

**Faculty of Computer Science and Information Technology
University of Malaya
Kuala Lumpur**

Leave Management System (LMS)

**being a Dissertation submitted in partial fulfillment
of the requirements for the Degree of
Bachelor of Information Technology**

in the University of Malaya

by

**Chan Yim Khim
WET 98071**

Under the supervision of

Mr. Ling Teck Chaw

January 2001

ABSTRACT

This project attempts to further enhance and develop new features and functionalities for the existing Leave Management System (LMS) in the Faculty of Computer Science and Information Technology (FCSIT). The existing LMS is a web-based application designed to automate leave processing system in the faculty.

Briefly, this project strives to test the efficiency of the system, to ease the process of leave application, to make the system user-friendlier, and to enhance the integrated system with the Attendance Management System (AMS).

Its significance includes higher quality of the system in terms of its efficiency and effectiveness, faster processing time, and having a centralized management of corporate resource holdings. In addition, LMS is hoped to project the main idea of a Generic Office Environment of promoting a thorough paperless environment with the routing of information through workflow applications.

The proposed methodology encompasses enhancements of the current system, as well as development of new functionalities. This came about from review of literature, whereby it has cited considerable feedback and recommendations from various sources. From there, a thorough analysis is done to find out the feasibility of this project. After determining that the project is feasible, the approach suggested would be the spiral model, where flexibility is the main advantage in coming up with enhancements and new developments. Systems designs with appropriate diagrams and tables are given to describe the system.

With the systems designs at hand, the system is implemented and developed accordingly. Step-by-step guidelines are provided to ensure clarity and avoid ambiguity. In order to test the accuracy and reliability of the system, testing is done in five phases; unit testing, module testing, interface testing, integration testing, and lastly system testing. Lastly, the system is evaluated in terms of its strength and limitations. Problems uncovered during the evaluation period are discussed, and solutions to overcome those problems are provided.

After overcoming a myriad of challenges, this project has smoothen the processing of leaves without any more hiccups, enable up-to-date information retrieval, complete an optimized database, and include more input controls in the system. As such, it is hoped that it will serve the faculty better and fulfill its aim in having a Generic Office Environment.

ACKNOWLEDGEMENT

The journey of executing this thesis was of particular significance to me as I had totally lost touch of programming during my industrial training days! The completion of this project was due to the effort and encouragement of the following people who have helped me garner so much of invaluable knowledge and hands-on experience.

I wish to extend my heartiest gratitude to:

- Mr. Ling Teck Chaw, my supervisor, for his tireless and invaluable guidance, and willingness to impart his knowledge and experiences in assisting me throughout the project.
- Miss Nor Aniza, my moderator, for her precious consideration and decision.
- Lecturers and staff at Faculty of Computer Science and Information Technology, for their assistance and guidance in enabling me to complete this arduous task.
- Miss Goh Sze Eng, my partner for the LMS and AMS integration, for her tireless and precious guidance, advice and assistance in helping me understand whenever I am blur!
- All my friends who have directly or indirectly assisted me in numerous ways in completing this thesis.

Last but not least, to my family members who have been so patient in my disappearance for the past few months! I am truly grateful to God who has provided me this wonderful opportunity to gain so much during the duration of the thesis. Thank God.....

TABLE OF CONTENTS

ABSTRACT	i
ACKNOWLEDGEMENT	ii
TABLE OF CONTENTS	iii
LIST OF FIGURES	vii
LIST OF TABLES	viii
 Chapter 1: INTRODUCTION.....	 1
1.1 Project Overview.....	2
1.2 Project Objectives.....	3
1.3 Project Scope.....	4
1.4 Significance of Project.....	6
1.5 Project Schedule.....	6
1.6 Report Organisation.....	8
 Chapter 2: LITERATURE REVIEW.....	 9
2.1 Purpose.....	9
2.2 Approach and Findings.....	9
2.2.1 Analysis of Existing System.....	9
2.2.2 Purposive Sample and Interview.....	10
2.2.3 Internet Research.....	11
2.2.3.1 Analysis on Similar Leave Management Systems.....	12
2.2.3.2 Analysis on Current Software Development Tools.....	14
2.2.4 Newsgroups.....	15
2.2.5 References.....	16
2.2.6 Group Discussions and Brainstorming Sessions.....	16
2.3 Analysis and Main Conclusions.....	16
 Chapter 3: METHODOLOGY.....	 18
3.1 Project Description.....	18

3.1.1	Enhancement.....	18
3.1.2	Development of New Functionalities.....	18
3.2	Motivation.....	19
3.3	Approach.....	20
3.4	Justification.....	21
3.4.1	Enhancement.....	21
3.4.2	Development of New Functionalities.....	21
3.5	Development Strategy.....	22
3.5.1	Functional Requirements.....	22
3.5.1.1	Applicant Module.....	22
3.5.1.2	Approver Module.....	23
3.5.1.3	Administrator Module.....	24
3.5.2	Non-functional Requirements.....	24
3.5.2.1	Security.....	24
3.5.2.2	Database Backup.....	25
3.5.2.3	Reliability.....	25
3.5.2.4	Flexibility.....	25
3.5.2.5	Scalability.....	25
3.5.2.6	Usability.....	25
3.5.2.7	Interoperability.....	25
3.5.2.8	Manageability.....	26
3.5.2.9	Users and Human Factors.....	26
3.6	Proposed Tools.....	26
3.6.1	Reasons to continue using ASP.....	26
3.6.2	Reasons to continue using SQL.....	27
Chapter 4:	SYSTEM DESIGN.....	28
4.1	System Functionality Design.....	28
4.1.1	LMS Three-tier Client/Server Architecture.....	28
4.1.1.1	Client Tier / First Tier.....	28
4.1.1.2	Middle Tier / Second Tier.....	28

4.1.1.3 Third Tier.....	28
4.1.2 System Structure Chart.....	29
4.1.3 Process Flow Model.....	30
4.2 Database Design.....	32
4.2.1 Database Structure.....	32
4.2.2 Data Dictionary.....	32
4.3 Graphical User Interface Design.....	34
4.4 Security Design.....	36
4.5 Database Backup.....	37
4.6 Users and Human Factors.....	38
Chapter 5: SYSTEM IMPLMENTATION AND DEVELOPMENT.....	39
5.1 System Implentation.....	39
5.1.1 Implementation Environment.....	39
5.1.1.1 Windows NT Server.....	39
5.1.1.2 Internet Information Server (IIS).....	40
5.1.1.3 Microsoft SQL Server 7.0.....	42
5.1.1.4 Microsoft Exchange Server 5.5.....	51
5.2 System Development.....	53
5.2.1 Coding.....	53
5.2.2 Interface.....	57
5.2.3 Database.....	60
5.3 Conclusion of System Development and Implementation.....	60
Chapter 6: TESTING OF PROGRAM.....	61
6.1 Unit Testing.....	61
6.2 Module Testing.....	69
6.3 Interface Testing.....	75
6.4 Integration Testing.....	75
6.5 System Testing.....	80
6.6 Conclusion of Testing.....	80

Chapter 7: EVALUATION AND CONCLUSION.....	81
7.1 Strength.....	81
7.1.1 Wide-accessibility.....	81
7.1.2 Interoperability.....	81
7.1.3 Coding Reusability.....	81
7.1.4 Confidentiality and Integrity of Information.....	81
7.1.5 Scalability.....	82
7.1.6 Reliability and Accuracy.....	82
7.2 Limitations.....	83
7.2.1 Browser and Platform.....	83
7.3 Problems and Solutions.....	83
7.3.1 Multiple Accounts for Second Approver.....	83
7.3.2 Ineffective Table.....	86
7.3.3 Integration Challenge.....	87
7.3.4 Lack of Time.....	88
7.4 Future Enhancement.....	88
7.4.1 Fine Tuning of System.....	88
7.5 Knowledge Gained.....	88
7.5.1 Ability to Set Up Windows NT Servers/Workstations.....	88
7.5.2 Understanding of Active-X Technology.....	89
7.5.3 Coding using Active Server Pages (ASP).....	89
7.5.4 Using SQL Server.....	89
7.5.5 Using Transact-SQL.....	89
7.5.6 Debugging Skill Improved.....	89
7.5.7 Requirements Capturing.....	90
7.6 Conclusion.....	90
 REFERENCES.....	 91

LIST OF FIGURES

Type	Description	Page
Figure 1.6	Project Schedule	7
Figure 3.2	Spiral Model	20
Figure 4.1.1.3	LMS Three-tier Client/Server Architecture	29
Figure 4.1.2a	System Structure Chart for LMS and AMS Integration of Interface	29
Figure 4.1.2b	LMS System Structure Chart	30
Figure 4.1.3	Process Flow Model for leave application process	31
Figure 4.2.1	Database Structure	32
Figure 4.3a	Function Buttons for Applying Future and Past Leave	35
Figure 4.3b	Button for 'List Of Approvers'	35
Figure 4.3c	Portion Interface of 'List Of Approvers'	36
Figure 5.1.1.2a	Creation of Virtual Directory	40
Figure 5.1.1.2b	Enter Alias To Access Virtual Directory	41
Figure 5.1.1.2c	Enter Physical Path of Directory	41
Figure 5.1.1.2d	Set Access Permissions	42
Figure 5.1.1.3a	Start SQL Server Agent	44
Figure 5.1.1.3b	Create A New Job In SQL Server Agent	45
Figure 5.1.1.3c	Enter General Information for Job	45
Figure 5.1.1.3d	Enter Job Steps	46
Figure 5.1.1.3e	Enter Job Schedule	47
Figure 5.1.1.3f	Edit Recurring Job Schedule	47
Figure 5.1.1.3g	Enter Job Notifications	48
Figure 5.1.1.3h	Create New Stored Procedure	49
Figure 5.1.1.3i	Type in Stored Procedure	50
Figure 5.1.1.3j	Architecture between SQL Mail and Mail Server	50
Figure 5.2.2a	Interface to Change Password and View Daily Attendance	57
Figure 5.2.2b	Old Interface	59

Figure 5.2.2c	New Interface	60
Figure 6.1a	Applicant from JBDEM Applies For Leave	64
Figure 6.1b	Leave Application Received by Approver, KJBDM	65
Figure 6.1c	Mail Notification of Approved Leave	66
Figure 6.1d	Error Message Indicating Overlap of Leave Application	67
Figure 6.1e	Error Message Indicating Incorrect Sequence of Date Range	68
Figure 6.1f	Error Message Indicating Mismatch in Applying for Future Leave	69
Figure 6.2a	Administrator Adds New Holiday	70
Figure 6.2b	Administrator Sets 1 st February, 2001 as a Public Holiday	71
Figure 6.2c	A Non-Working Date is Added	72
Figure 6.2d	Applicant Applies Leave on a Non-Working Date	73
Figure 6.2e	Error Message Informing Applicant of the Invalid Leave Date	74
Figure 6.4a	Applicant Logs In Through The Integrated Interface	76
Figure 6.4b	Applicant Enters LMS by Clicking 'Leave' on the Menu Bar	77
Figure 6.4c	Applicant is Directed to His/Her Own Leave Applicant Page	78
Figure 6.4d	Applicant Logs Out and is Redirected to the Integrated Main Page	79
Figure 7.3.1a	Previous Approval of Leave Application Flow	85
Figure 7.3.1b	New Approval of Leave Application Flow	85

LIST OF TABLES

Type	Description	Page
Table 4.2.2a	Some of the fields in <i>ApplicationInfo</i> table	33
Table 4.2.2b	Some of the fields in <i>Application</i> table	33
Table 4.2.2c	Fields in <i>Approver</i> table	33
Table 4.4	Description of Session Variables	36
Table 5.1.1.4	Installation Procedures for Internet Mail Service	52
Table 6.2	A Non-Working Date is Updated in the Database	72
Table 7.1.1a	<i>DepartmentCode</i> table	84
Table 7.1.1b	Departments for 'TDP'	87
Table 7.1.1c	<i>Approver</i> table	86
Table 7.1.2	<i>DeptCode</i> table	87

Chapter 1: INTRODUCTION

Paper is an inefficient media. It is not easily updated and the information contained on one piece of paper is not easily related to information on other pieces of paper. Except for the very gifted, it is difficult to access and retain all of the information that crosses our lives daily. In addition, the media that carries the words is no longer an economically viable form of transportation. The price of paper also rises every year. As environmental concerns become more critical and fewer trees are harvested, price for the cellulose to make paper is increasing dramatically. Contrast this picture with electronic publishing on computer floppy disks or CD-ROMs (Compact Disk - Read Only Memory). Packaging, distribution, and marketing costs are much lower than they are for paper-based books. Daily, new and innovative ways are being discovered to transfer information at dramatically lower prices.

Today, the communication industries offer nanotechnology, optical storage, virtual reality, interactive real time environments of work and play. The *Generic Office* is just one of these evolving environments, geared to take advantage of technology to support humans' daily needs to access information more efficiently and effectively. This vision of the evolving office environment is to provide the technological and managerial means to enable people to work smarter.

Today's modern office environment requires the establishment of a technology infrastructure that provides connectivity and the necessary tools to get the job done. Data requirements, having evolved into information requirements, have been further delineated into corporate knowledge requirements. Hence, the Generic Office is designed to support knowledge workers; individuals for whom the manipulation of information and knowledge is their business. This is possible, because all knowledge workers require the same core functionality to do their jobs; *to maintain timely access to information and knowledge in a form suited to the intended use.*

The **Generic Office Environment (GOE)** brings order to this dynamic, constantly changing world, by providing a structure that makes work processes easier and more effective. It provides knowledge workers with a quick-and-easy method to find the appropriate information and once found, they can be assured of knowing that what they have found is accurate, consistent and up-to-date. The Generic Office also assists the knowledge

worker develop information by enabling re-use of the existing corporate knowledge base and by ensuring that *the right information gets to the right people, at the right time*.

The move toward streamlining business processes, coupled with the objectives of reducing time to process, while improving overall quality, has brought about this proposal. Any organisations that are going to become paperless and sophisticated will definitely require this system.

1.1 Project Overview

Leave Management System (LMS) in the Faculty of Computer Science and Information Technology (FCSIT) evolved from the Generic Office Environment (GOE), where it forms an integrated workspace to enable office tasks and functions to perform with convenience and efficiency. This system enables users to apply for leave easily and effectively without the usage of paper documents.

The system needs to improve on its functional connectivity so that it can support knowledge workers in maintaining timely access to information in a form suited to the intended use. As such, the enhancement and development of functionalities for the existing system would further improve on the quality of data and information. Thus, the “paperless” GOE would be realised with an organised structure that makes leave application process easier and more effective.

Embedded in the LMS includes functions perform by the applicants, approvers, system administrator, and the super administrator. It is developed using Windows NT Server 4.0 as the platform and Microsoft Internet Information Server (IIS) as the web server. Microsoft SQL Server 7.0 serves as the database source and Microsoft Exchange Server 5.5 as the mail server.

The system is based on a three-tier client/server architecture. It consists of a client tier (web browser), a middle tier (web server and mail server) and a source tier (database server). Users may access the system through a web browser, preferably Internet Explorer version 4.0 and above. It provides the interface for users to process their application and maintain their data. The web server (IIS) processes web pages and the mail server (Exchange) supports communication within the LMS. The database server (SQL) maintains all data records.

Between the client tier and the middle tier, requests are sent and information is received. Between the middle tier and the source tier, database connectivity and communication objects are transferred.

1.2 Project Objectives

The objectives of this project are listed as below:

1. To test the efficiency of the existing LMS in GOE

At present, the current system has yet to meet the needs of every quarter of the staff at FCSIT. Hence, this has brought about the proposal to test the efficiency of the system.

2. To ease the process of leave application by enhancing workflows, functionalities, and features

The leave application process is the main function of this system, and that is why it needs to be enhanced in terms of its workflows to enable it to work efficiently and effectively. New functionalities and features that are necessary would be added in to make sure the process runs smoothly.

3. To verify, normalise, and optimise daily transactions

Daily transactions involve queries being sent from clients to servers and vice versa. To optimise the daily processes, the database has to be normalised and verified to prevent redundancies and inefficient transactions of data.

4. To make the system more user-friendly

This is to enable all staff to perform the functions in the system easily and without much help from the administrator. They should be able to use the system with ease and without any hassle.

5. To ease users with informative and easy-to-use graphical user interface (GUI)

Interfaces are important as a means of communication of the system with its users. Staff should not be confused with the use of words or phrases that relate to their requests in the system. In addition, staff should feel comfortable and at ease while navigating through the interfaces, with informative and understandable information.

6. To enhance the integrated system with Attendance Management System (AMS)

At present, staff has to log in twice in order to enter the LMS and AMS. With the objective mentioned above, staff would be able to access into the two systems only once, thus avoiding redundancy and also the inefficiency of logging in twice.

1.3 Project Scope

The project includes staff of FCSIT, but not the students. However, the scope that covers the integration with AMS involves students as well. This is to enable them to view the attendance of staff related to them, e.g. a certain lecturer that they would like to meet. The interface integration with AMS begins when users log in once via the first interface, and they are subsequently directed to a shared page, whereby they choose to enter AMS or LMS. At the main interface (before they enter either AMS or LMS), users are able to change their password setting and also view daily attendance record of all the staff.

The scope that covers the enhancement and further development of LMS involve the four modules:

1. Applicant (staff who wish to apply for leave)

Applicants are able to apply for leave and get a reply of confirmation or rejection in return. They will have a choice to apply either future leave or past leave that has yet to be applied. Besides that, applicants will be alerted if they attempt to apply leave that falls on non-working dates. Personal information, such as address during leave, can be configured as well. If leave must be revoked, applicants are able to perform this function. Applicants may generate reports on leave applied, and also the total days converted to cash and total days brought forward. At the end of the year, applicants may also make decisions on the

balance of leave applicable; whether they want to convert it to cash or bring it forward to the following year.

2. Approver (the Dean and/or Heads of Departments)

Approvers may approve leaves and revocations of their subordinates, and may view summaries of leave applied by the latter. In addition, they can also approve the total days converted to cash and total days brought forward of those who are under his/her respective department. When approvers are on leave, they may activate another approver or assistant to approve leaves on behalf of them. Personal information may be configured as well. Second approvers (i.e. the Dean and Deputy Dean) will not have separate accounts from being a first approver and a second approver. Instead, they will have one approver account for all departments that need their approvals to avoid any confusion.

3. Administrator/clerk

The administrator may generate reports, e.g. reports on list of holidays, departments, and applicants. For applicants who are not able to apply for leave, the administrator may apply emergency leave for them. Overall, the administrator manages users and approvers of the system through configuration of their details, e.g. log in names. Also, he/she is to administer the whole system, in the sense of making sure the system is working properly and users are able to refer to him/her when discrepancies arise.

4. Super administrator (the Dean)

The super administrator may add or delete the system administrator. He/she may also add the total days converted to cash and total days brought forward for the past two years.

1.4 Significance of Project

Much benefit would be gained in terms of:

1. Higher quality and faster time of revision

Reducing the time spent looking for information and documents, and ensuring the documents drawn for the latest information are always available can shorten overall review and revision process.

2. Enhancing speed of processing

To aim to be more productive in a short period of time and at a lower cost while improving quality, eliminating non-value-added-steps, enabling quicker decision making processes, increasing velocity of transactions, dissolving organisational boundaries, and communicating information promptly and accurately.

3. Centralised management of corporate resource holdings

By centralising the management and administration for shared resources, any information needed can be found easily, and the integrity of information would always be preserved. Additionally, with centralisation of management information sharing, time needed for retrieving information would be sharply reduced.

1.5 Project Schedule

The following graph indicates the project time expectations. The process specified in the proposal will take approximately 6 months to complete from date of concept definition to the date of the final testing. The project starts on June 2000 and is expected to end in December 2000.

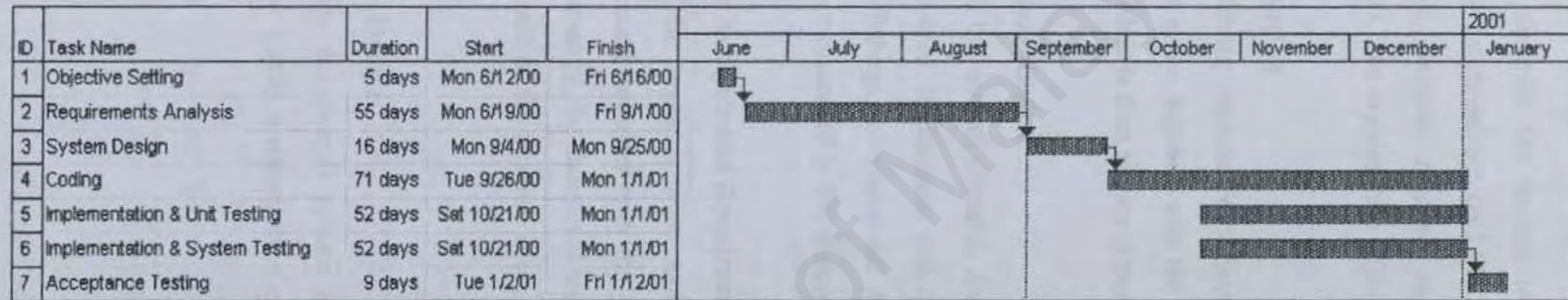


Figure 1.6 Project Schedule

1.6 Report Organisation

This report aims to describe the project intended to be carried out. It proposes the enhancements and new functionalities on LMS, whereby a proper research and analysis have been done to find out, compare, analyse, and propose the suggested methodologies and solutions for this project. The organisation of the report is as follows:

Chapter 1: Introduction

The chapter begins with the importance of having an effective working environment. A brief introduction to LMS is given, together with the objectives, scope, and significance of it to the faculty. A project schedule is then shown to portray the timelines.

Chapter 2: Literature Review

This chapter discusses the overall research done on this project. Firstly, the purpose of the literature review is covered. It proceeds with the approach used in conducting the research, and subsequently the findings are explained. After that, an analysis of what was found and the main conclusions are discussed in this chapter.

Chapter 3: System Analysis and Requirements

A brief description of the project is given, including the modules involved. The approach to be taken is also discussed, together with the justification of the particular approach/model. For the development strategy, the functional and non-functional requirements are listed out to ensure clear development of the system. Also, the chapter explains the proposed tools to be used in this project.

Chapter 4: System Design

It depicts and explains the overall system architecture, database design as well as the graphical user interface. Lastly, a statement of expected outcome of the project is given.

Chapter 2: LITERATURE REVIEW

2.1 Purpose

As this project mainly works on the enhancement of the existing system, the review of literature intends to find out the setback of it, and henceforth to seek ways of improving the system, as well as to determine whether the existing system and development tools could still be used. Additionally, new features and functionalities are searched upon to compare and to absorb them, so as to make the system as efficient and effective as possible.

2.2 Approach and Findings

2.2.1 Analysis of Existing System

The existing system was reviewed in terms of its interfaces, coding, and tables in the database. From the main web page of the Attendance and Leave Management System (<http://lms/default.htm>), the second option, which is the Leave Management System, was chosen. From there (<http://lms/lms>), a dummy login name was used to enter the LMS system. The subsequent web pages were scrutinised in order to catch the flow of the system, to find out if there were inefficiencies in terms of details, spellings, flow of interfaces, etc. This process was repeated for each module, namely as an applicant, an approver, an administrator, as well as a super administrator.

From the SQL database, the tables related to the system were looked through, e.g. *ApplicantInfo*, *DepartmentCode*, *Application*, etc. The field names, data types, sizes, and keys of all tables were documented for reference purposes. Besides that, the tables were checked to see if there were repetitive fields that could cause inefficiency of data retrieval.

The coding of the whole system was looked through for each of the module. Using Microsoft Visual Interdev, the coding was read to further understand the flow of the system and the details of each module. From there, all the inefficiencies in terms of coding were enquired, and then "book marked" to ensure they were corrected during the development stage.

Findings

Overall, the present interfaces could still be improved in terms of ease of use for all users. Other imperfections that were found were function buttons sometimes create ambiguity, spelling and grammar mistakes, no exact interface for certain function/web page, no caching to see the latest content, not fully integrated with AMS, etc.

In the database, too many queries are passing between IIS and SQL, resulting in data traffic. For example, three queries are passed. The first query is first passed to the second query, and subsequently the second query would be passed to the third. As such, the actions are rather repetitive.

Coding was found to be redundant for repetitive functions, names of files are the same but they perform different functions or stored in different folders, etc.

2.2.2 Purposive Sample and Interview

Purposive sample is based on judgment, whereby a group of individuals is chosen based on criteria, but upon choosing them, it is a nonprobability sample. The criteria chosen was based on the mentioned four modules. This group of people was interviewed to gather their thoughts, ideas, feelings, and suggestions that they might have for the enhancement and development of the system.

Findings

Constructive feedback was garnered from all three levels of users, and all of them were found to be doable. The feedback is as follows:

- **Applicants**

For non-academic staff, it was requested that the system shows overtime balance to enable application of replacement leave. Also, the system should enable conversion of cash from replacement leave. This option of replacement leave must therefore be transferred from AMS to LMS. To avoid redundancy of leave application, the system should ensure leave that has been approved is recorded in the database.

- **Approvers**

Approvers emphasised on having an automatic detection of leave application, i.e. system is locked when one tries to approve the same application again. Also, the system should improve on the time lapse when applying and approving for leave. Besides that, approval of leave that has just been applied (i.e. less than 2 weeks before leave date) should be enabled, whereby a message should appear to inform approvers of this urgency. The approvers also requested for viewing of staff who are present and on leave.

- **Administrator**

The administrator requested for deletion of those who have resigned from each New Year's day, and converting overtime to cash. The system should also be able to detect leave of contract staff and bring it forward according to their appointed date. Changes that affect approval of leave must also be update automatically, e.g. new appointment of Deputy Dean because it affects the approval of leave of certain staff.

2.2.3 Internet Research

Researched on the World Wide Web was done to look out for similar LMSs and new technologies of the current software development tools. The Internet search engines that were useful in the quest are as follows:

- <http://www.google.com>
- <http://www.altavista.com>
- <http://www.lycos.com>

Keywords such as "leave management system", "leave processing system" and "leave application system" were used. Upon finding these sites, the features were compared and ideas were absorbed to suit the current LMS. However, not all the web sites were useful, as some use expensive resources that are inadequate for the faculty. Besides comparing systems, research was also done to find out if the current development software tools could support this project, since the hardware tools are already fixed.

2.2.3.1 Analysis on Similar Leave Management Systems

A few examples of the web sites are as follows:

- “Prosoft Leave Management System”
<http://www.prosoft.com.sg/lms.htm>
- “Web-based Travel-Leave Management System”
<http://www.chttl.com.tw/pub/major/te88-007.htm>
- “Skyhi Software”
<http://www.skyhi.co.za/products.htm>

Some of the ideas that can be considered were:

- Calculation of staff leave from the day that they joined the faculty
- To seamlessly handle a change in public holidays
- To enable creation of a dynamic new kind of leave
- To enable staff to view leave policies before applying for leave

Below are examples of similar leave management systems that are available in the Internet, and are worth comparing with the current LMS in FCSIT:

➤ **LotusLeaves (Leave Management)** (<http://www.groupware.itil.com/leave.html>)

LotusLeaves is a lotus notes/Domino server based application designed to automate the leave processing system in an organization. The application uses the inherent email environment provided by lotus notes and is the need of any office that plans to be paperless in the future. An organization can define its leave policies using this application and a total approval cycle can be provided for its employees. With this system at the helm of affairs:

1. The organisation need not go to print their leave application forms to any paper printer company.
2. The cost of paper and print charges is out of question in this case.
3. The employees need not wait outside the office of their approver to wait for their leave application to approve.
4. The employees do not have to approach HR department for their pending leave status.

5. Nobody can fiddle with the pending leaves of the employees.
6. The HR department need not have to calculate the leaves of any employee; this will be done automatically.
7. The HR Department need not have to maintain bundle of papers in their files or drawer.
8. The approver can at anytime look for the employees who are on the leave.
9. The HR department can at any time look for the employee's details.
10. The idle time of the server would be better utilised.
11. The employee can at any time look for his past leaves.

Features of this application include:

1. The application contains a full approver cycle.
2. The HR department decides the approval cycle for the employees.
3. The leave policies is in the hands of HR department and the department can specify their policies in the application and the it takes care of the policies and makes the leaves documents of the employees as per the policies (eligible in most of the cases).
4. The employees who do not want to take leaves their leave document will not be created resulting in saving of disk space.
5. The employees do not have to wait for approval of their application if the approver is absent; the application can take care of such eventualities and can be customised accordingly.

➤ **NexTrak Leave Management System** (<http://208.233.127.171/product2.htm>)

NexTrak™ Leave Management software from Kronos® automates the process of managing and tracking employee leave. The *Policy Manager* allows the automation of an unlimited number of leave policies. Policy rules are applied to leave requests, and the appropriate responses are generated when leave is approved or denied.

NexTrak Leave Management software will collect the required data from the Timekeeper® system, apply the policy rules to leave requests, verify eligibility and availability, and issue the required documentation. Fitness-for-duty and other medical certifications can be tracked, as well as employee insurance premium payment while on extended unpaid leave.

Features of this system are as follows:

- ***Application Flexibility***

The system allows use in client/server environment with an Oracle or SQL database, or as a stand-alone application.

- ***Leave Policy Customisation***

It permits a variety of rules to be combined into an unlimited number of administered leave policies. Calculation methods, accumulators, actions required, and concurrencies may also be customized according to users' needs. Once the policies have been defined, the *Policy Manager* evaluates leave requests, reason codes, schedules and accrual balances to determine which policy or policies apply. The system alerts when action is required, such as obtaining medical certifications, leave status updates, or setting up insurance payment schedules. Notices are easily customized using the included document editor, and may even be archived for later retrieval and reprinting.

- ***User-definable Data***

It allows fields to be added to the employee database for use as additional policy, grouping or report criteria.

2.2.3.2 Analysis on Current Software Development Tools

For web sites related to SQL and ASP, they are as follows:

- "Active Server Pages"
http://idm.internet.com/articles/200002/wd_02_25_00aa.html
- "Internets and intranets: ASP"
http://www.cetus-links.org/oo_active_server_pages.html
- "Internet Information Server"
<http://www.furman.edu/~ggallowa/MIIS.html>
- "SQL Magazine"
<http://www.sqlmag.com/>

- “Stored Procedures”

<http://www.sqlmag.com/Articles/Index.cfm?ArticleID=6113&SearchString=stored%20procedures>

Besides the websites mentioned above, the “MSDN Online Web Workshop” web site was found. It provides the latest information about Internet technologies, including reference material and in-depth articles on all aspects of Web site design and development. It can be accessed through:

- <http://msdn.microsoft.com/workshop/>

2.2.4 Newsgroups

Newsgroups were also useful to discuss FAQs, topics such as development tools, system architectures, databases, etc. Questions can be posted and respondents would give their ideas and suggestions. Other search engines to search for newsgroups, besides those mentioned above are as follows:

- <http://www.infoseek.com>
- <http://www.hotbot.com>
- <http://www.about.com>

Findings

This source of information was not exactly that useful in terms of comparing similar LMSs, but many more features of the current development tools were found. A few examples of useful newsgroups sites are as follows:

- <http://www.liszt.com/news/comp/>
- <http://www.liszt.com/select/Computers/>
- <http://agma.ca/~leisen/mlnh/mlnhntables.html>
- <http://www.bookcase.com/library/faq/by-newsgroup.html>

2.2.5 References

Materials such as books, magazines, journals, newspapers, and thesis were read through for new ideas and to make comparisons. New technologies were analysed to see if they are suitable in the current system's environment. Apart from that, since all the system and development tools are currently being used, it was important to know if they can still support the system during development of new functionalities.

Findings

Examples of references are as follows:

Books – Buser, Kauffman, Llibre, Francis, Sussman, Ullman, & Duckett (1999)

Magazines – Tech

Newspapers – In Tech (The Star - Tuesdays), Computimes (New Straits Times - Thursdays)

These sources offer much information regarding the latest technologies currently in the market, but most of them are not applicable to the LMS. Nevertheless, the information is useful for future development of the system.

2.2.6 Group Discussions and Brainstorming Sessions

This method proved to be a productive way in garnering workable suggestions, solutions, and new ideas. Sources of information were from project supervisor, lecturers, peers, and certain staff of FCSIT.

2.3 Analysis and Main Conclusions

In terms of the system's coding, they need to be reviewed again to ensure that the codes are not too lengthy. The codes should be flexible in allowing easy addition of new items or categories. To make sure they are meaningful, codes should indicate the values of the attributes of the entity where possible. Most importantly, codes should be able to adapt to changing user requirements where possible. Tables in the database would have to normalised to ensure easy reference.

Much of the new technologies have to be put off because of lack of funds, or the current environment is inadequate to implement them. Nevertheless, the information is useful for the system if it were to reach higher grounds in the future.

The system should be able to be accessed through the two main web browsers: Internet Explorer (IE) and Netscape Communicator. Since the coding is in Active Server Pages (ASP) language, only IE is able to view the web pages of LMS. Instead, Java language would enable the system to be viewed in the two web browsers. But after considering the limited time frame, Java would not be used since it is anticipated that much time would be spent in changing the coding.

A revamp needs to be done on the interfaces to ensure users are clear with all the functions that are available, and functions that are going to be added in the system later. Also, the Malay version would be eliminated since majority of users are comfortable with the English version. This is also to maintain consistency after full integration with AMS.

Chapter 3: SYSTEM ANALYSIS AND REQUIREMENTS

3.1 Project Description

This project would be carried out in two parts: enhancements based on users' requirements and development of new functionalities. Enhancements would be done in all modules of the LMS, whereas the new functionalities are added in between those modules. In other words, they are hidden between the modules or in the database.

3.1.1 Enhancement

There are altogether about 15 user comments and feedback, in which they have been outlined in *Findings* of section 2.2.2 (Purposive Sample and Interview). All feedback would be taken into account in carrying out enhancements. Apart from that, continuous feedback would also be reviewed and considered during the duration of project implementation.

3.1.2 Development of New Functionalities

Referring to sections 2.2.3, 2.2.3.1, and 2.2.3.2, the new functionalities proposed would:

- Block people from taking leave when important functions arise. This is to ensure faculty activities are not disrupted.
- Avoid redundancy of coding by using stored procedures. This is because stored procedures are an efficient method of encapsulating statements for repeated execution. Besides that, they are quicker to run because they are stored in a precompiled form in the database.
- Fully integrate with AMS, whereby users access both systems by logging in once only.
- Enable deletion of resigned staff automatically at every end of the year and update of leave for contract staff according to their appointed date using cursors.

3.2 Motivation

Using the current software and hardware technologies, the identified user needs and the proposed new functionalities could be satisfied due to a list of motivations as below:

- **Current system is flexible, scalable, and interoperable**

Based on sections 2.2.3.2 and 2.2.5, the hardware and software technologies that the system is currently using are found to be able to support enhancements and new developments. The development tools and systems are also able to interact well. For example:

- Active Server Pages 3.0 partners well with SQL query language
- Combination of SQL Server 7.0 and IIS 4.0 provides users with complete database publishing capabilities
- IIS 4.0 revolutionises the web capabilities of Microsoft Windows NT

- **LMS needs to speed up on data processing and save on time**

Tables in the database can be normalised to make data retrieval more efficient. Data processing is also slow in conversion of overtime to replacement leaves as it is currently being processed manually. Resources are also wasted in terms of paper and manpower in producing necessary documents, when it can be automated and processed faster. This can be referred in section 2.2.5.

- **Current modules in LMS are expandable**

This is an important aspect for it to accommodate changes in enhancements and new functionalities. As previous developments of the system have taken this into account, all proposed modules could be added in the system.

- **Proposed modules will not incur extra costs**

The proposed system will be cost-effective from a business point of view and it can be developed given existing budgetary constraints. This is because major hardware and software technologies are already in use, and as such, no major costs would be incurred.

3.3 Approach

The methodology used in this project would be the *spiral model*. The spiral encompasses loops where each loop represents a phase of the process. The identified phases that fit the two parts of the project are objective setting, development and validation, and planning. The development model shown below does not depict a real spiral model, but a combination of the waterfall, prototyping, and incremental models.

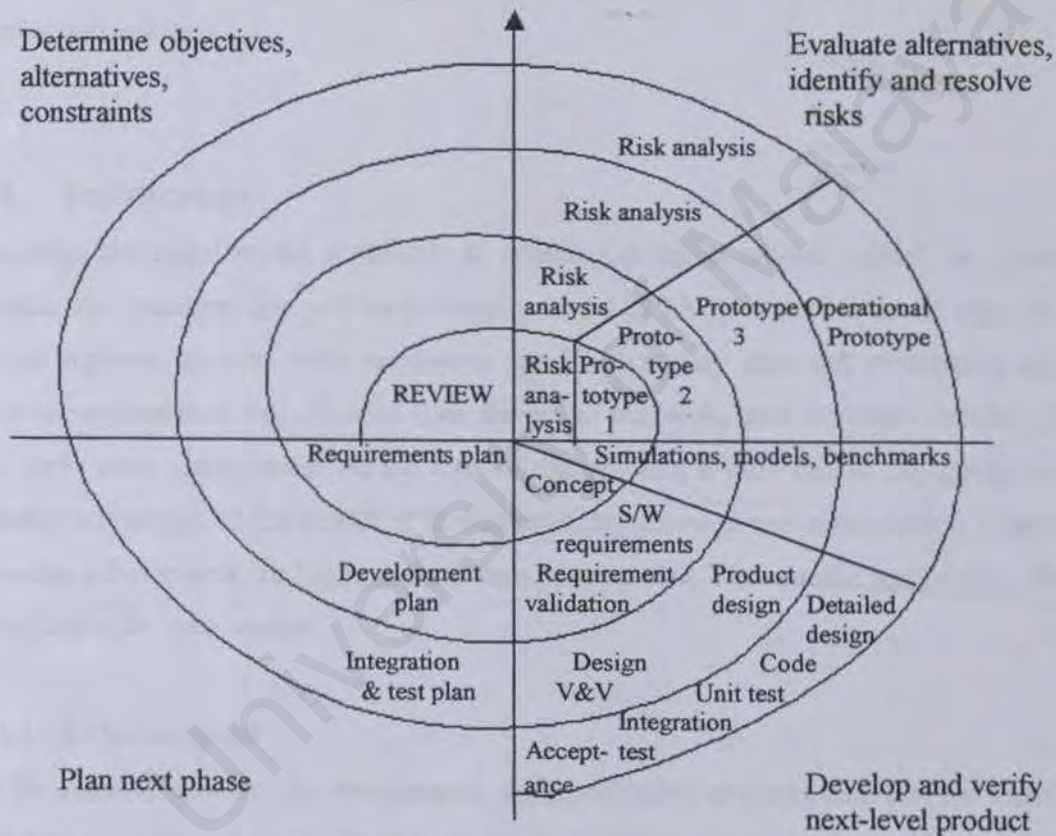


Figure 3.2 Spiral Model

There is no one fixed cycle in the spiral, but it encompasses other process models as well, as it deems fits. As such, the *enhancements* of the project would use its iterative characteristic, whereby the system is exposed to user comments and refined through many versions until an adequate system has been developed. For this particular part of the project, the repetitive process starts from coding until acceptance testing.

For the development of *new functionalities*, it is structured into phases to ensure thorough completion, right from the start of the spiral. An initial implementation is developed, and this is exposed to user comments and then refined through phases in the spiral until a desired system is developed.

3.4 Justification

Basically, the spiral model is flexible in adopting different process models in a particular system. For example, for well-understood systems in LMS, waterfall model may be used. Those high-risk systems with incomplete specification may then use prototyping model to resolve requirements risk. Besides that, the spiral model focuses attention on reuse options and early error eliminations. At the start of the process, it puts quality objectives up front. Another advantage of this model is, it integrates development and maintenance. Generally, it provides a framework for hardware/software development. For specific justification, they are described in the next section.

3.4.1 Enhancement

As the spiral is iterative, the development starts with parts of the system that are understood, and then experimenting with these parts of users' requirements. With that, users' inputs are concurrently being taken into account to achieve the desired end result.

3.4.2 Development of New Functionalities

The model advocates flexibility in adopting process models in each cycle of the spiral, and it is suitable for this part of the system. For example, in the integration with AMS, each interface can be prototyped and tests are run to see which is preferred. Two different

interfaces in the design can be chosen too, so that users can select an interface when they log on. This will ensure ease of use, and also minimise the risk of choosing an interface that will prevent productive use of the new system.

3.5 Development Strategy

With the spiral model adopting different process models, the basic concept is still to start from the requirements specification, all the way to integration and testing. The functional and non-functional requirements are described as below:

3.5.1 Functional Requirements

3.5.1.1 Integration with AMS

In this integrated interface, all users of LMS and AMS will access to only one page, where they log in only once. After that, they will be directed to another page based on their access rights. Below are the requirements that bound the integration:

- **Logging into the system**

Users of AMS and LMS (applicants, approvers and administrator) need to key in their login names and passwords in order to access their respective pages. After verification of the two items, users of AMS and LMS (except approvers and administrator) will be directed to a welcome page. They will then be provided with an option menu on the left. With a click on 'Leave', the applicants will then access their respective leave interface. For approvers and administrator, they will be directed to their respective default leave interfaces right after providing their log in names and passwords.

- **Changing of password**

All users can change their passwords before logging into the system, provided they are authorised users. With this function, applicants and approvers do not need to change their passwords in the 'Configure Personal Information' function.

- **Viewing daily attendance**

All users are able to view the attendance of FCSIT staff; the academic staff, non-academic staff and tutors. Users can find out whether a certain person is present, absent or on leave.

3.5.1.2 Applicant Module

Applicants should be able to apply for leave without any hassle, as well as perform other basic functions, as stated by the requirements below:

- **Application of leave**

Applicants should be provided with a complete, error-free and easy to use interface. They should be able to apply for future and also past leave, where they might not get the chance to apply at that point of time. For non-working dates, the system should be able to block people from taking leave.

- **Conversion of leave**

Applicants should be able to submit the number of leave days desired to be converted to cash. This option should only appear in the month of November and December.

3.5.1.3 Approver Module

Approvers should be able to approve leave applications for those under his/her own department. Besides that, they should be able to perform other basic functions as well. These requirements are stated below:

- **Processing of leave application**

Approvers should be provided with a complete, error-free and easy-to-use interface. The system should be able to detect approval of leave, whereby the system is locked when one tries to approve the same application again. Second approvers' will only have one account and not separate approver accounts if they are also first approvers.

- **Assignment of assistant**

A certain approver would appoint an assistant when the approver is not available to appoint him/her.

3.5.1.4 Administrator Module

Administrator should be the only one who can administer, configure, and monitor the whole system.

- **System configuration**

- *List of applicants*

Administrator should be able to find all applicants by position also. For contract staff, the system should be able to bring forward their leave according to their date of appointment, and not bring them forward at the beginning of the year. From New Year's Day, the system should automatically delete those who have resigned.

- *List of departments*

The system should update changes in editing automatically, especially names of the Dean, the Deputy Dean, and the Heads of Departments.

3.5.2 Non-functional Requirements

3.5.2.1 Security

The system should not show any potential leakage of information. It should be equipped with sufficient security features, as stated below:

- Access by users should be authenticated and validated
- Each user has his/her own unique login name
- Password should be encrypted before sending it online
- Password in the database should be encrypted

3.5.2.2 Database Backup

The system should be able to:

- Automatically launch daily backup
- Restore easily from any potential disaster
- Automatically delete leave applications that have been stored for years after backing up the records

3.5.2.3 Reliability

The system should perform its intended functions with high accuracy. Thus, the system should be reliable to perform its daily operations with required consistency. It should be able to operate 24 hours a day, 7 days a week, and 365 days a year.

3.5.2.4 Flexibility

The system should have the capability to take advantage of new technologies and resources, such as stored procedures. It should be able to be implemented in changing environments.

3.5.2.5 Scalability

The system should be capable in migrating as a client or server to machines of greater or lesser power, depending on requirements, with little or no change to underlying components. Database scalability issues can be resolved using distributed database architecture whereas web application scaling can be addressed by increasing bandwidth or additional web servers.

3.5.2.6 Usability

The system should be developed in such a way that it is easy to use. It will enhance and support processes, instead of limit or restrict them. Interfaces should be intuitive and consistent with itself and AMS.

3.5.2.7 Interoperability

The system should be capable of working with different types of applications to share data and processes.

3.5.2.8 Manageability

The modules in LMS should be easy to manage. This enables easier work of maintenance and enhancement.

3.5.2.9 Users and Human Factors

The system should be user-friendly with easy-to-use interface. Also, the skill needed to execute the functions should be minimal. It should automatically detect any discrepancies, for example:

- Appropriate error messages should be displayed when errors are made
- Data entered should not be truncated
- Format of data should be specified properly to avoid confusion

3.6 Proposed Tools

As mentioned in the earlier sections, this project will be dealing with enhancements and development of new functionalities. Hence, all of the existing tools are going to be used. The development and platform tools are as follows:

- Windows NT Server 4.0 as the platform
- Microsoft Internet Information Server 7.0 as the web server
- Microsoft SQL Server 7.0 as the database server
- Microsoft Exchange Server 5.5 as the mail server
- Internet Explorer 4.0 (and above) as the browser

As for the development software tools, the Active Server Pages (ASP) 3.0 and SQL will continue to be used. The reasons for using the existing software are as below:

3.6.1 Reasons to continue using ASP

- It is able to customise web pages according to specific needs of each individual user
- It takes less time to write and debug (no compilation), thus less “down” time for web sites
- It outperforms (by 5-to-1 ratio) other conventional web page design methods (CGI, etc.)
- It allows amazing power design in developing database applications

- It is able to support the overall development of this project.

3.6.2 Reasons to continue using SQL

- It delivers a powerful platform that scales to terabyte-size databases
- Functions available to perform stored procedures and queries
- Seamless integration with Windows NT that provides security, a web application environment, and Microsoft Transaction Server support
- Integration with Microsoft Exchange Server provides reliable and scalable Internet and Intranet collaboration and messaging, supporting SQL server initiated trigger and stored procedure-based messaging and replication of Exchange public folders
- The database can easily be mapped to other processing server through Open Database Connectivity (ODBC 32bit)
- It is a complementary language for ASP

3.7 Conclusion

With all the stated functional and non-functional requirements, development model, and proposed tools, all these would be used in the system design which is presented in the next chapter.

Chapter 4: SYSTEM DESIGN

This phase is concerned with how the system functionality is to be provided by the different components of the system.

4.1 System Functionality Design

4.1.1 LMS Three-tier Client/Server Architecture

4.1.1.1 Client Tier / First Tier

The client tier is made up of computers with Internet Explorer (4.0 and above). It will provide the interface for clients to process their application and maintain their data.

4.1.1.2 Middle Tier / Second Tier

The middle tier consists of the web server, Internet Information Server (IIS) 4.0, and the mail server, Microsoft Exchange Server (Exchange) 5.5. All application programs or files will be resided at the middle tier. The web server processes requests from the clients, and then returns the required results in HTML format. The mail server will generate mails to the users. This tier will always be linked to the database server to assist in authenticating and validating users who log into the system. Most of the processing role would be resided in the web server.

4.1.1.3 Third Tier

It consists of the database server, Microsoft SQL Server 7.0, which maintains all data records. Every query requested from the web server will be authenticated first from the database server, and the results will then be passed back to the web server, before passing back to the clients.

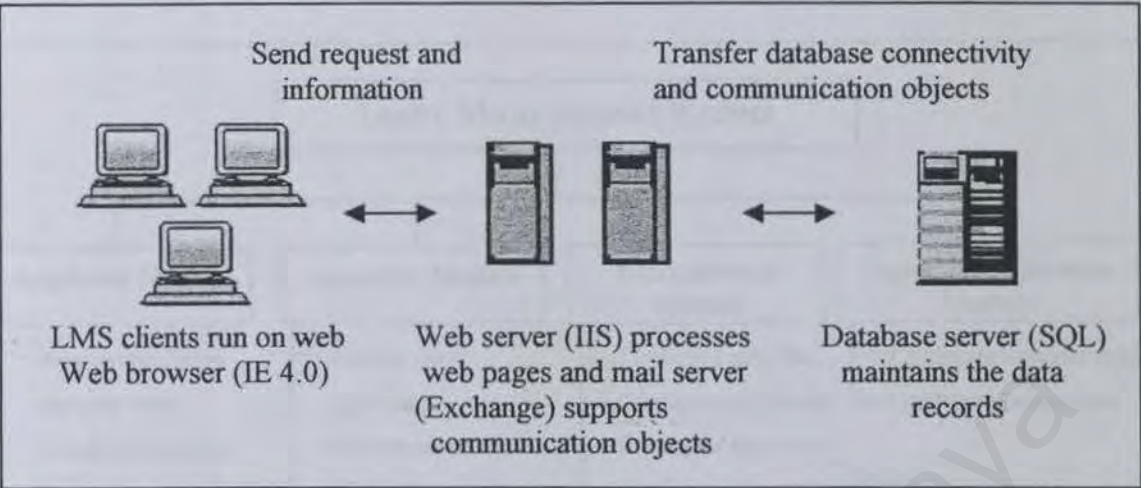


Figure 4.1.1.3 LMS Three-Tier Client/Server Architecture

4.1.2 System Structure Chart

The system structure chart for the full integration with AMS can be represented as below:

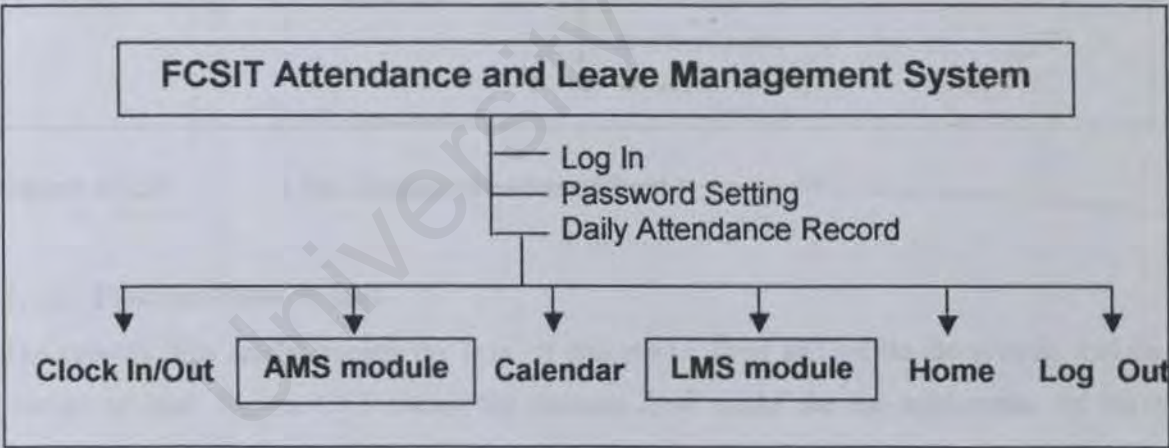


Figure 4.1.2a System Structure Chart for LMS and AMS Integration of Interface

The following system structure chart shows the hierarchical representation between the modules and sub modules in LMS.

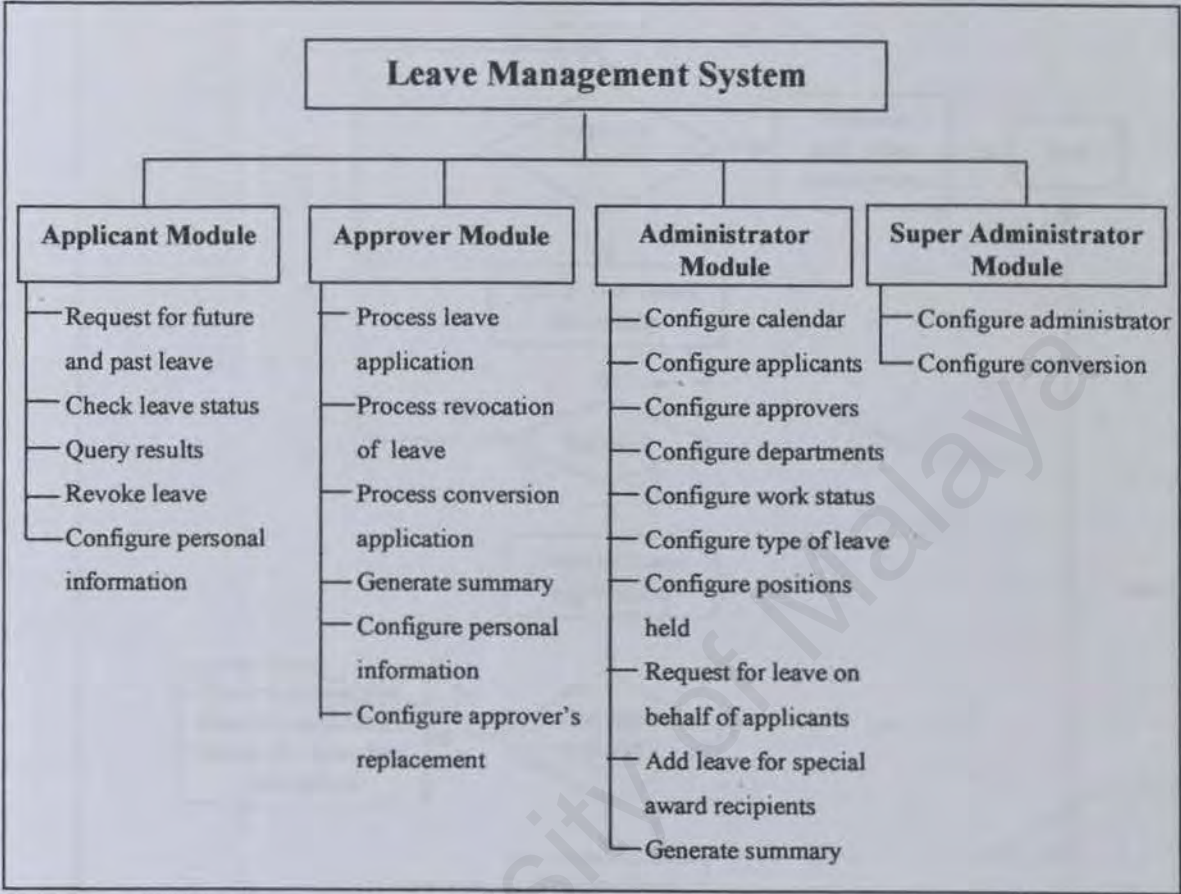


Figure 4.1.2b LMS System Structure Chart

4.1.3 Process Flow Model

The process flow model depicts the flow of process to, from and within the system, and the storage of data. Figure 4.1.3 shows the process flow model for the application for leave module.

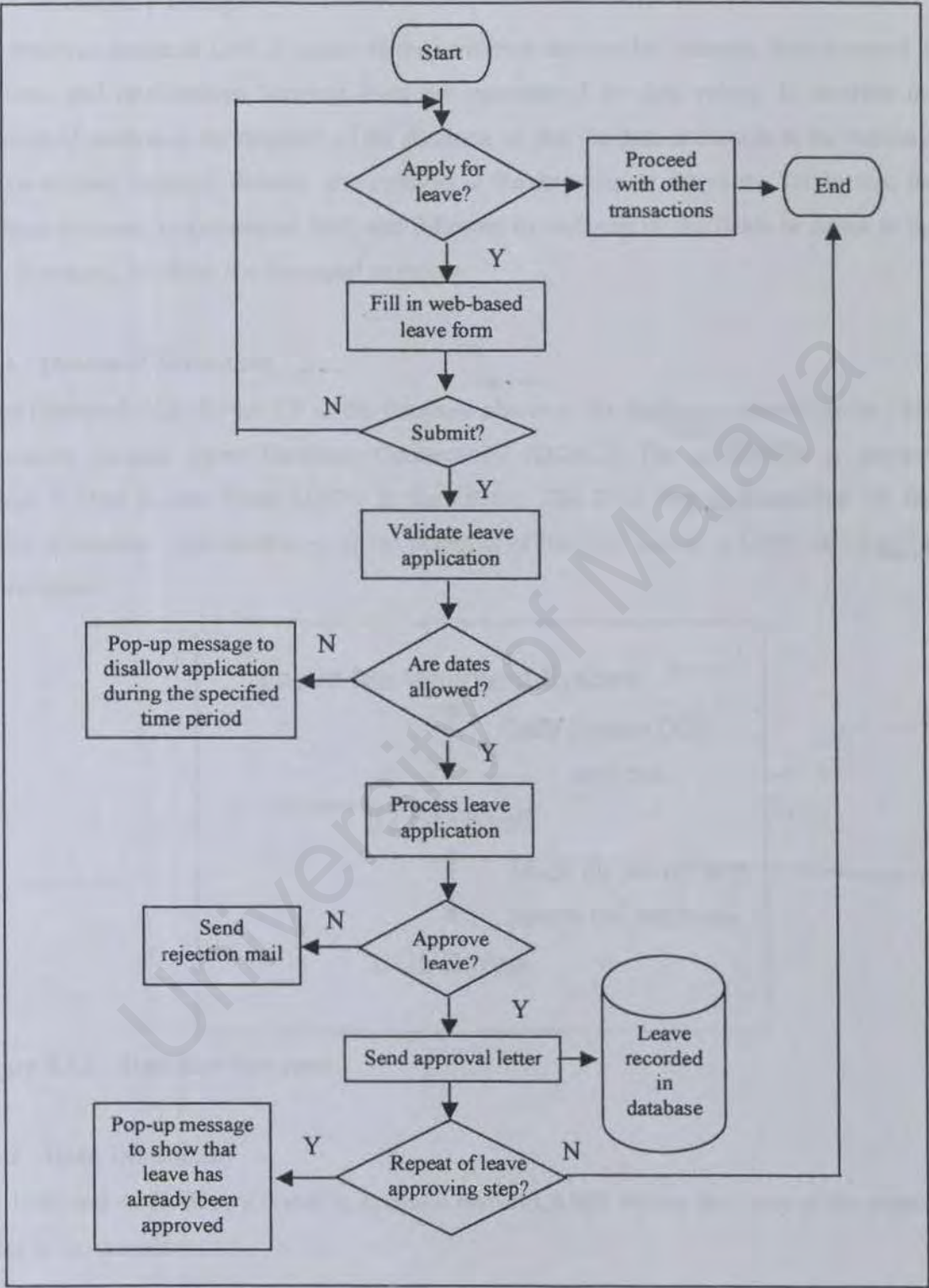


Figure 4.1.3 Process Flow Model for leave application process

4.2 Database Design

The database design in LMS is based on the relational data model, whereby data is stored in relations, and relationships between rows are represented by data values. It involves the activities of modelling the structure of the database so that the data or records in the database can be created, updated, deleted, or displayed at the direction of the users. To do this, the database structure is determined first, and followed by defining all the fields in tables in the data dictionary. All these are discussed as below.

4.2.1 Database Structure

Using Microsoft SQL Server 7.0 as the database platform, the database connects to the LMS application through Open Database Connectivity (ODBC). The connection is mapped through a Data Source Name (DSN) in the ODBC. The DSN that is designated for this project is **amsdev**. An illustration of the mapping of the SQL server to LMS via ODBC is shown below:

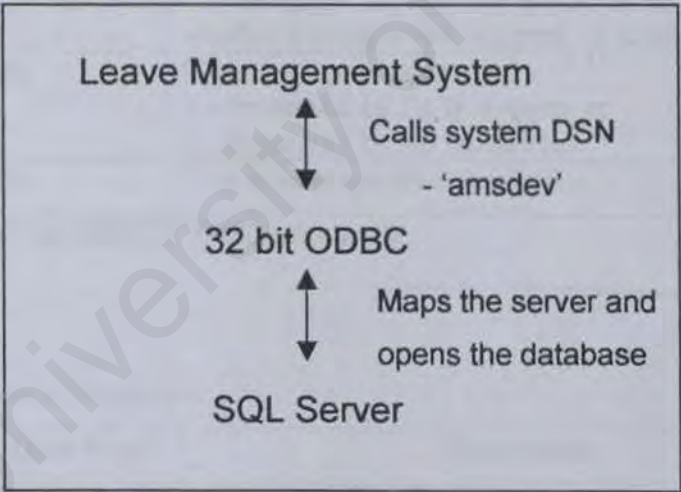


Figure 4.2.1 Database Structure

4.2.2 Data Dictionary

The LMS and AMS share a common database named **LAMS**. Below are some of the existing tables in the database:

Field Name	Data Type	Description
ApplicantID	Int	Key of the table
DepartmentCode	Varchar(10)	Code of department; to determine who is the approver
AccessRight	Varchar(10)	Access right can be user or administrator
BalanceAdded	Int	Balance added at the end of the year because of promotion or seniority, and it will be transferred to become permanent leave balance
PermanentAddOn	Int	Annual leave balance added permanently to the total leave balance because of promotion or seniority

Table 4.2.2a Some of the fields in *ApplicantInfo* table

Field Name	Data Type	Description
ApplicantID	Int	Key used to link to the ApplicantInfo table
ApplicationID	Int	Key of the table
Status	Varchar(20)	To determine the current status of the application, whether it is approved, rejected, or in progress
Processed	Int	0 – Not yet processed 1 – Processed by the first approver 2 – Processed by the second/last
Revoking	Bit	True if leave has been revoked

Table 4.2.2b Some of the fields in *Application* table

The following new table is introduced:

Field Name	Data Type	Description
LoginName	Nvarchar(30)	Approver's log in name
ApplicantID	Int	Act as a reference to the <i>ApplicantInfo</i> table
PositionCode	Nvarchar(20)	Approver's position code that links to the <i>PositionCode</i> table
DepartmentCode	Nvarchar(10)	Approver's department code
Email	Nvarchar(40)	Approver's email address
FullName	Nvarchar(50)	Approver's full name
Password	Nvarchar(20)	Approver's password

DepartmentName	Nvarchar(50)	Name of the department
ApproverName	Nvarchar(50)	Name of the approver, according to the post
Route	Int	0 – no replacement for approver 1 – replacement for first approver 2 – replacement for second approver
Status	Int	1 – first approver 2 – second approver
RloginName	Nvarchar(50)	Replacement's log in
Rstatus	Int	Replacement's status 1 – replacement for first approver 2 – replacement for second approver
Remail	Nvarchar(40)	Replacement's email
RfullName	Nvarchar(50)	Replacement's full name
Rpassword	Nvarchar(20)	Replacement's password
ID	Int	Approver's ID

Table 4.2.2c Fields in Approver table

Note: The purpose of Approver ID is to keep track on the number of approvers, and it also acts as a control to differentiate between the approvers.

4.3 Graphical User Interface Design

The interface of a system works as a central communication between the processing functions and the user requests. The objective of an interface is to enable the user to grab information that they need or to act as a medium for them to supply more information to the system. The interface is aimed to improve efficiency and effectiveness of the user when using the entire system. In addition, a good interface shall not cause the user to remember a large number of commands or codes, but should be as user-friendly as possible.

In LMS, there are four kinds of interfaces for four different types of users. The functions in navigation bars for each interface are stated in Lee (1999)^{pg 47}. The enhancements and development of new functionalities would affect certain pages:

Applicants

- **Application of future and past leave**

Applicants have two buttons to choose before they apply for leave: ‘Future Leave’ to apply for leave after the date of application, and ‘Past Leave’ whereby applicants are able to apply for previous leave that has yet to be applied.

The interface should be informational and yet not confusing. Applicants should be able to differentiate the two types of leave application when clicking on either of the function buttons. The suggested interface should look the screen below:

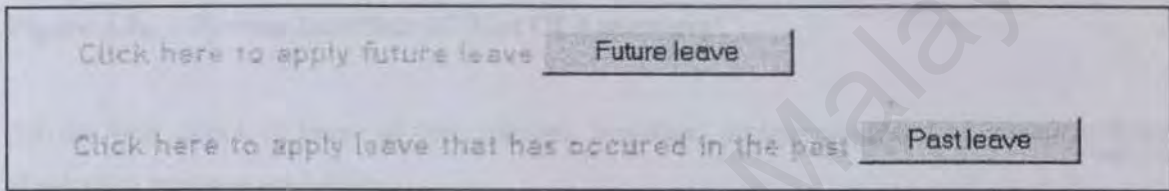


Figure 4.3a **Function Buttons for Applying Future and Past Leave**

Administrator

- **List of Choices**

An additional button, ‘List of Approvers’ would be included to add clarity and functionality. (Previously, approvers’ information was put together in the ‘List of Departments’ section).

The suggested button would look like this:



Figure 4.3b **Button for ‘List Of Approvers’**

Under ‘List Of Approvers’, the design of the interface would be similar to ‘List Of Departments’. A portion of the interface is shown below:

Approver Information Configuration

Approver's Code	Approver's Name	Approver's Status
<input type="checkbox"/> KJKP	PM Ow Siew Hock	1

Create New Approver

Delete Approver

Clear Selection

Figure 4.3c Portion Interface of ‘List Of Approvers’

All the web pages in terms of font, design, language, arrangement and alignment will be checked to improve readability.

4.4 Security Design

Every user will be verified through his/her log in name and password. A few session variables will be created to determine the access right of the user. They are:

Session Variable	Description
Session(“Authenticate”)	True if the user is authorized
Session(“ApplyLeave”)	True if the user is allowed to apply for leave
Session(“AccessRight”)	To determine the access right of the user, whether he/she is the first or second approver, administrator, or the super administrator

Table 4.4 Description of Session Variables

These sessions will be used to verify the access right of the user who logs into the system. This will prevent unauthorized users to break into it by just typing the URL address.

To ensure that every user has a unique log in name, information regarding new users that will be added into the database will first be checked for any duplication. If there is any duplication, then the user will be prompted with an error message.

Passwords that are sent through the network will be encrypted to ensure security and confidentiality. Password information in *ApplicantInfo*, *Approver*, and *SuperAdmin* tables are encrypted to avoid possible exposure of applicants' passwords to malicious users.

4.5 Database Backup

With the job scheduling function provided in Microsoft SQL 7.0, any job can be scheduled to execute at any time.

- Automatically launch the daily backup

The database will be backed up at 12:00 am everyday to the web server. The T-SQL coding is as below:

```
USE LAMS
GO
BACKUP DATABASE LAMS
TO DISK = '\\LMS\lms\lmsbackup.bak'
WITH FORMAT,
NAME = 'Full Backup of LMS Database.'
GO
```

Execution time can be scheduled by adding a new job in the SQL Server Agent Service.

- Restore easily from any potential disaster

Upon any unwanted happenings/disaster, the system will automatically restore the database. The T-SQL code that will be used is:

```
RESTORE DATABASE LAMS
FROM DISK = '\\LMS\lms\lmsbackup.bak'
GO
```

- Automatic deletion of past leave application records

The records that have been stored for more than three years will be deleted, after they have been backed up. Data transformation service will be used to copy the related records to another database named 'BACKUP'. Then with T-SQL, the records will then be deleted from the database.

4.6 Users and Human Factors

- Appropriate error messages should be prompted for the errors made.

Each error will be identified and different error message will be prompted for different kinds of error. For example, if the user tries to apply leave on a non-working date, an error message will be shown.

- Data entered should not be truncated

The maximum length of the fields for user to key in will be limited according to the settings in the database.

- Date format should be specified properly to avoid confusion

The use of Microsoft Active-X Calendar Control helps to avoid this kind of confusion. The date format that appears in the control would follow the configuration in the regional setting on each computer. Thereafter, the date format in the generated summary will be displayed in 'dd/mm/yyyy' format.

Chapter 5: SYSTEM IMPLEMENTATION AND DEVELOPMENT

5.1 System Implementation

During this phase, the design model of LMS is transformed into reality; from the start of determining the hardware and software to be used, to enhancing and adding new functionalities into the system. The proposed development software tools would first be installed into the machine. Next, coding of the programme begins, and simultaneously testing and debugging is carried out.

5.1.1 Implementation Environment

5.1.1.1 Windows NT Server

During the installation of Windows NT Server, the hard disk was formatted using NT File System (NTFS) format to ensure a more stable and secured NT transaction across the platform.

For the multi-tier architecture in LMS, three servers were needed; a web server (Microsoft Internet Information Server), a mail server (Microsoft Exchange Server 5.5) and a database server (Microsoft SQL Server 7.0). All these were installed in the Windows NT Server environment to develop LMS. The web server would be the Primary Domain Controller (PDC) and the database server as Backup Domain Controller (BDC). BDC can be set up during the installation of Windows NT or by using Windows NT Server Manager in the PDC.

The following steps were involved in installing the development tools:

1. Install Windows NT 4.0
2. Install NT Option Pack 4 which includes NT Service Pack 3, Internet Information Server 4.0 and Microsoft Transaction Server.
3. Install NT Service Pack 5
4. Install Microsoft SQL Server 7.0
5. Install Microsoft Exchange Server 5.5

5.1.1.2 Internet Information Server (IIS)

Creation of virtual directory

To enable users to access the system through the Internet, a virtual directory has to be created in the web server. The virtual directory corresponds to the actual directory where all system files are found. The default document is **frontframe.html**. In the IIS security setting, this directory should be granted “allow anonymous access”. All users can access the application through **http://lms/**. Here is a step-by-step process of creation of a virtual directory.

Step 1: Under the Internet Service Manager, right click on ‘Default Web Site’. Under ‘New’, choose ‘Virtual Directory’.

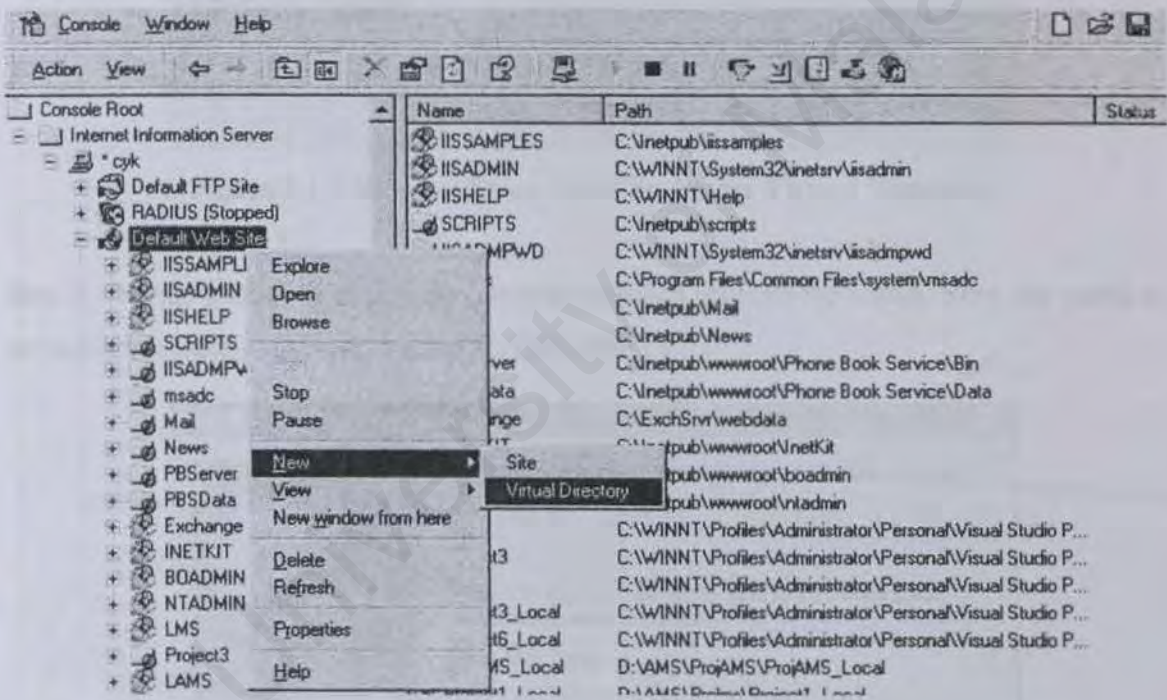


Figure 5.1.1.2a Creation of Virtual Directory

Step 2: Provide a name to access the virtual directory. In this example, the name “LAMS” is chosen.

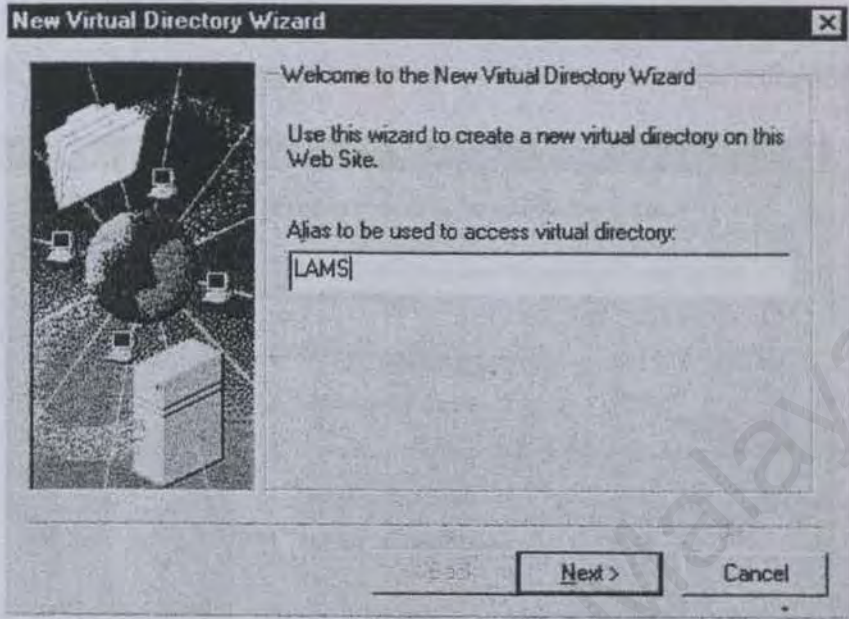


Figure 5.1.1.2b Enter Alias To Access Virtual Directory

Step 3: Provide a name to access the physical path that contains the codes. Here, the codes to publish are located in D drive, named “LAMS2000”.

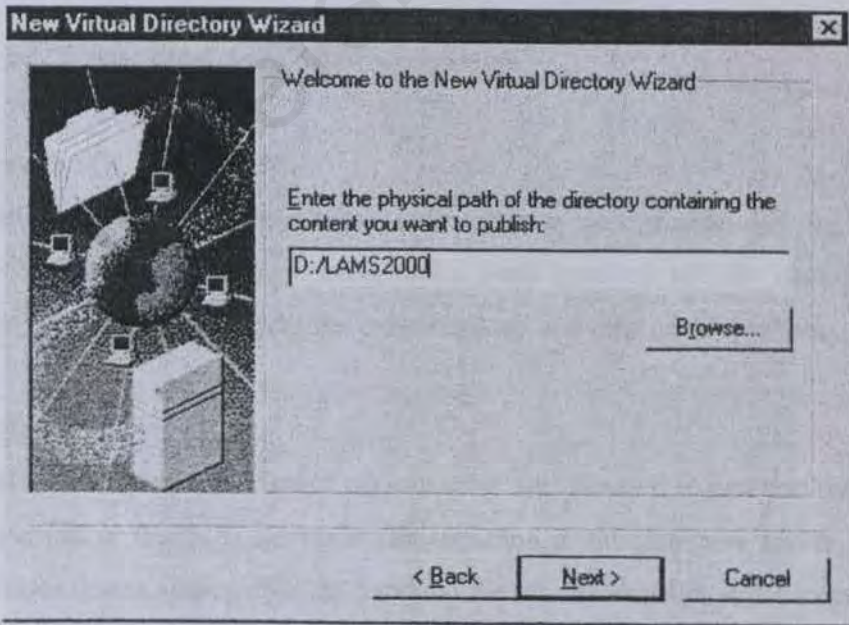


Figure 5.1.1.2c Enter Physical Path of Directory

Step 4: Specify the access permissions for this virtual directory. In order to maintain security, just leave the default settings, whereby 'Allow Read Access' and 'Allow Script Access' are clicked. If this virtual directory is accessed only by the developer of the system or by a trusted source, then click on the rest of the options, depending on which is applicable.

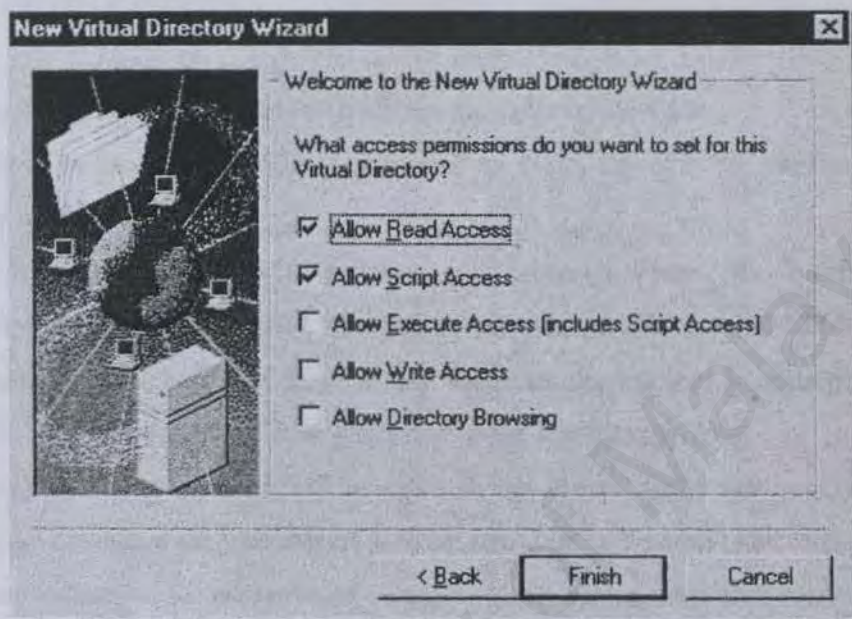


Figure 5.1.1.2d Set Access Permissions

Step 5: Click on 'Finish' once every detail is confirmed.

5.1.1.3 Microsoft SQL Server 7.0

After the database server was installed, a new database was created. For this project, the shared database is named **LAMS**. Existing tables were imported into the new database, and new tables are created to complement the enhancements and new developments.

SQL Server Agent

Besides creating the tables, SQL Server cursors were also inserted to perform scheduled jobs. These jobs are run in the SQL Server Agent, residing in the database server. It is like an auxiliary operator that is responsible for handling the repetitive tasks and exception handling conditions. The server agent is responsible for:

- Running SQL Server tasks that are scheduled to occur at specific times or intervals.
- Detecting specific conditions for which administrators have defined an action, such as alerting someone through pages or e-mail, or a task that will address the conditions.

In LMS, the system administrator will be responsible and will be informed if any of the scheduled jobs fail to perform.

A few scheduled jobs were created to maintain the system. They are:

- Database backup – the database is backed up every day at 12:00 am from the database server to the web server.
- Daily checking – if any new leave application is found, the particular head of department would be informed. This job is scheduled at 2:00 pm and 12:00 every day.
- Unprocessed checking – if there is any leave application that is unprocessed for more than 3 days, an e-mail would be generated to the related approver.
- November mail – on every 1st November, the server agent will send mails to every applicant to remind them to submit their desired days to be converted to cash.
- January update – on every 1st of January, leave applications that have occurred three years ago will be automatically archived. All related records will be copied to another database named 'Backup'. After the backup process is completed, all related records will be deleted from the existing database. Besides, all balance added based on promotion or seniority on the previous year will be converted to become permanent add-ons to their annual balance. Additionally, contract staffs' leave balances will be adjusted according to their appointed date. Records of staff who have resigned will be deleted also.
- December reminder message – if there is any unprocessed application of the total days to be converted to cash, the server agent will generate a reminder mail to the related approver. This job is scheduled to be executed every day at 12:00 pm, starting from 1st December.

An example of creating a job in SQL Server Agent is shown in the next page:

Step 1: Under the respective SQL Server Registration (provided by default by the system once the database server is installed), click on the folder named 'Management'. Make sure the SQL Server Agent is running. If not, right click on it and choose 'Start'.

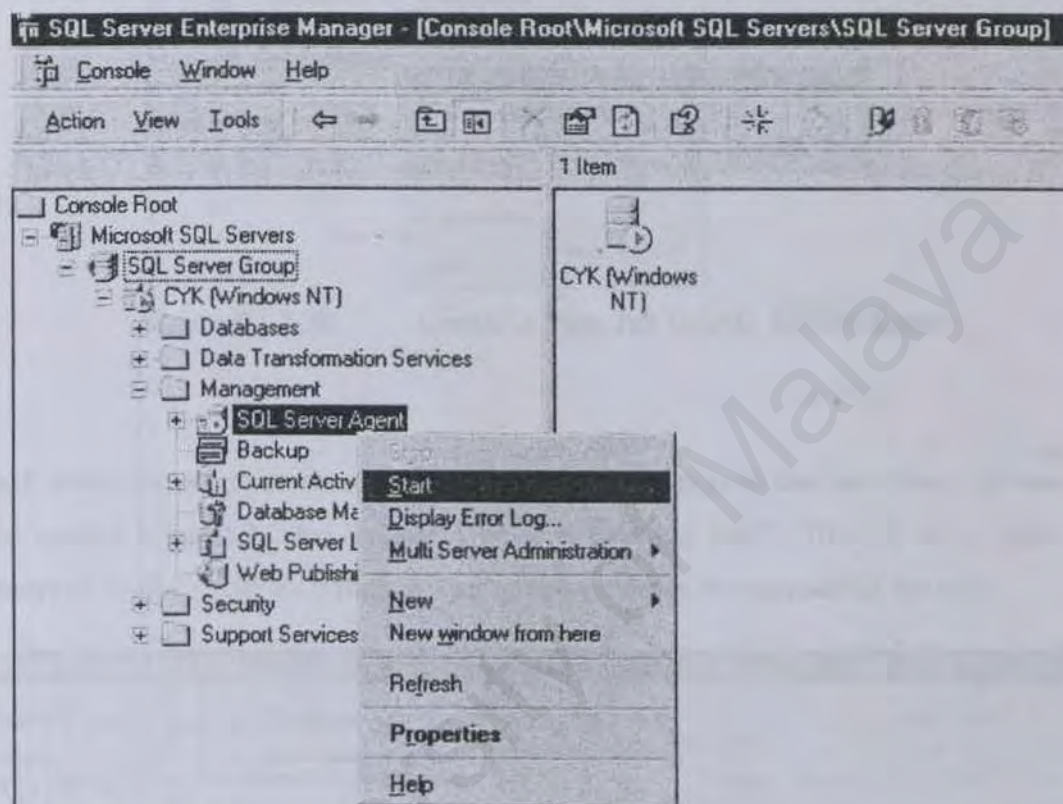


Figure 5.1.1.3a Start SQL Server Agent

Step 2: Right click on the server agent again. Click on 'New' and choose 'Job'. This is to create a new job to be performed by the server agent.

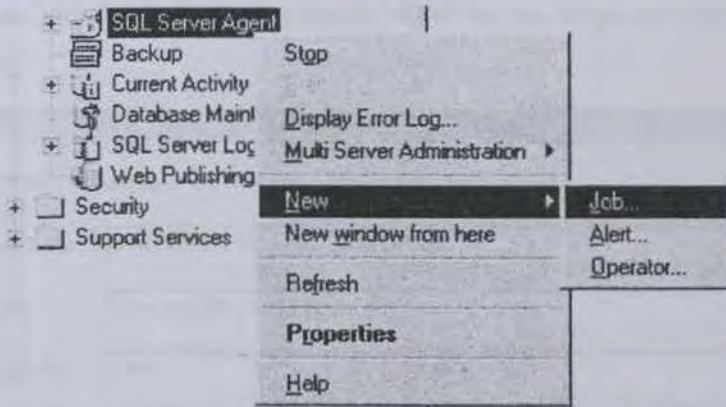


Figure 5.1.1.3b Create A New Job In SQL Server Agent

Step 3: Name the job to be created. Leave the default settings as they are. Here, the new job to be created is named “Dec Balance Update – Contract Staff”. This job is to adjust the balances of contract staff according to their appointed dates, at every end of the year.

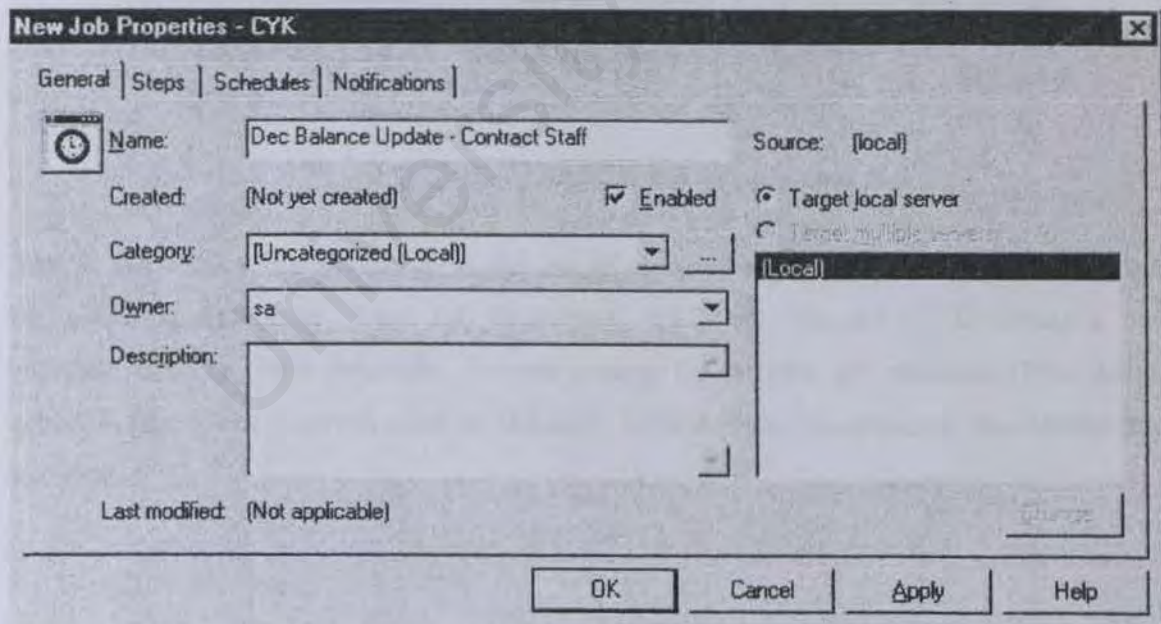


Figure 5.1.1.3c Enter General Information for Job

Step 4: Click on ‘Steps’, and then click on ‘New’. Provide the step name for the new job. Next, type in the steps to be performed by the server agent in the ‘Command’ space. The steps are written ideally using SQL cursors. After all the steps are typed in, click ‘Apply’ followed by ‘OK’.

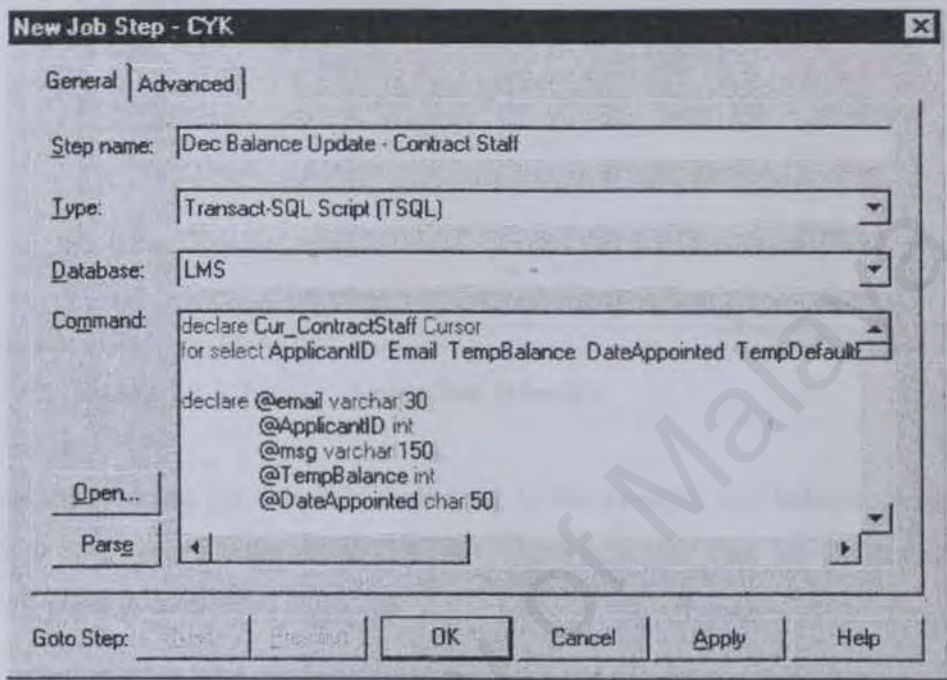
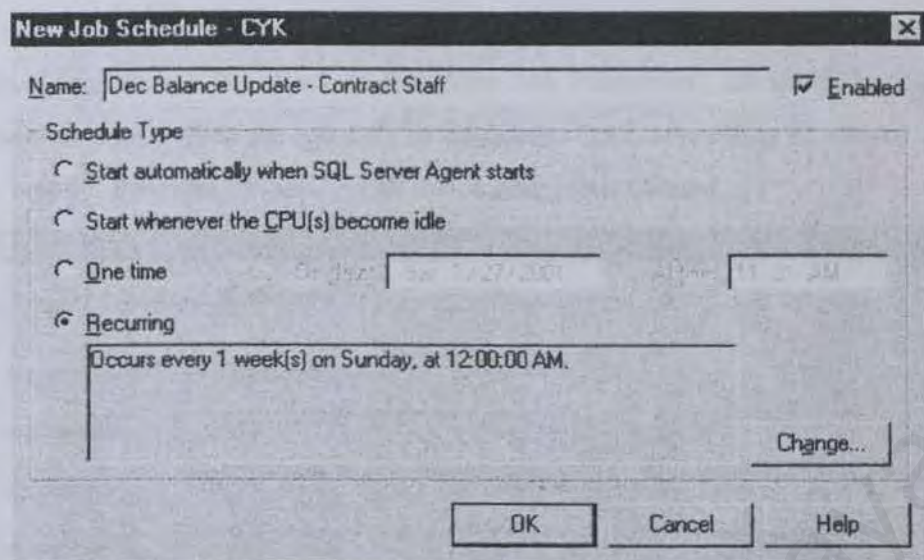


Figure 5.1.1.3 d Enter Job Steps

Step 5: Since this job is performed at the end of every year, a schedule must be entered into the server agent. Under ‘New Job Properties’, click on ‘Schedules’. To create a new schedule, click on ‘New Schedule’. Provide a name for the new job schedule. If the default schedule type is not accurate, click on ‘Change’ to change the recurrence to the desired time and date.



New Job Schedule - CYK

Name: ☒ Enabled

Schedule Type

☐ Start automatically when SQL Server Agent starts

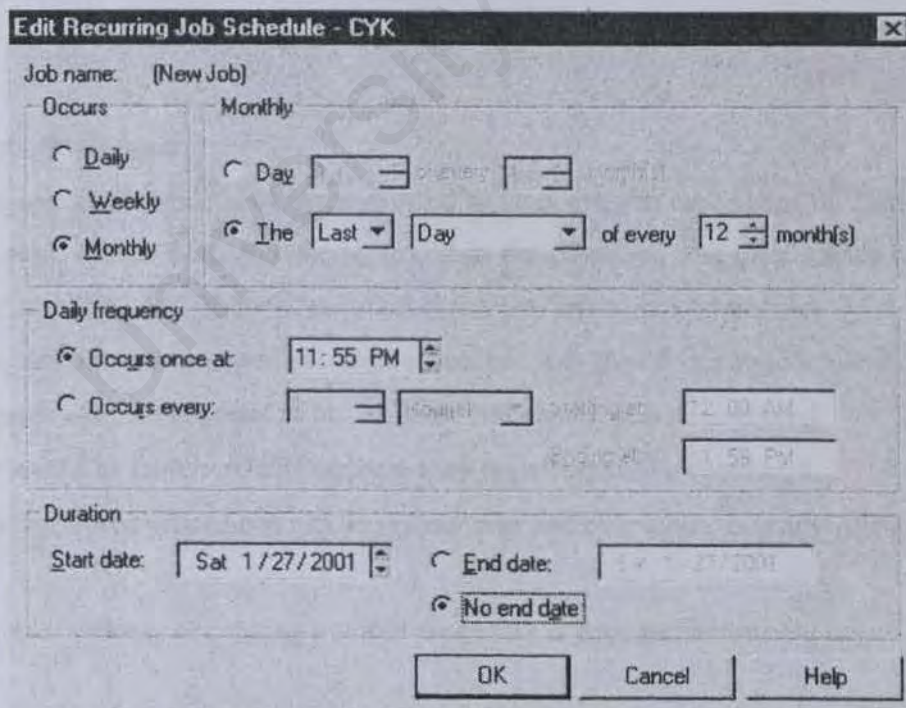
☐ Start whenever the CPU(s) become idle

☐ One time

☒ Recurring

Figure 5.1.1.3e Enter Job Schedule

Step 6: Edit the recurring job schedule as desired. In this example, this balance update occurs at the last day on every December at 11.55pm. There is no end date for this particular job. When every detail is confirmed, click 'OK'.



Edit Recurring Job Schedule - CYK

Job name:

Occurs: Monthly

☐ Daily

☐ Weekly

☒ Monthly

☐ Day month(s)

☒ The Day month(s)

Daily frequency

☒ Occurs once at:

☐ Occurs every:

Duration

Start date:

☐ End date:

☒ No end date

Figure 5.1.1.3f Edit Recurring Job Schedule

Step 7: In the ‘New Job Schedule’ dialogue box again, check to make sure that the recurrence is correct. Then, click ‘OK’. Back to the ‘New Job Properties’, click on ‘Notifications’ to specify who to notify when the job fails or succeeds. Once everything is entered as desired, click on ‘Apply’, followed by ‘OK’. With this, a new job is created.

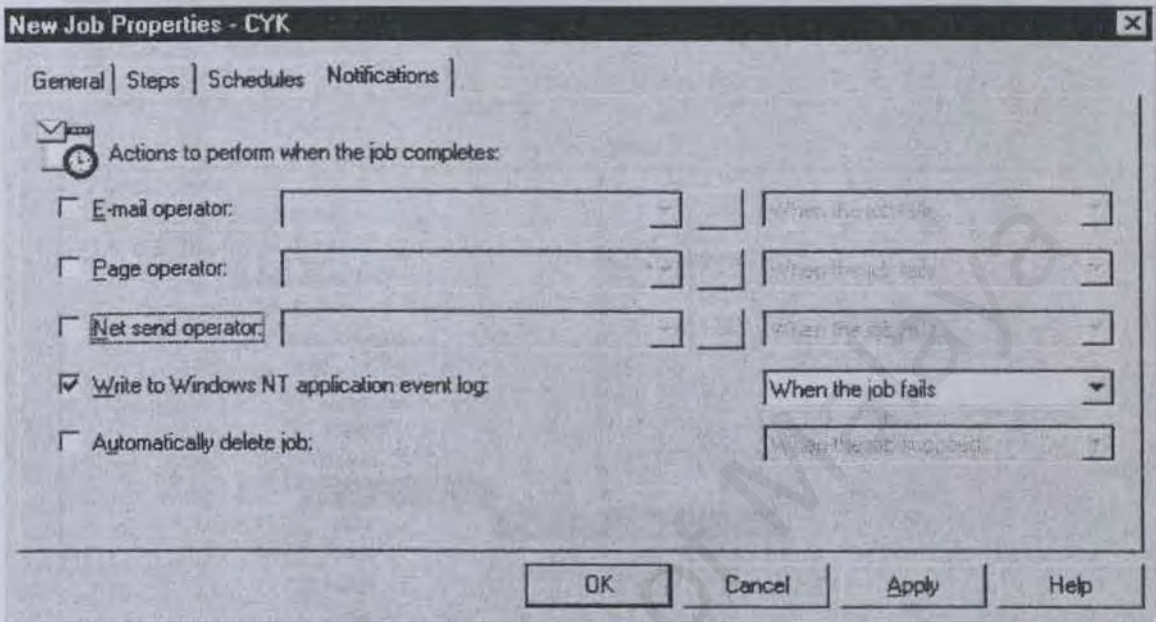


Figure 5.1.1.3g Enter Job Notifications

SQL Stored Procedures

A number of stored procedures were created to optimise the time spent on retrieving data. They are ready-to-run SQL statements stored in the database, and their names can be used repeatedly in ASP codes. There a few good reasons to use stored procedures:

- It is quicker to run a stored procedure because it is stored in a precompiled form on the database. They do not need to be compiled before they can be executed.
- Code would be more readable because they are effectively a shorthand.
- The same stored procedures can be reused over and over again, in many ASP pages.

A step-by-step process of creating a stored procedure is depicted in the next page:

Step 1: In the SQL Server Registration, click on 'Databases', followed by 'LMS' (depending on which database that is being used), and lastly 'Stored Procedures'. Right click on 'Stored Procedures' and choose 'New Stored Procedure'.

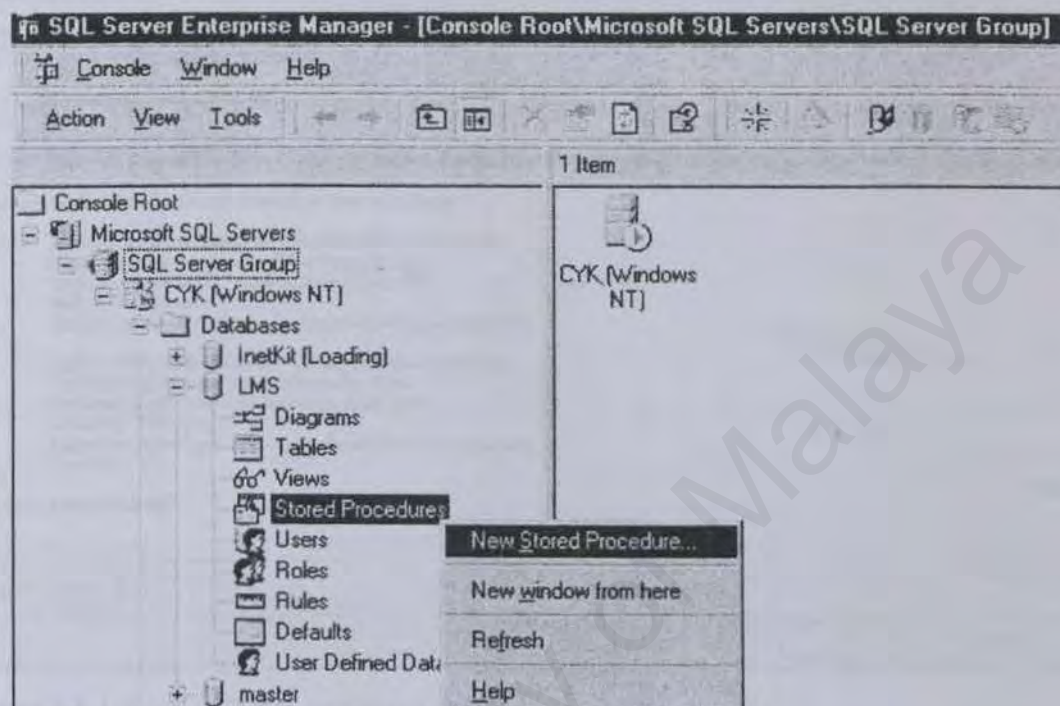


Figure 5.1.1.3h Create New Stored Procedure

Step 2: Using SQL stored procedures, type in the desired codes to be performed. Click 'Check Syntax' once all the codes are typed in. If the codes' syntax is correct, click on 'OK'. A new stored procedure is created.

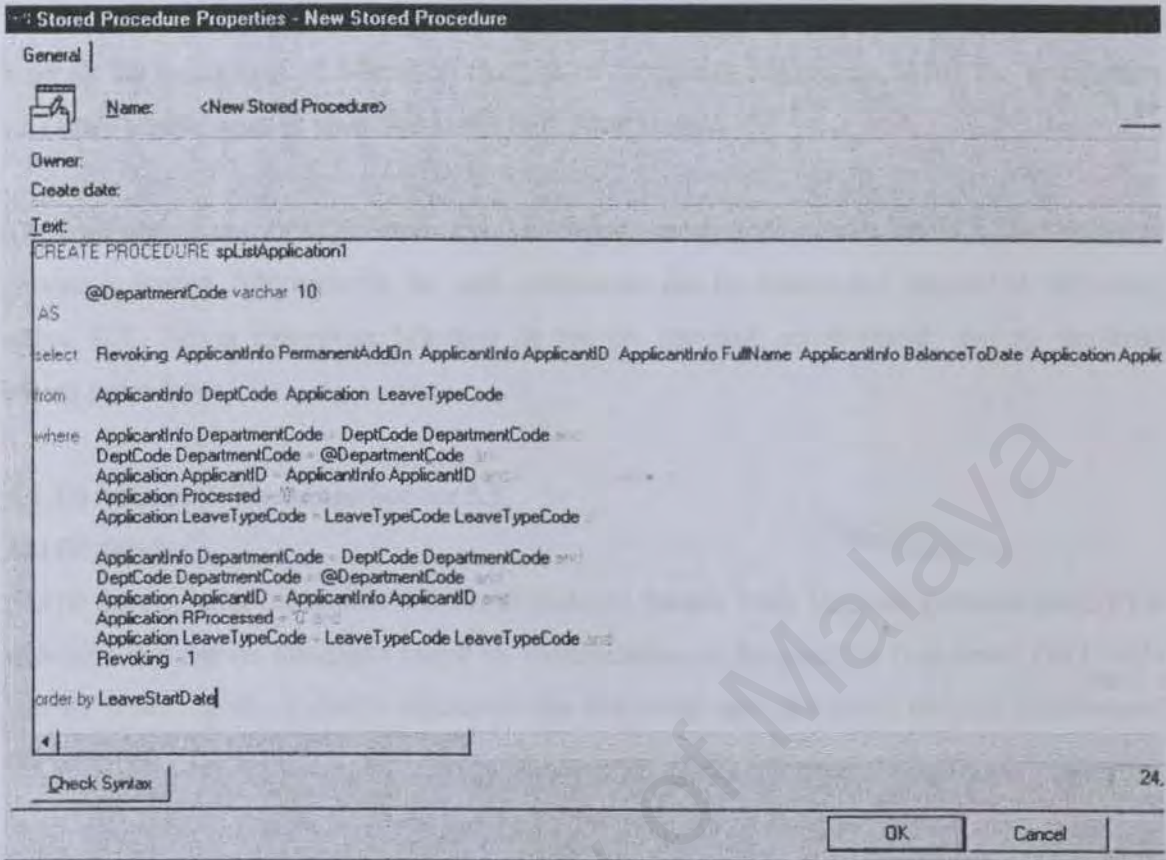


Figure 5.1.1.3i Type in Stored Procedure

SQL Mail

Microsoft SQL Server 7.0 provides a set of extended stored procedures that allow SQL Server to operate as a workgroup post office for MAPI-enabled e-mail system.

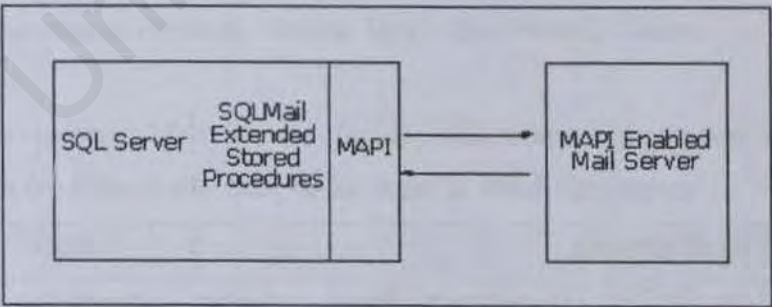


Figure 5.1.1.3j Architecture between SQL Mail and Mail Server

The computer running SQL Server must be set up as an e-mail client. This can be done through the installation of Microsoft Outlook or Microsoft Messaging. After the installation, an e-mail profile must be specified in the SQL Mail service.

The mail component of SQL Server can be started automatically when the SQL Server Agent service is started. Alternatively, the mail component can be started and stopped at will, using either SQL Server Enterprise Manager or the `xp_startmail`, `xp_stopmail`, and `xp_sendmail` stored procedures.

5.1.1.4 Microsoft Exchange Server 5.5

SMTP Service

SMTP Service uses the standard Internet protocol Simple Mail Transfer Protocol (SMTP) to transport and deliver messages based on specifications in Request for Comments (RFC) 821 and RFC 822. It also includes enhancements that build upon the basic delivery functions of the protocol.

In LMS, this service is mainly used to deliver mails to the applicants to inform them about the decisions made by the heads of department to their applications.

Installation of Internet Mail Connector

Internet Mail Service routes messages between Microsoft Exchange Server and SMTP-based systems, including Internet mail servers and Post Office Protocol version 3 (POP 3) and Internet Message Access Protocol, Version 4rev1 (IMAP4rev1) clients.

To set up the Internet Mail Service for the minimum configuration requirements, the following list of procedures was used, in the order in which they appear.

Task	Description
Installing and configuring Transport Control Protocol/Internet Protocol (TCP/IP)	Install Windows NT TCP/IP, and specify the host and domain name. If DNS will be used, enter the IP address of the DNS servers.
Adding the Internet Mail Service computer to DNS	If using DNS, add the host name, domain name and IP address of the computer that serves the Internet Mail Service to the DNS.

Updating the Hosts file when not using DNS.	If not using DNS, use the Hosts file to specify the hosts name and IP address of the hosts to which the Internet Mail Service will forward mail.
Verifying the site address	From the Configuration container in the Administrator window, select Site Addressing
Running the Internet Mail wizard to install the Internet Mail Service.	From the Connections container in the Administrator program, choose File, New Other, and Internet Mail Service.
Testing the connection	Test the connection to ensure that the Internet Mail Service is configured correctly.

Table 5.1.1.4 Installation Procedures for Internet Mail Service

Windows Messaging

After the installation of Internet Mail Service, with Windows Messaging e-mail can then be sent and received through the mail server. To start Windows Messaging, double-click the Inbox icon on window’s desktop. If the Inbox icon is not on the desktop, then Windows Messaging is not installed. It can be installed manually through the Add/Remove Programs in Control Panel.

MAPI profile is a set of configuration options used by Microsoft Exchange and other MAPI compliance messaging applications that contains essential information. This information includes the location of Inbox, Outbox, and address lists, and the personal folder files available to the account for storing and retrieving messages and files. It is needed for SQL Mail Service to send and receive e-mail. Every mail sent or received by SQL mail will go through the MAPI profile created with Windows Messaging.

In LMS, an Administrator account has been created in Exchange Mail Server. During the setting up of Windows Messaging, the user profile name would be ‘Administrator’. The default port number is used for POP3 and SMTP protocols.

5.2 System Development

5.2.1 Coding

For the *enhancement* of the application, the development process started with understanding the existing codes first, by which it proved to be a highly demanding task. This is because each and every programmer has their own style of writing, and for LMS, there is no exception. The iterative spiral started with scrutinising the codes for all modules to ensure the whole system was well understood. After that, all the critical functions were corrected first. This was subject to users' requirements, from which users' inputs were concurrently being taken into account to achieve the desired end results.

As an example, for **approving leave**, there was an inefficient way of querying from the database. A set of long codes was written to select the desired fields from the tables in the database. To solve this, stored procedures were used to minimise the time taken to retrieve information from the database, and also to optimise the codes. The old codes are shown below:

```
if AcRight = "Approver1" or AcRight = "Approver" then
StrSql1 = "select
Revoking,ApplicantInfo.PermanentAddOn,ApplicantInfo.ApplicantID,Application.ApplicationID,
LeaveTypeCode.LeaveTypeCode,ApplicantInfo.FullName, LeaveTypeCode.LeaveTypeName,
ApplicantInfo.BalanceToDate, convert(char(10),Application.LeaveStartDate,103) LeaveStartDate,
convert(char(10),Application.LeaveEndDate,103) LeaveEndDate, Application.NoDaysTaken,
Application.ReasonForLeave from ApplicantInfo , DepartmentCode , Application , LeaveTypeCode where "&_
"ApplicantInfo.DepartmentCode = DepartmentCode.DepartmentCode and "&_
"DepartmentCode.DepartmentCode = " & DepartmentCode & " and "&_
"Application.ApplicantID = ApplicantInfo.ApplicantID and "&_
"Application.Processed = '0' " &_
"and Application.LeaveTypeCode = LeaveTypeCode.LeaveTypeCode " &_
" or ApplicantInfo.DepartmentCode = DepartmentCode.DepartmentCode and "&_
"DepartmentCode.DepartmentCode = " & DepartmentCode & " and "&_
"Application.ApplicantID = ApplicantInfo.ApplicantID and "&_
"Application.RProcessed = '0' " &_
"and Application.LeaveTypeCode = LeaveTypeCode.LeaveTypeCode " &_
"and revoking = 1 " &_
" order by LeaveStartDate"
set qcode = db.Execute (StrSql1)
```

With stored procedures, the codes above would be optimised and written in just a few lines, as shown below:

if AcRight = "Approver1" or AcRight = "Approver" then

objComm.CommandText = "spListApplication1" 'codes are stored in 'spListApplication1'

Set objParam = _

objComm.CreateParameter("@DepartmentCode", adVarChar, adParamInput, 10)

objComm.Parameters.Append objParam

objComm.Parameters("@DepartmentCode") = strDepartmentCode

In approving for leave also, the system needs to differentiate between first approvers, second approvers, and the only approvers. This is a complicated task whereby when users log in, the system must capture their status before approving any leave application. Hence, array was used to store approvers' status and department codes. A portion of the codes is shown below.

if Balance >= 0 then

rs.fields("status") = "Approved"

rs.fields("ActionBy") = session("AName")

rs.fields("actionDate") = Date()

ArrayFA = session("DepartmentCode1") 'array for first approver

ArraySA = session("DepartmentCode2") 'array for second approver

ArrayOA = session("DepartmentCode0") 'array for only approver

Dim i, status

if session("Fcount") <> 0 then

for i = LBound(ArrayFA) to (session("Fcount")-1) step 1

'session("DepartmentCode1") = ArrayFA(i)

if ArrayFA(i) = rs1.fields("DepartmentCode") then

status = 1

exit for

end if

next

end if

if session("count") <> 0 then

for i = LBound(ArraySA) to (session("count")-1) step 1

if ArraySA(i) = rs1.fields("DepartmentCode") then

status = 2

exit for

end if

next

end if

if session("Ocount") <> 0 then

for i = LBound(ArrayOA) to (session("Ocount")-1) step 1

if ArrayOA(i) = rs1.fields("DepartmentCode") then

status = 2

exit for


```

end if
next
end if

```

For development of *new functionalities*, an initial completion was developed, and this was exposed to users' comments and then refined through testing and debugging. For example, in the integration with AMS, each interface was prototyped and tests were run to see which was preferred. The development for new functionalities were highly iterative also and there was always a requirement to trace back the previous stages if any errors were found. It ends with a complete workable module.

For the additional function of **creating and deleting approvers**, the codes were written to enable the function to work properly, as well as to accommodate the new table, *Approver*, as shown in **Table 4.2.2c**. Whenever the codes iterate each and every approver, three values must be captured, namely approver's department code, full name and status. This is to differentiate between approvers who might approve staff from multiple departments. Hence, an optimal way of capturing more than one variable is to use array. A portion of the codes are shown below:

```

<%
i = 0
rsApprover.movefirst
do while not rsApprover.eof

    myArray (0, i) = rsApprover.fields.item("DepartmentCode")
    myArray (1, i) = rsApprover.fields.item("FullName")
    myArray (2, i) = rsApprover.fields.item("Status")
    i = i + 1
    rsApprover.movenext
    session("ArrDept") = myArray

loop
%>

```

Using array, the three variables are stored in the respective arrays and are used in other parts of the files.

For **detecting non-working dates**, the system needs to recognise if the date of application of leave is valid. The system would bar users from applying if they choose first Saturday, third Saturday, or a public holiday. Below is a sample of the codes to detect non-working dates.

```
<%
    startdate = cdate(request.form("startdate"))
    enddate = cdate(request.form("enddate"))

' This function count the number of days (inclusive of weekends & holidays)
public function countdays
if (len(startdate)<>0) and (len(enddate)<>0) and request.form("Startdate") <> "" then
    countdays = 0
    startd = startdate
    endd = enddate
    Do until (cdate(startd)>cdate(endd))
        numd = Weekday(startd)
        set lsDate = server.CreateObject("ADODB.Recordset")
        StrDate = "SELECT * from NonWorkingDate where NonWorkingDate='" & startd & "'"
        lsDate.open StrDate, Application("goedb"), adOpenKeyset, adLockOptimistic, adCmdText

        if WeekdayName(Weekday(startd), True)="Sat" and (day(startd) < 8 or (day(startd)>=15 and
day(startd)<=21)) then

            FirstnThirdSaturday=true      ' Check if it is the first or third saturday of the week
        else
            FirstnThirdSaturday=false
        end if

if ( ((numd<>1) and (lsDate.eof)) and not FirstSaturday ) then
    countdays = countdays + 1
elseif (not lsDate.eof) or FirstSaturday then
    Session("LeaveApplicationIsValid")="false"
    response.write("<hr>")
    response.write("<br><center><font face='\"'\"'Trebuchet MS'\"'\"' size=2><B>You have chosen a non-
working date. Please choose a different set of dates.</B></center><br>")
    response.write("<b><center><font face='\"'\"'Trebuchet MS'\"'\"' size=2>Press BACK to reconfirm your
application date</center><b>")
    response.end
end if
lsDate.close
startd=DateAdd("d",1,startd)
loop
else
    countdays = 0
end if
end function
```


5.2.2 Interface

For the integration with AMS, a common interface was developed for users of both modules to log in with their log in names and passwords. Two links were developed:

- To enable authorised users to change their passwords without logging into the system first
- To enable all users, be it staff or students of FCSIT to view the attendance of staff.

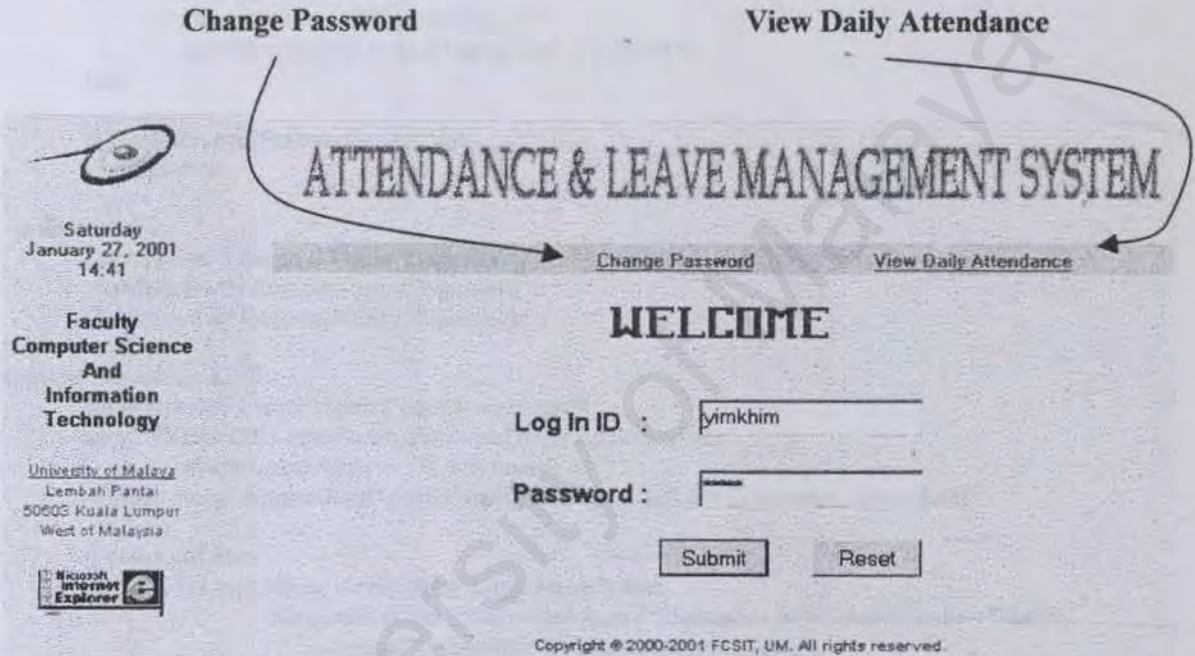


Figure 5.2.2a Interface to Change Password and View Daily Attendance

The changing of password involves four parties altogether. They are users of AMS, namely the staff and AMS's administrator, and the rest of the users of LMS which consist of approvers and administrator. When writing the codes for changing passwords, the passwords of the four parties must be carefully articulated so that the same information does not overlap and the system must be able to update the information before users log in to the system. An example of the codes is shown on the next page.

```

<%
function fnGenEncryptedPassword (Password )
'Function to generate a encrypted password

Dim i,iLen,AscVal,TempPass,TempChar,ReversedStr

HexStr=""
retVal = 0
TempPass = Trim( Password )
ReversedStr = StrReverse(TempPass)
iLen = len(ReversedStr)

for i=1 to iLen step 1
    TempChar = mid(ReversedStr,i,1)
    AscVal = AscVal + Asc(TempChar) + 10041970
next

fnGenEncryptedPassword = AscVal
End function

update = false
LoginName = Request.Form("LoginName")
NewPassword=Request.Form("Password")
OldPassword=Request.Form ("Password0")

'Applicant & Admin LMS
set rs = server.CreateObject ("adodb.recordset")
strsql="SELECT LoginName, Password from ApplicantInfo
        where LoginName = " & loginname & " "
rs.Open strsql, Application("goedb"),adOpenKeyset ,adLockOptimistic , adcmdtext

if not rs.eof then
    if LoginName <> rs.fields("LoginName") then
        Response.Write ("<br><font face=""Trebuchet MS"" size=3 color=""dark
        red""><center><b>Incorrect login name.")
    elseif trim(fnGenEncryptedPassword( OldPassword ))<> trim(rs.fields("Password")) then
        Response.Write ("<br><font face=""Trebuchet MS"" size=3 color=""dark
        red""><center><b>Incorrect old password.")
    else
        temppassword = rs.fields("Password")

        if temppassword <> NewPassword then
            rs.fields("Password") = fnGenEncryptedPassword(NewPassword)
            rs.Update
            update = true
        end if
    end if

    rs.Close
    set rs = nothing

end if
%>

```


For LMS, the four interfaces for users, approvers, administrator and super administrator were remained, except changes were made to the grammar, alignment, arrangement and font face. Additional function buttons were added to complement the new functionalities developed. All these were done to make the interfaces as user-friendly as possible.

For the previous interface, users need to relatively spend more time and effort in order to accomplish a certain task. As an example, users need to scroll down a number of pages in order to see the bottom part of the interface. This can be optimised and adjusted by eliminating extra space and using a smaller font. A comparison of the old and new interfaces is shown below.

Step 3 of 5 : Select The Type Of Leave

Type Of Leave	-
Balance Left	9
Days Applied	1
Leave Range	31/1/2001 To 31/1/2001
Apply On	27/1/2001

Step 4 of 5 : Select Reason Of Leave

☒ Rest

☐ Other

Figure 5.2.2b Old Interface

In this old interface, users need to scroll down using the scroll bar on the right in order to catch the whole screen. Unnecessary space can be eliminated by minimising the space needed to separate between sections of the interface. The new interface would look like the screen on the next page.

Step 3 of 5: Select Reason Of Leave

Type Of Leave	Annual Leave
Balance Left	9
Days Applied	1
Leave Range	8/1/2001 To 8/1/2001
Apply On	28/1/2001

Step 4 of 5: Select Reason Of Leave

☒ Rest

☐ Other

Step 5 of 5: Click To complete Application

Figure 5.2.2c New Interface

Comparatively, in this interface, users need not scroll down to see the end of the document. Hence, this would ease the users of having to manoeuvre around the interface.

Client-site input checking was performed to validate the correctness of inputs, as well as to improve the performance of the system. Besides this, a Microsoft Active-X control calendar is used to avoid users from entering invalid date formats.

5.2.3 Database

The database was initialised with some actual data, imported with the existing database. This is to ensure the length of the data is correct so that the data entered would not be truncated. For example, the input boxes on the interfaces were designed with limited lengths. Moreover, with the actual data, it helped to test the accuracy and reliability of the system.

5.3 Conclusion of System Development and Implementation

After developing and implementing the system, testing has yet to be done to test the accuracy and reliability of the application. Testing of the system is discussed in the following chapter.

Chapter 6: TESTING OF PROGRAM

The testing process of a program ensures that every function implemented works correctly, accurately, and reliably. The main objective of testing is to identify as much defects as possible and to eliminate them. Therefore, all testing for LMS was run in parallel with system development. Five testing strategies were conducted for LMS: unit testing, module testing, interface testing, integration testing, and system testing. Each of the phases is discussed as below.

6.1 Unit Testing

Unit testing aims at the verification of the smallest unit within a program. Each unit was tested independently to assure accuracy. In LMS, each module contains sub-modules, which in turn consist of different functions and units. These functions were individually tested before the entire application was tested. The white box and black box approach was used to carry out unit testing.

White box testing deals directly with the structure of codes within a module or segment. Some of the code coverage are discussed below:

- Segment coverage testing

Every segment of the code in LMS was rechecked to ensure that all of them are supposed to be executed at least once

In the *logon.inc* file, when users enter their login names and passwords, the system will authenticate their identities by checking in the database. Even though their identities might not concern other modules, but along the way the users might perform some other functions that concern other modules as well (e.g. An applicant enters the system. The system still needs to check his/her approver/s). As such, the system queries each and every identity of LMS users from the database. Testing was done to see if the four different type of users would have data (applicant, approver, administrator and super administrator).

The best way to debug the flow of the codes and retrieve the data was to use the debugging function in ASP. To debug line by line from the start, press **F11** from the keyboard. If one wants to perform debugging only at a specific page, then one should set that page as a start page by right-clicking the file, and choose 'Set As Start Page'. When debugging, as soon as the yellow line passes a line of code, bring the cursor near the line of code. The value of the code would be shown in a small yellow box. For example, if the code is suppose to select data from the database, as soon as the yellow line passes the code, the selected values from the database would be shown. This proves that the codes are performing and running well. Below is an example of the codes executing all segments of users of LMS from the database.

```
Public Function BAuthenticateUser
username = trim(replace(request.form("LoginName"), "", ""))
Password = fnGenEncryptedPassword(request.form("Password"))
set dbs = server.CreateObject("AMSDB.connection")
dbs.open Application("amsdev")

Strapplicant = "SELECT ApplyLeave, Accessright, ApplicantID from ApplicantInfo where LoginName=" &
               username & " and Password = " & Password & " ,"
set db= dbs.execute(Strapplicant)           'retrieve applicants' data

StrAdmin = "SELECT * from SuperAdmin where AdminID=" & username & "" &
           " and Password = " & Password & " ,"
set dbAdmin= dbs.execute(StrAdmin)         'retrieve administrator's data

StrApprover2 = "SELECT A.Email, A.DepartmentCode, A.DepartmentName, A.LoginName, A.Route,
                  A.Status, A.ApproverName from Approver A, DeptCode D where A.LoginName=" &
               username & "" &
               " and A.Password = " & Password & " and A.DepartmentCode =
               D.DepartmentCode and A.Status = 2;"
set Approver2 = dbs.execute(StrApprover2)   'retrieve second approvers' data

StrApprover1 = "SELECT A.Email, A.DepartmentCode, A.DepartmentName, A.LoginName, A.Route,
                  A.Status, A.ApproverName from Approver A, DeptCode D where A.LoginName=" &
               username & "" &
               " and A.Password = " & Password & " and A.DepartmentCode = D.DepartmentCode
               and A.Status = 1;"
set Approver1 = dbs.execute(StrApprover1)   'retrieve first approvers' data

RStrApprover2 = "SELECT A.REmail, A.RFullName, A.DepartmentCode, A.RLoginName, A.Route,
                  A.RStatus from Approver A, DeptCode D where A.RLoginName=" & username & "" &
               " and A.RPassword = " & Password & " AND A.Route=2 and A.RStatus = 2 and
               D.DepartmentCode = A.DepartmentCode;"
set RApprover2 = dbs.execute(RStrApprover2) 'retrieve data of replacement for second approvers

RStrApprover1 = "SELECT A.REmail, A.RFullName, A.DepartmentCode, A.RLoginName, A.Route,
                  A.RStatus from Approver A, DeptCode D where A.RLoginName=" & username & "" &
```



```
" and A.RPassword = "" & Password & "" AND A.Route=1 and A.RStatus = 1 and
D.DepartmentCode = A.DepartmentCode;"
set RApprover1 = dbs.execute(RStrApprover1) 'retrieve data of replacement for first approvers
```

- Compound condition coverage

For multiple conditions that appear in the codes, every possible combination of conditions was tested.

As an example, there are multiple conditions for the approval of leave function. The system needs to check whether the approver is a first approver, second approver, or the only approver. Hence, each unit was tested to see if the particular approver would enter a certain part of the code meant for the approver. Below is an example of the approval function. To check for a certain value, ASP is able to add 'watch' whereby a certain value would be displayed when the code is highlighted.

To show how to add watch, refer to line 9. Highlight 'rs.fields("Processed")' and press Shift + F9. If the watch value shows '2', this means that the field in database is updated with the value '2'.

```
if ((acright = "Approver2" or acright = "Approver" or trim(acright) = "") and status = 2) 1
or (acright = "Approver" and status = 2) then 'final approver 2
3
if rs.fields("Processed") = 2 then 4
session("tag") = 1 'avoid approving more than once 5
6
elseif rs.fields("Processed")= 1 or ((acright = "Approver" or trim(acright) = "") and 7
rs.fields("Processed")= 0) then 8
rs.fields("Processed") = 2 'approved by final approver 9
if rsleave.fields("Counted") = true then ' it is a recorded leave 10
..... 11
12
elseif rs.fields("Processed") = 1 then 'approved by first approver 13
session("tag") = 1 14
else 15
rs.fields("Processed") = 1 16
end if 17
end if 18
end if 19
```

- Data flow testing

This is meant to reflect dependencies which are mainly caused by sequences of data manipulations. For example, after an application of leave is submitted, it is supposed to reach the appropriate approver instantly. If the application has been processed, a mail should be correctly sent to the right applicant.

In the screen below, applicant ‘Chan Yim Khim’ applies for leave. Ideally, this leave application should reach her approver, who is the ‘Ketua Jabatan Demo LMS’.

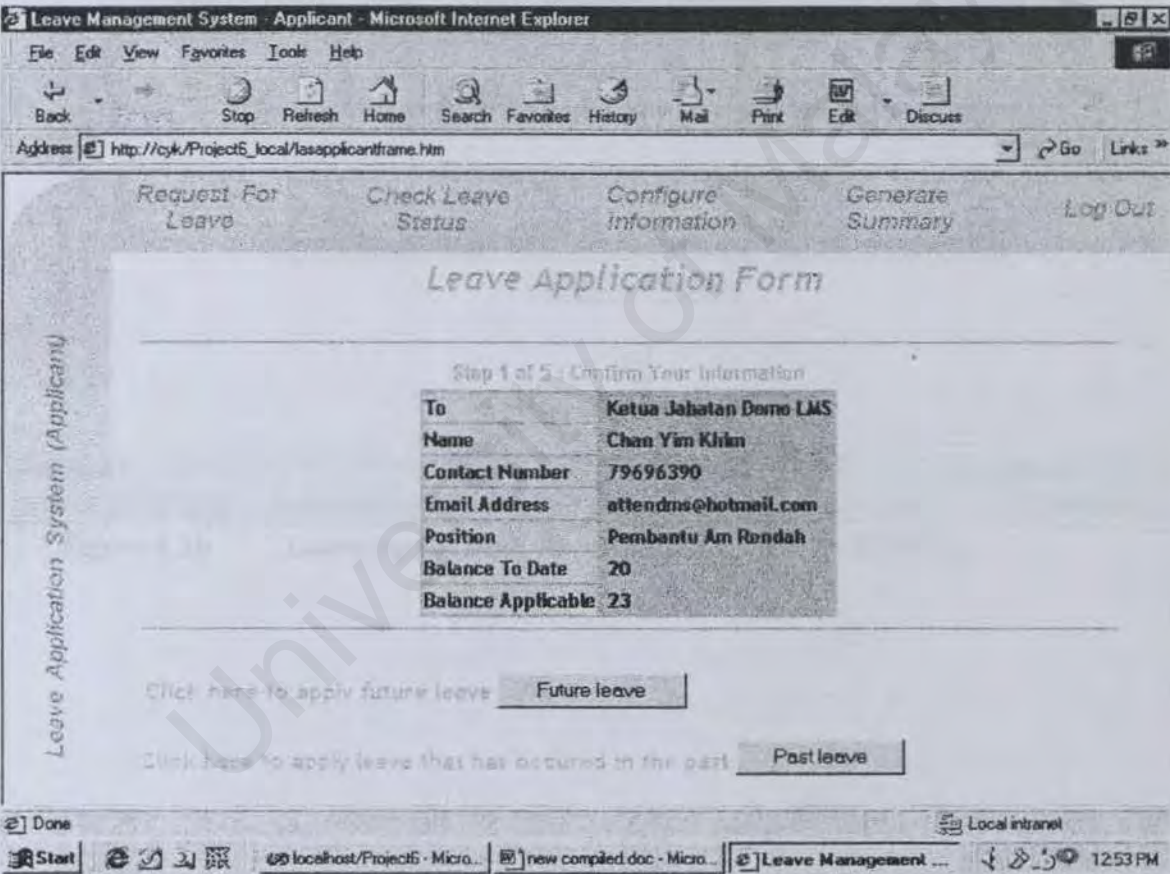


Figure 6.1a Applicant from JBDEM Applies For Leave

This applicant applies for one day of annual leave. The application should reach the approver's screen, as shown below:

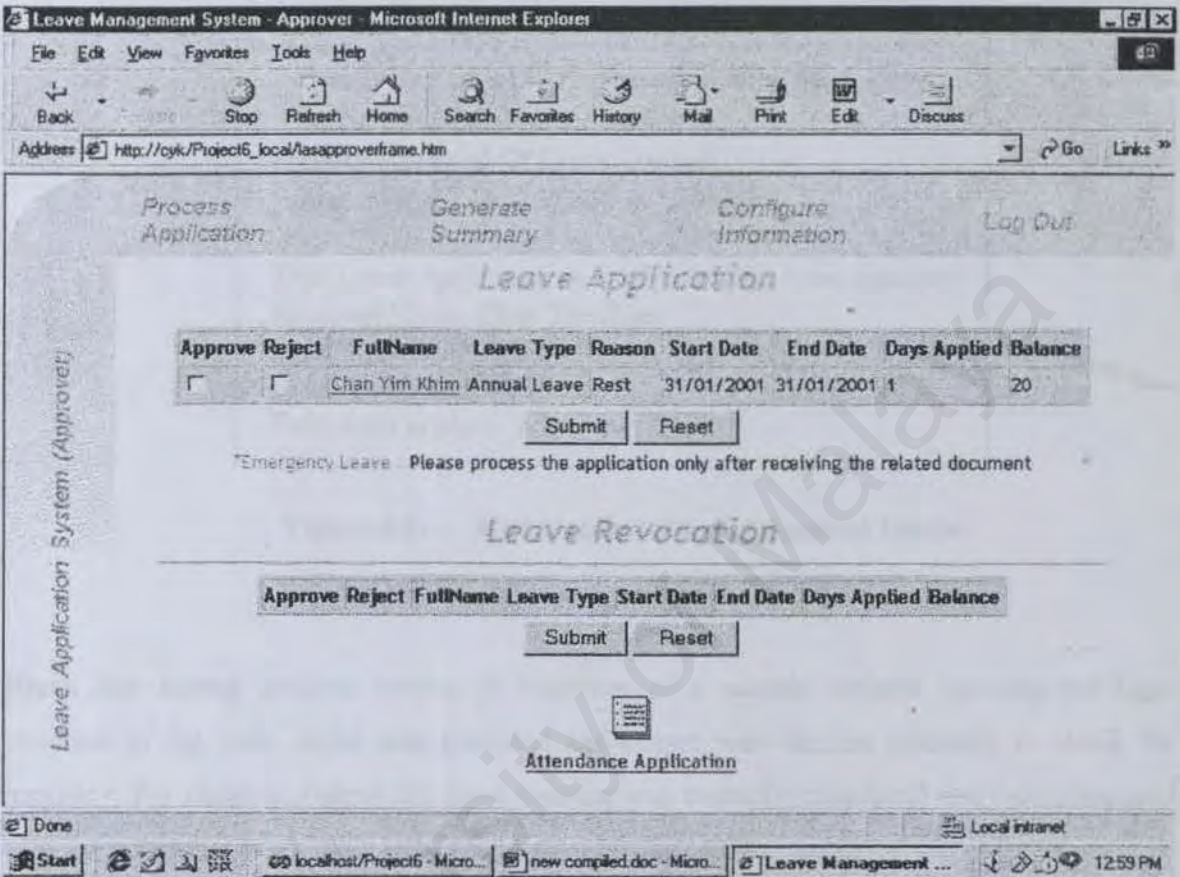


Figure 6.1b Leave Application Received By Approver, KJBDM

When the approver approves this application, a mail should be sent to the applicant, informing the applicant that the application of leave has been approved.

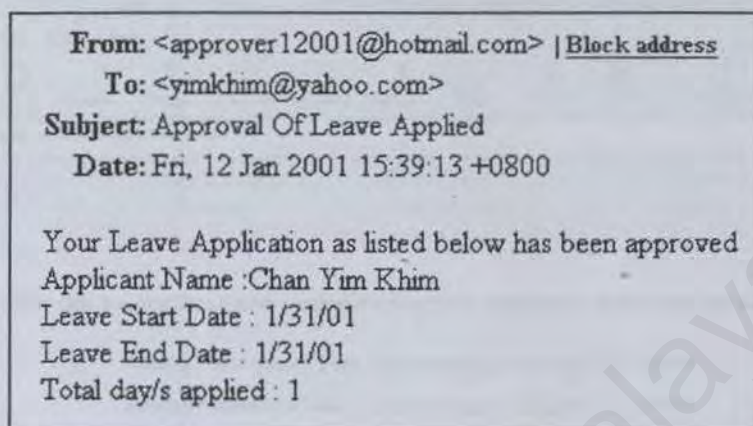


Figure 6.1c Mail Notification of Approved Leave

Black box testing involves testing of functions of a module without knowing the logic structure of the code. Input was provided and output was verified manually to check for accuracy. For example, output like leave balance was manually calculated and then compared with the generated result. Any error found must be corrected.

Input analysis

A tester deliberately makes some mistakes or keys in value that is out of acceptable range. This was done to see how the system will react and to determine whether the system will prompt the tester with the appropriate error message. For example, the tester, an applicant, is asked to:

1. Put in leave periods that overlap with the existing leave periods that have already been applied.

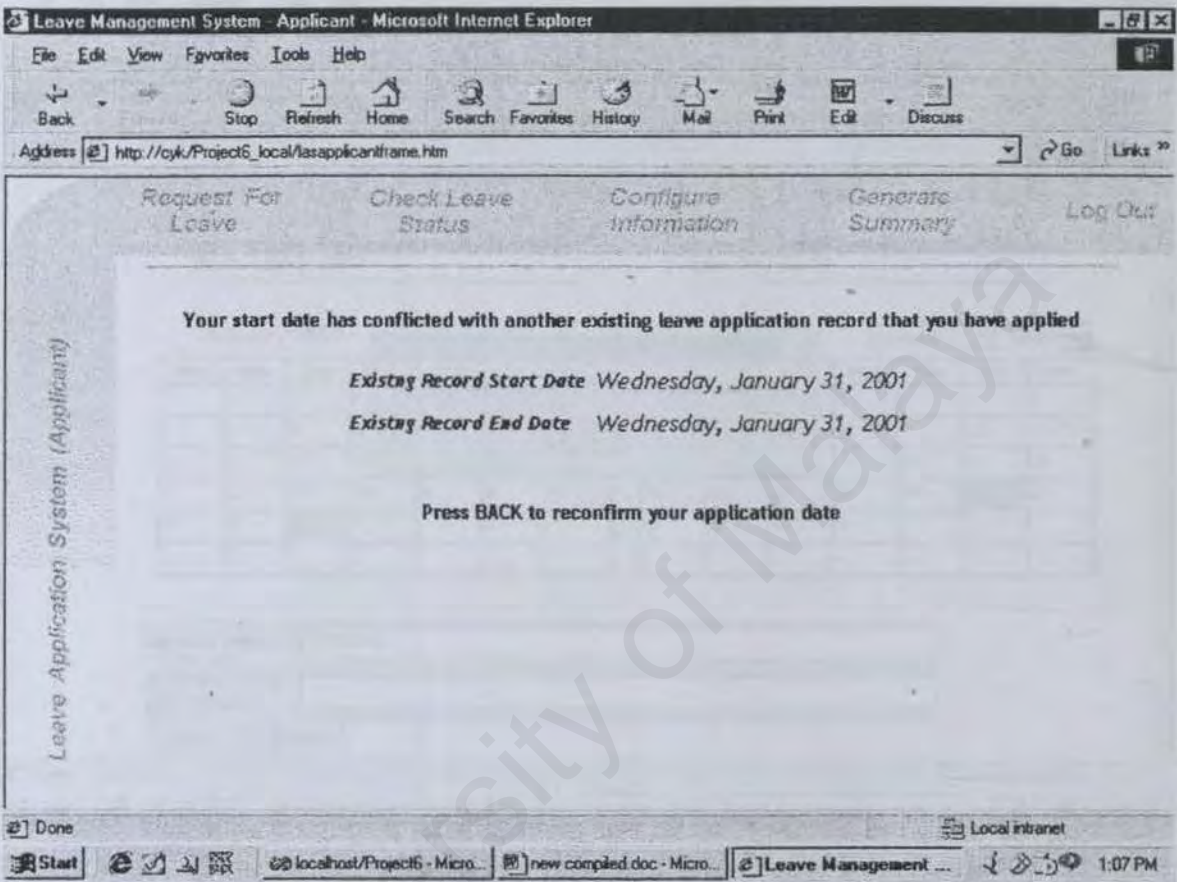


Figure 6.1d Error Message Indicating Overlap of Leave Application

- 2. Put in incorrect date sequence so that leave end date is earlier than leave start date.

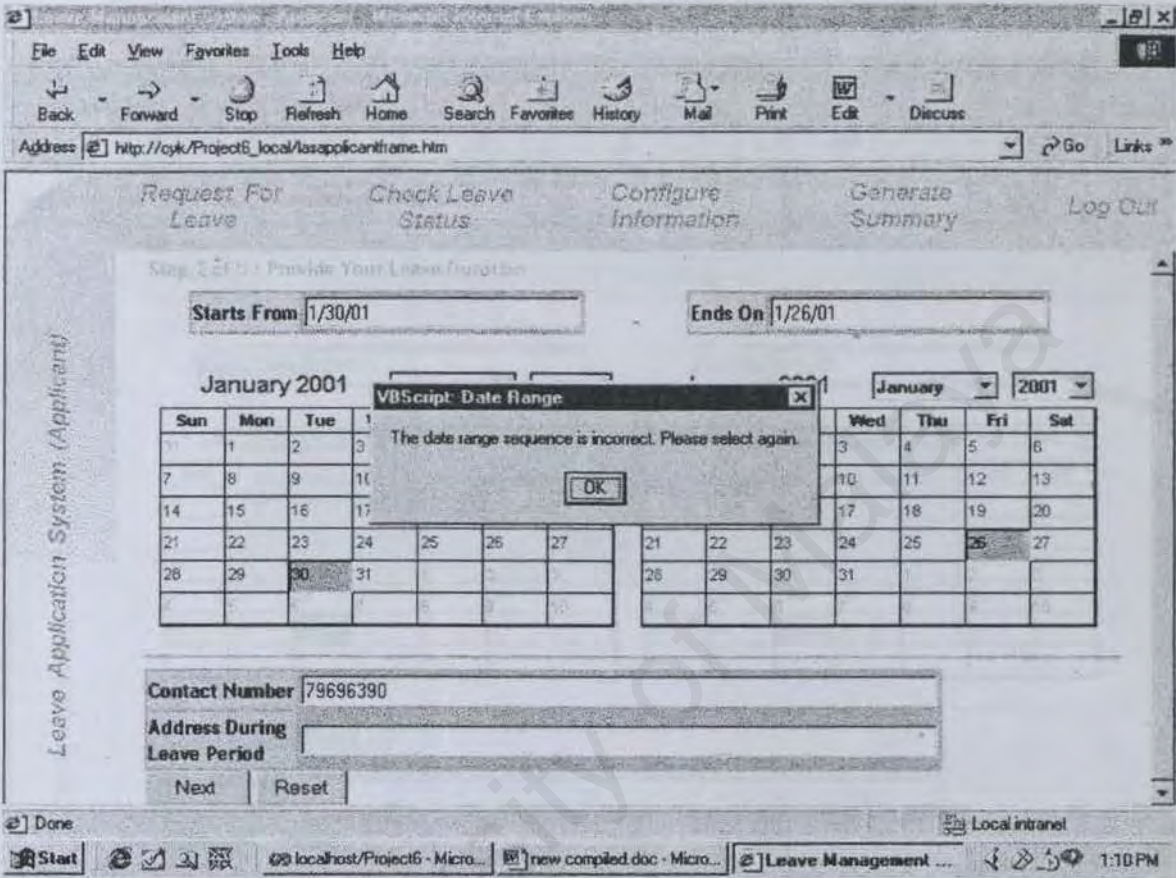


Figure 6.1e Error Message Indicating Incorrect Sequence of Date Range

- 3. Put in leave start date after the current date, but the test for the application is suppose to be for past leave.

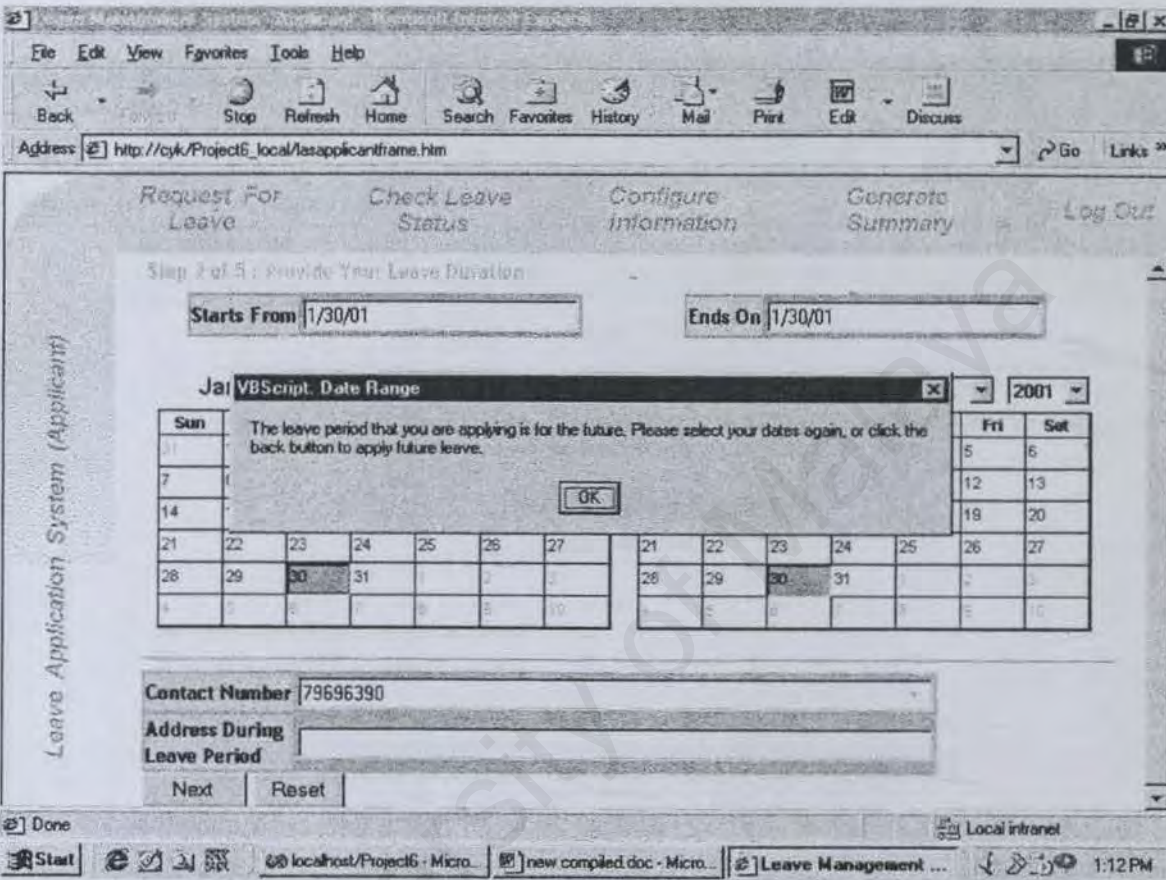


Figure 6.1f Error Message Indicating Mismatch in Applying for Future Leave

6.2 Module Testing

Module testing includes testing of all the four modules in LMS; the applicants, approvers, administrator and super administrator. Units in different modules are tested together to ensure the flow between the codes of the modules are not disrupted. The integrated units of codes must be checked intricately so that any error can be identified between the flow of the modules. If any error occurs, unit testing is done to check on the specific unit or function of the program. In the following, an example is shown how an administrator sets a non-working date which will affect the leave application of an applicant.

Administrator

A person acting as an administrator clicks on ‘Add New Holiday’ to add a non-working date.

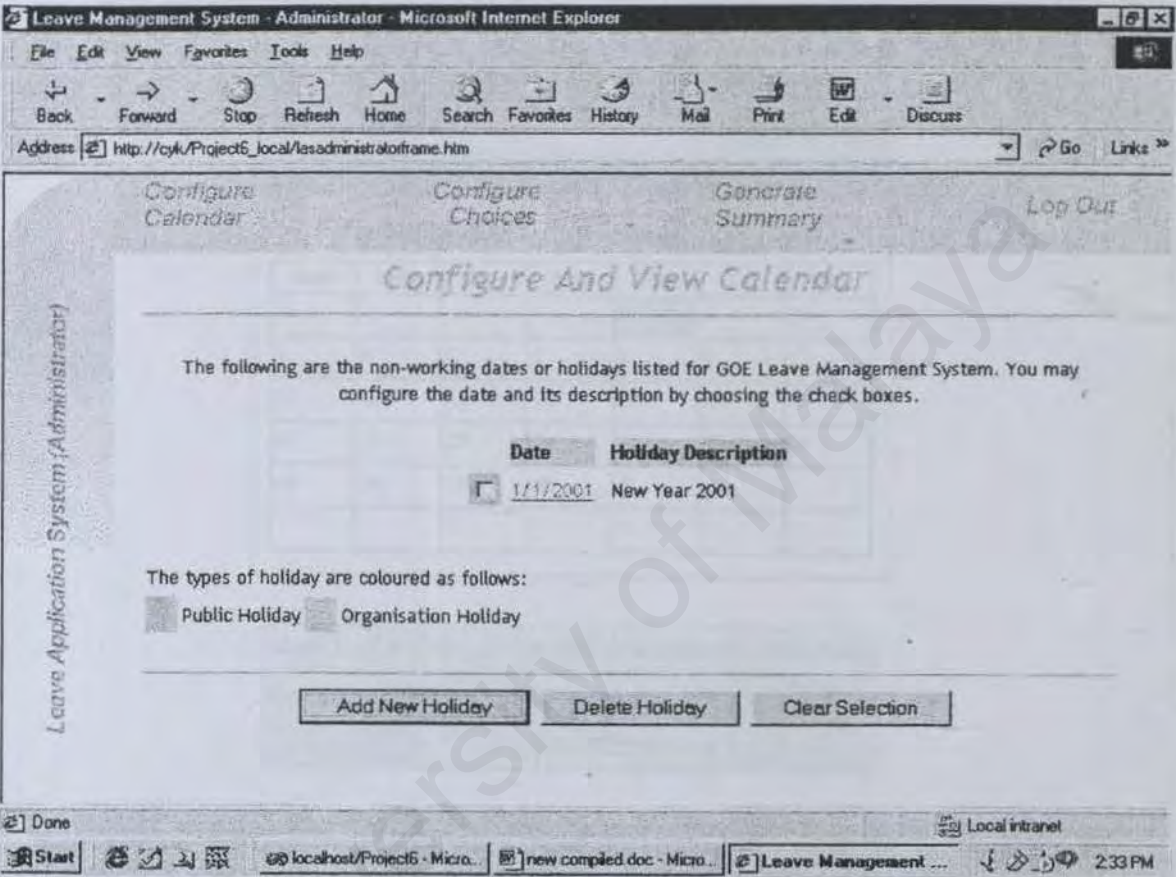


Figure 6.2a Administrator Adds New Holiday

The acting administrator sets 1st of February, 2001 as a non-working date (Federal Territory Day).

Add New Holiday - NonWorking Date

Step 1 of 3 Click on the calendar to choose the date

Holiday Date

February 2001

February

2001

Sun	Mon	Tue	Wed	Thu	Fri	Sat
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	1	2	3
4	5	6	7	8	9	10

Step 2 of 3 Choose the type of holiday for this date

Holiday Type ☒ Public Holiday

☐ Organisation Holiday

Step 3 of 3 Type a short description about the holiday.

Holiday Description

Add Holiday

Clear Selection

Figure 6.2b Administrator Sets 1st February, 2001 as a Public Holiday

The acting administrator clicks on ‘Add Holiday’ and the value would be updated in the database.

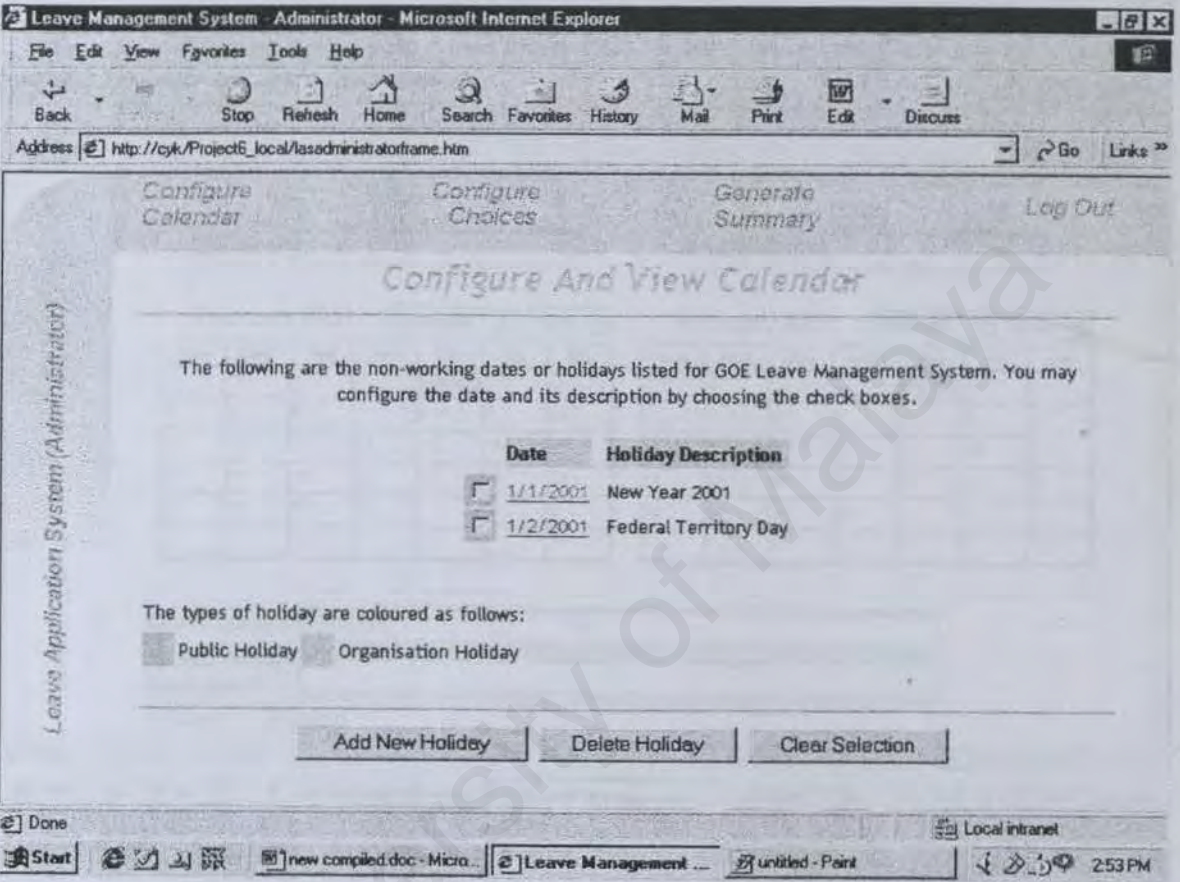


Figure 6.2c A Non-Working Date is Added

SQL Server Enterprise Manager - [2:Data in Table 'NonWorkingDate']		
Console Window Help		
SQL		
NonWorkingDate	Description	HolidayType
1/1/01	New Year 2001	PUB
2/1/01	Federal Territory Day	PUB
*		

Table 6.2 A Non-Working Date is Updated in the Database

An applicant, who tries to apply leave on that day, carries on the test.

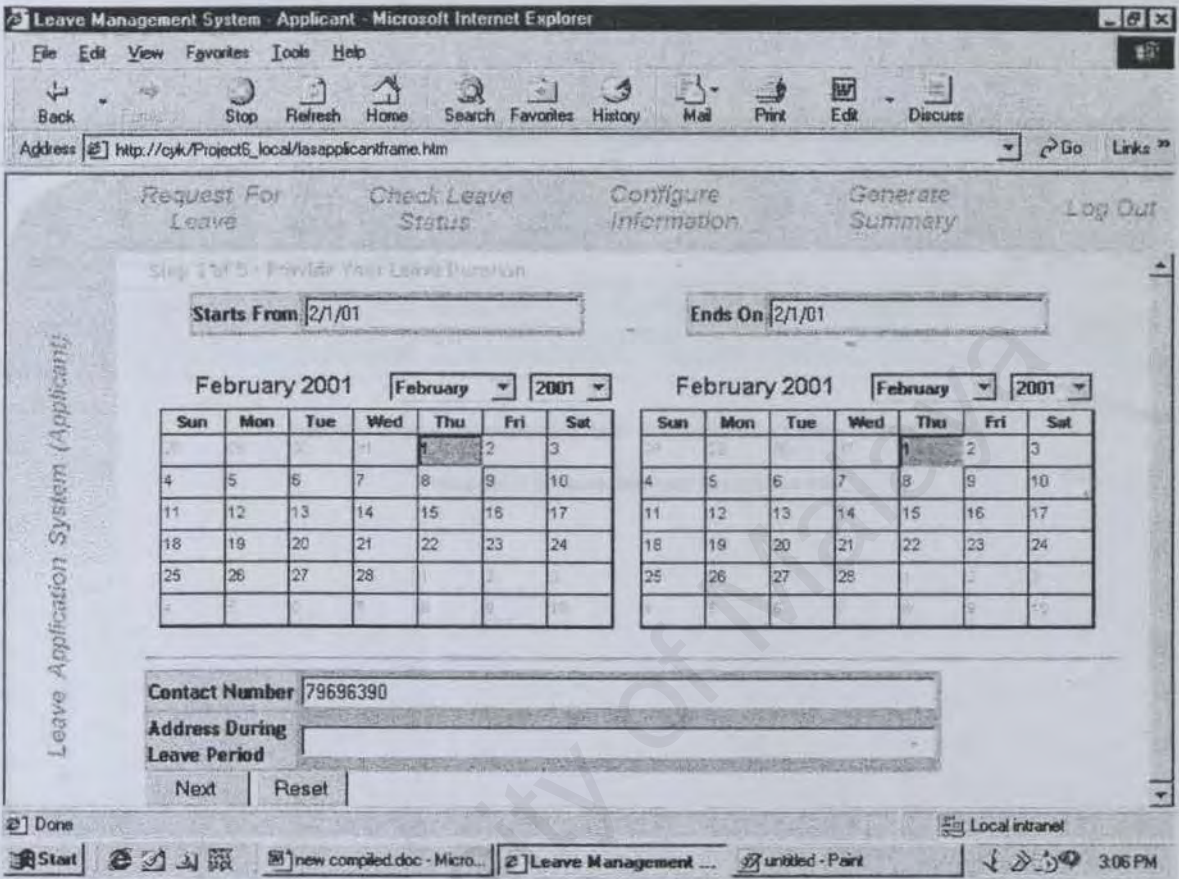


Figure 6.2d Applicant Applies Leave on a Non-Working Date

The applicant should be shown an error message indicating that the date of application is a non-working date. Hence, the applicant should choose another set of dates by pressing the ‘Back’ button.

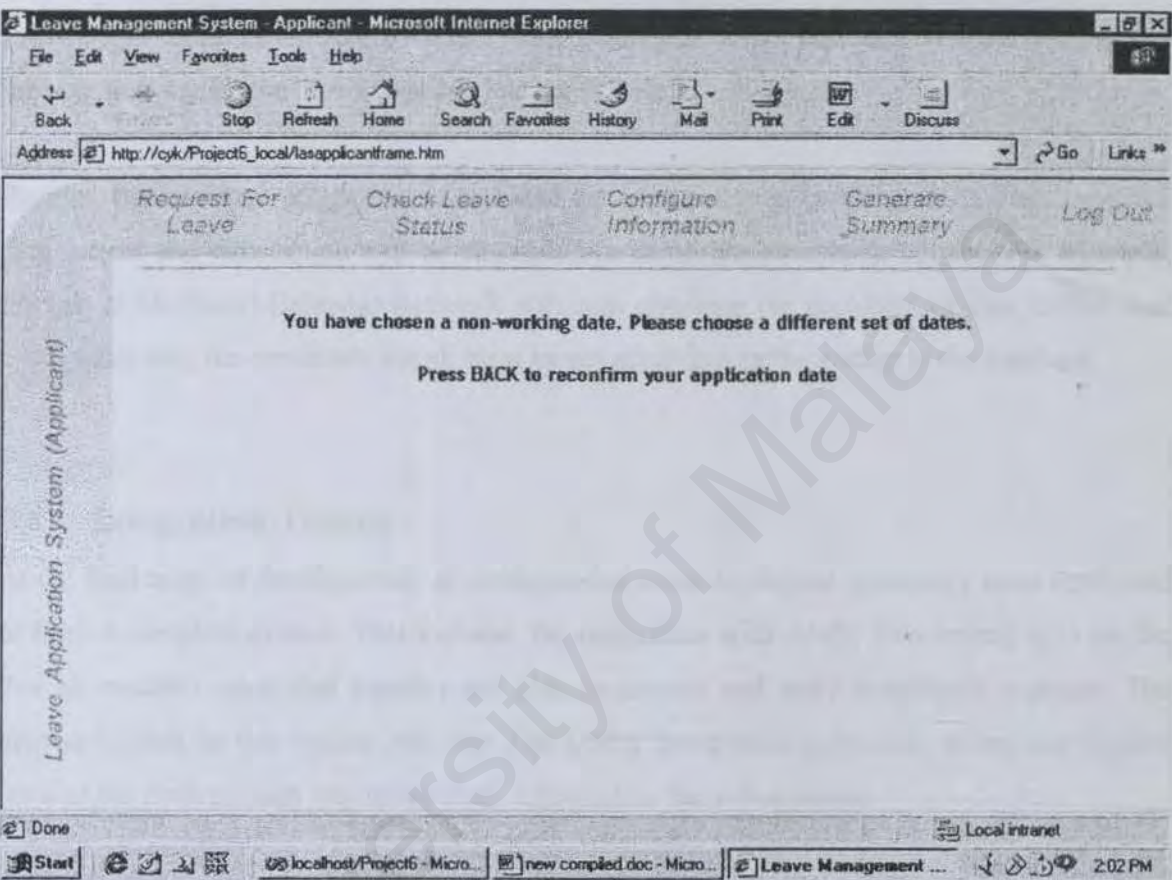


Figure 6.2e Error Message Informing Applicant of the Invalid Leave Date

6.3 Interface Testing

The interface should be user-friendly and not misleading. It is crucial to ensure the user understands what he/she is doing and what is the expected outcome. Instructions must be given in an appropriate manner and at the appropriate time. Error messages should be clear and straight to the point. However, the error messages should not incur any bad feelings on the user, leaving he/she discouraged to use the system.

Besides, the interface design should not lead the user to key in invalid entries. Data type like date format and data length will be controlled to avoid unnecessary problems. For example, the use of Microsoft Calendar Active-X will help eliminate the problems of date format. For every input text, the maximum length must be set according to the setting in the database.

6.4 Integration Testing

At the final stage of development, all modules that were developed separately were combined to form a complete system. This includes the integration with AMS. This testing is to ensure that all modules integrated together are able to interact and work seamlessly together. The approach used in this testing was the Top-Down Integration approach, where the highest level of the main module was tested first, followed by the sub-modules.

Every link to all modules was tested and all components must be tested again after the integration. White box and black box testing was repeated and every output was verified again. The flow of information between modules was validated for accuracy and completeness. In the example below, the flow of logging in function of a LMS user (applicant) is shown.

Applicant logs in through the system using the integrated interface.

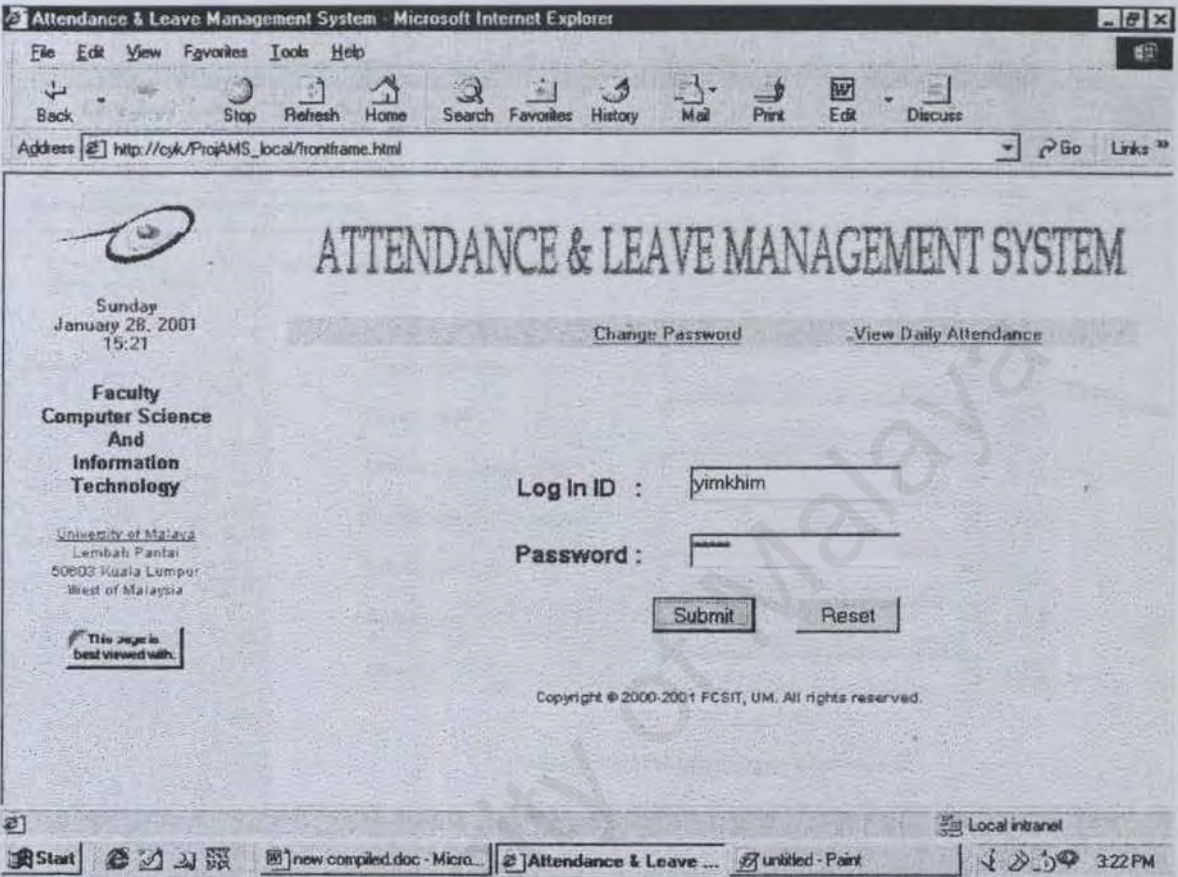


Figure 6.4a Applicant Logs In Through The Integrated Interface

The applicant enters the welcome page. To enter the individual leave application page, the applicant clicks 'Leave' on the menu bar.

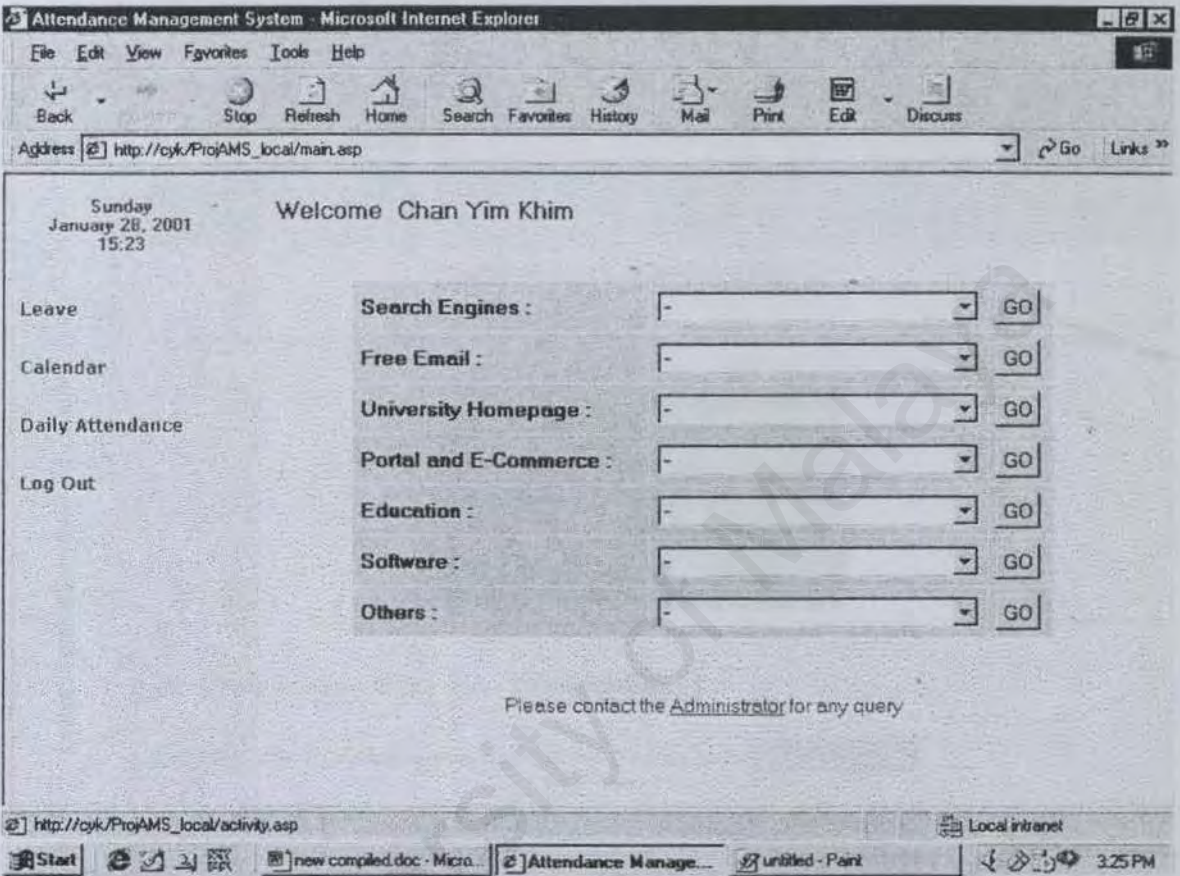


Figure 6.4b Applicant Enters LMS by Clicking 'Leave' on the Menu Bar

The applicant should be able to enter his/her own leave application page via a different screen.

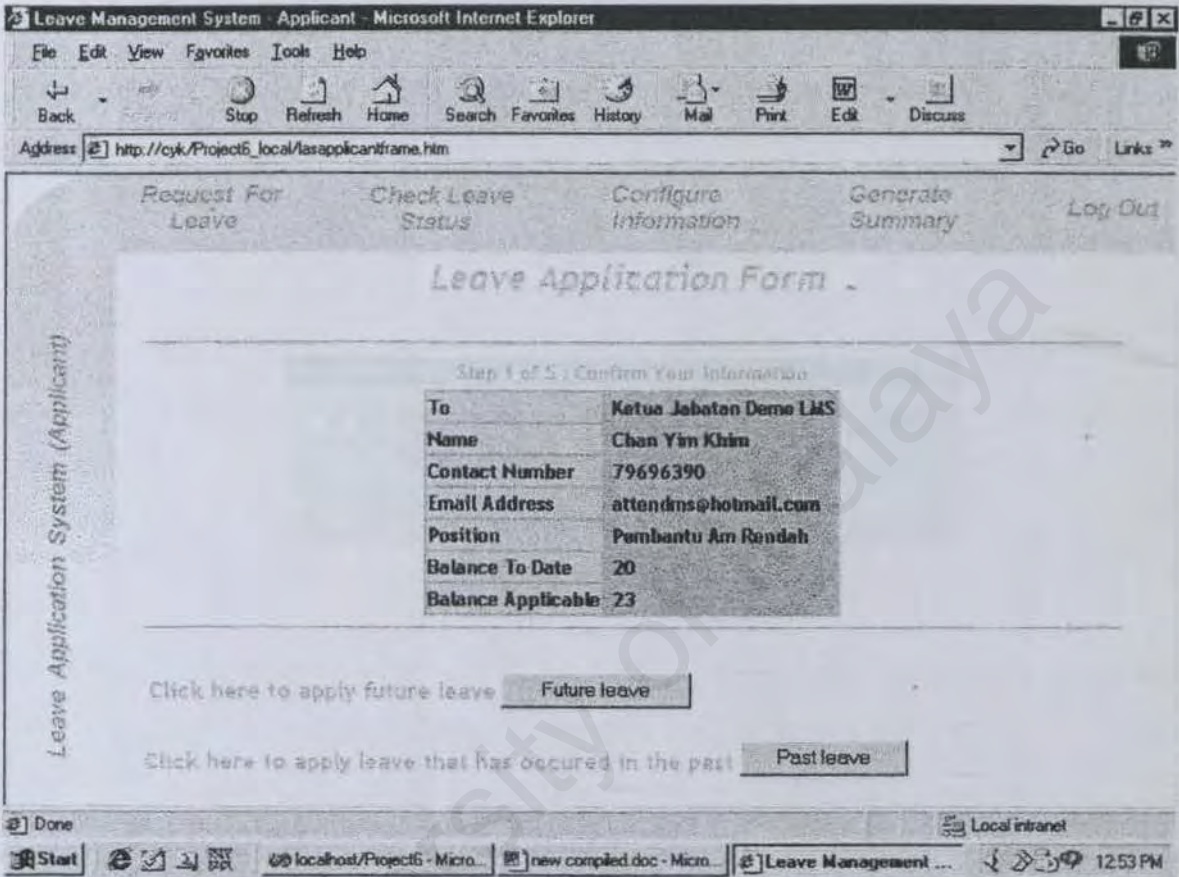


Figure 6.4c Applicant is Directed to His/Her Own Leave Application Page

When the applicant logs out, the applicant should be prompted with a dialogue box, requesting the user if he/she really wants to close the window.

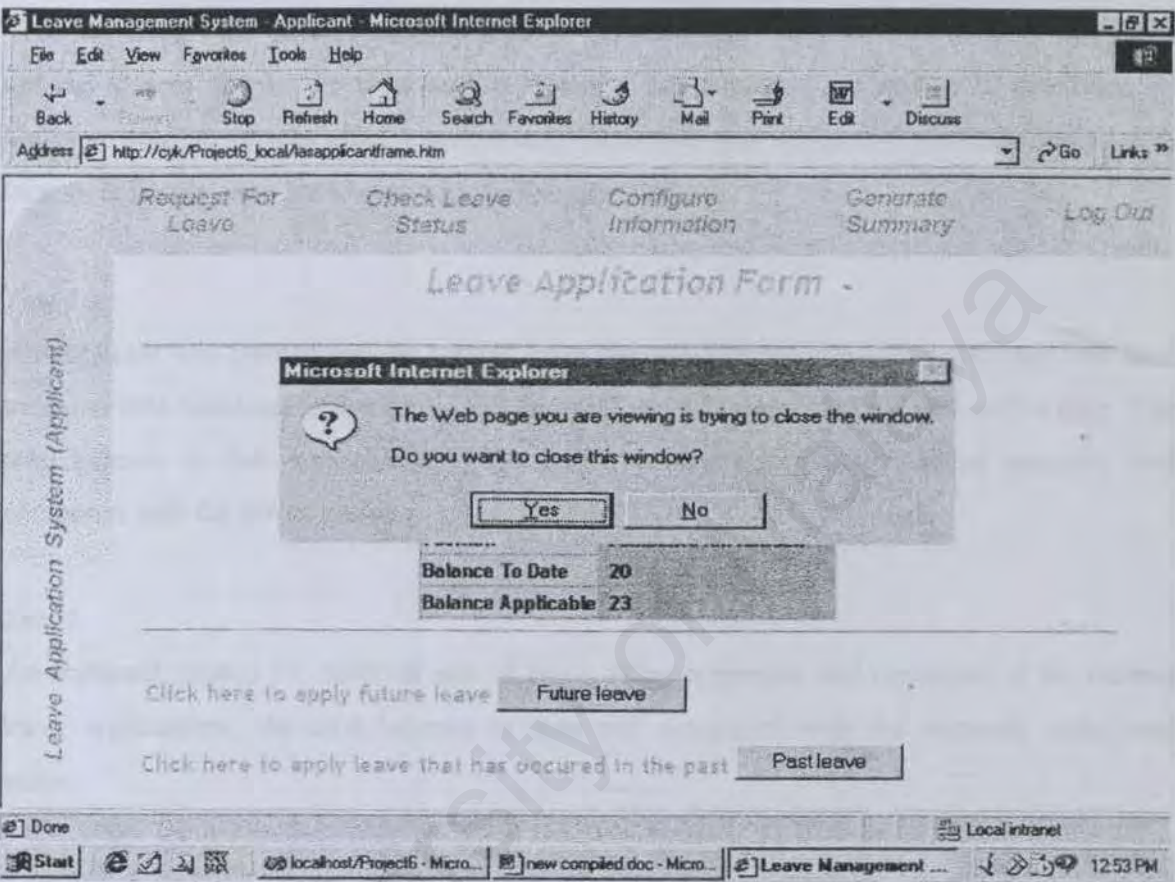


Figure 6.4d Applicant Logs Out and is Redirected to the Integrated Main Page

After the applicant clicks 'Yes', the applicant will be redirected to the main page, as depicted in Figure 6.3a.

6.5 System Testing

System testing was the final test to be carried out. A system test is a series of different tests designed to fully exercise the system to uncover its limitation and to measure its capabilities. This test covers the performance, reliability, accuracy and other criteria. Testers who were not the system developers tried on the system. They accessed the system to determine its ability to suit the current office environment. Each user was assigned to different modules to be tested. Errors were jotted down for further correction.

Test 1

An applicant was created and the current leave balance was recorded. The applicant was then awarded with additional leave days, and the total leave brought forward was added later. The total balance to date was checked. The leave balance was then calculated manually and compared with the actual output.

Test 2

An applicant applies for different sets of leave. After approvals and rejections of the various leave applications, the total balance to date was compared with the manually calculated value.

Test 3

An administrator adds an approver. The new approver should be able to log in and view the default approver web page.

Test 4

An approver purposely approves the same application twice on different computers. An error message should appear to let the approver know of this mistake.

6.6 Conclusion of Testing

The results of the various tests were taken into account and changes were done where needed. The next chapter evaluates and concludes the project.

Chapter 7: EVALUATION AND CONCLUSION

7.1 Strength

7.1.1 Wide-accessibility

Since LMS is deployable on the Internet, users can always access the system virtually from anywhere in the world. The basic tool needed on the client-side PC is just Internet Explorer 4.0 or above, which is already preinstalled with Windows 98.

7.1.2 Interoperability

The distributed nature of server-side application has contributed to the interoperability of LMS. The application files are located in the web server while the database server is located in another machine. Communication through the machines from different domain is possible through the establishment of trust relationship. This also means that LMS is highly portable and the implementation of distributed server can be established easily, either to a new machine or an existing server.

7.1.3 Coding Reusability

The simplicity and modularity of the functions make them reusable. Almost all functions are located in the “*.inc” (include) files. Therefore, developers are able to trace them easily. With remarks and high modularity of the functions, they are easy to understand and reuse. For example, some functions like *countdays* and *checkholidays* can be used without any modification.

Additionally, the stored procedures can be reused as often as possible (as it deems fit) for the same function. They are stored in a precompiled form in the database and are quicker to run.

7.1.4 Confidentiality and Integrity of Information

Passwords and log in names are needed for users to log into the system. There are headers in every file to check for user authentication. This is to protect the confidentiality and integrity

of information so that only information which the users have access rights to can be published. Besides that, passwords encryption has fully enhanced the security of the system.

In addition, SQL Server will validate the users' accounts written in "*global.asa*" file before returning any results to the users. For Internet Information Server, "*global.asa*" is a file with global variables and is protected by the web server from remote access. Also, directory browsing is disabled and the access rights of anonymous users who log in to the web server are controlled.

Furthermore, the database will be backed up every day at 12:00 am automatically. This acts as a precaution for unexpected problems, such as power break down.

7.1.5 Scalability

LMS is very scalable. For example, it is possible to add another machine dedicated as a mail server in the future. The changes needed are the inbox setting for Windows Messaging in the Control Panel and the DNS settings to "recognise" the mail server. A trusted relationship must exist for machines in different domains.

7.1.6 Reliability and Accuracy

The interface design is consistent throughout the whole system. A navigation bar is always in top and brief instructions are given for every process. Besides, the current leave balances will be calculated again after some changes, such as adding or deleting holidays. With automated calculation, the accuracy of leave balances is guaranteed.

LMS is also reliable because of the error tolerance provided by the system. For any errors made by users such as duplication of log in names, deleting empty records or submitting changes twice, an error message for each process will be prompted stating the particular mistake.

In terms of accessibility, LMS is easily accessed online; 24 hours a day, 7 days a week.

7.2 Limitations

7.2.1 Browser and Platform

LMS is limited to certain platforms only. It can only run in Windows 95/98/NT with Internet Explorer 4.0 and above. This limitation is due to the usage of Microsoft Calendar Active-X control. Active-X control can only work properly in Netscape Navigator's browser with the Active-X plugs installed. Besides that, VB Scripts that are used in LMS can be supported only in Internet Explorer browser.

7.3 Problems and Solutions

During the entire development of LMS, many problems were encountered. Some of the problems were overcome through certain solutions and some were unfortunately not solved due to unforeseen circumstances. The problems are discussed as below:

7.3.1 Multiple Accounts for Second Approvers

Before the start of this project, there were two accounts for the Deputy Dean (TDP) because the table in the database could not resolve the particular post as a single approver. The Deputy Dean is the second approver for the Technical Divisions of UNIX and PC. Hence, the system could not differentiate between the approver of the former from the latter.

Solution

A new table has to be created in the database to suit the different needs of approvers; be it the first approver, second approver, first and second approver, or just an approver, who approves only once. This proved to be a challenge as all conditions pertaining to the different approvers were widespread throughout the codes. Besides creating the new table, the codes were rewritten to complement the new table. A comparison of the old and new table is shown on the next page.

DepartmentCode	DepartmentName	FName	FLoginName	SName	SLoginName
DEEM	Jab. Demo LMS	kjbdm	kjbdm	kkjbdm	kkjbdm
TDP	Timbalan Dekan Pembangunan	Timb. Dekan Pembangunan	tdp	Dekan	dekan
KJSKT	Ketua Jabatan Sains Komp. & Tek.	Ketua Jabatan Sains Komp & Tek	kjskt		
Test	testing purposes	testing approver	lee1	Lee	lee
NB	Naib Canselor	b/p Naib Canselor	bpnc		
KJSM	Ketua Jabatan Sains Maklumat	Ketua Jabatan Sains Maklumat	kjsm		
PEJ	Pejabat - Agnes	Penolong Pendaftar pp2	pp2		
KJKP	Ketua Jabatan Kejuruteraan Perisian	Ketua Jabatan Kej. Perisian	kjkp		
TKPC	Bahagian Teknikal (PC)	Peg. Sis. Maklumat	psmpc	Timb. Dekan Pembangunan	tdp2
Dekan	Dekan	Dekan	Dekan		
KJKB	Ketua Jabatan Kepintaran Buatan	K. Jabatan Kepintaran Buatan	kjkb		
TKWS	Bahagian Teknikal (UNIX)	Peg. Sis. Maklumat	psmws	Timb. Dekan Pembangunan	tdp2
PEJAM	Pejabat Am - pp1	Penolong Pendaftar pp1	pp1		
TDA	Timbalan Dekan Akademik	Timb. Dekan Akademik	tda	Dekan	dekan

Table 7.3.1a DepartmentCode table

The description of the Table 7.3.1a is as below:

- FName - First approver's name
- FLoginName - First approver's log in name
- SName - Second approver's name
- SLoginName - Second approver's log in name

As shown by the arrows, *tdp* is a first and also a second approver. However, *tdp* has different log in names for different departments:

Approver	Department
First approver – <i>tdp</i>	Deputy Dean
Second approver – <i>tdp2</i>	Technical Division (UNIX) Technical Division (PC)

Table 7.3.1b Departments for 'TDP'

As such, a new idea or solution must come up to solve this dragging problem. Besides, the flow that permeates the codes must be correct in order to ensure the process of approving leave applications was not compromised.

Hence, a new flow of approving applications was thought of. A comparison of the flows is explained on the next page.

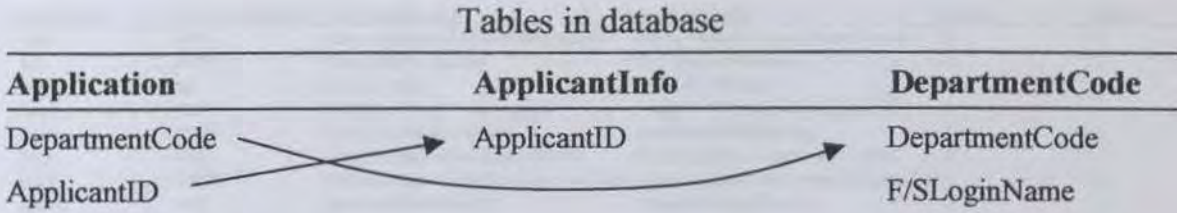


Figure 7.3.1a Previous Approval of Leave Application Flow

Previously, the flow started from the *Application* table, whereby the system will detect the department code of the applicant, and check the item in the *DepartmentCode* table. This was inefficient as the item should not be in the *Application* table in the first place.

The new flow that would replace the old one would be:

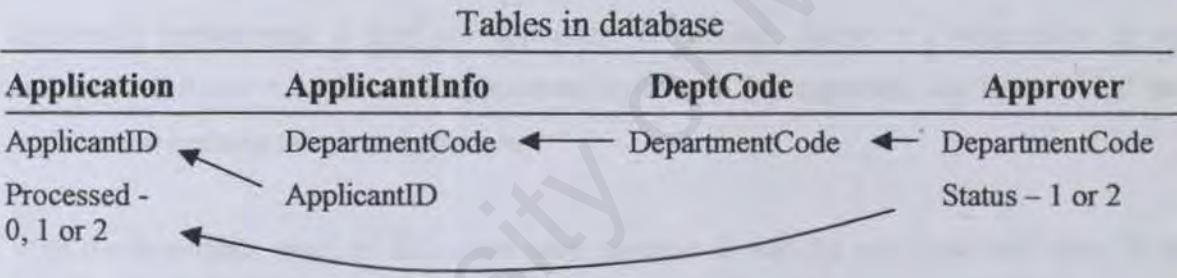


Figure 7.3.1b New Approval of Leave Application Flow

Notice that the flow now flows in the opposite direction. This is because the system will detect the status of the approver first, whether he/she is a first (Status = 1), second (Status = 2), both (Status = 1 and 2) or just an approver (Status = 1 or 2). With that, the system will check whether the leave application is not processed (Processed = 0), processed by the first approver (Processed = 1) or already approved by the second or final approver (Processed = 2). This would eliminate any discrepancies surface from who should approve which application. Simultaneously, the system would validate and verify the approver's department code with the applicant's. The new table is shown on the following page.

LoginName	DepartmentCode	DepartmentName	ApproverName	Route	Status
kjlp	KJLP	Ketua Jabatan Kejuruteraan Perisian	Ketua Jabatan Kejuruteraan Perisian	0	1
pp1	PEJAM	Bahagian Pejabat Am	Peg Sistem Maklumat	0	1
bpnc	NB	Naib Canselor	b/p Naib Canselor	0	1
tdp	TDP	Timbalan Dekan Pembangunan	Timbalan Dekan Pembangunan	1	1
psmws	TKWS	Bahagian Teknikal (UNIX)	Penolong Pendaftar pp1	0	1
kjkb	KJKB	Ketua Jabatan Kepintaran Buatan	Ketua Jabatan Kepintaran Buatan	1	1
kjskt	KJSKT	Ketua Jabatan Sains Komputer and Tekr	Ketua Jabatan Sains Komputer and Tekr	1	1
psmpc	TKPC	Bahagian Teknikal (PC)	Penolong Pendaftar pp2	1	1
pp2	PEJ	Pejabat-Agnes	Peg Sistem Maklumat	0	1
dekan	Dekan	Dekan	Dekan	1	1
kjsm	KJSM	Ketua Jabatan Sains Maklumat	Ketua Jabatan Sains Maklumat	1	1
tda	TDA	Timbalan Dekan Akademik	Timbalan Dekan Akademik	0	1
tdp	TKPC	Bahagian Teknikal (PC)	Timbalan Dekan Pembangunan	2	2
tdp	TKWS	Bahagian Teknikal (UNIX)	Timbalan Dekan Pembangunan	2	2
dekan	TDP	Timbalan Dekan Pembangunan	Timbalan Dekan Pembangunan	2	2
dekan	TDA	Timbalan Dekan Akademik	Timbalan Dekan Akademik	2	2

Table 7.3.1c Approver table

In this new table, some records might be duplicated as the table must show and differentiate the first approvers from the second. As an example, the Dean is the only approver for that department, and the Dean is also the second approver for the Deputy Dean departments. However, this is differentiated by the field 'Status'. The field 'Route' represents the approver's replacement, if they are appointed. By default, Route = 0 when there is no replacement, Route = 1 when the replacement replaces a first approver, and Route = 2 if the replacement replaces the second approver.

With the new table, much of the codes were changed to suit the new flow and table. With that, each approver only has one account; no matter he/she is an approver of one or multiple departments.

7.3.2 Ineffective Table

The previous *DepartmentCode* table (Table 7.3.1a) housed the details of not only the various departments, but also all the approvers' information. The name of the table itself should reflect only the departments' information, and not other unrelated information. Besides, this would cause confusion for the system administrators as to where to look for information as the name of the table might be misleading.

Solution

As such, the approvers' information should be exported out of the table. The approvers' information would be stored in *Approver* (Table 7.3.1c) table and the departments' information will be stored in *DeptCode* table, as shown below.

DepartmentCode	DepartmentName
JBDEM	Jabatan Demo LMS
TDP	Timbalan Dekan Pembangunan
KJSKT	Ketua Jab Sains Komputer & Teknologi
NB	Naib Canselor
KJSM	Ketua Jabatan Sains Maklumat
KJKP	Ketua Jabatan Kejuruteraan Perisian
TKPC	Bahagian Teknikal (PC)
Dekan	Dekan
KJKB	Ketua Jabatan Kepintaran Buatan
TKWS	Bahagian Teknikal (UNIX)
PEJAM	Bahagian Pejabat Am
TDA	Timbalan Dekan Akademik

Table 7.3.2 *DeptCode* table

Any changes made in this *DeptCode* table would be changed and in sync with other tables that have information regarding the departments. With this, information is stored in the correct table and irrelevant details are not included.

7.3.3 Integration Challenge

The integration with AMS was also a challenge as initially different persons developed the two applications. As a result, the style of writing is different and functions were written in a different way from each other.

Solution

Codes must go in sync with each other, at least at the log in section. In the "*global.asa*" file, both codes from AMS and LMS were combined, but it was made sure that there was no duplication as this would affect the flow of the system. Besides that, to identify and authenticate users, the AMS module would be checked first as it would have to detect the non-working dates and differentiate between academic and non-academic staff.

After that, for LMS, the applicants would enter into the welcome page of the integrated web page first, and then followed by their individual leave application page with just a click of a

button at the menu. The file that they log in to is "*logon2.inc*". For approvers, they are directed to their own leave approval page right after they key in their log in names and passwords. This is to avoid getting the message "unauthorised user" when they click on the menu at the welcome page. This was unavoidable because after the person has already log in, the system will not check the "*logon2.inc*" file again.

After scrutinising all the codes, besides checking the connection to ODBC for both applications, finally the integration was workable.

7.3.4 Lack of Time

Due to the lack of time, one module cannot be tested thoroughly, which is the conversion of leave balance. The scope of this project was substantial enough and time was spent on enhancing other modules. At the turn of the year, problems were encountered by the staff who have converted their leave balances to cash. The codes were rechecked, and it turned out that there were certain conditions that the codes did not consider. More time was needed to fix this problem.

7.4 Future Enhancement

7.4.1 Fine Tuning of System

The system should be fine tuned in such a way that users can easily add, delete or make any changes without compromising on the integrity of the flows. For example, a third approver might exist and the system administrator should be able to add "Status = 3" easily and the approving process should proceed without any hiccups.

7.5 Knowledge Gained

7.5.1 Ability to Set Up Windows NT Servers/Workstations

For a distributed system, knowledge of setting up a new domain, server and workstation is inevitable. This project provided invaluable knowledge on setting up a network of NT domains, trust relationships, configuring domain users, server manager in NT, etc.

7.5.2 Understanding of Active-X Technology

Active-X technology has contributed to a higher level of the object-oriented technology. Active-X technology stresses on maintainability and reusability. Usage of Active-X has resulted in a shorter development time. This is one of the major factors which has encouraged the growth of Active-X technology.

7.5.3 Coding using Active Server Pages (ASP)

ASP provides very powerful features, enabling one to create highly interactive and dynamic web pages. With its strong integration with Internet Information Server, Active-X technology and SQL Server, ASP currently is one of the most prominent web developing programming language.

7.5.4 Using SQL Server

SQL Server 7.0 is the best database for Microsoft Windows platform. It provides a comprehensive platform that makes it easy to design, build, manage and use data warehouse solutions. This enables any organisation to make effective business decisions based on timely and accurate information. Therefore, knowledge of SQL Server is of great value.

7.5.5 Using Transact-SQL

Transact-SQL is the standard language for communicating between applications and SQL Server. The language is an enhancement to the Structured Query Language (SQL), the ANSI-standard relational database language. It provides a comprehensive language for defining tables; selecting, inserting, updating, or deleting information stored in tables. Also, the language controls access to data stored in the tables. Extensions such as stored procedures make Transact-SQL a full programming language.

7.5.6 Debugging Skill Improved

Doing enhancement on someone else's project can greatly increase debugging skills, especially for a beginner who has no experience in the programming language. Learning about the language and debugging on the existing codes went hand-in-hand, and knowledge was gained from the whole process of enhancing and developing new codes.

7.5.7 Requirements Capturing

The development process of LMS enabled capturing of requirements in an actual office application. Through interviews, prototyping, discussions, and brain storming sessions, one can be trained to analyse and evaluate the basic requirements of a system. Knowledge was gained also through identifying system deficiencies, generating alternative solutions and evaluating the feasibility of the system.

7.6 Conclusion

This project has met the aim of solving the critical parts of the Leave Management System. Though most of the work was done through back-end jobs, much knowledge was gained as to how it should meet the needs of different users as well as to fine tune the whole system accordingly. At the completion of this project, the LMS should enable all users to use the system without much fuss.

In addition, the whole nine credit hours dedicated to this thesis provided a whole new perspective of how to develop a certain system, right from the start till the end of the project. Enhancing and developing new functionalities were no exception, and in actual fact it was even more challenging because the first step was to understand another person's codes. Besides that, invaluable knowledge was gained on the development tools such as Windows NT Server, Internet Information Server, SQL Server, and Microsoft Exchange Server.

LMS was designed in a proper manner so that it can be adopted at any time for any organisation, and only minor changes need to be done to customise the system. The enhancement of LMS in FCSIT has complemented the GOE in an actual office environment.

REFERENCES

Anderson, Blexrud, Chiarelli, & others (1999). *Professional Active Server Pages 3.0*. Wrox Press Ltd., Birmingham.

Buser, David & Kauffman, John, & others (1997). *Beginning Active Server Pages 3.0*. Wrox Press Ltd., Birmingham.

Choong, Raymond Khang Yen (1999). *Generic Office Environment (Leave Application System)*. Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur.

Kendall, Kenneth E., Kendall, Julie E. (1998). *Systems Analysis And Design*. Prentice Hall International, Inc., New Jersey.

Lee, Chun Hoo (2000). *Leave Management System*. Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur.

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

Sommerville, Ian (1998). *Software Engineering*, 5th ed. Addison-Wesley, England.

Vieira, Robert (1999). *Professional SQL Server 7.0 Programming*. Wrox Press Ltd., Birmingham.

“**Active Server Pages**”. Retrieved on 21 June, 2000, from the World Wide Web:
http://idm.internet.com/articles/200002/wd_02_25_00aa.html

“**Internets and intranets: ASP**”. Retrieved on 21 June, 2000, from the World Wide Web:

http://www.cetus-links.org/oo_active_server_pages.html

“Internet Information Server”. Retrieved on the 16 June, 2000 from the World Wide Web:
<http://www.furman.edu/~ggallowa/MIIS.html>

“Internet Information Server 4.0 Data Sheet”, *Microsoft Windows NT Server*. Retrieved on 12 June, 2000, from the World Wide Web:
<http://www.microsoft.com/ntserver/web/exec/feature/Datasheet.asp>

“Leave Management System”, *Professional Software Systems*. Retrieved on 12 June, 2000, from the World Wide Web:
<http://www.prosoft.com.sg/lms.htm>

“Leave Management Software”, *Skyhi Software*. Retrieved on 25 July, 2000, from the World Wide Web:
<http://www.skyhi.co.za/products.htm>

“Leaves Project”. Retrieved on 13 June, 2000, from the World Wide Web:
<http://ais.cern.ch/projs/leaves/welcome.html>

“Lotus Leaves Leave Management System”, *Lotus Notes Groupware Solutions*. Retrieved on 12 June, 2000, from the World Wide Web:
<http://www.groupware.ital.com/leave.html>

“MSDN Online Web Workshop”. Retrieved on 16 June, 2000, from the World Wide Web:
<http://msdn.microsoft.com/workshop/>

“NexTrak Leave Management System”, *NexTrak*. Retrieved on 12 June, 2000, from the World Wide Web:
<http://www.nextrak.com/product2.htm>
<http://www.nextrak.com/nextrakleavemanagementsystem.htm>

<http://www.mypage.bluewin.ch/queloz/vactrac5.htm>

“SQL Magazine”. Retrieved on 25 June, 2000, from the World Wide Web:

<http://www.sqlmag.com/>

“Web-based Travel-leave Management System”. Retrieved on 18 July, 2000, from the World Wide Web:

<http://www.chttl.com.tw/pub/major/te88-007.htm>