly, the key send to the user is a session key that will be expired after some time, so the act of replay is minimize.

When the Authentication Web Services gets the user ID and the encrypted message, it will verify the user identity by retrieving the password using the user ID, then decrypt the encrypted message using the password and the compare with the session key. If both match, then it proves that the user is real and a ticket will be grated to the user.

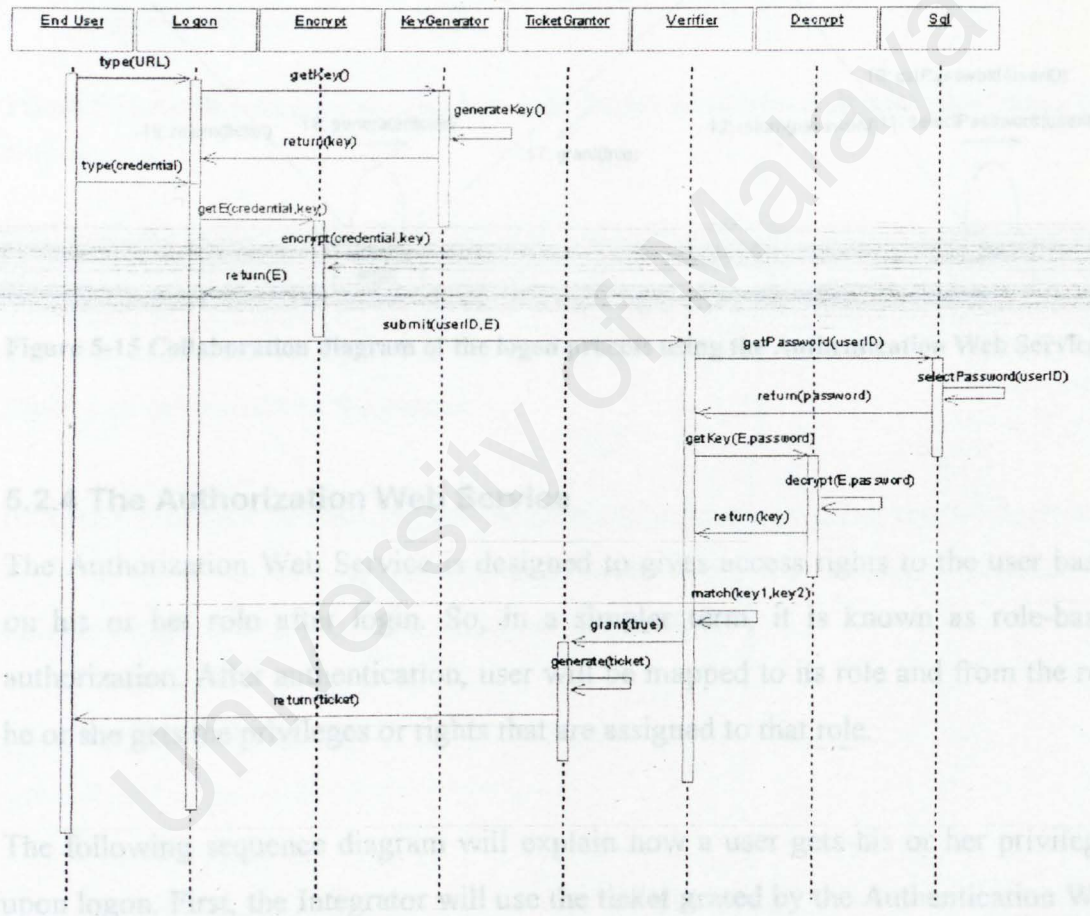


Figure 5-14 Sequence diagram showing the details process when user logon

Besides providing the primary functionality of authentication, the Web Service also allows the administrators to add new user or to remove user from account. Meanwhile, the normal user will be allowed to change his or her password.

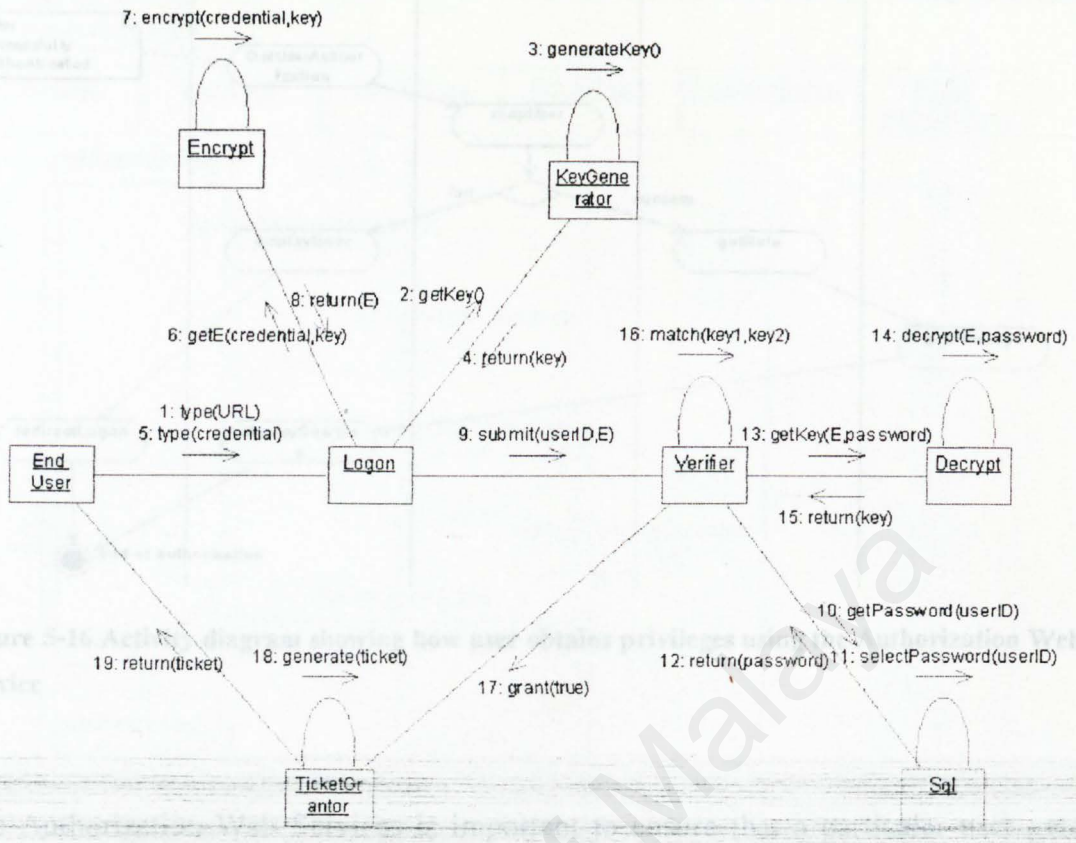


Figure 5-15 Collaboration diagram of the logon process using the Authentication Web Service

5.2.4 The Authorization Web Service

The Authorization Web Service is designed to gives access rights to the user based on his or her role after login. So, in a simpler term, it is known as role-based authorization. After authentication, user will be mapped to its role and from the role he or she gets the privileges or rights that are assigned to that role.

The following sequence diagram will explain how a user gets his or her privileges upon logon. First, the Integrator will use the ticket grated by the Authentication Web Services to invoke the Authorization Web Service.

The ticket will be match to the user ID and then maps to the role he or she holds. Then, the Authorization Web Services will retrieve his or her privileges based on the role he or she holds.

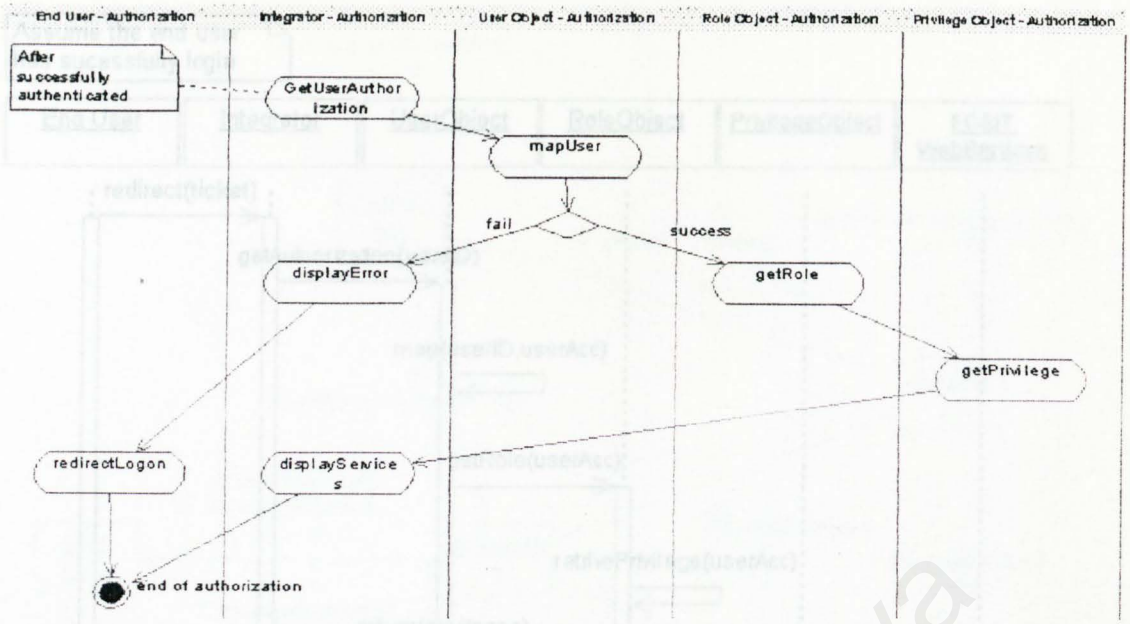


Figure 5-16 Activity diagram showing how user obtains privileges using the Authorization Web Service

The Authorization Web Services is important to ensure that a particular user gets only his or her access right but not more. This is also to ensure the security and privacy of other users of the system.

Besides the benefit of ensuring the security and privacy, by using the Authorization Web Services, the developers need not custom-made the necessary interfaces in order to assign and restrict what a user can or cannot do.

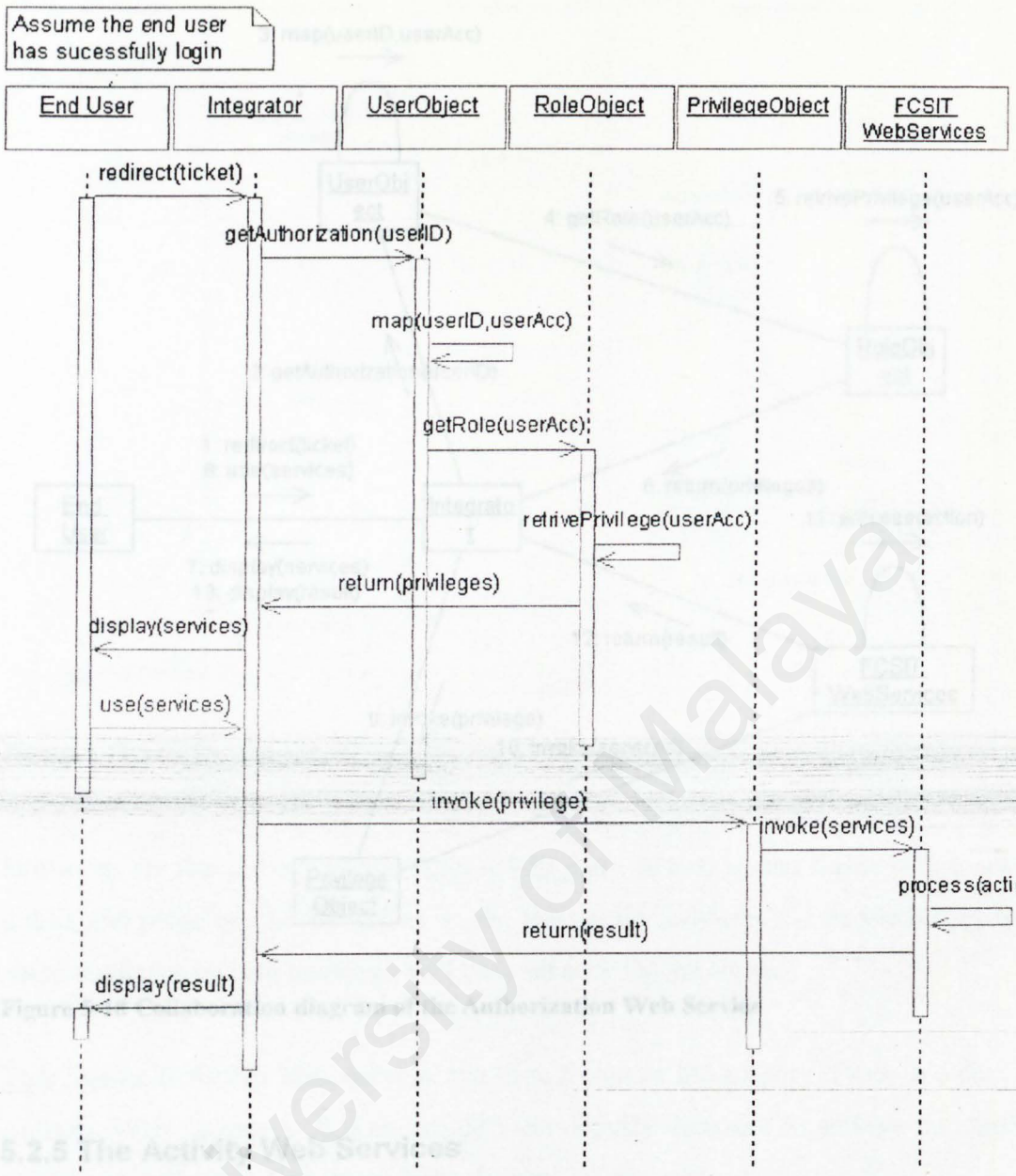


Figure 5-17 Sequence diagram showing the details in time frame on how user gets his or her access rights from the Authentication Web Service

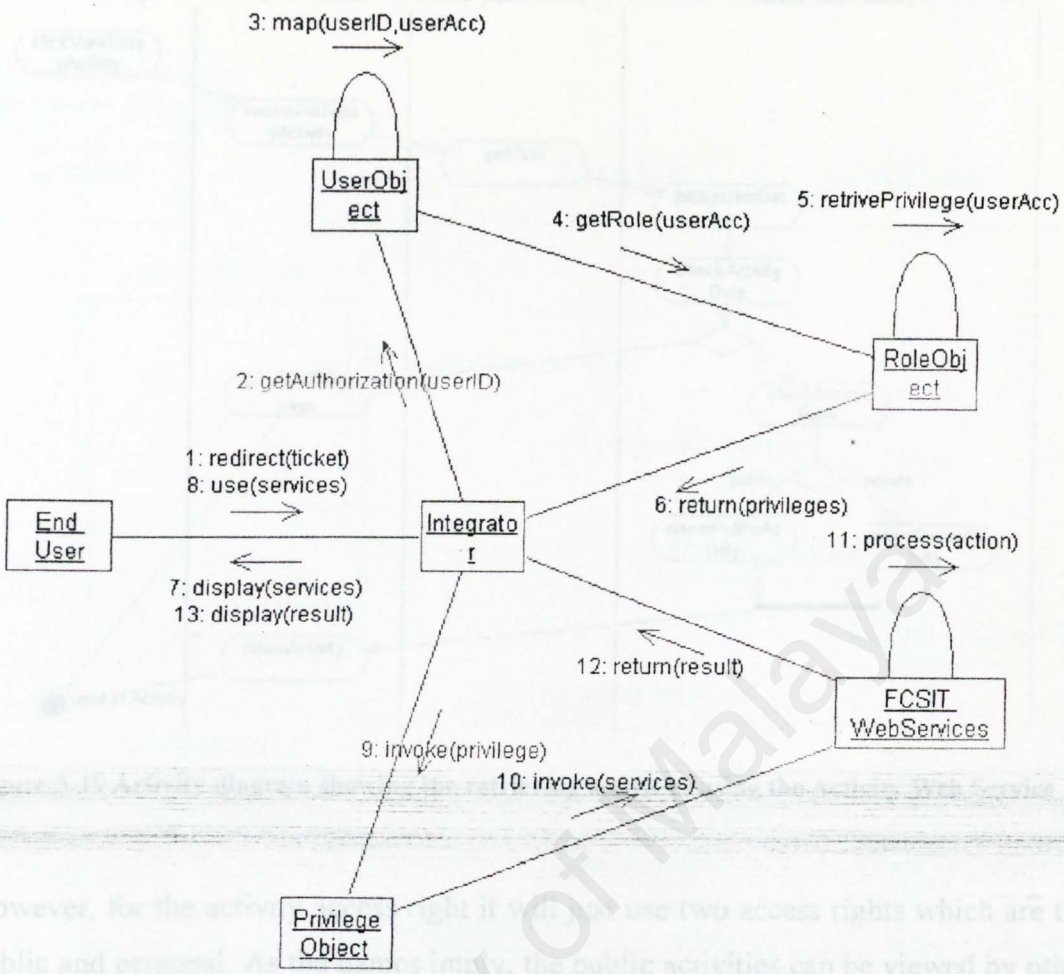


Figure 5-18 Collaboration diagram of the Authorization Web Service

5.2.5 The Activity Web Services

The Activity Web Services that is going to be designed is a very straight forward with the activity setting, removing editing and viewing functionalities. In the background, the Activity Web Services will check for archiving old activities and broadcasting activities of the day.

The new feature that is added over the previous ALMS system is the access right of the activities. The access right of the activity somehow mimics the access right in the object oriented programming such as public, private and protected.

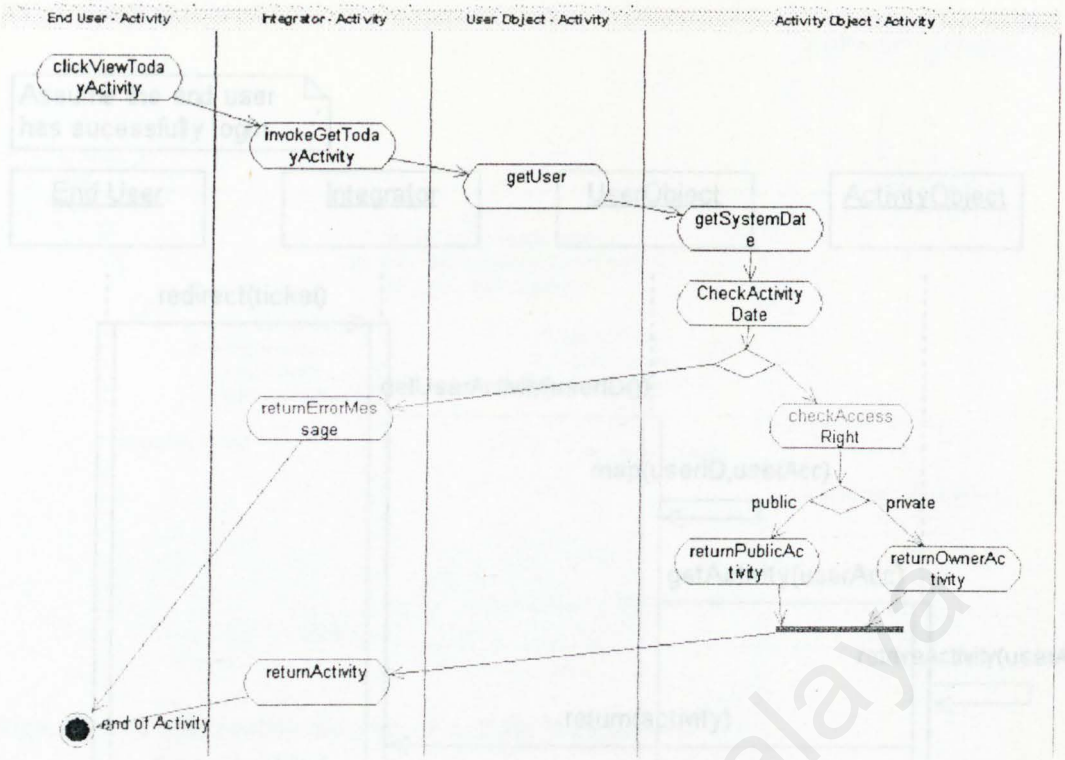


Figure 5-19 Activity diagram showing the retrieving activities using the Activity Web Service

However, for the activity access right it will just use two access rights which are the public and personal. As the names imply, the public activities can be viewed by other users while the private activities will only be seen by the owner.

This feature is further noticeable in the time frame of the activity. Once in a day, the Activity Web Service will scan through the activity database to archive the expired public activities based on the activity date set by the user.

The expired public activities are all the public activities set by the users of Activity Web Service in which the broadcast date of these activities is expired as compare to the system date. When a public activity is expired, the creators of these activities are not allowed to remove or edit the activities anymore.

The expired private activities are all the private activities set by the users of Activity Web Service in which the broadcast date of these activities is expired as compare to the system date. These activities will not be archived by the system. So, creators of these activities will remain the rights to remove or edit the activities.

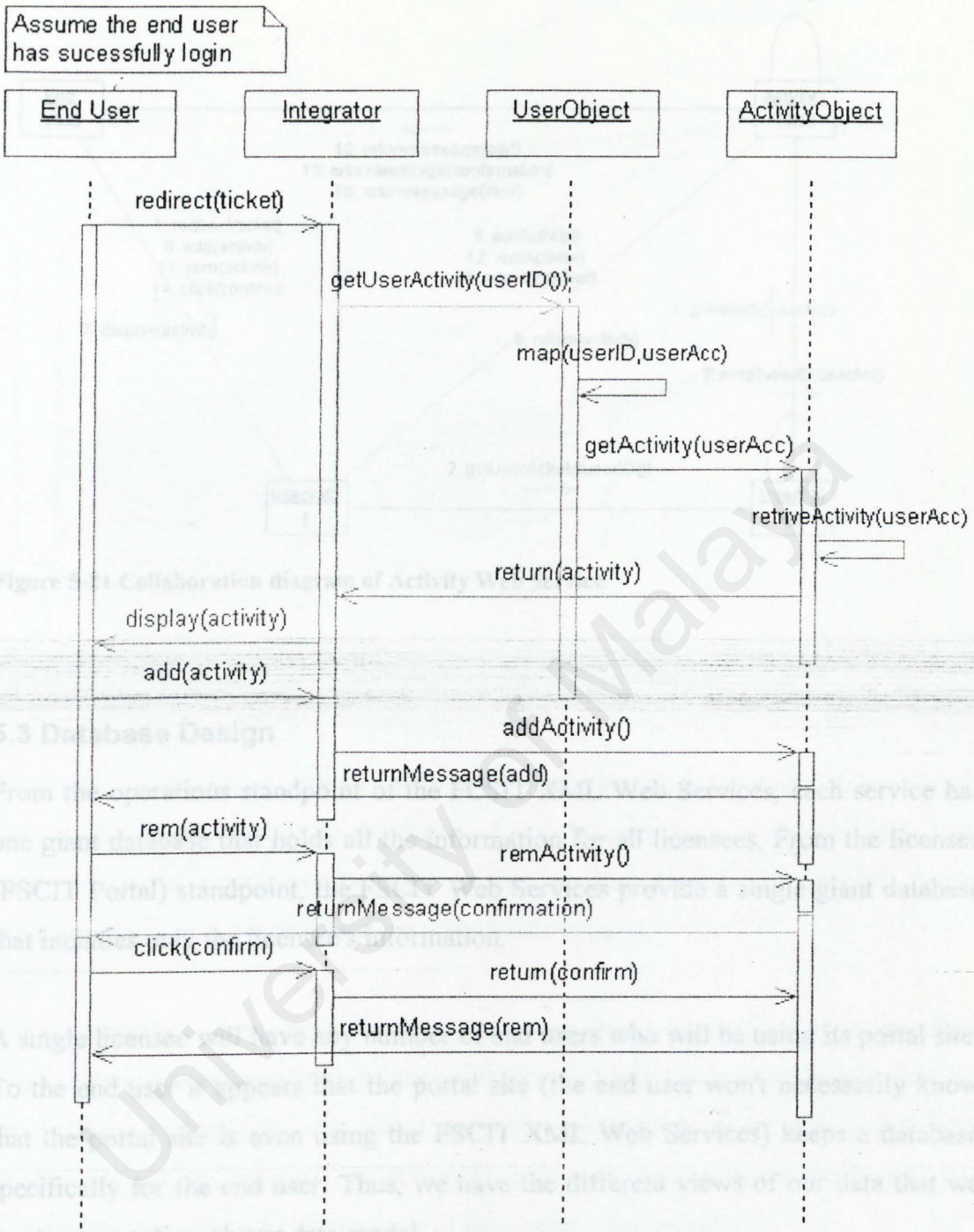


Figure 5-20 Sequence diagrams showing hot the Activity Web Services retrieve the activities

As similar to the current ALMS activity subsystem, once in a day, the Activity Web Services will scan through the activity database for the activity date of all the activities. The public activities will be broadcasted which means that other user will be able to see these activities while the private activities will be listed and only be seen by the owner of these activities.

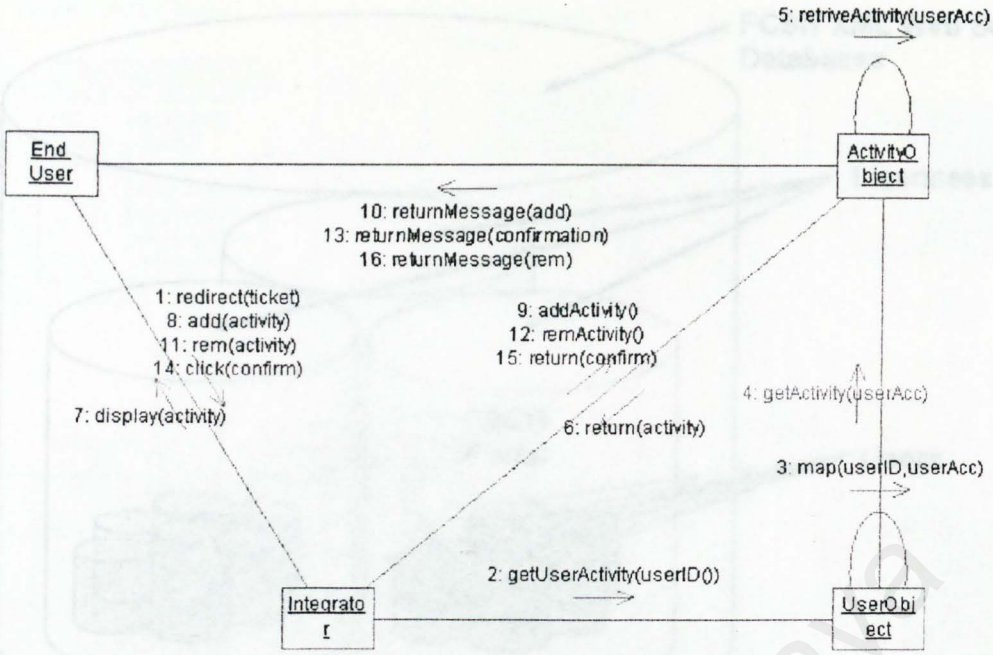


Figure 5-21 Collaboration diagram of Activity Web Service

5.3 Database Design

From the operations standpoint of the FCSIT XML Web Services, each service has one giant database that holds all the information for all licensees. From the licensee (FSCIT Portal) standpoint, the FSCIT Web Services provide a single giant database that includes only the licensee's information.

A single licensee will have any number of end users who will be using its portal site. To the end user it appears that the portal site (the end user won't necessarily know that the portal site is even using the FSCIT XML Web Services) keeps a database specifically for the end user. Thus, we have the different views of our data that we must support through our data model.

By partitioning the each database into two conceptual partitions, the end user's data is effectively accessed and managed through the actual FCSIT XML Web Service operations. End users can uses the web services offered without knowing this.

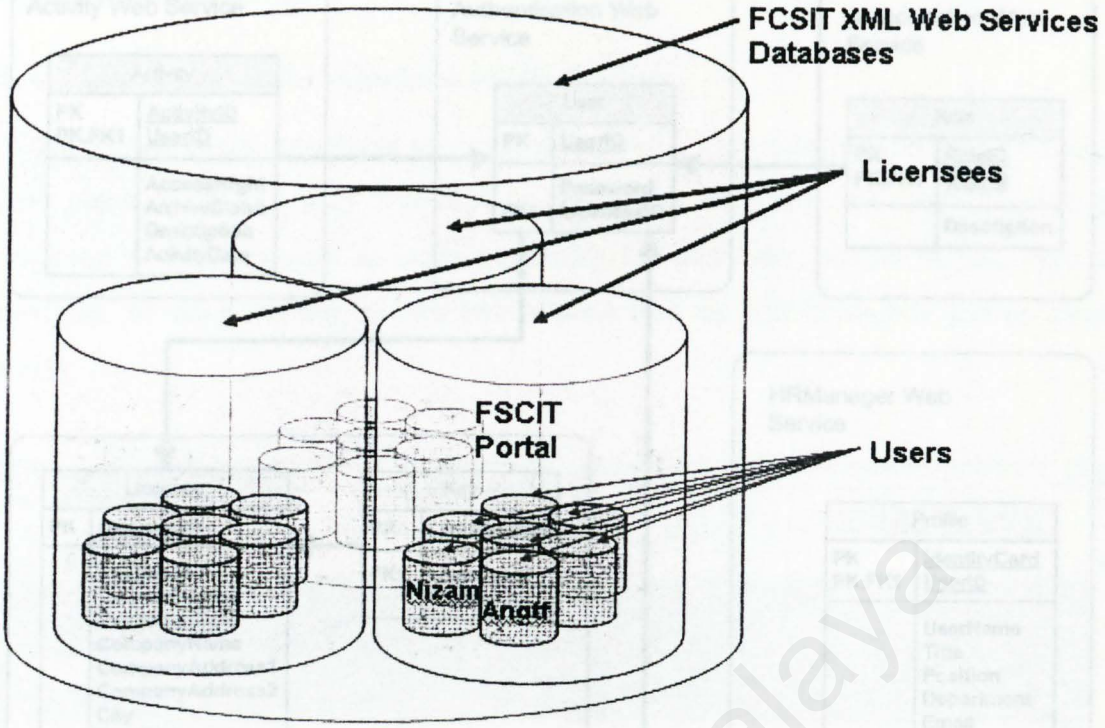


Figure 5-22 Partitioning of the FCSIT Database to provide abstraction

Therefore, requests to update any of the back-end data at this level always require the licensee key (which identifies the licensee) and the user name of the end user. User names must be unique per licensee so that we can identify the specific list of favorites that should be provided for a particular request.

From a licensee's perspective, its data is accessed through a number of means. First of all, it sees the end user's interactions with the end user's data as essentially its own interactions, because the end users probably know little about the actual Web Service. It is the licensee (FSCITPortal), after all, that performs the logon and retrieves the key that will be used for all the normal service calls.

As you can see, almost all of the data points back to the UserID and thus to the LicenseeId. This should not be too surprising because we want to make sure that a licensee sees only its own information. As for users, we only want them to get their own information when a licensee performs actions on their behalf. Typically, the end user delegates the actions to the licensee.

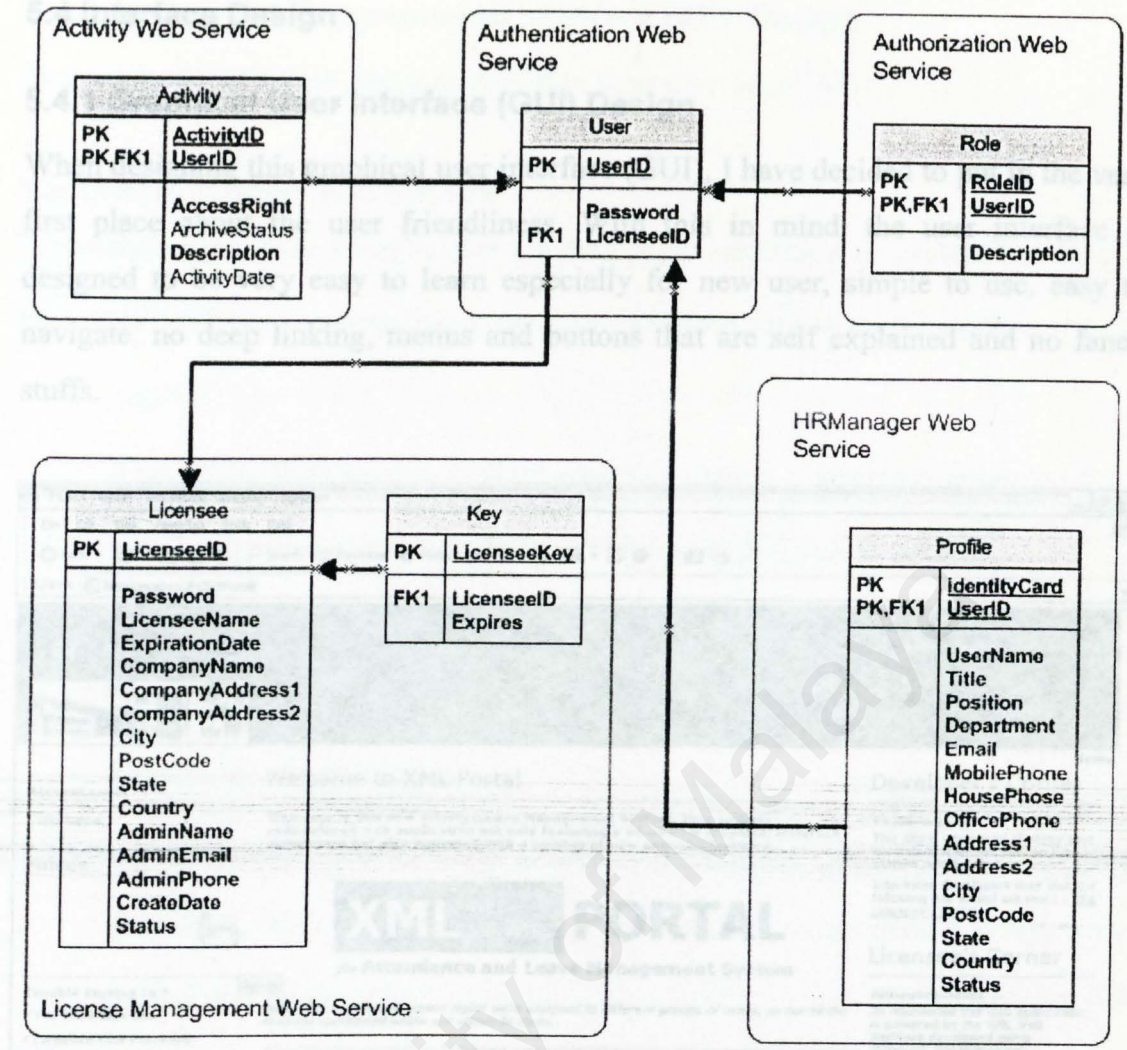


Figure 5-23 Logical view of the FCSIT Web Services database design. Each web service **manages its own database.**

For example, the end user adds an activity using the Activity Web Service. In turn, the FCSITPortal automatically forwards that request to the Activity Web Service and the activity is stored. An administrator that holds the role as licensee cannot alter or store information for end users that are not associated with its licensee ID.

As mentioned in the detailed design earlier, the FCSITPortal will just act as an interface between the end user and the FCSIT XML Web Services. So, the FCSITPortal will not store or manipulate any user data. In this case, therefore the system does not need a database.

5.4 Interface Design

5.4.1 Graphical User Interface (GUI) Design

When designing this graphical user interface (GUI), I have decided to put in the very first place about the user friendliness. With this in mind, the user interface is designed to be very easy to learn especially for new user, simple to use, easy to navigate, no deep linking, menus and buttons that are self explained and no fancy stuffs.

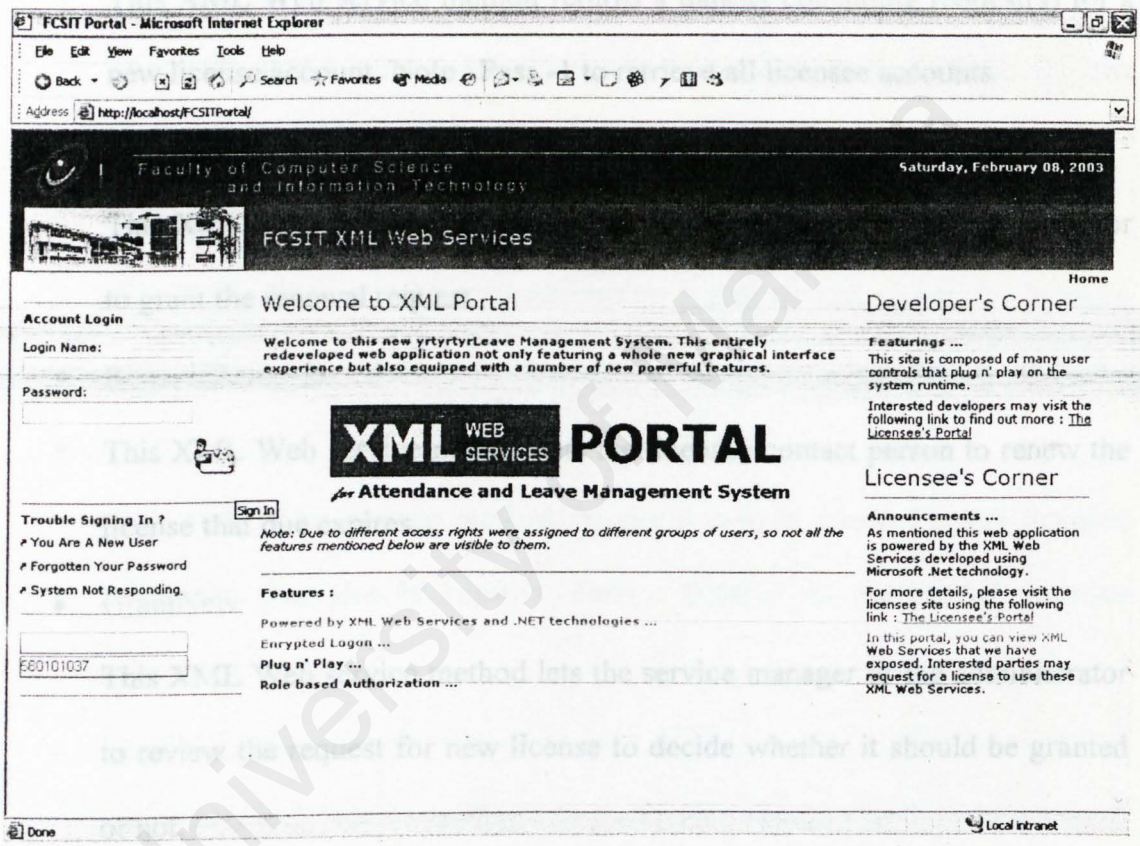


Figure 5-24 The FCSITPortal default page

The GUI design was also taking the consideration of easy integration for the Web Services. As FCSIT XML Web Services will be enhance and more new services will be provided to the user, I hope that the GUI design that is flexible and scalable to integrate these new services as well.

5.4.2 Application Programming Interface (API) Design

Each of the FCSIT Web Services will have their own set of APIs.

The License Manager Web Services

LicenseManager class encapsulates all data logic necessary to protect the FCSIT Web Services from unauthorized usage by anonymous sites or applications.

- GetReqNew

  This XML Web service method returns a dataset containing request(s) for a new license account. Note : Pass -1 to retrieve all licensee accounts.

- GrantRenew

  This XML Web service method lets the service manager or the administrator to grant the renewal request.

- RequestRenewal

  This XML Web service method lets the licensee contact person to renew the license that due expires.

- GrantNew

  This XML Web service method lets the service manager or the administrator to review the request for new license to decide whether it should be granted or not.

- RetrieveDetails

  This XML Web service method returns the information about the licensee in the form of a dataset based on the licenseeName and password supplied.

- GetSuspendedAcc

  This XML Web service method returns a dataset containing the suspended licensee account(s). Note : Pass -1 to retrieve all licensee accounts.

- RequestChanges

  This XML Web service method lets licensee to make a request to change its particulars.

- SuspendLicense

  This XML Web service method lets the service manager or the administrator to mark the licensee in the licensee database as suspended.

- ChangePassword

  This XML Web service method the contact person to change his / her account password.

- GetNearExpiryAcc

  This XML Web service method returns a dataset containing the licensee account(s) that near expiry. Note : Pass -1 to retrieve all licensee accounts.

- GetExpiredAcc

  This XML Web service method returns a dataset containing the licensee account(s) that already expired. Note : Pass -1 to retrieve all licensee accounts.

- ApproveChanges

  This XML Web service method lets the service manager or the administrator to decide if the request for changes is permitted or denied.

- GrantReinstate

  This XML Web service method lets the service manager or the administrator to approve or reject the request for license reinstatement.

- GetReqRenew

  This XML Web service method returns a dataset containing the request(s) for renewal. Note : Pass -1 to retrieve all licensee accounts.

- IsLicensed

  This XML Web service method checks if the licensee key supply is valid and not expired.

- ResolveID

  This XML Web service method resolves the license ticket to its corresponding licenseeID.

- RequestReinstate

  This XML Web service method lets the licensee contact person to make a request to reinstate his / her license that has been suspended.

- ResetPassword

  This XML Web service method the service manager or the administrator to reset the password of the particular account.

- GetNormalAcc

  This XML Web service method returns a dataset containing the normal licensee account(s). Note : Pass -1 to retrieve all licensee accounts.

- GetReqReinstate

  This XML Web service method returns a dataset containing the request(s) for reinstatement. Note : Pass -1 to retrieve all licensee accounts.

- GetReqChanges

  This XML Web service method returns a dataset containing request(s) for changes. Note : Pass -1 to retrieve all licensee accounts.

- GetLicenseTicket

  This XML Web service method lets the licensee to login and gets license key in order to use the Web Services under its protection.

- RequestNew

  This XML Web service method lets prospect licensees to request for a new license.

The Ticket Manager Web Services

TicketManager class encapsulates all data logic necessary to authenticate users that have registered with the Ticket Manager Web Service.

- AddUser

  This XML Web service method inserts a new user account into the Ticket Manager Web Services database table.

- RemoveUser

  This XML Web service method enables the site administrator to remove a user from the Ticket Manager Web Services database.

- GetAllUsers

  This XML Web service method returns a DataSet containing the userIDs and loginNames of all the registered users.

- GetCurrentUserSessionKey

  This XML Web service method returns the current user's session key that can be used for secure data transfer.

- Logout

  This XML Web service method removes the current user's session.

- GetSingleUser

  This XML Web service method returns a DataSet containing the userID and its information.

- Login

  This XML Web service method lets users to login with session support enabled. Note: User agent's HTTP Cookie must be enabled in order for this authentication Web Service to track a particular user properly.

- ResetKey

  This XML Web service method lets the TicketManagerWS's administrator to reset the encryption and decryption key.

- ResetPassword

  This XML Web service method enables admin to reset the password of a user account.

- GetCurrentUserLoginName

  This XML Web service method returns the current user's LoginName.

- GetCurrentUserID

  This XML Web service method returns the current user's ID.

- ChangePassword

  This XML Web service method enables user to change his/her account password.

- GetServerTimeStamp

  This XML Web service method returns the server's timestamp.

- GetCurrentUserLicenseeID

  This XML Web service method returns the current user's LicenseeID.

- IsAuthenticated

  This XML Web service method checks if the user already login by checking at the user's session. The session must not be null.

The Role Manager Web Services

RoleManager class encapsulates all data logic necessary to manipulate roles within the RoleManagerWS database.

- GetUserRoles

  This XML Web service method returns a Dataset containing the roles that have been assigned to the given userID.

- GetRoleUsers

  This XML Web service method returns a Dataset containing all the users that have been assigned to the given roleID.

- GetAllRoles

  This XML Web service method returns a dataset of all role names for the current licensee.

- AddRole

  This XML Web service method creates a new role for the specified licensee and returns the new RoleID value.

- DeleteRole

  This XML Web service method deletes a role from the specified licensee and returns the RoleID value of the deleted role.

- AddUserRole

  This XML Web service method adds a user to a specified role and return the roleID to which the user has been succesfully added.

- IsInRoles

  This XML Web service method checks if the given userID is in one of the collection of roles supplied; each role is separated by a semicolon.

- DeleteUserRole

  This XML Web service method deletes a user from a specified role and return the roleID to which the user has been succesfully removed.

- UpdateRole

  This XML Web service method updates an existing role definition and returns the RoleID value of the updated role.

The HR Manager Web Services

HRManager class encapsulates all data logic necessary to store information regarding the human resource.

- GetSingleEmployeeData

  This XML Web service method gets the employee data from the HRManagerWS based on employeeID.

- EditDepartments

  This XML Web service method lets the administrator to edit a department in the HRManagerWS database.

- AddWorkStatus

  This XML Web service method lets the administrator to add a new work status into the HRManagerWS database.

- RemoveDepartment

  This XML Web service method lets the administrator to remove a department from the HRManagerWS database..

- GetInitials

  This XML Web service method retrieves all the initials from the HRManagerWS.

- RemoveEmployeeData

  This XML Web service method lets the administrator to remove an employee from the HRManagerWS database..

- EditInitials

  This XML Web service method lets the administrator to edit an initial in the HRManagerWS database.

- AddPosition

  This XML Web service method lets the administrator to add a new position into the HRManagerWS database.

- AddEmployeeData

  This XML Web service method lets the administrator to add a new employee into the HRManagerWS database.

- EditEmployeeData

  This XML Web service method lets the administrator to edit an employee's data that exists in the HRManagerWS database..

- GetWorkStatus

  This XML Web service method retrieves all the work status from the HRManagerWS.

- EditPositions

  This XML Web service method lets the administrator to edit a position in the HRManagerWS database.

- RemovePosition

  This XML Web service method lets the administrator to remove a position from the HRManagerWS database..

- EditWorkStatus

  This XML Web service method lets the administrator to edit a work status in the HRManagerWS database.

- RemoveWorkStatus

  This XML Web service method lets the administrator to remove a work status from the HRManagerWS database..

- GetAllEmployeeIdentification

  This XML Web service method gets the identification of all employees from the HRManagerWS.

- GetPositions

  This XML Web service method retrieves all the positions from the HRManagerWS.

- GetEmployeeData

  This XML Web service method gets the employee data from the HRManagerWS based on loginID.

- GetAllEmployeeData

  This XML Web service method gets the data of all employees from the HRManagerWS.

- RemoveInitial

  This XML Web service method lets the administrator to remove an initial from the HRManagerWS database..

- AddDepartment

  This XML Web service method lets the administrator to add a new department into the HRManagerWS database.

- GetDepartments

  This XML Web service method retrieves all the departments from the HRManagerWS.

- AddInitial

  This XML Web service method lets the administrator to add a new initial into the HRManagerWS database.

The Activity Manager Web Services

The LicenseManager class encapsulates all data logic necessary to set, edit, delete and retrieve activities / reminders.

- RemoveActivity

  This XML Web service method lets the author of the activity deletes his / her own activity. However, the activity with public access right must not be expired in order to be deleted.

- GetUserActivities

  This XML Web service method returns a dataset of all the private and public activities created by the user.

- GetPublicActivities

  This XML Web service method returns a dataset of all the public activities selected for the selected date range and sorting..

- AddActivity

  This XML Web service method lets user adds a new activity. The activity can be either private or public access. Private activity can only be viewed by the author while public can be viewed by other users belonging to the same

licensee. Every activity can only be edited or removed by the author of the activity.

- EditActivity

  This XML Web service method lets the author of the activity edits his / her own activity. However, the activity with public access right must not be expired in order to be edited.

- GetActivityDetail

  This XML Web service method returns a dataset of the activity details for the activityID supplied.

6 SYSTEM IMPLEMENTATION AND DEVELOPMENT

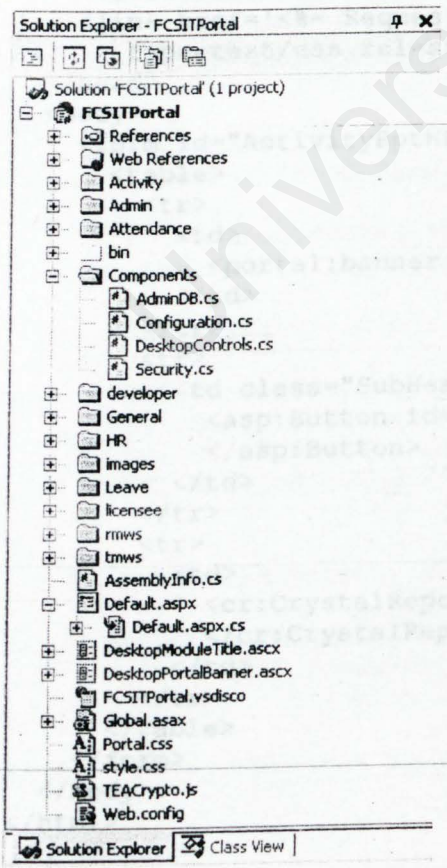
All the hard work done this while is useless unless it can be successfully converted to the software that will eventually meet all the requirements specified and solve the problems stated. That is, we must turn the designs into programs.

While this task can be very tedious and daunting for several reasons, so I have set a standard for structuring and documenting my codes so there are easy to understand, modified and reuse. Also, I always keep in mind on some of the good programming practices that I have been taught, theoretically.

In this documentation, it is not possible for me to explain every lines of codes that I have written as there are more than thousands lines of codes there. So, I have picked some of the important snippets, algorithms and patterns that I have used in my programs.

6.1 Program Structuring

Likewise in the design, divide and conquer technique is used when writing the source codes. Large system is divided into subsystem, smaller classes / modules and then methods.



By doing so, I can concentrate on solving one problem domain at each time without distracting to others problems. Also, this technique promotes manageability and reusability.

Debugging process is also made much easier when anything goes wrong in the code. I can easily tackle the bugs with the hassle to go through all the codes as if it was a monolithic program.

Figure 6-1 The use of components and classes in a complex system such as the FCSITPortal.

Another technique used is the separation of code between interface, business logic and data access layer. In realizing this technique, the Cascading Style Sheet (CSS), Hypertext Markup Language (HTML), code behind (C#) and stored procedure (SQL) were used throughout the system source code.

The HTML is mainly used for structuring the interface in the form of web pages and user controls while CSS is used to define the style and format the display. The new C# language is used to program the code behind and the business logic layer of the system and the stored procedure written in the Structured Query Language (SQL) is used in the data access layer.

Below is the code snippet of HTML code used.

```
<%@ Page language="c#" Codebehind="ActivityRptWF.aspx.cs"
AutoEventWireup="false"
Inherits="FCSITPortal.Activity.ActivityRptWF" %>
<%@ Register TagPrefix="cr" Namespace="CrystalDecisions.Web"
Assembly="CrystalDecisions.Web, Version=9.1.3300.0, Culture=neutral,
PublicKeyToken=692fbea5521e1304" %>
<%@ Register TagPrefix="portal" TagName="Banner"
Src="~/DesktopPortalBanner.ascx" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<html>
  <head>
    <title>FCSIT Portal</title>
    <link href='<%= Request.ApplicationPath + "/Portal.css" %>'
       type=text/css rel=stylesheet>
  </head>
  <body>
    <form id="ActivityRptWF" method="post" runat="server">
      <table>
        <tr>
          <td>
            <portal:banner id="Banner1" runat="server" />
          </td>
        </tr>
        <tr>
          <td class="SubHead">Click here to regenerate report :
            <asp:Button id="Button1" runat="server" Text="Send">
            </asp:Button>
          </td>
        </tr>
        <tr>
          <td>
            <cr:CrystalReportViewer id="CRV1" runat="server">
            </cr:CrystalReportViewer>
          </td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

Below is the code behind for the above aspx page used:

```
using System;
using System.Net;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace FCSITPortal.Activity
{
  public class ActivityRptWF : System.Web.UI.Page
  {
    protected CrystalDecisions.Web.CrystalReportViewer CRV1;
    protected System.Web.UI.WebControls.Button Button1;
    int tabId = 0;
    int tabIndex = 0;

    private void Page_Load(object sender, System.EventArgs e)
    {
      tmws.TicketManager tm = new tmws.TicketManager();
      tm.CookieContainer = new CookieContainer();

      if (Session["tmCookiesContainer"] == null)
        Session["tmCookiesContainer"] = tm.CookieContainer;
      else
        tm.CookieContainer =
          CookieContainer)Session["tmCookiesContainer"];

      if ( tm.IsAuthenticated() == false)
        Response.Redirect("~/Admin/AccessDenied.aspx");

      if (Request.Params["tabid"] != null)
        tabId = Int32.Parse(Request.Params["tabid"]);

      if (Request.Params["tabindex"] != null)
        tabIndex = Int32.Parse(Request.Params["tabindex"]);

      ActivityRptCR cr = new ActivityRptCR();

      CRV1.ReportSource = cr;
      CRV1.BestFitPage = true;
      CRV1.DisplayGroupTree = false;
      CRV1.EnableViewState = true;
    }

    private void Button1_Click(object sender, System.EventArgs e)
    {
      Response.Redirect("~/Default.aspx?tabindex=" + tabIndex +
        "&tabid=" + tabId);
    }
  }
}
```

Below is the code snippet of the CSS used:

```
/* ================================
    CSS STYLES FOR FCSIT PORTAL
   ================================
*/


/* PAGE BACKGROUND */
/* background color for the header at the top of the page  */
.HeadBg {
    background-color: #9d0000;
}

/* background color for the content part of the pages */
Body
{
    background-color: white;
}


/* PAGE BANNER */
/* style for the text of the site title */
.SiteTitle {
    font-family: Verdana, Helvetica, sans-serif;
    font-size: 18px;
    text-indent: 15px;
    letter-spacing: 1px;
    color:#cccc99;
}
```

Below is the code snippet of the stored procedure used:

```
/*
Author: Wong Kok Chuan WEK000119 wkc80@hotmail.com.
Created: 01 Jan 2003
Version: 1.0.0
*/
CREATE PROCEDURE GetAuthRoles
(
    @PortalID     int,
    @ModuleID     int,
    @AccessRoles nvarchar (256) OUTPUT,
    @EditRoles    nvarchar (256) OUTPUT
)
AS

SELECT
    @AccessRoles = Tabs.AuthorizedRoles,
    @EditRoles   = Modules.AuthorizedEditRoles

FROM
    Modules
```

```
    INNER JOIN
        Tabs ON Modules.TabID = Tabs.TabID
    WHERE
        Modules.ModuleID = @ModuleID
      AND
        Tabs.PortalID = @PortalID
    GO
```

Below is the code snippet of the source code written in C# used:

```
//************************************************************************
//
// UpdateTabOrder Method
//
// The UpdateTabOrder method changes the position of the tab with
respect
// to other tabs in the portal.
//
// Other relevant sources:
//      + UpdateTabOrder Stored Procedure
//
//************************************************************************

public void UpdateTabOrder (int tabId, int tabOrder)
{

    // Create Instance of Connection and Command Object
    SqlConnection myConnection =
        new sqlConnection(ConfigurationSettings.AppSettings[
        "connectionString"]);
    SqlCommand myCommand =
        new SqlCommand("UpdateTabOrder", myConnection);

    // Mark the Command as a SPROC
    myCommand.CommandType = CommandType.StoredProcedure;

    // Add Parameters to SPROC
    SqlParameter parameterTabId =
        new SqlParameter("@TabId", SqlDbType.Int, 4);
    parameterTabId.Value = tabId;
    myCommand.Parameters.Add(parameterTabId);

    SqlParameter parameterTabOrder =
        new SqlParameter("@TabOrder", SqlDbType.Int, 4);
    parameterTabOrder.Value = tabOrder;
    myCommand.Parameters.Add(parameterTabOrder);

    myConnection.Open();
    myCommand.ExecuteNonQuery();
    myConnection.Close();

}
```

6.2 Header Block

Every program that I have written starts with a header block which states the program title, the author, the date created, a short description on the purpose of the program, the version of the program and lastly the program copyright.

```
//*********************************************************************
//
// ActivityManager Class
//
// Created: 01 Jan 2003, Wong Kok Chuan WEK000119 wkc80@hotmail.com
// Summary: The ActivityManager class encapsulates all data logic
//          necessary to set, edit, delete and retrieve activities.
// Version: 1.0.0
//
// Increment by 1.0.0 for full modification on the class
// Increment by 0.1.0 for any functional modification on the class
// Increment by 0.0.1 for any debug modification on the class
// Increment by 0.1.1 for the combination of above two modifications
//
// Modified: DATE, PERSON
// Summary : SUMMARY ON THE REASON(S) AND THE MODIFICATION(S) MADE.
// Version : VERSION NUMBER
//
// Copyright 2003 All Rights Reserved
//
//*********************************************************************
```

Also, I have created a template for those who empowered to modify my programs. This template acts as a standard on how to document any modifications or changes made to the programs since the first release.

This template is important as to standardize the documentation so that any developers can easily understand the evolution of the program without the hassle to search through the program for changes made. I have also set a simple set of rules on how to version the program so the version number is not incremented simply.

Besides having header block in every starting of a program, I also include a header block comment to every starting of the methods that I created. This is to refine the header block comment that of greater granularity.

```
//*********************************************************************
//
// LicenseManager.AddEmployeeData() Method
//
// Created: 01 Jan 2003, Wong Kok Chuan WEK000119 wkc80@hotmail.com
// Version: 1.0.0
//
// Summary:
// AddEmployeeData() method lets user adds a new activity. The
// activity can be either private or public access. Private activity
// can only be viewed by the author while public can be viewed by
// other users belonging to the same licensee. Every activity can
// only be edited or removed by the author of the activity.
//
// Other relevant sources:
//     AddEmployeeData Stored Procedure ( v1.0.0 )
//
//*********************************************************************
```

The header block comment in the starting of every method describes the method name, the author, the date created, the method version, the method summary which describe the functionality of the method and lastly the method other related resources and version.

```
/*
Author: Wong Kok Chuan WEK000119 wkc80@hotmail.com
Created: 01 Jan 2003
Version: 1.0.0
*/
```

Equally important is header block comment for each of the stored procedure which I have created. This header block describes the author, the date created and the version of the stored procedure.

6.3 Inline Comment

Inline comment is the most widely used documentation method used in any programs written. This technique enables the readers of the program understand a particular statement or block of statements that the program is trying to perform.

```
protected override void Render(HtmlTextWriter output)
{
    // If no caching is specified, render the child tree and return

    if (_moduleConfiguration.CacheTime == 0)
    {
        base.Render(output);
        return;
    }

    // If no cached output was found from a previous request, render
    // child controls into a TextWriter, and then cache the results
    // in the ASP.NET Cache for future requests.

    if (_cachedOutput == null)
    {

        TextWriter tempWriter = new StringWriter();
        base.Render(new HtmlTextWriter(tempWriter));
        cachedOutput = tempWriter.ToString();

        Context.Cache.Insert(CacheKey, _cachedOutput, null,
            DateTime.Now.AddSeconds(_moduleConfiguration.CacheTime),
            TimeSpan.Zero);
    }

     // Output the user control's content

    output.Write(_cachedOutput);
}
```

Above is a code snippet which shows the importance of inline comment. It serves a great help in documenting the purpose of the code written and the logic which the author of the program trying to expressed.

So, inline documentation is not only useful for others developers but also for the authors themselves as to recall what the particular codes or algorithms serves.

6.4 Self Explained Code

Documentation is indeed very important. However, excessive documentation in a program makes the program unpleasant to read. If developers have to refer every code written with the help of the program's comment, this would be a very tiring job.

So, I have decided to make the code itself self explained. In many cases, the developers will be able to understand the code just by reading at them without the help of the program documentation.

Below is a code snippet show how the code is self explained.

```
private void BindData( DateTime selectedDate )
{
   tmws.TicketManager tm = new tmws.TicketManager();

   if ( tm.IsAuthenticated() == true )
   {
      amws.ActivityManager am = new amws.ActivityManager();
      DataSet dstActivity = new DataSet();

      dstActivity = am.GetUserActivities(
         Application["LicenseeKey"].ToString(),
         tm.GetCurrentUserID(), selectedDate);

      DataGrid1.DataSource = dstActivity;
      DataGrid1.DataBind();

      if ( dstActivity.Tables[0].Rows.Count == 0 )
      {
         Message.Text = "No activity found in " +
            selectedDate.ToLongDateString() + ".";
         DataGrid1.Visible = false;
      }

      else
      {
         Message.Text = tActivity.Tables[0].Rows.Count.ToString() +
            " activity found in " + selectedDate.ToLongDateString();
         DataGrid1.Visible = true;
      }
}
```

If you have observed the code clearly, you will find out that the method BindData(), as its name implies, is used to bind a dataset to a datagrid based on the date passed in to this method. First, this method declares a new object of class TicketManager() from the FCSIT XML Web Services.

Then, it uses the IsAuthenticated method from the class TicketManager to check if the user has already been authenticated or not. If you have noticed the named of this method, you may easily guess out that this method returns a Boolean data type. If the return value is true, then a new object of class ActivityManager also from the FCSIT XML Web Services is declared.

Also, this method will declare a dataset named dstActivity. This identifier is made self explained by which the identifier start with dst which is the acronym for dataset and Activity which somehow the developers can guess that it is somehow related to the activity data.

Then, the following statement calls the method GetUserActivities from the class ActivityManager. Again, the identifier which used to name the method is self explained. From the method, one can guess out that this method is used to get or obtain some information about the user activity on a particular date.

As a convention, the identifier used to name a method that begin with prefixes such as add, set, edit, remove or delete results in noticeable effect to the data while identifier with get, retrieve or is make no changes to the data.

Then, special object type usually starts with the acronym of the class name, for example dstUser means this variable is of dataset object, btnSignIn means this variable represents a button and so on.

However, variables that represent the primitive data types such as the integer, char, double and so forth are normally not prefixes with their respective acronym as for recommended by Microsoft programming practice.

7 SYSTEM TESTING

Testing is not first place where faults finding occur. In fact, as early as requirement analysis, design and coding phases, problems and fault identification and correction have been carried out.

However, in this phase, testing is carried out to demonstrate the existence of faults but not to demonstrate the correctness of a program as many developers feel. The final result is not to produce a perfect system, but to minimize the occurrence and impacts of faults during usage.

Testing is a very complex task. Careful test planning is needed to help design and organize the tests, so that programs are tested thoroughly. So, I have decided to perform three stages of testing.

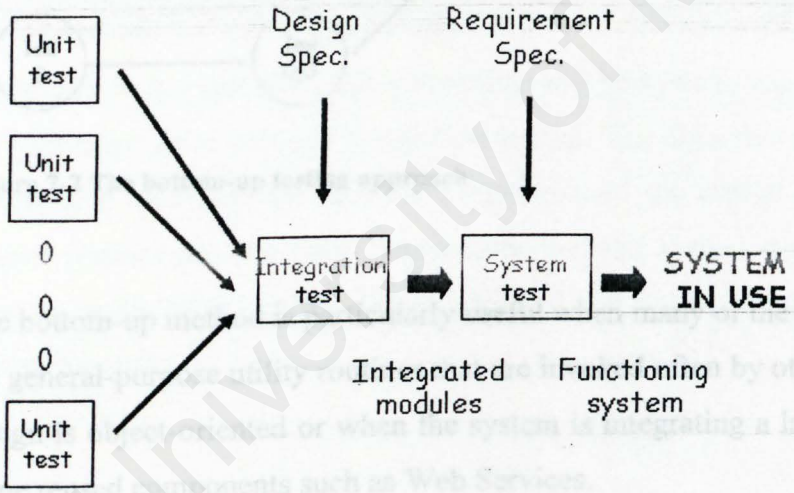


Figure 7-1 The three stages of testing planned

First is the unit test. In the unit test, coding for each modules are reexamine thoroughly by reading through it. After that, I tested each of the modules with different sets of test cases to prove the correctness of the result returned.

Next is the integration test. In the integration test, each tested modules are combined together in an organized way and then is tested as a whole. I have chosen the bottom up integration testing approach.

In this approach, each component at the lowest level of the system hierarchy is tested individually first. Then, the next components to be tested are those that call the previously tested ones. This approach is followed repeatedly until all components are included in the testing.

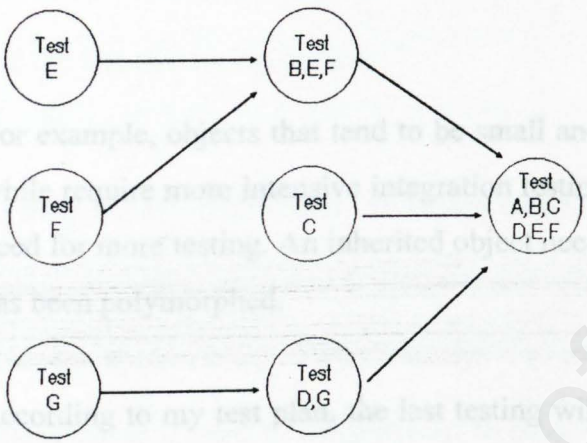


Figure 7-2 The bottom-up testing approach

The bottom-up method is particularly useful when many of the low level components are general-purpose utility routines that are invoked often by others such as when the design is object-oriented or when the system is integrating a large number of stand-alone reused components such as Web Services.

Also, as the system developed is fully object-oriented system, so there are some additional test considerations that I must take in account as stated by Graham (1996a). Graham notes which aspects of an object-oriented make testing easier and which make it harder.

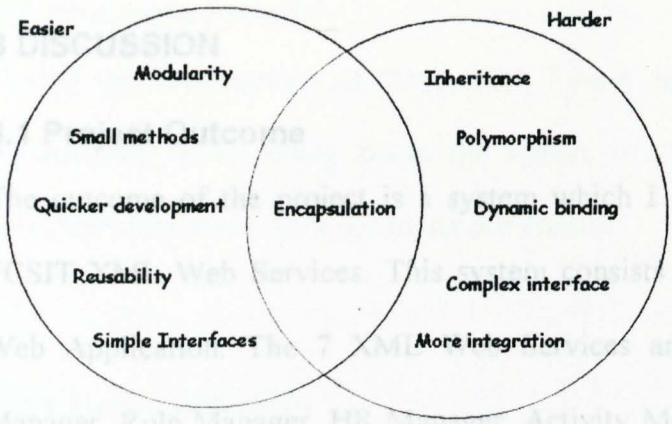


Figure 7-3 The easier and harder part when testing object-oriented systems [Graham1996a]

For example, objects that tend to be small and less complex require less unit testing while require more intensive integration testing. Similarly, inheritance introduces the need for more testing. An inherited object needs additional testing if it is redefined or has been polymorphed.

According to my test plan, the last testing will be system test. Testing the system is very different from unit and integration testing. The objective of unit and integration testing was to ensure that the code implemented the design properly while in the system testing, the objective is to ensure that the system does what the customer wants it to do. Also, during system testing, I and my partner have jointly tested the system together.

8 DISCUSSION

8.1 Project Outcome

The outcome of the project is a system which I and my partner named it as the FCSIT XML Web Services. This system consists of 7 XML Web Services and a Web Application. The 7 XML Web Services are the License Manager, Ticket Manager, Role Manager, HR Manager, Activity Manager, Attendance Management and Leave Management XML Web Services. The first five were developed by me and the last two by my partner.

As their names implies and mentioned earlier in this document, the License Manger is used to protect the FCSIT Web Services from unauthorized usage by anonymous sites or applications, TicketManager is used to authenticate registered users, RoleManager is used to manage user to roles mappings, HRManager is used to store information regarding the human resource, ActivityManager is used to set, edit, delete and generate report for user activities / reminders, AttendanceManager is used to manage staffs' daily attendance and LeaveManager is used to manage staffs' leave applications.

Also, I have also developed a web application, which I have named it as the FCSIT Portal, is used to consume and integrate the XML Web Services, demonstrates the functionality and interoperability of the XML Web Services developed and also provides the graphical user interface for the end users while making the service transparent to them when interacting with the services.

8.2 Problems Faced

During the development of the project, I have encountered several problems and obstructions which really made the matter worse. Some of the problems were solvable while some were not under our control.

8.2.1 Time Limitation

This problem is considered as my most haunting obstacle which I faced throughout the project. As the proposed project scope was quite large and the duration give was too short, I frequently faced the problem of late completion.

As the technologies that I have incorporated in the system are very new not only to me, so I have to learn and apply at the very same time without time for practice. So, the development time was not fully utilized for implementation only but also for learning.

Also, concurrently I was taking several other subjects, so my time is cannot be fully devoted to this project. I have to divide my time for other subjects as well especially the exams season and other projects from those subjects taken.

8.2.2 Inexperience

During the time of development, I was really inexperience in handling such a large scale project. In the previous semesters, what were taught to us were just minor projects and programs which can be done in couple of days or even couple of minutes but this project was totally different.

This project is a full length software development project which comprises all the phases such as the requirements captures analysis, design, and implementation, testing and finally delivering the full system.

So, the project management was not done professionally though we have designed the project schedule nicely and received full cooperation from my partner. Also, inexperience in the new technologies which is the .NET Framework makes the design and implementation phase even more difficult to handle.

8.2.3 Requirement Doubts and Design Ambiguities

During the design phase, I faced a lot of doubts and ambiguities in determining the final system requirements and finalizing the best system architecture. This is most probably the requirements capture phase was not done thoroughly which then leads to those uncertainties occurred during the design phase.

In the system design phase, several architectures have been proposed and suggested. Each of them has their pros and cons over the others. Thus, I have faced the difficulties in choosing the final design for the system to be implemented.

8.2.4 Programming Bugs and Runtime Errors

During the implementation phase which the system design is turned to source code, I was struggling to look for the simplest algorithm to accomplish the stated requirements.

Syntax errors were not threatening anymore with the advent of powerful tool like Microsoft Visual Studio .Net as the IDE pointed out the syntax errors on coding time and also during compilation time.

However, the logic errors were still a headache to me when solving complex problem which requires complex algorithm. This type of error can only be revealed during testing of the method or module and can be very tedious to locate and fix the error.

And lastly were the runtime errors which were my worse nightmare. Some of these errors can be solved but some were totally out of my sense and beyond my ability due to my knowledge and experience.

8.3 Solutions Formulated and Applied

8.3.1 Follow Schedule Closely

The project schedule that I have made is followed as close as possible. This is because the duration given to the project must not be extended. So, I always kept in mind that the schedule must be met whenever possible.

When a module is about to overdue, I will try to work overtime and work through my weekends to accomplish the module so that the impacts could be reduced and the subsequent schedule could still be met.

Also, as time was really pushing, some of the extra features were taken out from certain modules to spare out some extra time for other more important features which must be achieved.

8.3.2 Refers to Various Resources

The technology used was the Microsoft .NET Framework which includes the ASP .NET, the new C# and VB .NET languages, XML and the new Web Services architecture. Each of them was so new to me and even to the developers' community around the world.

So everything is made to be self learning. And as time was a major constraint, I have to self learned and then apply them to the project immediately which much time to practice.

The resources which I referred to vary widely ranging from the Internet to e-book, white papers, product documentation, help files, online forums, reference books, and magazines till attending free seminars.

8.3.3 Utilization of the Proper Tools and Techniques

When designing and implementing the system, appropriate tools and techniques were used. In the designing phase, the object-oriented modeling technique such as the UML and Rational Rose were used.

In the designing and testing phase, the Microsoft Visual Studio .NET, notepad and few graphics tools were used. The Microsoft Visual Studio .NET was really a great tools with many features that assisted me in developing, debugging and testing the system.

So, choosing the correct techniques and tools for different development phases really helps in reducing the burden and eases many tedious tasks. They also helps in reducing development time, increases productivity, reduces errors and produce better results.

8.4 Product Strength

8.4.1 Modularity

This system is consists of separate independent XML Web Services but interrelated. This makes the system very modular in the sense that new XML Web Services can be added, modified or even removed without affected the others.

The modularity of the system is also being further enhanced with the design of the FCSIT Portal which makes use of web user controls (.ascx) to build the content of the site. The web user controls can be added, modified or removed dynamically at the runtime that.

8.4.2 User friendly

User friendliness is heavily emphasized when designing the graphical user interface. The GUI is designed so that it is easy to understand and use even for new users. This feature is future enhance with the help on page (HOP) which helps the user right from the spot.

As I realized that in the user point of view, the attractive of the interface design is also very important in rating a system. So, I have tried my best to made the system more attractive with the use of some pictures and icons.

8.4.3 Interoperability

The system has proven to be able to interoperate with different languages and environments. This is because the system employs the XML and SOAP which is complies with the industry standard.

As a proof, the system now is build up of two languages, which are the VB .NET and C#. Also, I have tested the system with Java applet and Java application that consume the FCSIT XML Web Services that I have developed.

8.4.4 Reusability

The FCSIT XML Web Services are made to be as generic as possible. This is to promote the reusability and adaptability for various environments. The reusability of the FCSIT XML Web Services is also made possible with database partitioning which virtually separate the data of different licensees.

The system is a fully object oriented system. So, the system is made up of classes and components that can be reused for other software development project. These components are also fully documented for the ease of reusability.

8.4.5 Security

The system also emphasizes high security feature which protect the credentials from malicious users. Encryption was used in various parts of the system to protect the credentials especially during transmission over the network and data storage.

Also, role based authorization is used so that user with different roles or privileges view only the data that he / she assumes. This user to role mapping was also been further enhanced with the support of multiple roles to a single user account.

8.5 Product Limitations

When the time the product was delivered, there are still few limitations which the system is still lacking of. These limitations are not corrected before delivery because of time constraint and are not the necessary features that this system must have.

8.5.1 License Manager Web Service

An internal web application is not provided for the service manager to govern or administer the XML Web Services. So, the service manager has to use the .asmx interface for this purpose.

Another limitation which I discovered is the lacking of usage reporting and auditing features that can be used by service provider to charge their licensees and see the performance, abuse and so on.

8.5.2 Ticket Manager Web Service

Again, an internal web application is not provided for the service manager to govern or administer the XML Web Services. So, the service manager has to use the .asmx interface for this purpose.

In fact during the design phase, the encryption algorithms that I proposed were the TEA, DES or 3DES encryption algorithm. However, the implementation was not successfully as the ciphertext produced by Javascript differs from C#. So, I was forced to come out with my own encryption algorithm. This algorithm obviously is not strong and not up to enterprise standard.

This authentication XML Web Services is currently limited to only web application with support for http session and cookie enabled user agent. So, other applications that do not support for http session and cookie will not able to use this services properly.

8.5.3 Role Manager Web Service

This is the same problem again. An internal web application is not provided for the service manager to govern or administer the XML Web Services. So, the service manager has to use the .asmx interface for this purpose.

This XML web services currently has no control over the privilege assignment and the usefulness of this web service heavily dependent on how the client side utilizes it.

8.5.4 HR Manager Web Service

Again, an internal web application is not provided for the service manager to govern or administer the XML Web Services. So, the service manager has to use the .asmx interface for this purpose.

Currently, this XML Web Service only provides the very basic HR functionality such as create, edit and remove an employee profile. So, it is lacking of full features that normally enterprise standard HR systems provide such as employee performance analysis, employee training management features and many more.

8.5.5 Activity Manager Web Service

This is the same problem again. An internal web application is not provided for the service manager to govern or administer the XML Web Services. So, the service manager has to use the .asmx interface for this purpose.

Currently, the code that used to retrieve user activity and bind it into a datagrid was not optimized for performance. On every activity viewed by user, this code will get the data all over again from the database.

8.5.6 FCSITPortal web application

No major limitations were discovered so far. Most of the limitations are concerned with future scalability and user friendliness issues.

Currently, this web application support only one level tab. So, if more and more pages are added in, the tab will overflow and cause the GUI to be elongated horizontally.

Also, the input validation such as invalid values and user action verification such as deletion confirmation alert are not implemented in full scale. So, this lacking might cause some unrecoverable action made.

Currently, when new modules are added into the system, dialog box which prompt user for file location was not implemented in point and click feature. So, user has to type in the path correctly.

8.6 Future Enhancements

There is still a lot of future enhancements that can be made upon the system that I have developed. This is due to limited time which I have to develop a really complete system of this large scale alone.

This system is made to be easily enhanced because of their modularity and object oriented feature. So, changes to the logic will not affect the others as long as their interface remains the same. Below are some suggested future enhancements:

8.6.1 License Manager Web Service

Develop an internal administrative web site for the service manager to maintain the XML Web Services. Also, features such as better ways of licensing technique, usage auditing and reporting, license management and charging and many more can be added into this XML Web Services to make it better.

8.6.2 Ticket Manager Web Service

Develop an internal administrative web site for the service manager to maintain the XML Web Services. Then, try to implement the industry standard encryption algorithm such as TEA, DES or 3DES which are strong enough to prevent crackers. Also, extend the support for clients that do not support for http session and cookie.

8.6.3 Role Manager Web Service

Develop an internal administrative web site for the service manager to maintain the XML Web Services. Also, try to include privilege assignment feature in this system and less dependent on the client side programming.

8.6.4 HR Manager Web Service

Develop an internal administrative web site for the service manager to maintain the XML Web Services. Try to refer to some enterprise standard HR systems such as system developed by SAP and Peoplesoft, and then include those powerful features into this HRManager Web Service.

8.6.5 Activity Manager Web Service

Develop an internal administrative web site for the service manager to maintain the XML Web Services. Try to implement the crystal report for the activity as a web service. Currently, the crystal report was done in this way.

8.6.6 FCSITPortal web application

Enhancement to this web application can be divided into two categories. First by redevelop the framework itself or secondly by adding in new modules.

As for the first option, try to enhance the tab to support multi levels tab organization. Also, try to make this portal framework to support drag and drop feature. As for the second option, it is up to your personal creativity or future requirements to decide on what module to be created to enhance the existing feature of this FCSIT Portal.

8.7 Conclusions

After two semesters of hardship, finally this project has come to an end. However, it also means the starting of another phase which is the maintenance phase which the system will be going through a pre-go-live simulation before it is really used in my faculty.

Overall, I am really happy and feel proud of the system which I have developed so far, though with some minor deficiencies. After delivering the system, I will have to make some refinements to the system in order to fix some defects and refine some of the features that my supervisor, Mr. Ang and moderator, Mr. Nizam have discovered during my viva.

The major regret is not being able to see the system running in production environment. Also, because of the time constraints, I have to limit the system to only fulfilling the basic requirements and needs only. I am lacking of time to implement the extra features which I think they would definitely makes the system more powerful, interactive, user friendly, reliable, scalable, secure and flexible.

As a conclusion, I have learned a lot of new ideas, technical and non-technical skills, extra knowledge and invaluable experience from this project. I am sure they will be very useful and will act as an added advantage to me in my future as I am about to be graduating soon this year.

And lastly, I really hope that the system can make itself through the pre-go-live simulations and testing and be used in production.

GLOSSARIES

Web Services

- A Web service is a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via internet-based protocols.
- A Web Service is programmable application logic accessible using standard Internet protocols. Web Services combine the best aspects of component-based development and the Web.
- Web Services are self-contained, modular applications that can be described, published, located, and invoked over a network, generally, the Web.

SOAP

- The **Simple Object Access Protocol (SOAP)** is a lightweight, XML-based protocol for exchanging information in a decentralized, distributed environment. SOAP supports different styles of information exchange, including: Remote Procedure Call style (RPC) and Message-oriented exchange.
- **RPC style** information exchange allows for request-response processing, where an endpoint receives a procedure oriented message and replies with a correlated response message.
- **Message-oriented** information exchange supports organizations and applications that need to exchange business or other types of documents where a message is sent but the sender may not expect or wait for an immediate response.

UDDI

- The Universal Description, Discovery, and Integration (UDDI) specification is an online electronic registry that serves as electronic *Yellow Pages*, providing an information structure where various business entities register themselves and the services they offer through their WSDL definitions.
- There are two types of UDDI registries, public UDDI registries that serve as aggregation points for a variety of businesses to publish their services, and private UDDI registries that serve a similar role within organizations.

WSDL

- The Web Services Description Language (WSDL) is an XML format for describing network services containing RPC-oriented and message-oriented information.
- Programmers or automated development tools can create WSDL files to describe a service and can make the description available over the Internet.
- Client-side programmers and development tools can use published WSDL descriptions to obtain information about available Web Services and to build and create proxies or program templates that access available services.

REFERENCES

The Microsoft Website
URL: http://www.microsoft.com

The Microsoft Developer Network (MSDN) Library
URL: http://www.msdn.microsoft.com

The IBM Website
URL: http://www.ibm.com

The Java Official Website
URL: http://ww.java.sun.com

The W3C Organization Website
URL: http://www.w3.org

The Rational Website
URL: http://www.rational.com

The Apache Website
URL: http://www.apache.org

The MySQL Website
URL: http://www.mysql.com

BIBLIOGRAPHY

Agosti, M., Smeaton, A. F. (1996). *Information Retrieval and Hypertext.* 2nd ed. Kluwer Academic Publishers.

Shari, Lawrence, Pfleeger (1998). *Software Engineering Theory and Practice.* International Edition. Prentice Hall Inc.

Coulouris, G., Dollimore, J., Kindberg, T., (2001) *Distributed Systems Concepts and Design.* 3rd ed. Adison Wesley.

Mohd. Noorman, Kamarulariffin A. Jalil, Safawi A. Rahman, (2001). *Analisis & Rekabentuk Sistem Maklumat.* McGraw Hill.

Kendall, S., (1999). *UML Explained.* Adison Wesley.

Schmuller, J., (1999) *Teach Yourself UML in 24 Hours.* SAMS.

P. Sellapan, (2000), *Software Engineering Management & Methods.* Sejana Publishing.

Abdullah Embung, (2000), *Sistem Pangkalan Data Konsep Asas, Rekabentuk dan Perlaksanaan.* 2nd Ed. Tradisi Ilmu Sdn. Bhd.

Forouzan, B. A., (2000), *Data Communication and Networking.* 2nd Ed. McGraw Hill.