BOSS PAYROLL MANAGEMENT SYSTEM

Prepared by : Ong Siew Kheng
Matrik No : WQT000028

ABSTRACT

YTL Power Services is a multi-disciplinary services company registered in Malaysia, specialising in the operation and maintenance of power stations. The enterprise is jointly owned by YTL Corporation Berhad and Siemens AG of Germany up to the end of 2001. From 2002 the company ownership is 100% YTL Corporation. The company had its origins in Malaysia's newly competitive power industry, which was transformed in the early 1990's by privatisation and the arrival of independent power producers.

The company manages the entire range of operation and maintenance (O&M) activities for YTL Power's two Malaysian power stations, which together generate about 12% of the electricity consumed in Peninsular Malaysia. YTL Power Services also provides engineering expertise to other projects in the region. Technology transfer is built into the structure of the company and has been the guiding philosophy behind its formation and growth.

YTL Power Services is a modern company operating in a high-technology environment and the use of the latest equipment and techniques for maintaining and running combined cycle gas turbine power plants is integral to the company's organization.

The primary objective of YTL Power Services: to develop a sustainable payroll management system in Human Resource Department due to increasing of employee in the organization. The current database is using Microsoft Access 97 to update and controlling employee data for the total of 314 employee was facing difficulties where the backup is not sufficient enough to track the back record.

Therefore, I am a staff in Human Resource Department mainly incharge for the payroll of the organization and my intention is to build a better system to contribute for my department.

ACKNOWLEDGEMENT

First and foremost, I would like to give my sincerely thanks to my supervisor, Mr.Mustaffa Kamal Bin Mohd Nor for his unwavering guidance, advise, time and thoughtful contribution to my project. He is so kind by showing me every steps need to be included without failure.

I am also very grateful to thank my modulator Dr. Rosli Bin Salleh for his suggestion and advise in the project by explaining some example for further understanding. Besides, he explain the steps in software development, analysis and design and even giving some software that is user friendly to most of us.

Lastly, I would like to express my deepest gratitude to my coursemates from Penang, Ipoh who had also given substantial contribution idea to success in producing this project proposal.

TABLE OF CONTENTS

LIST OF FIGURES

1.0 INTRODUCTION

Payroll Management System (PMS)

Payroll Management System is defined as the coordination and supervision of supply as well as storage and distribution and recording of materials (software / hardware / equipment) to maintain adequate quantities for current needs. If will provide users with critical information that will help user's to analysis their business and make informed product and purchasing decisions.

The Payroll Management System tracks calculation, amendment figures or type of report printed and returned form Human Resource Department. This system provides useful data. If will monitor and track payroll report at serial number level and maintain complete control over the database. Moreover, the system's ability to count and physical inventory ensures inventory integrity besides avoiding shortages and maximize the impact of out-of-stock conditions to keep us on track. In addition, this system could generate notification via email for items that is near or have passed their due dates.

1.1 Project Objectives

The main objective of the system are :

- Computerized the payroll management system in Human Resource Department.
- Provide an on-line system for users (management, head of department and staffs level) to access the system more easily.
- Reduce the manual work and save management, head of department, staffs time.
- To increase quality and accuracy in data keeping. Keep all payroll record information in the database for tracking purpose.
- Improve management control on payroll processing, returns and requisitions by supplying up-to-date information.
- Avoid data shortage, which is to maintain a good balance ratio between stock usage.
- Improve productivity of inventory control system and reduce mistakes or errors.
- To develop an attractive user interface that will help to increase and quicken the user usage.
- To enhance the security level of the system to prevent data from any modification from unauthorized users.

1.2 Project Scope

Payroll Management System is a static networking database system. It is used within organization between Head Office; Paka Station and Pasir Gudang Station. This system is useful to manage the up-data transaction for computer item such as basic salary; overtime claim; medical reimbursement; loan subsidies; EPF deduction of portion on employee and contribution from employee; Socso deduction of portion on employee and contribution from employee; PCB (Income tax) monthly deduction, etc.The scope of Payroll Management System is divided into 3 categories that are the payroll utilities; employee management centre; payroll transaction system.

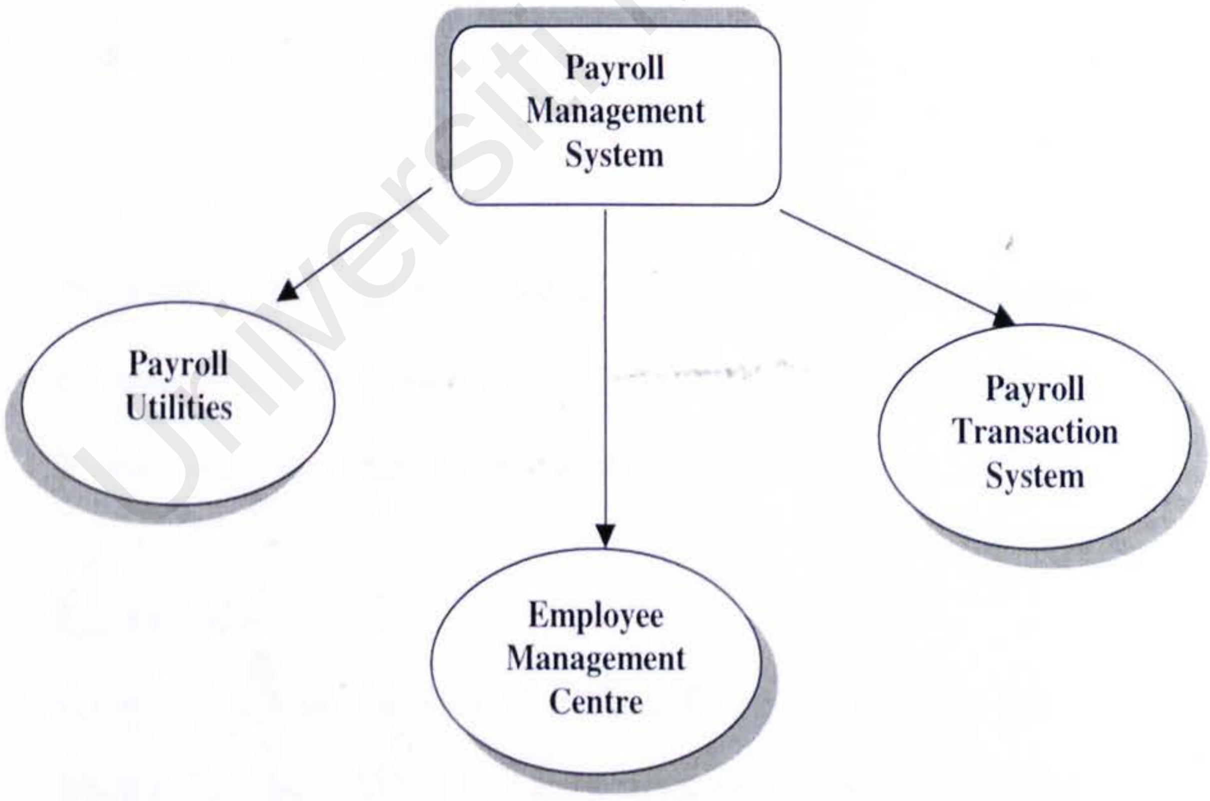

Payroll Management System
Payroll Utilities
Payroll Transaction System
Employee Management Centre

Figure 1.1 : Three main categories in Payroll Management System

1.3 Project Modules

Payroll Management System can be divided into 6 categories as below:

1.3.1 Authentication module

This module will be responsible with the system security with password setting, changing password and authentication checking to avoid data modification by unauthorized user.

1.3.2 Item Module

This module will store the information of the payroll itemize detail. This module allows adding, updating of deleting details of an item.

1.3.3 Loan module

This module will store the information of every type of loan such as study loan, car loan, housing loan, personal loan, etc. It will stores amount of loan need to be deducted and the balance after current month payroll processing.

1.3.4 Return module

This module will store the return information of the payroll transaction. The details from the loan from will be stored to return module when items returned.

1.3.5 Report module

This module will provides comprehensive reporting where users can choose to view their payslip if they want such as the figures that employee contribution or employer contribution on EPF, SOCSO or PCB Deduction for that current month is accurately. If not, they could inform Human Resource Department immediately and get the backpay in next month later.

1.3.6 Email notification module

This module allows the system could generate notification via email to the borrowers and management as a reminder for those items that are near or have passed their due dates.

1.4 Target User

The main target user for this project is the staff of Head Office, Paka Station and Pasir Gudang Station. There are two main parties in YTL Power Serives that will be using the payroll management system.

1.4.1 Low Level User

The low level users are personnel staff at Paka Station and Pasir Gudang Station.they have the authority only in viewing payroll information by department but that is not by individual. This authorized personnel could only viewing the payroll statement in total figures and not in details. They could able to print out the payroll statement report but they are not entitled to modify the date inside the payroll management system or key in transaction of any payroll transaction. The system itself wills automatic lock up the entire field when user login with low level user status. This is important to prevent users for changing some important information and data inside the payroll database.

1.4.2 High Level User of Administrator

The high level users are persons who have the authority to key in, modify, update information for all payroll transaction, delete relevant records, create records for new staff, change the statutory contribution which follow the government adhoc decision, restore the past record for submission to government department as per require. Besides that, they also have authority same as the low level users, for example viewing payroll statement, printing out their individual payslip. In shortly, they are entitling to fully using the whole payroll management system to fulfill their daily task.

1.5 Expect Strengths of Payroll Management System

Every system has the strengths and limitations. Below are some of the strengths of the system that are expected to be achieved as the payroll management system is fully completed and installed:

I. More attractive and interactive user interface to ease users usage in this system

II. Reduce log in errors

III. More easy to use without much completed functions

IV. This system will be much cheaper than other system in the market

1.6 Expected Limitations of Payroll Management System

Below are some of the expected limitations of the system as the payroll management system is fully completed are installed:

I. Must have a good response time

II. Need to have Internet access in certain parts

III. Communication link must be reliable

1.7 Expected Outcome

Before coming out with the expected outcome from this project, there are a few factors that need to be considered such as the amount of time to complete the project and the resources available. Below are some of the expected outcomes of the project:

I. System can perform some basic functions and meet some important criteria such as stability, consistency, reliability and user friendly.

II. The system will be able to show out the result of all the transaction flow.

III. The system provides a good database for keeping the current month payroll and employee records. The database will help users to keep track the records easier.

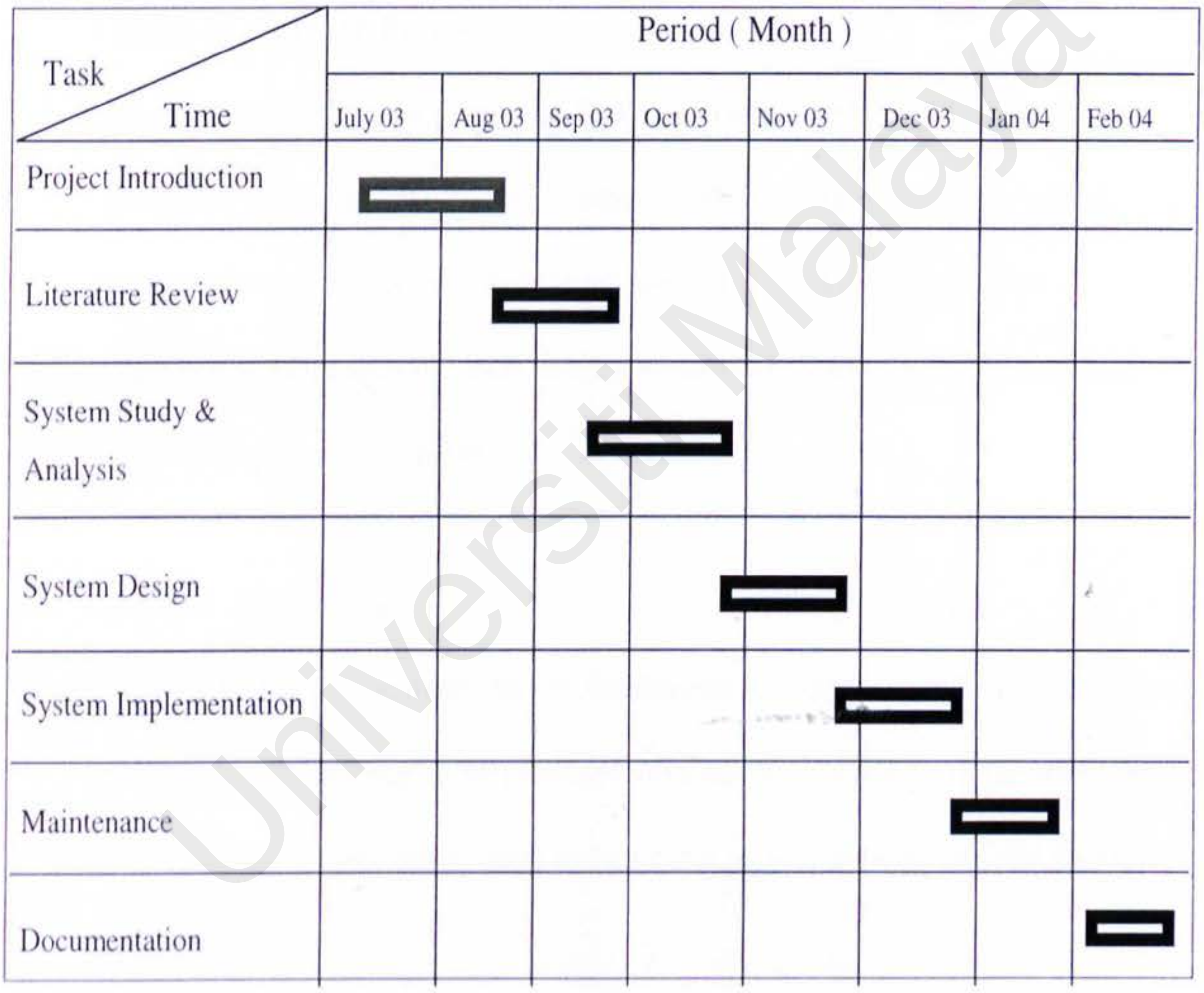
IV. Able to add security to the system so that the system can be only access by authorized users.

V. Provides an on-line system for staffs so that transaction can do through their desktop without going anywhere.

1.8 Project Schedule

In order to achieve the project on time, proper planning is essential to ensure smooth running of the process. The most crucial part of project planning is estimating the time it takes to complete each task or activity. Project scheduling specifies all the activities or discrete tasks involved in the software development cycle and the duration of time for each activity to be successfully implemented.

The schedule of this project is shown in the Gantt Chart below:



1.9 Report Layout

Chapter 1 : Introduction

It is an overview of the payroll management system project. The project definition objectives, scopes, target users, expected strengths, limitations and outcome of this project are being described and listed. Besides, the project schedule is shown in a chart.

Chapter 2 : Literature Review

This chapter describes the process of finding and related information about the system that intends to develop. Information about the current available systems, system tools review that are operating system, database server, application programming languages and tools that are needed to develop a web-based system will be discussed in this chapter.

Chapter 3 : Methodology

Chapter 3 highlights the methodology, mechanism and approach to be adopted. Then the quality and efficiency of the proposed tools will be described. All the software tools, operating system and strategy to tackle this project are analyzed and decided in this section.

Chapter 4 : System Analysis and Design

Chapter 4 focuses on the analysis of this project. Content of this chapter are analyze the user and system requirements such as functional requirements, non functional requirements, hardware and software requirements. Besides, it also will cover the payroll management system architecture design, database design, data flow and process / functional design as well as user interface design of the project. The various components of proposed system will be identified and explained. Expected outcome of this project can be view through the interface design.

Chapter 5 : System Implementation

This chapter will mention the software and hardware configuration, which are the basic preparation for the system development. It will also cover the interface development, database development, database connection and development of system documentation. Example of code will be shown to explain the development of the system.

Chapter 6 : System Testing and Coding

Every system must be tested before it be using. This chapter will show some essential testing such as Unit Testing, Integration Testing, System Testing and Fundamental Testing. The system maintenance will also be presented too.

Chapter 7 : System Evaluation and Conclusion

Evaluation is a process that occurs continuously at all phases of the system development. The last chapter will cover the system evaluation. Problems encountered and recommended solutions will be included.

Chapter 2 : LITERATURE REVIEW
YTL POWER SERVICES SDN BHD

2.0 Company Background

Malaysia economic growth created a dramatic rise in electricity demand. To address the situation, in 1990 the Government corporatised the National Electricity Board to form Tenaga Nasional Berhad – and paved the way for independent power producers (IPPS) to enter the market.

The first IPP licence was granted to the infrastructure group YTL Corporation Berhad. In November 1993, its subsidiary YTL Power Generation Sdn Bhd began construction of two power plants, with a combined generating capacity of 1212 MW, at Paka, Terengganu amd Pasir Gudang, Johor.

The turnkey contractor was a joint venture between Syarikat Pembenaan Yeoh Tiong Lay Sdn Bhd (SPYTL) and Siemens AG – one of the world's largest turbine manufacturers and power plant suppliers.

For the operation and maintenance of the plants, Siemens and YTL jointly established YTL Power Services – the first time Siemens had become directly involved in power plant O&M after construction, commissioning and handling over to the client.

YTL Power Services was incorporated in late 1993. During the construction phase the company rapidly built up its infrastructure – recruiting, developing procedures and systems, and training staff intensively. By October 1994, less than a year after the start of construction, the operation and maintenance teams had seen assembled and commercial production of electricity in open cycle began at both stations.

For its scale, the project was one of the fastest ever built. With just 22 months from start to finish, it set a world record even for Siemens. Such rapid progress added to the challenge of O&M mobilisation, but YTL Power Services kept pace with this performance, and began commercial operation seven months ahead of the original schedule.

2.1 O&M Experience

Since the start of commercial operation on October 1st 1994, YTL Power Services engineers and technicians have managed all O&M activities at both the Paka and Pasir Gudang power stations, operating them in open cycle mode prior to completion of the water/system cycle.

The plants are gas-fired using the CCGT (combined cycle gas turbine) concept for optimum efficiency and minimal environmental impact, and consist of three 404 MW generating blocks; two at Paka and one at Pasir Gudang. The main components of each block are:

- Two Siemens V94.2 gas turbines and generators;
- Two single pressure drum type heat recovery boilers;

- One Siemens E30-16-L1x6.3 steam turbine with generator.

The experience of YTL Power Services includes all the areas of competence involved in operating and maintaining such power plants.

- Recruitment and selection.
- Induction and training.
- Infrastructure development.
- Internal procedure development: health and safety, maintenance, engineering, administration.
- External procedure development: grid interconnection procedures, production schedules, fuel supply issues.
- Plant commissioning.
- Plant data gathering and analysis.
- Maintenance planning.
- Space parts planning and monitoring, receipt and storage.
- 24-hour base-load operation of gas and steam turbines, boilers, generators and ancillary equipment .
- Managing output disposition between both stations to meet targets and changing circumstances.
- Handling daily communications with grid company and fuel supplier.
- Corrective and preventive maintenance for electrical, mechanical and C&I equipment.
- Ensuring compliance with statutory requirements.
- Chemical engineering and water/stream chemistry.

- Environmental impact monitoring.
- Continuous strategic assessment of spare parts requirements.
- Continuous training needs analysis.
- Health and safely management.
- Evolution of maintenance strategy through Reliability Centred Maintenance analysis and Condition Monitoring.

2.2 Performance Record

A measure of the performance of YTL Power Services can be seen in the results from the Paka and Pasir Gudang plants over the first years of commercial service. As base-load stations they provide around the clock, so they must achieve high reliability and availability. The time availability achieved at both stations is good by industry standards.

Combined Cycle Operation				
	Time Availability (%)			
	Paka		Pasir Gudang	
	Combined Cycle	Gas Turbine	Combined Cycle	Gas Turbine
1997 -98	95.13	96.03	97.78	98.09
1998 - 99	91.88	94.77	96.50	97.28
1999 - 00	97.75	98.39	94.43	96.03
2000 - 01	94.11	95.75	96.21	97.78
Average	**94.65**	**96.19**	**96.12**	**97.22**

2.3 Quality

Since the formation of YTL Power Services, the company has recognized that Quality Assurance is not a diversion from its work, but an integral part of its proper management and vital to its success.

The QA function was built into YTL Power Services from the beginning. Throughout the build-up and development of the company, procedures and systems were designed in parallel with the evolving quality system.

In May 1995 YTL Power Services decided to apply for ISO 9002 registration. SIRIM, the Standards and Research Institute of Malaysia audited the company in 1995 and early 1996.

In June 1996 the quality system officially received the coveted ISO 9002 certification from SIRIM, making YTL Power Services the first power station O&M company in Malaysia to achieve company-wide registration.

Starting in 1998 the quality system has been changed continuously to comply with the newly developed ISO 9000 : 2000 on the basis of released drafts. As the result of this YTL Power Services became the first company in Malaysia to get certified to this new standard on 16 December 2000.

Now the company is looking beyond ISO 9000 with the goal of integrating all its business and technical activities by introducing TQM – Total Quality Management.

YTL Power Services is committed to the continuous review and improvement of the quality system so that it continuous to do its job effectively – contributing to YTL Power Services' aim to achieve the highest possible standards.

2.4 Maintenance Strategy

Reliability Centred Maintenance was adopted as a long-term maintenance strategy for the two power stations in 1995. All that time, the stations were in the final commissioning stages and just starting full-scale commercial operation. Even though this was an extremely busy time for the power station personnel it was decided to give RCM a high priority so that the principles could be established at an early stage and built into the station procedures.

The methodology of RCM analysis is to consider the function and failure modes of each item of equipment and to devise a maintenance strategy that is directed towards maintaining equipment only when it needs maintenance and maintaining only those items which need to be maintained.

The RCM programme was combined with a condition-monitoring programme to reinforce the data behind the analysis.

Three techniques are being used to monitor equipment conditions:

- Vibration monitoring
- Thermography
- Oil monitoring and analysis

The implementation of RCM and condition monitoring, largely carried out by our own staff, has led to increased plant availability, shorter duration shut downs and improved reliability of equipment.

2.5 Human Resource Department

2.5.1 Features of Human Resource Department

The high level of automation in modern power plants means that while staff numbers are relatively low, personnel must be highly skilled and versatile. YTL Power Services employs around 314 people, but all are chosen with careful regard to their experience and / or qualifications.

The great majority of staffs are Malaysian. Coming from a variety of technical background, they have been carefully selected to build up a multi-skilled and flexible team of O&M specialists, including:

- Experienced power plant operation personnel.
- Technicians with skills in troubleshooting and maintaining complex control and instrument systems.
- Machine specialists with operation and maintenance knowledge of turbines and other rotating equipment.
- Graduates highly-qualified in appropriate areas of engineering. There staff also form an investment in the long-term future of the company and the industry.

For the early years at Paka and Pasir Gudang, the company also assembled an international team of key engineers with in-depth experience of modern power plant. Their role has been to guide operations, and to facilitate technology transfer in order to build a highly-skilled, entirely Malaysian team.

In a competitive employment market, YTL Power Services has a very low staff turnover rate. The result: a high level of experience retention and team continuity. The company's personnel are predominantly young; they are in a position to grow with YTL Power Services to become Malaysia's PIP experts in years to come.

As a service company in a high-tech field, YTL Power Services recognises the critical importance of training. All staff have received intensive training in combined cycle gas turbine technology by Siemens, Technical staff assisted closely with plant commissioning to gain on-the-job training. Training is a continuous process. A Training Needs Analysis programme regularly monitors and address employees' training needs to ensure that they will continue to be well equipped for the job. Engineers and technicians are encouraged to apply for and trained for government issued Certificates of Competency, above and beyond the statutory requirements.

2.5.2 Objective of Human Resource Management system

Human Resource Strategic Business Consulting

- Strategic development and mission planning
- Business performance planning

- Diagnostic audits of management effectiveness
- Human Resource Audits
- Employee Attitude surveys
- Executive Search and recruitment
- Executive coaching and leadership development
- Corporate communications
- Outsourcing strategies
- Employee handbooks, policies and procedures

Organizational Development & Change Management

- Organizational analysis and design
- Career progression and succession planning
- Conflict resolution
- Coaching and counseling
- Team-building
- Change management
- Development of metrics and measures
- Individual skill development

Staffing Services

- Develop strategies recruiting solutions
- Conduct detailed reporting and measurement of recruiting including costs

per hire

- Recruitment advertising including internet/web-based sear
- Interview training
- Background and reference checking
- Employment application design

<u>Compensation & Rewards Programs</u>

- Analysis of existing compensation practices
- Design of job analysis techniques and job descriptions
- Design of global compensation programs including executive incentive compensation
- Market survey input and analysis
- Design of comprehensive compensation system including planning and salary structure
- Develop communication program and train managers

<u>Benefits Administration</u>

- Plan design and development of global benefits programs
- Conduct vendor searches, evaluations and implementation
- Integrate short-term disability, long-term disability and women compensation strategies
- Benchmark plan design, cost, quality and performance
- Develop communication plan

- Assistance with governmental fillings and reporting
- Claims audits
- COBRA administration

<u>Training & Development</u>

- Management training including Managing within statutory requirement, prevent sexual harassment, and Performance Management
- Needs assessment
- Benchmarking and metrics studies
- Managerial assessment and development programs
- Leadership training

<u>Performance Management</u>

- Design of performance management systems based on company objectives
- Design of performance appraisal forms
- Effective management and coaching training and assistance

<u>EEO/AAP Compliance</u>

- Organizational profile / workforce analysis
- Job group analysis
- Availability factor and analyses
- Utilization analysis
- Goal attainment report
- Adverse impact analysis

2.6 Malaysia Employment Law

2.6.1 Definition of Employment

Employment means work or service performed by an individual to the task at hand for another person or entity in exchange for wages or other remuneration.

2.6.2 Employment Law

Employment Law is the law which regulates the operation of the labour market in general and the employment relationship between employers and employees in particular. Examples include hiring process, suspension from work, maternity rights, layoff and wages.

The obligations and rights of an employment contract are covered by the Employment law.

When an offer for employment is made by an employer to an employee, the law governing the relationship between an employee and an employer begins.

2.6.3 Basic Terms Used To Describe The Employment Relationship

An employer in relation to an employee or a workman, means a person by whom the employee or workman is employed. An employer obtains the services of another to perform work and has direct control of the way in which the work is to be done.

An employer shall provide the means through which the services will be performed such as providing a place where the work is to be performed and tools required to get the job done.

Contract of service means any agreement, whether express or implied, and if express, whether oral or in writing where the employer agrees to employ and the employee agrees to be employed and includes an apprenticeship contract.
In general, an employee means a person who has entered into or works under a contract of employment. There are different interpretations for the term, employee, in different Acts governing the labour market in Malaysia. Those interpretations will determine whether you are an employee protected under the scope of an Act.

Where an employee begins employment with an employer and for a period of more than one month, such employee must be given a written contract of employment with particulars of the terms and conditions of employment including the notice period required to terminate the contract of employment

2.6.4 What Are The Laws Governing Employment In Malaysia?

The principal legislation governing the labour market and employment relationship in Malaysia is the Employment Act 1955. However, the application of these rules to Sarawak and Sabah references made under the Act shall be substituted by references to the Sarawak Labour Ordinance (Cap. 76) and Sabah Labour Ordinance (Cap. 67) or other written laws in force in Sarawak or Sabah, as the case may be.
Some other legal regulations include :

- Pensions Act 1980

  For the administration of pensions, gratuities and other benefits for officers in the public service and their dependants.

- Employees Social Security Act 1969 (ESSA)

  For social security protection to all employees and their dependants as well as the employers. This Acts is administered by the Social Security Organization (SOCSO) or Pertubuhan Keselamatan Social (PERKESO), Malaysia.

- Employees Provident Fund Act 1951

  For the provision of financial security to its members particularly after retirement, through a compulsory savings scheme. This Act is administered by the Employees Provident Fund (EPF) or Kumpulan Wang Simpanan Pekerja (KWSP), Malaysia.

- Occupational Safety and Health Act 1994 (OSHA)

  For the safety, welfare and health of persons of workplaces or in the operation of high risk machinery against risks to safety or health. This Act is administered by the Department of Occupational Safety and Health or Jabatan Keselamatan dan Kesihatan Pekerjaan, Malaysia.

- Private Employment Agencies Act 1981

  This Act is administered by the Manpower Department, Ministry of Human

Resources or Jabatan Tenaga Rakyat, Kementerian Sumber Manusia, Malaysia.

- Human Resources Development Act 1992

  This Act is administered by the Human Resources Development Council or Majlis Pembangunan Sumber Manusia, Malaysia.

- Factories and Machinery Act 1967

  This Act is administered by the Department of Occupational Safety and Health or Jabatan Keselamatan Dan Kesihatan Pekerjaan, Malaysia.

- Petroleum (Safety Measure) Act 1984

  This Act is administered by the Department of Occupational Safety and Health or Jabatan Keselamatan Dan Kesihatan Pekerjaan, Malaysia.

- Trade Unions Act 1959

  This Act is administered by the Trade Union Affairs Department or Jabatan Hal Ehwal Kesatuan Sekerja, Malaysia.

- Workmen's Compensation Act 1952

  For compensation to foreign workers injured in the course of their employment and to worker's dependents in the event of fatal accident. This Act is administered by the Department of Labour or Jabatan Buruh - Peninsular Malaysia, Sabah and

Sarawak.

- Industrial Relations Act 1967

  An Act which governs the relationship between employers and workmen or employees and their trade unions and generally deal with trade disputes. This Act is administered by the Industrial Relations Department or Jabatan Perhubungan Perusahaan, Malaysia.

- Wages Council Act 1947

  This Act is administered by the Department of Labour or Jabatan Buruh - Peninsular Malaysia, Sabah and Sarawak.

- Employment Information Act 1953

  This Act provides the Department of Labour with power to obtain and collect information and data on employment, terms and conditions relating to an employment, from any industries in the private sector. This Act is administered by the Department of Labour or Jabatan Buruh - Peninsular Malaysia, Sabah and Sarawak.

- Employment (Restriction) Act 1968

  This Act is administered by the Department of Labour or Jabatan Buruh Peninsular Malaysia.

- Worker's Minimum Standards of Housing and Amenities Act 1990

  This Act is administered by the Department of Labour or Jabatan Buruh Peninsular Malaysia.

- Weekly Holidays Act 1950

  This Act is administered by the Department of Labour or Jabatan Buruh Peninsular Malaysia.

- Children and Young Persons (Employment) Act 1966

  This Act is administered by the Department of Labour or Jabatan Buruh Peninsular Malaysia.

2.6.5 Whom May Employ

You can employ almost any local citizens. A person whom you employ under a contract of service is an employee.

In Peninsular Malaysia, only employees as being defined in the Employment Act 1955 is protected under the Act. In Sabah and Sarawak, an employee is defined in the Labour Ordinance (Sabah Cap. 67) and Sarawak Labour Ordinance (Cap. 76) respectively.

An employee under the Employment Act 1955 means :

- any person whose wages does not exceed RM1,500 per month under a contract of service with an employer
- any person who irrespective of the wages earns in a month has entered into a contract of service with an employer and disengaged in

- manual labour
- engaged in the operation of mechanically propelled vehicles

■ one who supervises and oversees employees in manual labour

■ any person engaged in any capacity in any vessel registered in Malaysia with certain exception

■ An amendment has been made to the Act on 1st August 1998 to include employee earning between RM1,500 and RM5,000 per month. However, these persons are only covered in terms of wages, allowances or other cash benefits.
Any complaints on employment shall be made by an employee to the nearest Department of Labour Office or Jabatan Buruh.

Where there are disputes such as unfair dismissal, employees covered and not covered under these Acts or Ordinance can voice their complaint to the Industrial Relations Department.
You are not permitted to employ women to carry out underground, industrial and agricultural work between 10:00 PM to 5:00 AM without exemption from the Directory General of Department of Labour.

Subject to certain requirements, you can only employ foreign workers in the plantation and manufacturing sectors. Only nationals of Cambodia, Indonesia, Philippines, Sri Lanka, Thailand, Bangladesh and Pakistan are permitted to be employed in these sectors.
A foreign company is only permitted to employ expatriate personnel in areas where there are lack of trained local to do the job under certain requirements.

A company with foreign paid-up capital of US$2 million and above will automatically be allowed 5 expatriate posts including key posts. Where such company requires additional expatriate posts, a request has to be made and posts requested will be given when necessary.
A company with foreign paid-up capital of less than US$2 million will be considered for expatriate posts on certain basis.

2.6.6 Termination Contract of Service by Either the Employer or Employee

Where a contract of service is considered broken, an employer can dismiss an employee. A contract of service is considered to have been broken when an employee has been absent from work for more than 2 consecutive working days without prior leave from the employer or without informing or attempting to inform the employer at the earliest opportunity during such absence with reasonable excuse.

An employer may terminate the contract of service where the employee is found guilty of misconduct, misdemeanor or negligence.
An employee has the right to terminate the contract of service, where an employer fails to pay wages within seven days after the wages period.

A contract of service can also be terminated without notice :

■ by paying to the other party or indemnity in lieu of notice

- if there is a willful breach by the other party of a term or condition of the contract of service

Where the contract of service has expired or work being completed, the contract may also be terminated. Written notice being given by either party may also terminates a contract of service.

2.6.7 The Notice Period Required to Terminate A Contact of Service

An employee may resign by giving notice of resignation or termination to the employer to terminate the contract of service. An employer may also dismiss an employee by giving notice of termination to such employee. In both situations, the length of notice shall be the same pursuant to the contract of service.

Where the period of notice of termination is not specified in the contract of service, the notice period shall be as follows :

- less than 2 years of service - minimum 4 weeks
- 2 years or more but less than 5 years of service - minimum 6 weeks
- 5 years of service or more - minimum 8 weeks

2.6.8 Other Than Termination, What Actions Can Am Employer Takes Against An Employee On The Grounds Of Misconduct On the Part Of The Employee ?

Where an employee is found guilty of misconduct by an employer, the employer may take the following actions :

- downgrade the employee

- impose any other lesser punishment as the employer considers just and fit

2.7 Employees Provident Fund (EPF)

Employee Provident Fund is a compulsory savings scheme in Malaysia. Its primary aim is to provide a measure of security for old age retirement to its members. It also provides supplementary benefits to members to utilize part of their savings for house ownership and other withdrawal schemes.

EPF is the abbreviation for Employees Provident Fund. Employees Provident Fund is commonly known in the Malay term as KWSP or Kumpulan Wang Simpanan Pekerja.
Employees Provident Fund Act 1991 is the act governing the Employees Provident Fund in Malaysia. This Act is administered by the Employees Provident Fund, Malaysia.

2.7.1 EPF Statutory Contribution

All employees in Malaysia who have reached the age of 16 and employed under a contract of service whether express or implied, and whether oral or in writing must be registered as a member of the Employees Provident Fund.

An employer will contribute 12% of the employee's wages and the employee contributes 11% of the monthly wages towards the employee's account.

Prior to 1st August 1998, expatriates and foreign workers were not required to contribute to the EPF although they may elect to do so.

However, with effect from 1st August 1998, all foreign workers and expatriates earning less than RM2,500 per month are also required to contribute to EPF with the exception of certain categories.

Those who are exempted from making the compulsory contribution are

- employees or workers holding Employment Pass or expatriates holding Visit Pass (Temporary Employment) whose monthly wages is not less than RM2,500
- Thai workers who enter Malaysia with a Territorial Pass
- Seamen
- Foreign domestic maids
- Self-employed persons
- Out-workers who do cleaning and alteration repair works
- Persons detained in custody, in prison, Henry Gurney School and mental hospital
- Pensioners

Nevertheless, the above can choose to contribute to the fund.

Where a member continues employment after withdrawing the contributions upon retirement, such member may opt to continue contributing to the EPF by submitting the KWSP 20/20A Form.

The statutory rates of contributions are as follows :

Type of Employee	**% of contributions of employees' wages**	
	Employer	Employee
All except expatriates and foreign workers	**12%**	

Employers and employees are, however, allowed to elect to contribute at higher rates.

2.7.2 Registration Procedure of an Employee with EPF

Employers must register their employees with the EPF within 7 days of employment under law.

Under section 41(2) of the EPF Act 1991, an employer who contravenes the above shall be found guilty of an offence and shall be liable, on conviction to imprisonment for a term not exceeding 3 years or fine not exceeding RM10,000 or to both.

An employer shall register the company or firm with the EPF by submitting the KWSP 1 Form. This can be obtained from the nearest EPF branch office.

Thereafter, for each employee, the employee and the employer is required to complete the KWSP 3 (AHL) Form. Generally, an employee will also be required to submit the Nomination Form KWSP 4 (AHL) which is attached together with the KWSP 3 (AHL) Form.

Once the application is approved, the employee will be sent a Membership Card. The EPF contribution by employer and employee shall commence on the first month of salary payment pursuant to section 45(2) of the EPF Act 1991.

2.7.3 Employer Contribution to EPF

An employer shall before the end of the first week in the first month in which he is paying required to pay contribution to the Employees Provident Fund.
An employer shall prepare and furnish a statement of wages to each employee.
An employer who fails to make contributions to EPF shall be guilty of an offense and shall be liable on conviction to imprisonment for a term not exceeding 3 years or to a fine not exceeding RM10,000 or to both.

2.8 Social Security Organization (SOCSO)

The Social Security Organization is an organization set up to administer, enforce and implement the Employees' Social Security Act, 1969 and the Employees' Social Security (General) Regulations 1971.

The Social Security Organization provides social security protection by social insurance including medical and cash benefits, provision of artificial aids and rehabilitation to employees to reduce the sufferings and to provide financial guarantees and protection to the family.

SOCSO is the abbreviation for Social Security Organization. It is commonly known in the Malay term as PERKESO or Pertubuhan Keselamatan Sosial.

2.8.1 Who is compulsory to contribute to SOCSO?

An employee employed under a contract of service or apprenticeship and earning a monthly wages of RM2,000 and below must compulsorily register and contribute to SOCSO regardless of the employment status whether it is permanent, temporary or casual in nature.

An employee must be registered with the SOCSO irrespective of the age. SOCSO only covers Malaysian workers and permanent residents. As a result, foreign workers are protected under the Workmen's Compensation Act 1952.

Nevertheless, SOCSO does not cover the following categories of persons :

- A person whose wages exceed RM2,000 a month and has never been covered before.
- Government employees.
- Domestic servants employed to work in a private dwelling house which includes a cook, gardeners, house servants, watchman, washer woman and driver.
- Employees who have attained the age of 55 only for purposes of invalidity but if they continue to work they should be covered under the Employment Injuries Scheme.
- Self-employed persons.
- Foreign workers.

2.8.2 The purpose of SOCSO contribution due to wages

For the purpose of SOCSO contribution, wages mean all remuneration payable in money to an employee. The following payments are considered as wages :

- salary
- overtime payment
- commissions and service charge
- payment for leave, sick, annual, rest day, public holidays, maternity and others
- allowances, shift, incentive, housing, food, cost of living and others.

Payments made to an employee paid at an hourly rate, daily rate, weekly rate, task or piece rate are also considered as wages.

However, the following payments are not considered as wages :

- payments by an employer to any statutory fund for employees
- mileage claims
- gratuity payments or payments for dismissal or retrenchment
- annual bonus.

2.8.3 The coverage provided to an insured person by SOCSO under ESSA 1969

An insured person or dependants will be entitled to the following benefits :

- Periodical payments in the case of invalidity
- Periodical payments in the case of disablement suffered as a result of an employment injury

- Periodical payments to the dependants of an insured person who dies as a result of an employment injury
- Payments for funeral benefit or expense on the death of an insured person as a result of an employment injury
- Periodical payments to an insured person who is in receipt of invalidity pension or disablement benefit and is so severely incapacitated or disabled as to require the personal attendance of another person
- Medical treatments for the attendance on insured persons suffering from disablement
- Periodical payments to dependants of an insured person who dies while in receipt of invalidity pension

SOCSO provides coverage to eligible employees through 2 schemes namely

- Employment Injury Insurance Scheme
- Invalidity Pension Scheme.

These schemes are classified into 2 categories :

- First Category - Employment Injury Insurance Scheme and Invalidity Pension Scheme. The contribution payment is made by both the employer and employee
- Second Category - Employment Injury Insurance Scheme Only. The contribution is paid by the employer only. An employee who is not eligible for coverage under the Invalidity Pension Scheme is protected under this category.

These schemes provide the benefits of invalidity pension, invalidity grant, survivors pension, rehabilitation, funeral benefit, constant attendance allowance and educational loan.

2.8.4 The invalidity pension scheme cover and the benefits provided

Invalidity Pension Scheme provides a 24-hours coverage to employees against invalidity and death due to any cause not connected with employment before the age of 55 years.

The benefits provided under this scheme are Invalidity Pension, Invalidity Grant, Constant Attendance Allowance, Survivors Pension, Funeral Benefit, Rehabilitation and Educational Loan.

- Invalidity Pension

  For the purpose of Invalidity Pension, invalidity means a serious disease or disablement of a permanent nature that is either incurable or not likely to be cured, as a result of which an employee is unable to earn at least 1/3 of what a normally able person could earn.

  Heart attack, kidney failure, cancer, mental illness, chronic asthma and other similar conditions are chronic ailments or diseases that could be considered for invalidity.

  The following conditions must be fulfilled by an employee to be eligible for Invalidity Pension :

  - at the time the notice of invalidity is received, the employee has not completed the age of 55 years

- if the employee has completed the age of 55 years when the notice of invalidity is received, such employee has to provide proof that the invalidity occurred before 55 years and he had ceased employment at that time.
- certified as an invalid by a Medical Board or Appellate Medical Board
- has fulfilled the contribution qualifying conditions.

There are 2 contribution qualifying conditions:

- full qualifying condition
- reduced qualifying condition.

An employee is deemed to have fulfilled the conditions of full contribution qualification if

- before the month in which the notice of invalidity is received, an employee's monthly contributions within a period of 40 consecutive months must be at least 24 months.
- an employee has made monthly contributions for at least 2/3 of the number of full months in the period between the date of first coverage under the Invalidity Pension Scheme and the date the notice of invalidity is received by SOCSO. This is subject to the condition that the total number of monthly contributions made during the stated period, is at least 24.

- Invalidity Grant

This is an outright payment paid to worker or employee who does not qualify for the Invalidity Pension, as he does not meet any of the contribution qualifying conditions stated, but has made at least 12 monthly contributions. The Invalidity Grant is equivalent to the total amount of contributions paid by the employee and the employer for the Invalidity Pension Scheme including the interest thereof.

If an employee is severely incapacitated and requires constant personal attendance, the recipient of Invalidity Pension is also entitled to Constant Attendance Allowance. The Medical Board or an Appellate Medical Board will decide on the eligibility to receive this allowance and will pay the recipient of the benefit directly. Subject to a maximum of RM500 per month, the benefit is 40% of the rate of Invalidity Pension.

- Survivors Pension

Where an employee dies in any of the following situations, irrespective of the cause of death, the dependants will be paid Survivors Pension :

- while an employee is receiving invalidity pension irrespective of his age
- an employee who is not a recipient of the invalidity pension and has not reached the age of 55 years but met either the full contribution qualifying condition or the reduced contribution qualifying condition.

Where the deceased is a recipient of Invalidity Pension, the rate of the Survivors Pension is equivalent to the rate of the Invalidity Pension received by him.

Where the deceased is not a recipient of the Invalidity Pension and has met the full contribution qualifying conditions, the full rate of the Survivor's Pension is between 50% to 65% of the average monthly wage depending on the number of contributions made in his behalf.

The rate of Survivors Pension will be 50% of the average monthly wage for an employee who meets the reduced contribution qualifying condition. Dependants who are entitled for the pension are the same as those under the Dependent Benefit of Employment Injury Scheme.

- Funeral Benefit

  This benefit is paid to the eligible next-of-kind if an employee dies :

  - while receiving Invalidity Pension
  - before reaching the age of 55 but meets the full or reduced qualifying contribution conditions.

The amount and the persons qualified to receive this benefit are the same as those under the Employment Injury Scheme

- Rehabilitation Benefit

  An employer who suffers invalidity is also entitled to receive Rehabilitation Benefit as provided under the Employment Insurance Injury Scheme

The total contribution for the Invalidity Pension Scheme is about 1% of the wages of an employee and is shared by the employer and the employee equally.

2.8.5 The rate of SOCSO Contribution

The principal employer must make a monthly contribution for each eligible employee according to the rates specified under the Act.

The employer pays 1.75% for the Employment Injury Insurance Scheme and the Invalidity Pension Scheme while the employee's share of 0.5% of wages should be paid for coverage under the Invalidity Pension Scheme.

The rate of contribution is based on the monthly wage of the employee in accordance to 24 categories.

Wage	(First Category) Employment Injury & Invalidity			(Second Category) Employment Injury Only
	Employer Share	Employee's Share	Total (RM)	Employer Contribution Only
Up to RM30	0.40	0.10	0.50	0.30
Exceeds RM30 but does not exceed RM50	0.70	0.20	0.90	0.50
Exceeds RM50 but does not exceed RM70	1.10	0.30	1.40	0.80
Exceeds RM70 but does not exceed RM100	1.50	0.40	1.90	1.10
Exceeds RM100 but does not exceed RM140	2.10	0.60	2.70	1.50
Exceeds RM140 but does not exceed	2.95	0.85	3.80	2.10

RM200				
Exceeds RM200 but does not exceed RM300	4.35	1.25	5.60	3.10
Exceeds RM300 but does not exceed RM400	6.15	1.75	7.90	4.40
Exceeds RM400 but does not exceed RM500	7.85	2.25	10.10	5.60
Exceeds RM500 but does not exceed RM600	9.65	2.75	12.40	6.90
Exceeds RM600 but does not exceed RM700	11.35	3.25	14.60	8.10
Exceeds RM700 but does not exceed RM800	13.15	3.75	16.90	9.40
Exceeds RM800 but does not exceed RM900	14.85	4.25	19.10	10.60
Exceeds RM900 but does not exceed RM1,000	16.65	4.75	21.40	11.90
Exceeds RM1,000 but does not exceed RM1,100	18.35	5.25	23.60	13.10
Exceeds RM1,100 but does not exceed RM1,200	20.15	5.75	25.90	14.40
Exceeds RM1,200 but does not exceed RM1,300	21.85	6.25	28.10	15.60
Exceeds RM1,300 but does not exceed RM1,400	23.65	6.75	30.40	16.90
Exceeds RM1,400 but does not exceed RM1,500	25.35	7.35	32.60	18.10
Exceeds RM1,500 but does not exceed RM1,600	27.15	7.75	34.90	19.40
Exceeds RM1,600 but does not exceed RM1,700	28.85	8.25	37.10	20.60
Exceeds RM1,700 but does not exceed RM1,800	30.65	8.75	39.40	21.90
Exceeds RM1,800 but does not exceed RM1,900	32.35	9.25	41.60	23.10
Exceeds RM1,900	34.15	9.75	43.90	24.40

2.8.6 The responsibilities of an Employer

An employer must cover their employees even if the employees have other private insurance coverage.

A company or firm with one or more employees whose individual earnings do not exceed RM1,000 a month has to register with SOCSO. The contributions for such employees are borne solely by the employer.

Where an employee reached the age of 55 and continues to be employed after that age, only the employer shall contribute to SOCSO for such employees.

Under the Employees' Social Security Act 1969, it is the duty of an employer to make contribution to the SOCSO on behalf of the employees to insure them against employment injury and the contingencies of invalidity.

It is the liability of the principal employer to ensure all employees employed by the immediate employer have been registered and their contributions have been paid.

Under the Employees' Social Security Act 1969, the principal employer will be liable, in the event the employees have not been registered and the immediate employer cannot be located.

2.9 Inland Revenue Malaysia

Lembaga Hasil Dalam Negeri Malaysia (LHDN)

FORMULA FOR CALCULATION OF MONTHLY TAX DEDUCTIONS

STEP ONE
Determine the relevant employee's *CATEGORY* in accordance with the *SCHEDULE.*

STEP TWO
Calculate the employee's CHARGEABLE INCOME (P) in the following manner:-
CATEGORY 1 : P = [(Total Monthly Remuneration - *EPF) X 12] - RM8,000.00;

CATEGORY 2 : P = [(Total Monthly Remuneration - *EPF) X 12] - (No. Of Children X RM800.00) - RM11,000.00;

CATEGORY 3 : P = [(Total Monthly Remuneration - *EPF) X 12] - (No. Of Children X RM800.00) - RM8,000.00;
*(*EPF is restricted to maximum RM416.00 per month)*

STEP THREE
The monthly deduction is calculated by using the following formula, where:-
i) Remuneration *RM 10,000 or below:*

$$\left[\frac{(P - M) \times R + B}{12} \right] \times 0.8$$

ii) Remuneration is *MORE than RM 10,000:*

$$\left[\frac{(P - M) \times R + B}{12} \right]$$

Where the value of M, R, and B are determined in accordance with the following table:-

NOTE
i. The calculated value should be rounded up to the *nearest* Ringgit.
ii. Deduction is not required if the amount is *less* than RM20.00.

2.9.1 The rate of taxable income by categories

P	M	R	B [CATEGORY1 & 3]	B [CATEGORY 2]
(RM)	(RM)	(%)	(RM)	(RM)
2,500 - 5,000	2,500	1	-350	-700
5,001 - 10,000	5,000	3	-325	-675
10,001 - 20,000	10,000	5	-175	-525
20,001 - 35,000	20,000	9	325	-25
35,001 - 50,000	35,000	15	2,025	2,025
50,001 - 70,000	50,000	20	4,275	4,275
70,001 - 100,000	70,000	25	8,275	8,275
100,001 - 150,000	100,000	28	15,775	15,775
Above 150,001	150,000	29	29,775	29,775

2.10 Purpose of Computerized Payroll Management System

This review of literature describes the finding of various researches and technology in the effectiveness of payroll management system. The idea, knowledge and experience gained during the survey will be used in the development of payroll management system. Various good and relevant features are to be noted during the survey, particularly, the design and interface methods used by various systems.

The purpose of this section is to review the techniques that are used to collect information about the system requirements and the knowledge gained to develop this project. It is to evaluate and to analyze an existing system that has the similar concept of this system if one exists. Evaluation will help in determining the strength and weaknesses of the existing system and measures can be taken to improve the weaknesses. The information will provide a new approach and direction to determine the relevant requirements for the project and lead in producing the best solution.

It is also done to get a better understanding on the development tools that can be applied and also a better knowledge on the methodologies used in the process of developing it. This attempt does not only stop until this project where it might be helpful in the future project where it might be helpful in the future project or in the working world that applies the same system.

Thus, the literature review provides pertinent information and validity to the researches and the environment will be necessary to determine and implement the best solutions.

2.11 Current Available System

2.11.1 Manual key-in report

Human Resource's staff, personnel staff from Paka Station, personnel staff from Pasir Gudang Station used the manual documentation to manage the payroll calculation, overtime claim, deduction staff loan, key-in the calculation which followed the circular from EPF, SOCSO and Income Tax for each individual employee number. Head of department with first go to the Human Resource Department or check with the personnel staff in-charge to submission their staff attendance and overtime claim. Personnel staffs who in-charge of payroll need to compile all submission from various departments, do checking on overtime calculation, reimbursement medical claim and key-in according to the staff ascending number.

In the conventional way of transaction, there are a lot of problems that has emerged such as inefficiency, inaccuracy, insecurity, delay and others. Management needs to check the payroll print out manually in order to know its errors. This manual job difficult to do so as it would consume a lot of time and paperwork. Besides, it also takes borrower's time to wait the confirmation and approval from signatory. The payroll personnel staff has to key in all the payroll transaction manually in details for payroll report and update it when transaction occur intent to keep track of the payroll report. Payroll reports that is record in paperwork will cause mistakes or error thus brings wrong calculation. Figure 2.1 is shown data flow diagram in the existing manual system.

Figure 2.1 : Data Flow Diagram in Manual System

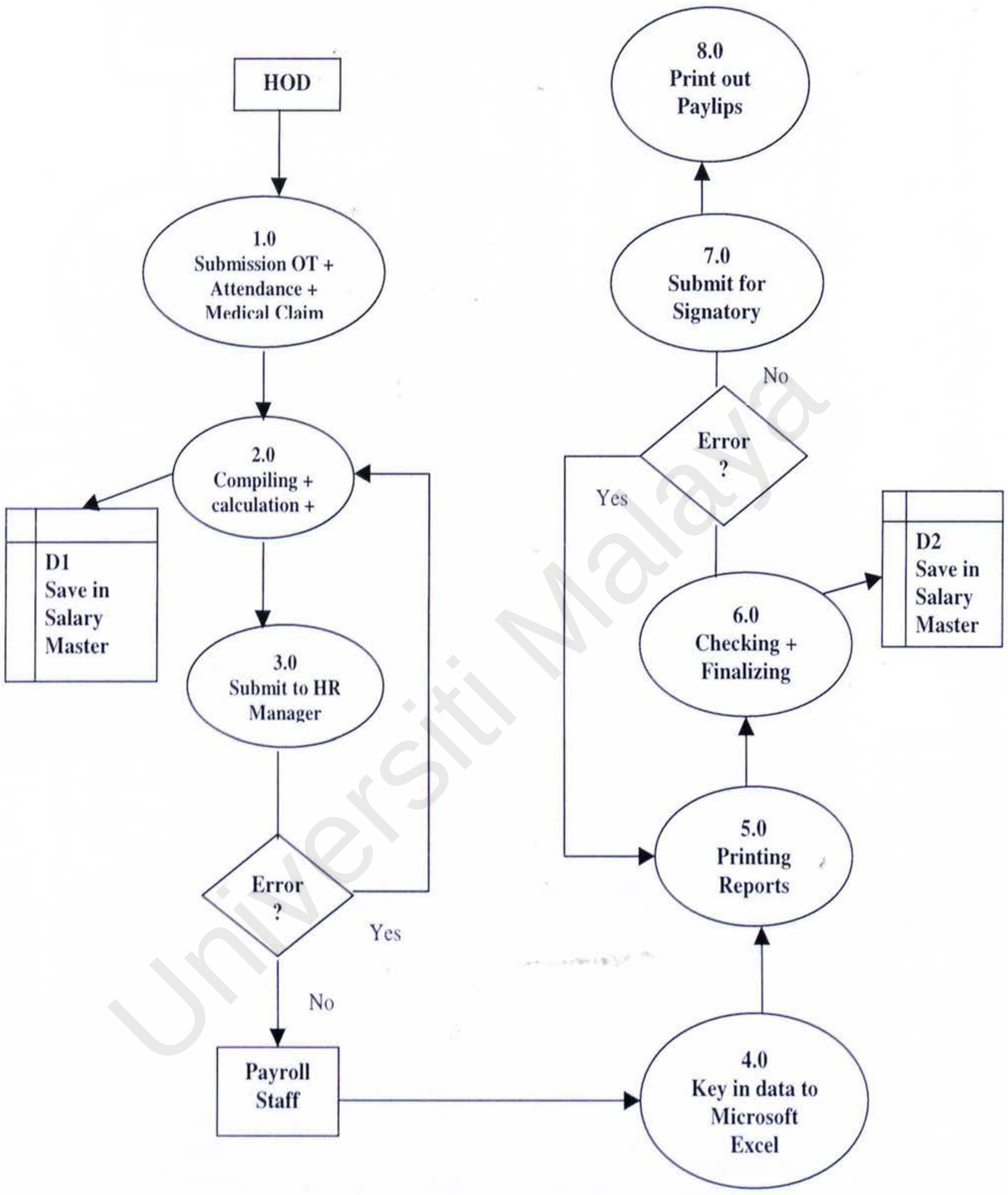
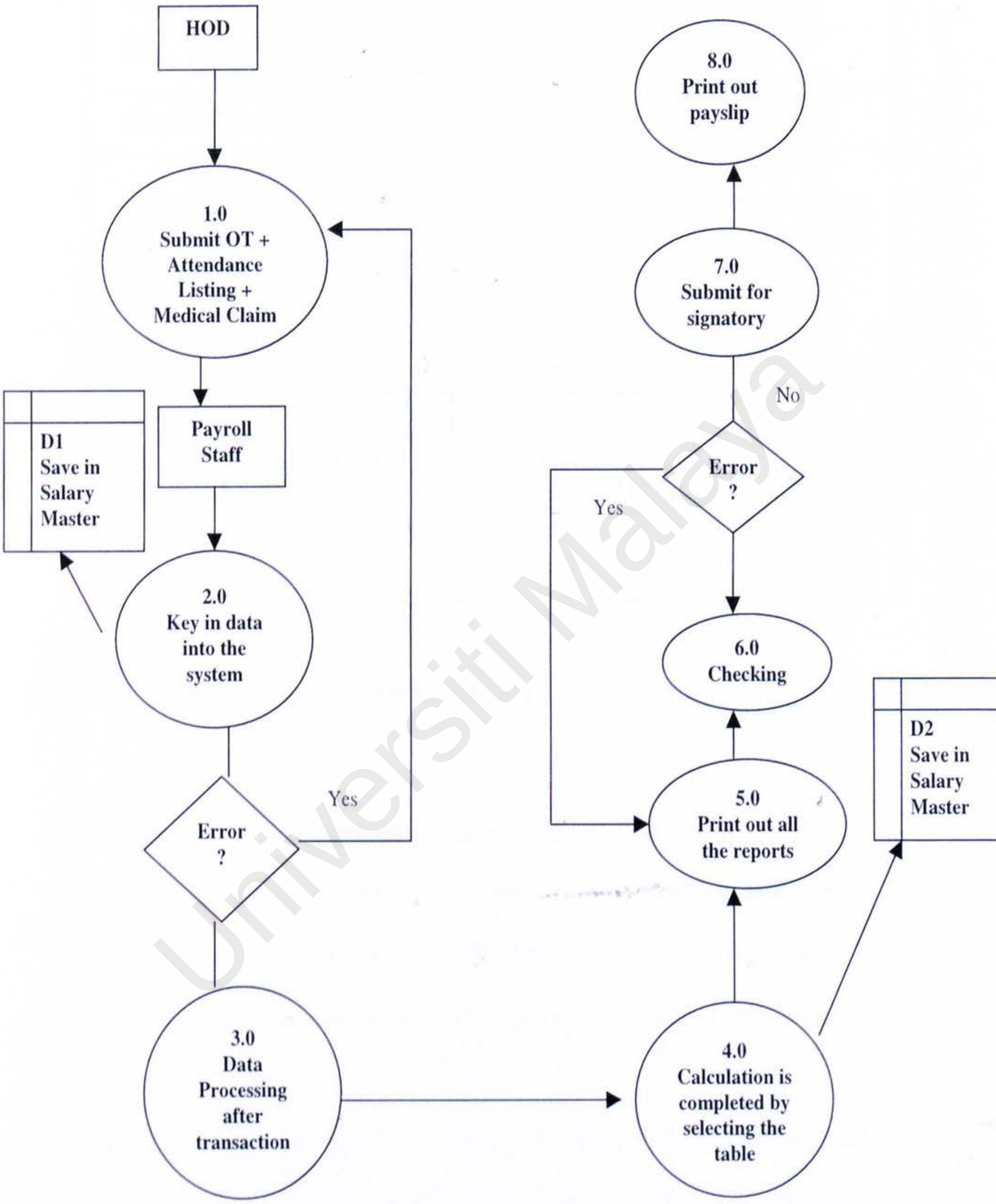


Figure 2.2 : Data Flow Diagram in Computerized System


HOD
1.0 Submit OT + Attendance Listing + Medical Claim
Payroll Staff
D1 Save in Salary Master
2.0 Key in data into the system
Error ?
Yes
3.0 Data Processing after transaction
4.0 Calculation is completed by selecting the table
D2 Save in Salary Master
5.0 Print out all the reports
6.0 Checking
Error ?
Yes
No
7.0 Submit for signatory
8.0 Print out payslip

YTL Power Services Sdn Bhd

END-MAR'03

Name :	Asmah Binti Salleh	SOCSO :	P8290982X
NRIC :	710102-10-5316		
EPF :	12339269		
EARNING	RM	DEDUCTION	RM
Basic Salary	2500.00	Employee EPF	225.00
		Employee SOCSO	9.75
Total Earning	2500.00	Total Deduction	234.75
Net Wage	2265.25	T-T-D	
E'R EPF	300.00	Gross Pay	5000.00
E'R SOCSO	34.15	E'E/E'R EPF	450.00/ 600.00
		E'E/E'R SOCSO	19.50/ 68.30

Figure 2.3 : A Sample of existing paylip by using Microsoft Excel

2.11.2 Comparison between manual system and computerized system

i. The manual system is not well organized compared to computerized system

ii. The manual system less efficient in identify the uncompleted payroll report unlike the computerized system just has to view at the screen with latest update.

iii. Manual system prove to human error since payroll personnel staff has to write in particular while computerized system able to alert to the users if there is any error made by them.

iv. Manual system waste payroll personnel staff to do a lot of paperwork unlike computerized system can share data with other modules.

v. A systematic recording will easier the control on the payroll thru computerized system.

vi. Computerized system can provide facilities the printing or viewing of reports effectively.

vii. Data flow diagram in Figure 2.2 shown all the process could be finished by one person.

2.11.3 Summary of the existing system

From the analysis that have been done on some of the existing system, a conclusion have been made regarding these existing system:

i. Most of the existing systems are more on payroll transaction control usage.

ii. Most of these systems are written using Visual Basic

iii. Most of these posed a poor graphical user interface design

iv. Some of these systems are very complicated to use by provides a complete function

v. Most of these system support many payroll transaction items

vi. Only a few of these system proposed a flexible searching

vii. Most of these systems provide comprehensive reporting

viii. Not many of these system support multiple branches function

ix. Staff involved in Figure 2.1 should need minimum 2 staff to finish it

2.12 Information Sources

Fact-finding techniques are important to determine the right requirements for the project.

A few research criteria have been set as the base to obtain all the necessary information for this payroll management system. This set of criteria is laid out so that the amount of information obtained would not get out of hand and would always stay on track with the project. As a result, the following research techniques have been chosen.

2.13 Analyzing the existing documentation

The existing documentation concerning permanent staff, staff registration number and kin family data, which are supposed to be carried out were analyzed to determine the required information, needed to create a database information system.

2.14 Research and Site Visits

Analyzing the existing staff database information system on the web. A study on the attractiveness of the interfaces was done. This technique helps to determine the appropriate software and architecture that is needed to build the system. Information can be obtained from system users, computer programs, procedure manuals and reports forms and documents and also directly from the Internet.

2.15 Interviews

Interviewing users is not probably the most direct method for getting information since opinions can be directly collected from people who are interested in using the system. Interview helps on finding facts, verify facts, clarify facts and identifying requirements. However the conversation between the interviewer and the interviewee may be unstructured and relatively long. As a result, the number of users that could be interviewed is limited.

2.16 Documentation Room

Existing documentation are taken as a research material for this project. This is because previously documented systems would give a very good picture about how research is carried out and how system designing is done in a proper manner.

2.17 Reference Books

The main areas of interest are books on Visual Basic and also books on Human Computer Interaction.

2.18 Questionnaires

This a basically a method of fact finding that is used to gather abstract information on things like the feasibility of this system and also general perceptions on the interface of the system. As it is deduced this method would involve the designer since recipient presumably needs different sets of questions for them to be relevant to their environment of responsibility. Therefore he questions must be general and wide reaching as possible.

2.19 Operating System Review

2.19.1 Microsoft Windows 2000 Server

Microsoft Windows 2000 is one of the leading operating system in the Internet and Intranet world. This popularity achieved by the Microsoft Skills in placing Windows 2000 Server as a useful corporate solution, with a variety of dynamic tools and the ability to standardize on both development and development in Windows 2000 Server platform. In the other hand, it also works smoothly and pretty well as an Internet platform, especially on small scale or fragmentizes the Internet level in departmental size.

The advantages in Windows 2000 Server are it is stable, huge variety of functional tools, less bugs and it is better security than the other versions of Microrsoft Windows. It also supports virtual domains and has the ability to delegate administration to other users. In the management console, browser based administration and common line scripting. In the security view, it features user authorization such as username and password, as configured by the system administrator and editing capabilities to meet auditing guidelines.

2.20 Database Management Review

The concept behind a database is simple where it is like a file cabinet. Database stores information, just as a file cabinet stones information. A specific application related to a set of information that is called a database. As for the relational database, Oracle, Microsoft Access 97 and Microsoft SQL Server were studied and below is the information about its:

2.20.1 Oracle

Oracle is the most popular database management system or rather relational database management system. The advantages of Oracle's architecture are:

- **Large Databases**

  Supports the largest of databases, potentially terabytes in size.

- **Many Concurrent Users**

  Oracle supports large numbers of concurrent users executing variety of database applications operating on same data. It minimizes data contention and guarantees data concurrency.

- **High Transaction Rates**

  It has fast processing performance.

- **High availability**

  At most production sites, Oracle works high hours (24 hours) per day with no down time.

- **Controlled Availability**

  An administrator can disallow use of a specific application so that the application's data can be reloaded, without affecting other application.

- **Manageable Security**

  Provides fail-safe security features to limit and monitor access.

- **Database Integrity**

  Oracle enforces data integrity, 'business miles' that dictate the standards for accepting data.

- **Distributed System**

  Oracle combines the data physically located on different computers into one logical database that can be accessed by all network users.

- **Portability**

  Oracle software is ported to work under different operating systems. Application developers for Oracle can be ported to any operating system with little or no modifications.

- **Connectibility**

  Oracle software allows different types of computers and operating systems to share information across networks.

- **Replicated Environments**

  Oracle software allows you to replicate objects to multiple sites. Oracle supports replication both at data and schema level.

2.20.2 Microsoft Access 97

One of the best and fastest selling relational database packages for Windows on the market Microsoft Access 97. Access comes in two different modes. The first one is an easy to use menu driven interface that lets you issue commands with only a basic knowledge of Access. The second mode program mode that lets the users to store instructions in a Visual Basic program file and execute them with one particular command.

Access has existed in five main versions. In the context of Access, a database can be viewed as a large repository in which tables, reports, queries, forms and other objects are stored inside.

Access allows the user to indicate how tables should be related to each other, a table can have one-to-one, one-to-many or many-to-many relationships. A table that has referential integrity allows only one parent record for each child record. Access allows the user to make changes to the structure of a database table, user can add, delete and rearrange fields in the table structure. Users can also control how data will be entered in a table using the properties sheet of a field.

As a database system Access has any good points and many bad points. Happily most of the bad points relate to the 'class' of application that Access it. It is important to clarity the 'class' that Access falls information. Access is a desktop database package and it has the first preceived bad point where it does not provide good performance when run across a network and when a lot of people using it at the same time. However, if one considers where the performance is good. In addition, Access can and does make a good front-end package larger engine such as Oracle and SQL Server.

Against other desktop database packages Access has a bigger advantage where it is likely if you are running Windows as your operating system and using Microsoft office as your application base, Access integrates well with these packages and data transfer between Access and the other office components is relatively easy. In addition, against the other desktop database Access is both rich in features and powerful, Access 2000 would be our choice to buy it.

2.20.3 Microsoft SQL Server 7.0

SQL Server is a special-purpose, nonprocedural language that could support the manipulation, definition and control of data in relational database management systems. It's special-purpose language, because we can use it only for handling databases and we can't write general-purpose applications with it. SQL is also known as a data sub language because to write an application, we have to embed SQL in some other language, and it's frequently used that way. A sub language can be used with applicator languages. A full-featured application language applicable because it usually includes semantics for procedures, where else SQL is nonprocedural. It only specifies what should be done so as conclusion SQL is concerned with results rather than procedures.

The most important features of SQL is that it provides access to relational databases, where it is so fundamental to SQL that many people think the terms SQL database and relational database are synonymous. But it is not like that and further more it doesn't even mention the term relation.

It makes giant strides in performance, reliability and scalability, giving the organization many opportunities to create intelligent, real world business solutions. These are following innovation that the SQL Server had made:

i. Scalable from laptop to multiprocessor cluster
ii. Dynamic row-level locking
iii. Dynamic self-management
iv. Wide array of replication options
v. SQL Server Desktop
vi. Integration with Microsoft office 2000 and Microsoft Visual Studio

These innovations, plus many more changes, make SQL Server 7.0 highly scalable and excellent for data warehousing. In addition, organisations that also run office 2000 can take advantages of new ways that office and SQL Server work together.

2.21 Application Programming Languages Review

2.21.1 Visual Basic

The programming language that has been chosen to be the base on which this intelligent system would be built is Visual Basic. This programming language was chosen due various reasons. Visual Basic is generally a compromise between "high- Level" programming and "Low-Level" programming. It allows all sorts business oriented problems to be carefully managed and it is very easy to produce forms and reports and do database maintenance.

The other reason why Visual Basic is the "right" language to use it that it is the flagship language of the Microsoft powerhouse. Due to this fact it is guaranteed to keep pace with windows developments. Besides all this, the other reason why Visual Basic was chosen is because it has the ability to perform other functions as well. The function mentioned here is basically it's ability to create and to produce graphical interfaces internally. This could be considered, as a very strong aspect due to the fact that well designed interfaces is a vital concern to this intelligent system. Visual Basic is also considerably easy to use, whereby it doesn't necessary require system developers of very high knowledge and programming skills.

Visual Basic is programming software that has many well identified features that could be of very good use. By using such strong features, this system could be developed much more cohesively. The last factor that led to choosing, Visual

Basic is because, Visual Basic has very good mapping functionally with Microsoft Access. Programming codes could easily be developed in order to access databases that are created with Microsoft Access.

2.21.2 Visual C++

Nearly all world-class software, from the leading Wen Browsers to mission-critical corporate applications, is built using the Microsoft Visual C++ development system. Visual C++ is the most productive C++ tool for the highest-performance development for Windows and the Web.

Visual C++ 6.0 brings a new level of productively to C++, without sacrificing flexibility, performance, and control. In addition to features like IntelliSence Technology and Edit and continue which significantly speed development time, Visual C++ 6.0 contains greatly improved support for Web and Enterprise development. With Microsoft Visual C++ 6.0 Standard Edition, you can learn to build and reuse components, create ActiveX Controls, and develop Internet application in tightly integrated, visual environment.

The benefits in using Visual C++ are as below:

- Get a jump-start on high-performance development for Windows and the Web with the step-by-step tutorial and online C++ instruction. You can also easily get up and running with the numerous wizards, including the built-in AppWizard.

- Enjoy a new level of productivity with new features that significantly reduce development time. Development will spend less time building applications, less time coding, less time compiling, and less time debugging, while enjoying greater component reuse.

- Develop for Windows and the Web. Now you can easily build the smallest ActiveX controls, take advantage of the latest user interface enhancements form Microsoft in your applications, and create multimedia-based highly interactive, Dynamic HTML Pages.

- Next generation:

  Standard features

i. Easy-to-use tutorials

   Master Visual C++ programming tasks quickly using the printed step-to-step tutorial and online C++ instruction.

ii. IntelliSence Technology

   Increase productivity and greatly simplify coding with auto list members, auto parameter information, auto type information, code comments, and complete word that eliminate the need to memorize complex syntax, parameters, and component properties.

iii. Edit & continue

Get faster turnaround with now Edit and Continue in the debugger. Developers can edit code while debugging without having to quit the debugging session, rebuild restart the debugger, and return the application to the state where the problem occurred.

iv. Dynamic Class View Updating

Easily navigate your code and save time as changes like adding a variable or method are reflected immediately in classView.

v. Active Document Containment

Easily and seamlessly integrate the functionality of Active Documents from Microsoft Word, Microsoft Excel, and other applications.

vi. Composite Controls

Get state-of-the-out ActiveX development with new composite controls-easily reuse and Active control within you own.

vii. Faster Compiler Throughput

Compiler throughput on debug projects is as much as 30 percent faster, and non-debug projects as much as 15 percent faster (without sacrificing any optimizations).

viii. Internet Explorer 4.0 Common Controls

Easily use the new common controls to give your user interfaces the compelling look of the new Internet Explorer 4.0.

ix. Web integration in MFC

Just choose CHTML View in the AppWizard to build custom web browsers into MFC front ends, with full support for dynamic HTML.

x. Dynamic HTML Support

Build multimedia-rich, highly interactive dynamic HTML Web Pages that fully exploit Internet 4.0 and capitalize on your existing skills and code.

xi. Multiple Monitor Support

Divide and conquer with the multiple monitor support, Run an application on one screen and debug on another (Requires Windows 98 or Windows NT 5.0).

Imported Features

i. AppWizard

Easily get up and running with the built in AppWizard chose from more than 10 different application types, including web applications.

ii. Extensive Sample code

Save time with examples and sample code written by Microsoft experts – including full source code for Microsoft Foundation Classes (MFC).

iii. Wizards

Save time with numerous new and enhanced wizards throughout the product for MFC, ATL, and more.

iv. Microsoft Foundation Classes (MFC)

Build world-class application with the most robust, productive, and widely used application framework available for Windows.

v. Active Template Library (ATL)

Quickly create the smallest, most scaleable Server-Side components and Acitve controls with the Active Template Library (ATL).

vi. ANSI C++ Support

Support for the latest ANSI/ISO specification, including array new and delete.

2.21.3 JavaScipt

JavaScipt is Netscape's cross-platform, object-oriented language. Core JavaSript contains a core set of obejcts, such as Array. Data, and Math, and a core set of language elements such as operators, control structues and statements. Core javaScipt can be extended for a variety of purposes by supplementing it with additional objects; for example:

- Client-side JavaScript extends the core language by supplying objects to control a browser (Navigator or another web browser) and its Document Object Model (DOM). For example, client-side extensions allow an application to place elements on an HTML, form and respond to user events such as mouse clicks, form input, and page navigation.

- Server-side JavaScript extends the core language by supplying objects relevant to running JavaScript on a server. For example, server-side extensions allow an application to communicate with a relational database, provide continuity of information from one invocation to another of the application, or perform file manipulations on a server.

JavaScript lets us create application that run over the Internet. Client applications run in a browser, such as Netscape Navigator, and server applications run on a server, such Netscape Enterprise Server. Using JavaScript, we can creat dynamic HTML pages that process user input and maintain persistent data using special objects, files, and relational databases.

Through JavaScript's Live Connect functionality, we can let Java and JavaScript code communicate with each other. From JavaScript, you can instantitate Java objects and access their public methods and fields. From Java, you can access JavaScript objects, properties, and methods.

Netscape invented JavaScript, and JavaScipt was first used in Netscape browsers.

2.21.4 HTML Use For Web Sites

HTML stand for Hypertext Markup Language, HTML is lingua franca for publishing hypertext on the World Wild Web, Web sites are written in HTML. Actually H-T-M-L are initials that stand for HyperText Markup Language (computer people love initials and acronyms!)

Let me break it down for you:

- **Hypertext** is the opposite of linear. It used to be that computer program had to more in a linear fashion. This before this, and so on. HTML does not hold to that pattern and allows the person viewing the World Wide Web Page to go anywhere, any time they want.
- **Text** is what you will use, Real honest to goodness English Letters.
- **Mark up** is what you will do. You will write in plain English and then mark up what you wrote
- **Language** because they needed something that started with "L" to finish HTML. Because it's a language and the language is plain English.

The frantic pace of progress on the Internet has slowed considerably in recent years as the medium has started to mature. Once the exclusive realm of academia and government, the Internet has wormed its way into our popular culture. Email and web URLs are becoming as common as business cards and brochures.

As part of the Internet, the World Wide Web is the predominant force in the growth of the global computer network. Its language, much richer than a few years ago, is still quite simple. The web interface is attractive and friendly and is adaptable to many uses. These are Web Sites for selling products, selling ideas, keeping up appearances, informing publics, distributing policy, delivering education, not to mention the vast array of sites devoted just to killing time.

Hypertext Markup (HTML) is the language that puts the face on the web. HTML is a non-proprietarty format, based upon SGML (Standard Generalized Markup Languages), for describing the structure of hypermedia documents – plain text (ASCII) files with embedded codes for logical markup, using tags like <A> and </A> to structure text into tables, hypertext links interactive forms, headings paragraphs, lists and more with HTML and the World Wide Web, you have the ability to bring together text, pictures, sounds , and links....all in one place! HTML files are plain text files, so they can be composed and edited on any type of computer.... Windows, Mac, UNIX, whatever.

SGML is the substratum on which HTML was conceived and, therefore, is responsible for many of HTML 's strengths and weaknesses. SGML stands for Standard Generalized Markup Language: this is a formal system designed for building text markup languages. It is not a markup system by itself, however; think of it as a programming language to build working programs (HTML being one of them) rather than a program by itself.

HTML contains commands, called elements or tags, to mark text as headings, paragraphs. Lists quotation, and so on. It also has tags for including images within the documents, for including fill-in forms that accept user input, and most importantly, for including hypertext links connecting the document being read to other documents or Internet resources such as WAIS databases or anonymous FTP sites.

It is this last feature that allows the user to dick on a string of highlighted text and access a new document, an image, or a movie file from a computer thousands of miles away. The HTML documents specify where this other document through a URL, which is included in the HTML markup instructions and which is used by the user's browser to find the designated resource.

URLs can point to other HTML documents, pictures, sound files, movie files, or ever database search engines. They can be downloadable programs in Java or other languages. They can be located on the user's computer or anywhere on the

Internet. They can be assessed from HTTP servers or from FTP, Gropher, WAIS or other servers.

The URL is an immensely flexible scheme, and in combination with HTML, yields an incredibly powerful package for preparing a web of hypertext documents linked to each other and to Internet resources around the world. This image of interlinked resources is the fact the vision that gave rise to the name, World Wide Web.

One of the best things about the World Wide Web is that it's just as easy to create Web Pages as it is to browse them. They key to publishing on the Web is having a form understanding of Hypertext Markup Language. (HTML) Despite the intimidating name, HTML is extremely simple to learn and use.

Before diving head-first into the language of HTML itself, it will help you to understand a little bit about the World Wide Web works. After all, HTML is designed to guide users through the vast and tangled resources of the Web. As an HTML author, you will need to understand some of the basics behind the architecture of the World Wide Web. Knowing how the Web works, as well as when it doesn't and why, can help you make important decisions, about how to construct your own Web pages.

HTML isn't the only way to present info on the Web, but it's the glue that holds everything together. Writing good HTML documents involves both technical issues (proper construction of the documents) and design issues (ensuring the info content is clearly presented to the user).

Which brings us to the first reason to learn HTML. Even the best editor don't support all the tags that are part of HTML at any given this. Sometimes it's necessary to directly modify the source of the page to add or change tags and attributes. To do this, you need to know how the tags relate to each other.

HTML is as much all organizational tool as it is a design tool. Even with WYSIWYG (What-You-See-It-What-You-Get) Editors, the rationale behind the tags is to give a structure and purpose to your page. Learning how HTML organizes your page leads to better planning and design for your readers.

Finally, HTML can be down right fun. You gain a certain satisfaction from building a Web Page from the ground up. It's like building your own house, you know every brick, every board, every nail and you know how to modify it so the result is just what you want. It also makes it much easier to take a look at someone else's page and know how they achieved their effect.

HTML as an Extensible Language

HTML is designed to be extensible. This simply means that new features, commands, and functionality can be added to the language without "breaking" older documents that don't use these new features. In fact, HTML is a rapidly evolving language, with new features being added on a regular basis.

Since HTML is constantly evolving, it is important to have a way of indicating the version the language being used. This is done through the version number of the HTML specification. The very first definition of HTML was call Version 1, or HTML 1.0. First generation pages use old-fashioned HTML 1.0, and the mostly text with a hokey picture or two stuck in the middle, they were the best you could do in 1989, but having a first-generation page today marks you as more technologically backward than having no Web Page at all.

This quickly evolved into the next "definitive" version of HTML, known as Version 2, or HTML 2.0, Second-generation pages use a few HTML 2.0 ticks, such as putting a pretty (or garish background behind a page, arranging text in tables, and offering an online order form. They can look nice, but rarely match the quality that people have come to expect from paper documents.

Then from HTML 2.0 to HTML 3.2. all browsers, at a minimum, support the HTML 3.2 standard. Third-generation pages are what the world is talking about how that HTML 3.2 is the standard. They use creative layout, custom color, fast graphics, fonts, and interactive feedback to make your Web Site more engaging than anything does on paper.

The newest "definitive" version of HTML is known as HTML 4.0 Note, however that many of the newest features of HTML 4 are not widely supported. As mentioned, HTML is a moving target, and there is much effort underway to add features and improve old ones.

2.22 Creating User Interface

The user Interface is the most important part of an application. As users are not usually aware of the coding behind an application, thus the useability of the application depends on the interface. User interface is how users interact with the system to provide inputs and outputs on users request.

Before designing a user interface, the criteria has to be considered. It is regarding the purpose of application, documents style, type of different form, command in the menu and the use of dialog boxes to interact with the users. Visual Basic is one of the tools that is used to create a great application for users.

2.23 Summary

Overall this chapter reviews on the existing system. The weaknesses of the existing system are detected. The comparison between the systems helps to determine the features of the new system.

It focuses on the specification that has to be the content in the new system, which is necessary for successful system development. The criteria of a user interface and the type of modeling in use have to be determined.

Chapter 3 : METHODOLOGY

3.1 Introduction

The objective of this phase is to define the Payroll Management System's methodology, information gathering techniques and requirements that are needed in order to fulfill the systems purpose. It is important to take in understanding, documenting and managing the systems requirements in order to avoid problems in the later stages of the systems development. Therefore, the requirements have to be correct, consistent, complete and realistic before proceeding to the design phase.

3.2` System Methodology

3.2.1 Waterfall And Incremental Prototyping Model

The system development methodology, show in figure 1 describes the sequence of stages of Payroll Management System development process. This method is a combination of the waterfall model and the incremental prototyping model. The waterfall model, shown in figure 2 is a helpful in presenting a high-level view of what will go on during the development of the BOSS and the sequence events that are expected to be encountered.

However, problems become understood as systems often evolve and alternatives are evaluated. Developers will then be required to backtrack to previous phases to make enhancements. It is not feasible for developers to journey through this entire waterfall model repeatedly for this purpose. Therefore, the waterfall model fails to reflect the way systems are really developed, that is with iteration. Realizing this, the incremental prototyping model was integrated information the developments method to overcome the rigidity of the waterfall model. The incremental prototyping model will be a more efficient and flexible way to develop the system's module. It will enable us to assess alternative module design and coding strategies and decide which is best for the system before integrating the different units. Revisions can be made as needed at these early phases rather than at the final system testing stage. This will help save time and cost.

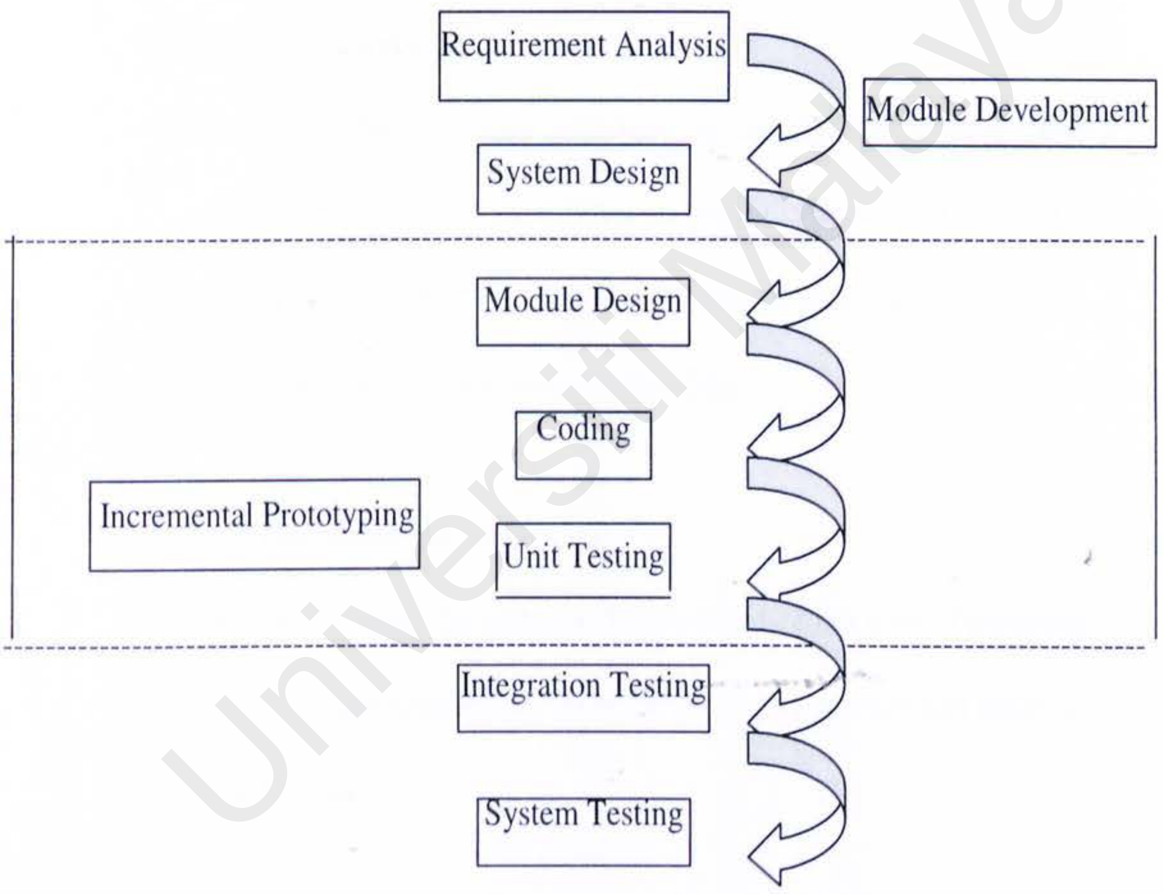


Figure 1 : Combination of the waterfall and Incremental Prototyping Model

3.2.2 Waterfall Model

The first published model of the software development process was derived from other engineering processes (Royce, 1970). This is illustrated in Figure 3.1.

because of the cascade from one phase to another, this model is known as the 'waterfall model' or software life cycle. This principal stages of the model map onto fundamental development activities:

1. Requirement analysis and definition

   The system service, constraints and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.

2. System and software design

   The system design process partitions the requirements to either hardware of software systems. It establishes an overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.

3. Implementation and unit testing

   During this stage, the software design is realized as a set of programs or program units. Unit testing involves verifying that each unit meets its specification.

4. Integration and system testing

   The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.

5. Operation and maintenance

   Normally (although not necessarily) this is the longest life-cycle phase. The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the life cycle, improving the implementation of system units and enhancing the system's services as new requirements are discovered.

   The figure 3.1 shown the software life cycle.

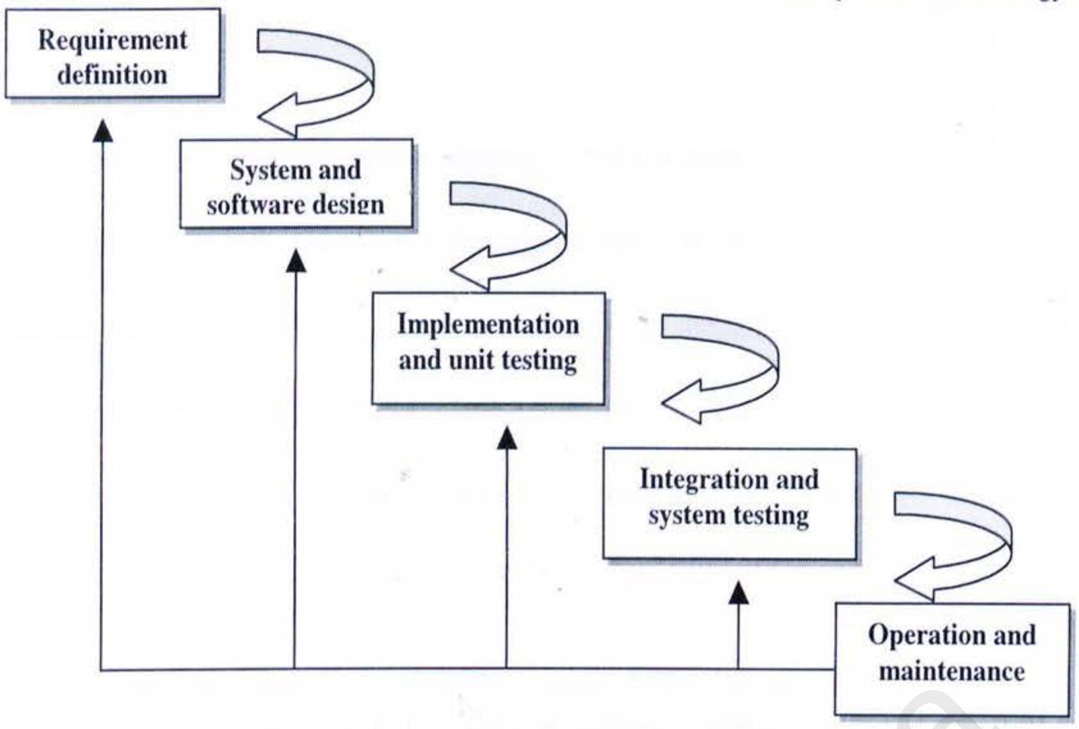

Requirement definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Figure 3.1 : The software life cycle

In principle, the result of each phase is one or more documents which are approved ('signed off'). The following phase should not start until the previous phase has finished. In practice, these stages overlap and feed information to each other. During design, problems with requirements are identified, during coding design problems are found and so on. The software process is not a simple linear model but involves a sequence it iterations of the development activities.

Because of the costs of producing and approving documents, iterations are costly and involve significant rework. Therefore, after a small number of iterations, it is normal to freeze parts of the development, such as the specification, and to continue with the later development stages. Problems are left for later resolution, ignored or are programmed around. This premature of requirements may mean that the system won't do what the user wants. It may also lead to badly structured system as design problems are circumvented by implementation tricks.

During the final life-cycle phase (operation and maintenance) the software is put into use. Errors and omissions in the original software requirements are discovered. Program and design errors emerge and the need for new functionally is identified. The system must therefore evolve to remain useful. Making these changes (software maintenance) may involve repeating some or all previous process stages.

The problem with the waterfall model is its influence partitioning of the project into these distinct stages. Commitments must be made at an early stage in the process and this means that it is difficult to respond to changing customer requirements. Therefore, the waterfall model should only be used when the requirements are well understood. However, the waterfall model reflects engineering practice. Consequently, software processes based on this approach are still used for software development, particularly, when this is part of a larger engineering project system.

The advantages of the waterfall model are:

(a) Best Model

The waterfall model is best when expectations and quality are more focused on compared to cost and time.

(b) Reliable and flexible

The system tends to be more reliable and flexible as each phase is analyzed thoroughly before they are built.

Problems/Challenges associated with the Waterfall Model

Although the **Waterfall Model** has been used extensively over the years in the production of many quality systems, it is not without its problems. In recent years it has come under attack, due to its rigid design and inflexible procedure. Criticisms fall into the following categories:

- Real projects rarely follow the sequential flow that the model proposes.
- At the beginning of most projects there is often a great deal of uncertainty about requirements and goals, and it is therefore difficult for customers to identify these criteria on a detailed level. The model does not accommodate this natural uncertainty very well.
- Developing a system using the **Waterfall Model** can be a long, painstaking process that does not yield a working version of the system until late in the process.

The advantages of the waterfall model:

a) The waterfall Model takes time to collect facts, draw models and validate those models as each phase is analyzed thoroughly

b) The waterfall model is not recursive, as each stage has to be completed before proceeding to the next step. Users tend to change their mind anytime.

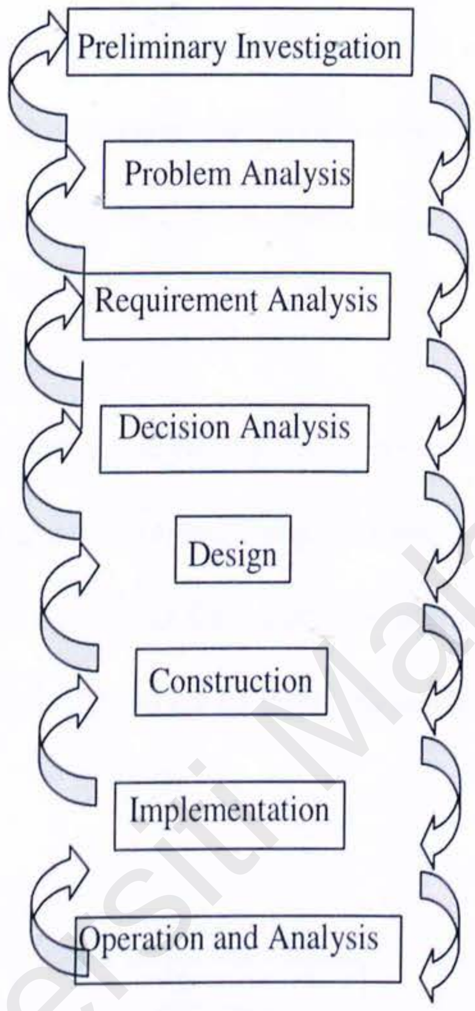


Figure 3.2 : Waterfall Model

3.3.3 Evolutionary Development

Evolutionary development is base on the idea of developing an initial implementation, exposing this to user comment and refining this through many versions until an adequate system has been developed (Figure 3.3). Rather than have separate specification, development and validation activities, these are carried out concurrently with rapid feedback across these activities.

These are two types of evolutionary development:

1) Exploratory development

Exploratory development where the objective of the process is to work with the customer to explore their requirements and deliver a final system. The development starts with the parts of the system which are understood. The system evolves by adding new features as they are proposed by the customer.

2) Throw-away prototyping

Throw-away prototyping where the objective of the evolutionary development process is to understand the customer's requirements and hence develop a better requirements definition for the system. The prototype concentrates on experiencing with those parts of the customer requirements which are poorly understood.

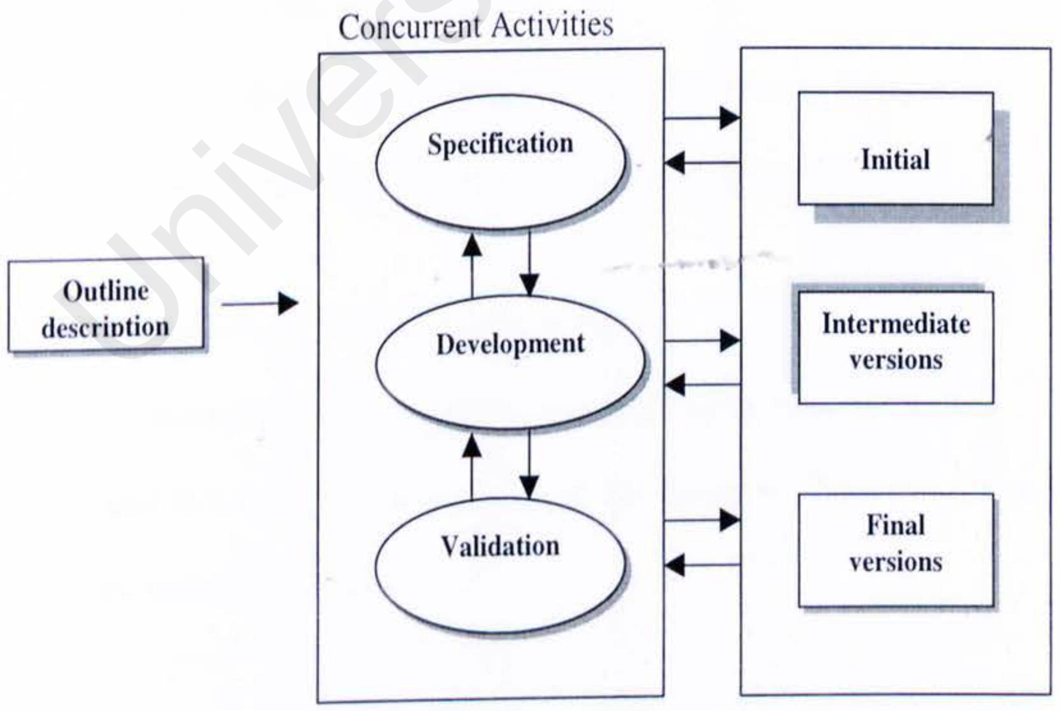


Figure 3.3 : Evolutionary development

An evolutionary approach to software development is often more effective than the waterfall approach in producing systems which meet the immediate needs of customers. The advantage of a software process which is based on an evolutionary approach is that the specification can be developed incrementally. As users develop a better understand of their problem, this can be reflected in the software system. However, from an engineering and management perspective, it has three problems:

I. The process is not visible

Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.

II. Systems are often poorly structured

Continual change tends to corrupt the software structure. Incorporating software changes becomes increasingly difficult and costly.

III. Special tools and techniques may be required

These allow for rapid development but they may be incompatible with other tools or techniques and relatively few people may have the skills that are needed to use them

For small systems (less than 100,000 lines of code) or for medium-sized systems (up to 50,000 lines of code) with a fairly short lifetime, I think that the evolutionary approach to development is the best approach. However, for large, long-lifetime systems, the problems of evolutionary development become particularly acute. For these systems, I recommend a mixed process that incorporates the best features of the waterfall and the evolutionary development models.

3.4 Waterfall Model with Prototyping

The Prototyping Model was developed on the assumption that it is often difficult to know all of your requirements at the beginning of a project. Typically, users know many of the objectives that they wish to address with a system, but they do not know all the nuances of the data, nor do they know the details of the system features and capabilities. The Prototyping Model allows for these conditions, and offers a development approach that yields results without first requiring all information up-front .

When using the Prototyping Model, the developer builds a simplified version of the proposed system and presents it to the customer for consideration as part of the development process. The customer in turn provides feedback to the developer, who goes back to refine the system requirements to incorporate the additional information. Often, the prototype code is thrown away and entirely new programs are developed once requirements are identified.
There are a few different approaches that may be followed when using the Prototyping Model:

- creation of the major user interfaces without any substantive coding in the background in order to give the users a "feel" for what the system will look like,
- development of an abbreviated version of the system that performs a limited subset of functions; development of a paper system (depicting proposed screens, reports, relationships etc.), or
- use of an existing system or system components to demonstrate some functions that will be included in the developed system.

3.4.1 Prototyping is comprised of the following steps:

- Requirements Definition/Collection

  Similar to the Conceptualization phase of the Waterfall Model, but not as comprehensive. The information collected is usually limited to a subset of the complete system requirements.

- Design

  Once the initial layer of requirements information is collected, or new information is gathered, it is rapidly integrated into a new or existing design so that it may be folded into the prototype.

- Prototype Creation/Modification

  The information from the design is rapidly rolled into a prototype. This may mean the creation/modification of paper information, new coding, or modifications to existing coding.

- Assessment

  The prototype is presented to the customer for review. Comments and suggestions are collected from the customer.

- Prototype Refinement

  Information collected from the customer is digested and the prototype is refined. The developer revises the prototype to make it more effective and efficient.

- System Implementation

  In most cases, the system is rewritten once requirements are understood. Sometimes, the iterative process eventually produces a working system that can be the cornserstone for the fully functional system.

3.4.2 The disadvantages of the Prototyping Model

Criticisms of the Prototyping Model generally fall into the following categories:

- Prototyping can lead to false expectations

  Prototyping often creates a situation where the customer mistakenly believes that the system is "finished" when in fact it is not. More specifically, when using the Prototyping Model, the pre-implementation versions of a system are really nothing more than one-dimensional structures. The necessary, behind-the-scenes work such as database normalization, documentation, testing, and reviews for efficiency have not been done. Thus the necessary underpinnings for the system are not in place.

- Prototyping can lead to poorly designed systems

  Because the primary goal of Prototyping is rapid development, the design of the system can sometimes suffer because the system is built in a series of "layers" without a global consideration of the integration of all other components. While initial software development is often built to be a "throwaway, " attempting to retroactively produce a solid system design can sometimes be problematic.

3.4.3 Variation of the Prototyping Model

A popular variation of the Prototyping Model is called Rapid Application Development (RAD). RAD introduces strict time limits on each development phase and relies heavily on rapid application tools which allow for quick development.

3.3.4 Incremental Prototyping Model

The Incremental Prototyping Model is a working model of the users requirement. In other words, it is a proposed design for a system. The purpose of prototype is to reduce the amount of time until the users begin to see a working system. It is a technique for quickly building a function but an incomplete model of the information using rapid application development tools and new parts are added on as the design cycle progresses.

The advantages of the Incremental Prototyping Model are

a) Encourages user's active involvement

As users do not usually know want they want, their involvement is very important in the prototyping model. It allows the user to see the model and make changes according to their needs. User's involvement throughout the phases helps to build a reliable project. It also helps to reduce risk and uncertainly.

b) Errors tend to be detected at an earlier stage

These are frequent user feedback and that helps the detection of errors at the early stage. To do modifications after developing a full system is a waste of time and expensive.

The disadvantages of the Incremental Prototyping Model are:

a) Fast approach

Prototyping is a fast approach and this can sometimes affect the quality of the system.

b) Problem analysis is ignored

This phase is ignored as prototype can easily solve any problems while developing the system.

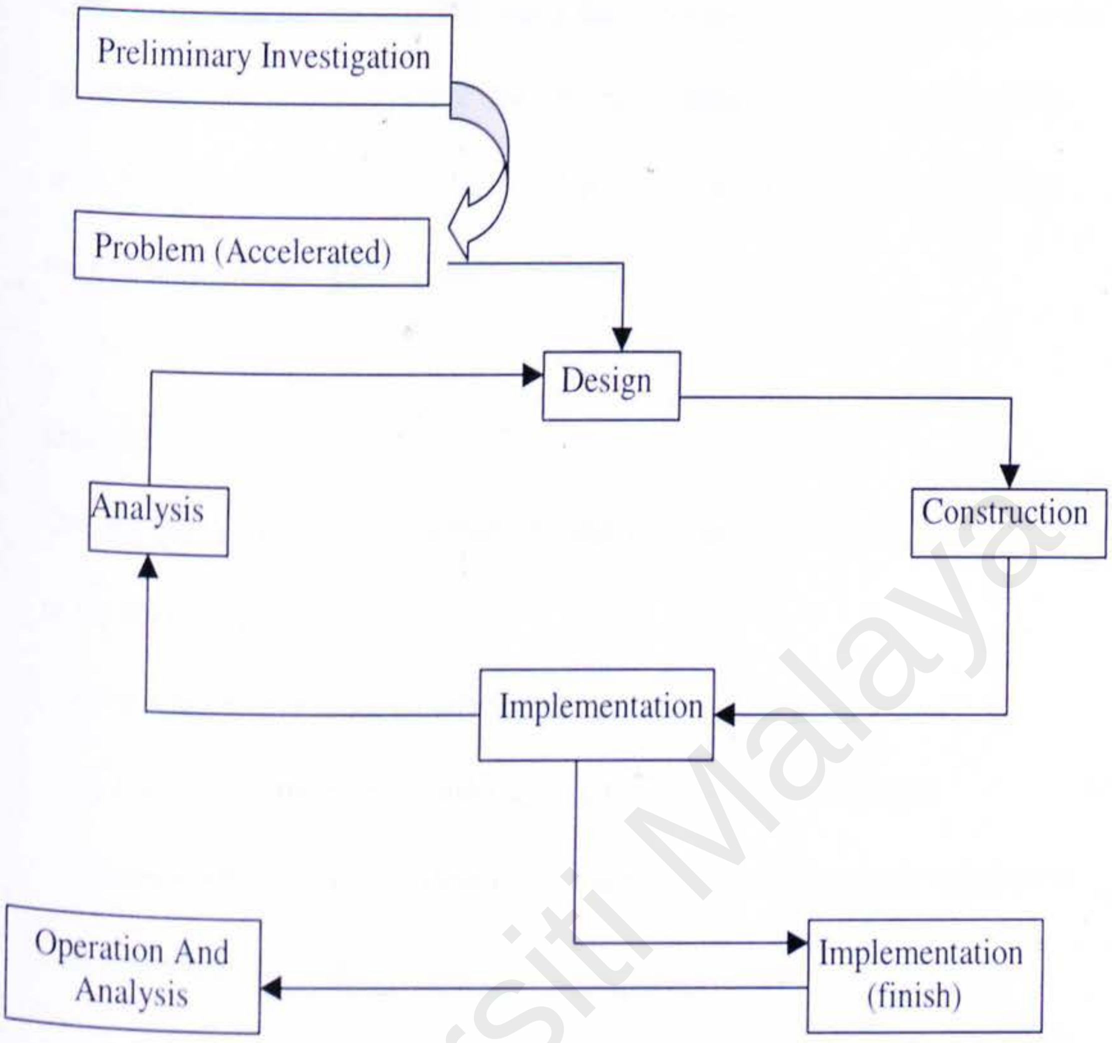


Figure 3 : Prototype Model

3.3.5 The Payroll Management System

The payroll management system (PMS) is a computer-based system that supplies users so they are able to carry out their activities of retrieving information gathered from a census in a fast, easy, safe, effective and efficient way.

Log On

Users are able to log on to (PMS) using their user name and password in order to get authentication to the system. The user name determines the category of the user. Either it is the administrator who has the full access to the system or the normal user who has a partial access to system.

Options

One the user's identity is identified, the user is given the following options to link to the following:-

i. Access or retrieve required information

As this section, the information required by the users are displayed automatically according to their requirements. Thus, user are only required to choose the given options.

ii. Analysis the information

Users are able to carry out computations according to their own requirements. They may carry out computation for totals and so on by using the BOSS software.

iii. Generation of reports, salary master, bank advise, EPF report, SOCSO report and PCB monthly submission.

Users are able to generate reports for the information that they have retrieved or analysis using the BOSS software.

iv. Password

Users are given the options to change their password anytime to protect their authentication according to their need. In short, they can change it anytime.

v. Printing

The option of printing is available for users that wish to print the reports for further references. They are also able to print the data that has been retrieved from the database. This option is given when the user activates the BOSS software.

3.4 Advantages of the Payroll Management System

a) PMS has the following advantages

PMS provides an online info concerning a census at ones fingertips. Users are able to access to the system anytime to get the relevant information and analysis the information. The PMS has high availability.

b) Data

Census's data such as the number of men and women working in different professions and data concerning similar indicators concerning gender is displayed according to the user requirements.

c) Computations and analysis

The user can also compute totals and other calculations that they required. This can be done by the user to analyze the data that has been retrieved and make decisions by using this data. This option is carried out by activating the BOSS software.

d) Various type of reports

Various type of reports can be generated according to the information that is required by the user. The user can determine the data that he/she requires to be displayed in a form of report. This option can be carried out by activating the BOSS software.

e) Time saving

The data and information, which are displayed automatically, save the time. It helps to decrease the time taken to manually gather and analyze the information.

f) Documentation

PMS provides electronics documentation, which consist of any data that is related to payroll for the staffs. Instead of looking through pile or files, users can just access PMS to locate the information that is much faster and convenient.

g) Printing

PMS gives the option of printing of payslip by individual, department checklsit by head of department by using the BOSS software. This will make it more convenient for the user to have a copy of the data for further references.

h) Interface

PMS also provides a user-friendly interface. It consists of drop down list, attractive menus of interfaces and dialogue boxes that interact with the users and the system.

3.5 Functional Requirements

Functional requirement is the features that have to be included in a system to satisfy the needs of the system. It describes the interaction between system and environment and how a system interacts with a request.

a) User Module

i) Login Menu

This is an introduction to the payroll management system. Users are required to key-in their user name and password in order to access the system. The username will determine the type of access for the specific user.

Once the user's identity has been identified, the user is allowed to link to:

I. Access or retrieve required information

At this section, the information required by the users are displayed automatically accordingly to their requirements. Thus, users are only required to choose the given options.

II. Analysis the information

Users are able carry out computations according to their own requirements. They may carry out computation for totals and so on by activating the BOSS software.

III. Generation of reports

Users are able to generate various reports for the information that they have retrieved or analyzed by activating the BOSS software.

IV. Password

User are given the options to change their password anytime to protect their authentication according to their needed. In short, they can change it anytime.

V. Print

This section provides the facility of printing the information that the user wishes to print such as reports, paylip for further references. They are also able to print the data that has been retrieved from the database by activating the BOSS software and choosing the option to print.

b) Administrator Module

I User Management

Records of the users can be edited, deleted and modified. New users can be registered.

II Database Management

To update any changes on the records of the database. Administrators are able to view all the reports of the database.

III Administrator Management

New administrators can be added and an existing administrator can be deleted.

IV Password

Administrator can change their password according to their needs.

3.6 Non Functional Requirement

Non functional requirements are the additional features that attribute to the system that may limit the proposed solution's boundaries.

a) Security

This requirement is to allow only the valid users to get access to the system. User name and password determine the type of access of the particular user.

b) Interface

Graphical User Interface allows the manipulation of the graphic presentation that is displayed on the screen. It is a representation of the user's input and ouput.

c) User Friendly

The facility helps the users to be comfortable with the application. Users are allowed to perform the operation in an easy way.

d) Response time

The average delay between a request and response has to be fast.

3.7 System Development Tools

There is many software available in the market to develop this system. Among the software that are reviewed and analyzed are Microsoft Visual Basic, Microsoft Visual C++, Oracle, Microsoft Access 2000, Microsoft SQL Server, HRMS software, UBS software, MCSB software and NCCS (The Number Crusher Statistical System).

3.7.1 Programming Language

The programming language that has been chosen to be the base on which this intelligent system would be built is Visual Basic. This programming language was chosen due various reasons. Visual Basic is generally a compromise between "high-level" programming and "low-level" programming. It allows all sorts of business oriented problems to be carefully managed and it is very easy to produce forms and reports and do database maintenance.

The other reason why Visual Basic is the "right" language to use it that it is the flasship language of the Microsoft powerhouse. Due to this fact it is guaranteed to keep pace with windows developments. Besides all this, the other reason why Visual Basic was chosen is because it has the ability to perform other functions as well. The function mentioned here is basically it's ability to create and to produce graphical interfaces internally. This could be considered, as a very strong aspect due to the fact that well designed interfaces is a vital concern to this intelligent system, Visual Basic is also considerably easy to use whereby it doesn't necessary require system developers of very high knowledge and programming skills.

Visual Basic is programming software that has many well identified features that could be of very good use. By using such strong features, this system could be developed much more cohesively. The last factor that led to choosing Visual Basic is because, Visual Basic has very good mapping functionally with Microsoft Access 97. Programming codes could easily develop in order to access databases that are created with Microsoft Access 97. Visual Basic is also to integrate well with the BOSS software.

Chapter 4 : SYSTEM ANALYSIS AND DESIGN

4.1 Introduction

System design is a process through which requirements are translated into representations of software. System design is a very important factor in system development as it determines the success of the system. The design specifications describe the features of a system, the components or elements of a system and their appearance to users. Requirements that are found in analysis stages are the ones actually translated into design specifications.

The objective of a system design is listed below:

1) Specify logical design elements

   Detailed design specification that describes the features of information system, input, output, files and databases and procedures.

2) Support planning activities

   Results of using the system help development planning processes

   Technology is secondary to the result procedure using the system

3) Meet user requirements

   Meet user needs in terms of

   - Presenting appropriate procedures correctly
   - Presenting paper form of information
   - Providing accurate results
   - Using appropriate method of interaction

- Providing overall reliability

4) Easy to use

   Favorable human engineering

In a system development life cycle, the design phase is the stage where the requirements are analyzed in the previous phase. System analysis phase is translated information system characteristics. The system design phase is the phase in which requirements produced in the system analysis phase are translated into a representation of the system.

This phase will be concerned with :

1. Architectural design
2. Database design
3. Functional design
4. Interface design

4.2 Architectural Design

Architectural Design represents the structure of data and program components tht are required to build the computer-based system. It also considers the architectural style that the system will take, the structure and properties of the components that constitute the system, and the relationship that occurs among all architectural components of a system.

For this project, architectural design is based on the modular decomposition approach. Decomposition is a structured system approach where designs are partitioned to smaller parts that are called modules or components. It is a top-down approach that is based on assigning function to components.

The developer or designer begin with a high-level description or explanation of the functions that are to be implemented and builds lower-level explanations of how each component will be organized and related to other components. In this method or approach, the system development begins from a high level description and goes down to a lower level description.

4.3 Database Design

File and database design occurs in two steps. We may begin by developing a logical database (Dummy database) model, which describes data using a notation that corresponds to a data organization used by a database management system. This is the system software responsible for storing, retrieving, and protecting data (such as Microsoft Access, Oracle, or SQL Server). The most common style for a logical database model is the relationship database model. Once we develop a clear and precise logical database model, we could ready to prescribe the technical specifications for computer files and databases in which to store the data ultimately. A physical database design provides these specifications.

We typically do logical and physical database design in parallel with other systems design steps. Thus, we collect the detailed specifications of data necessary for logical database design as I design system inputs and outputs. Logical database design is driven not only from the previously developed E-R data model for the application but also from form and report layouts. We study data elements on these system inputs and outputs and identify interrelaitonship among the data.

Relations:

User(User_ID, Password)

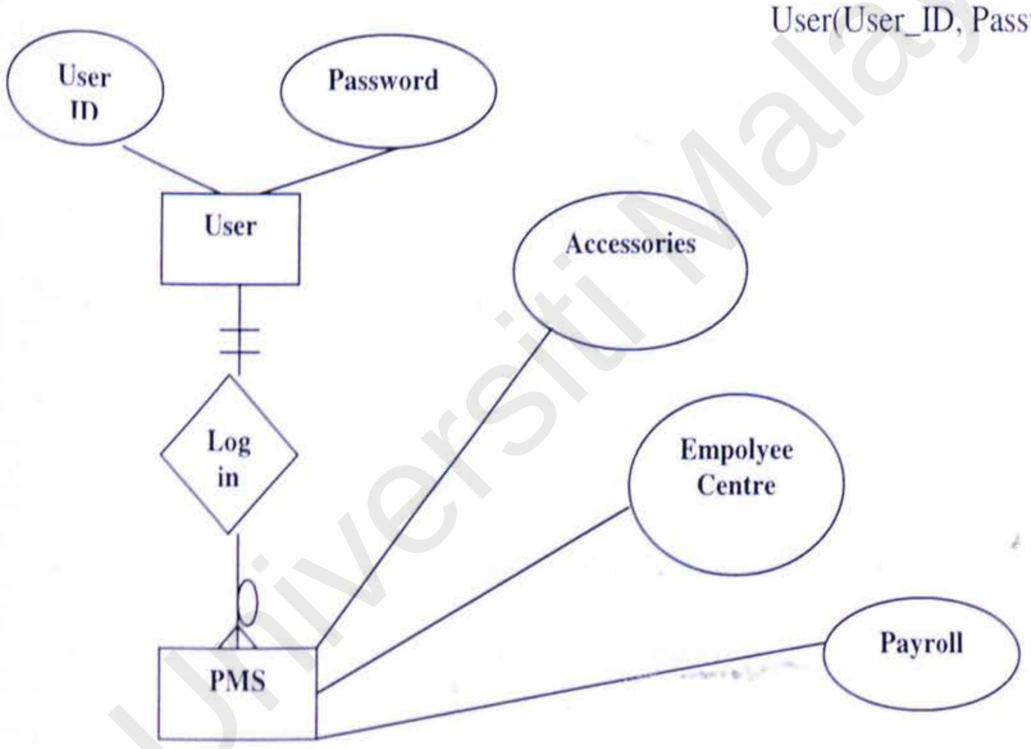


Figure 4.1 : E-R Diagram and Relations

4.4 Information System Analysis and Design

Information system analysis and design is a method to create and maintain information systems that perform basic business functions such as keeping tracks of customer names and address, processing orders, and paying employees. The main goal of system analysis and design is to improve organization system, typically through applying software that can help employees accomplish key business tasks more easily and efficiently.

4.5 Core Concepts

The main goal of system analysis and design is to improve organization system. Often this involves developing or acquiring application software and training employees to use it. Application software, also called a *system*, is designed it support a specific organization functions or process, such as payroll system in the organization. The goal application software is to turn data into information, for example, software for the payroll management in Human Resource Department may keep tracks of the changing payrates of employees.

In addition to application software, the following system includes

1. The hardware and systems software on which the application software runs. Note that the system software helps the computer function, whereas the application software helps the user perform tasks such as writing a paper, preparing a spreadsheet, and linking to the Internet.

2. Documentation and training materials, which are material created by the system analyst to help employees use the software they're helped create.

3. The specific job roles associated with the overall system, such as the people who run the computers and keep the software operating.

4. Controls, which are parts of the software written to help prevent fraud and theft

5. The people who use the software in order to do their jobs

The components of a computer-based information system application are summarized in Figure 4.1. I try to address all the dimensions of the overall system, with particular emphasis on application software development – My primary responsibility as a system analyst.

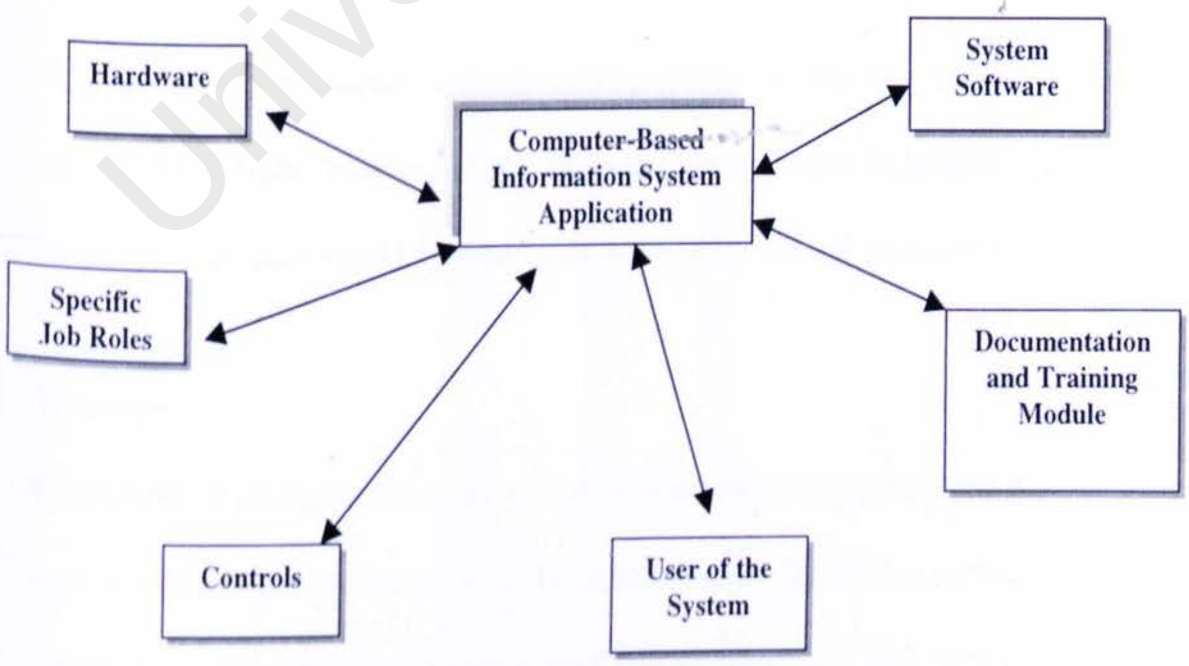

Hardware
System Software
Computer-Based Information System Application
Specific Job Roles
Documentation and Training Module
Controls
User of the System

Figure 4.2 : Components of a Computer-Based Information System Application

My goal is to help the organization to understand and follow the software engineering process that leads to the creation of information systems. As shown in Figure 4.2, proven methodologies, techniques, and tools are central to software engineering processes.

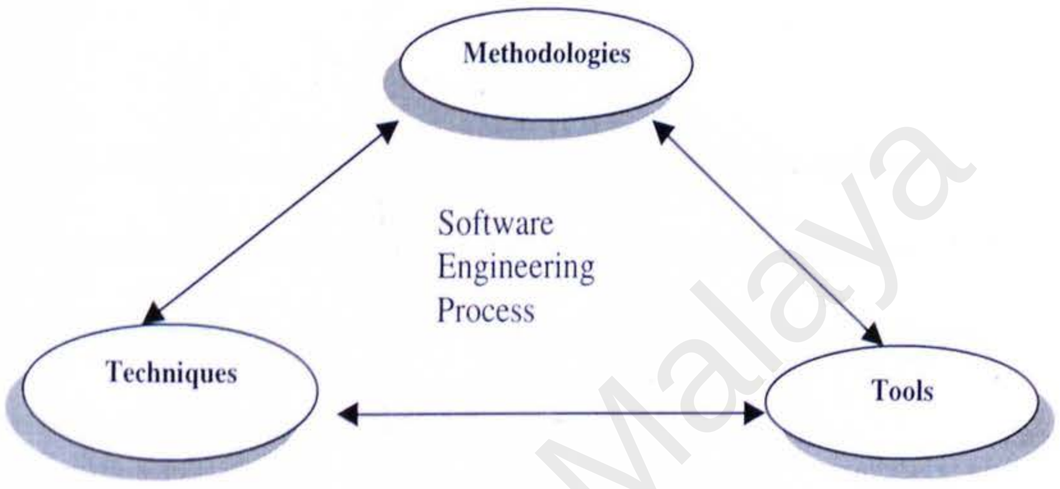


Figure 4.3 : The software engineering process user methodologies, techniques and tools

i. Methodologies

Methodologies are a sequence of step-by-step approaches that help develop the system. Most methodologies incorporate several development techniques, such as direct observations and interviews with users of the current manual system.

ii. Techniques

Techniques are processes that as an analyst, will follow to help ensure that the work is well thought out, complete, and comprehensible. Techniques provide support for a wide range of tasks including thorough interviews with current and future users of the information system to determine what the system should do, planning and managing the activities in a system development projects, diagramming how the system will function, and designing the report, such as government submission reports, the system will generate for its users to perform their jobs.

iii. Tools

Tools are computer programs, such as computer-aided software engineering tools, that make it easy to use specific techniques. These three elements – methodologies, techniques and tools – work together to form an organizational approach to systems analysis and design.

4.6 System and its Parts

A system is an interrelated set of business procedures (or components) used within one business unit, working together of some purpose. For example, a payroll management system in Human Resource Department keeps track of checks of employee data.

A system has nine characteristics, seven of which are shown in Figure 4.3

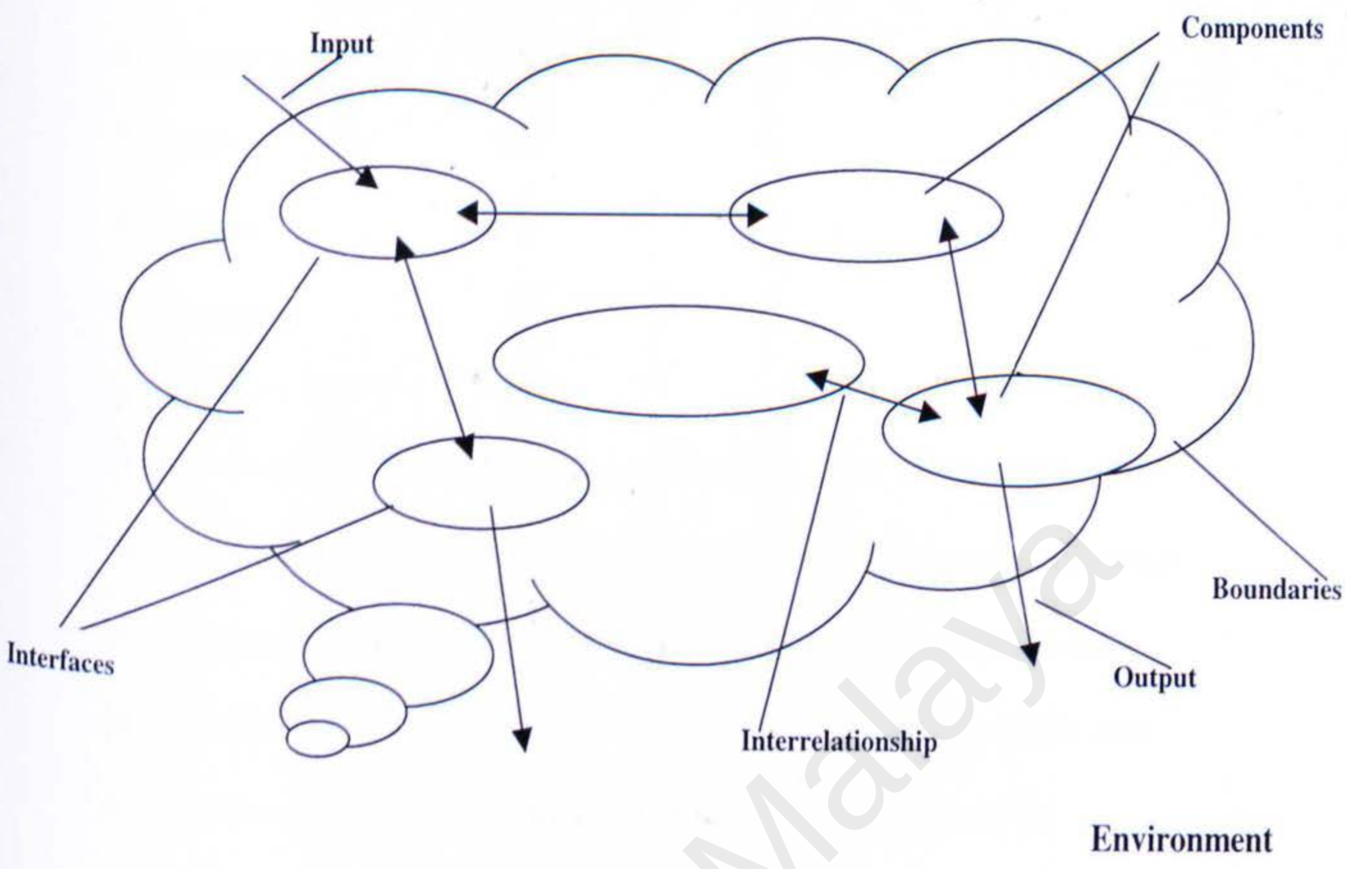


Figure 4.4 : Seven characteristics of a system

A detailed explanation of each characteristics follows, but from Figure 4.3 that we can see that a system exists within a larger world, an environment. A boundary separates the system from its environment. The system takes input from outside, processes it, and sends the resulting output back to its environment, the arrows in Figure 4.3 shown this interaction between the system and the world outside of it.

1. Components
2. Interrelated components
3. Boundary
4. Purpose

5. Environment
6. Interfaces
7. Input
8. Output
9. Constrains

A system is made up of components. A component it either an irreducible part or an aggregate of parts, also called a *subsystem*. The simple concept of a component is very powerful. For example, just as with a payroll management system with proper design, we can updating and changing the system by changing individual components without having to changes throughout the entire system. The components are interrelated, that is, the function of the one is somehow tied to the functions of the others. For example, the work of one component, such as producing a daily report of employee's particular may not progress successfully until the work of another component is finished, such as sorting the employee ID number by date of creation. A system has a boundary within which all of its components are contained and which establishes the limits of a system, separating it from other systems. Components within the boundary can be changed whereas system outside the boundary cannot be changed. All of the components work together to achieve some overall overall purpose for the larger system; the system's reason for existing.

A system exist within as environment – everything outside the system's boundary that influences the system. An information system interacts with its environment by receiving data (raw facts) and information (data process in a useful format). The points at which the system meets its environment are called interfaces, and there are also interfaces between subsystems.

A system must face constraints in its functioning because there are limits (in terms of capacity, speed, or capabilities) to what it can do and how it can achieve it purpose within its environment. Some of these constraints are imposed inside the system (e.g., a limited number of staff available), and others are imposed by the environment (e.g., due dates or regulations). A system takes input from its environment in order to the function. Finally, system returns output to its environment as a result of its functioning and thus achieves its purpose. The system is constrained if electrical power is cut.

4.7 System Development life Cycle

The system development life cycle (SDLC) is a common methodology for system development in many organization. It marks the phases or steps on informaiton system development. Someone has an idea for an information system and what it should do. The organization that will use the system decides to devote the necessary resource to acquiring its.

A careful study is done of how the organization currently handles the work the system will support. A strategy for designing the new system is developed, which is then either built or purchased. Once complete, the system is installed in the organization, and after proper training, the users begin to incorporate the new system into their daily work.

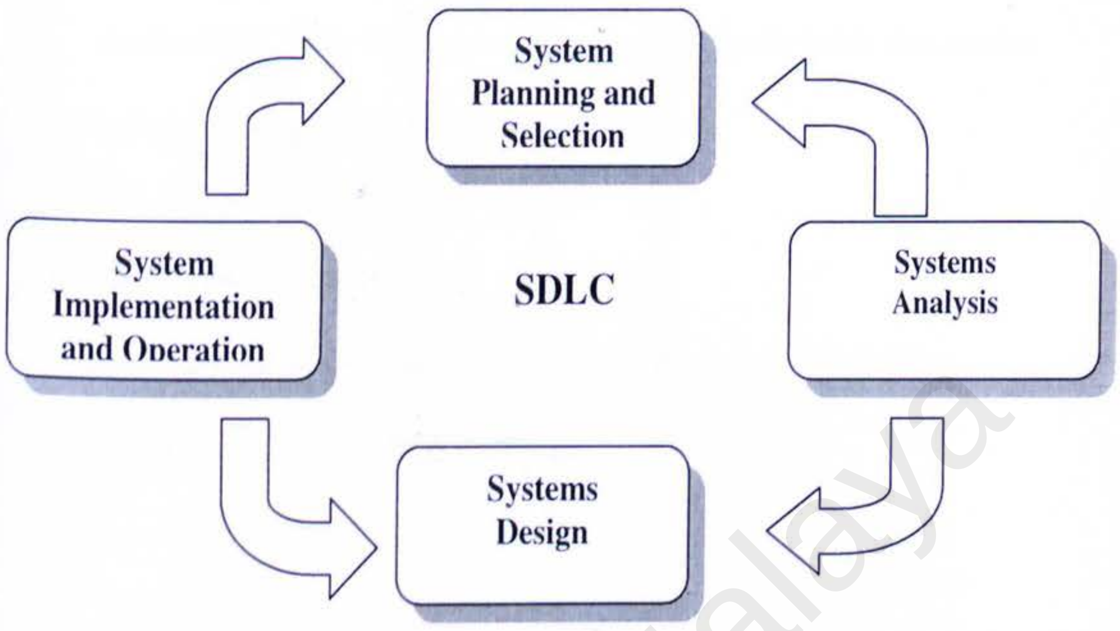

System Planning and Selection
Systems Analysis
Systems Design
System Implementation and Operation
SDLC

Figure 4.5 : The System Development Life Cycle (SDLC)

The Payroll Management System consists of two main modules and that are the :

a) User Module

b) Administrator Module

a) User Module

The users use this module. The functions of this module are to :

i) Access or retrieved required information

In this module, the information required by the users are displayed automatically accordingly to their requirements. Thus, users are only required to choose the given options.

ii) Analyzing the information

Users are able to carry out computations according to their own requirements. They may carried out computations for totals, frequencies and so on using the BOSS software.

iii) Generation of various reports

Users are able to generate reports for the information that they are retrieved by using the BOSS software.

iv) Printing

The option of printing is available for users that wish to print the reports for further references. They are also to print the data that has been retrieved from the databases. This is done using the BOSS software option to print.

v) Password

Users are given the options to change their password anytime to protect their authentication according to their needs. In short, they can change it anytime.

b) Administrator Module

This module is used by the administration for modification and updating the system.

i) User Management

Records of the user can be edited, deleted and modified. New users can be registered.

ii) Database Management

To update any changes to the records of the database, administrators are able to view all the reports in the database.

iii) Administrator Management

New administrator can be added and as existing administrator can be deleted.

iv) Password

Administrators can change their password according to their needs.

4.8 Structure Chart

Structure chart is used to depict high level of abstraction of a specified system. The use of a structured chart is to describe the interaction between independent modules. Major function forms are initial component part of the structure chart, which can be broken into detailed sub-component. Payroll Management System (PMS) is divided into two main components mainly the user module and administrator module.

Users of PMS are linked to all the modules using the hypertext links. Figure below shows the structure of the PMS.

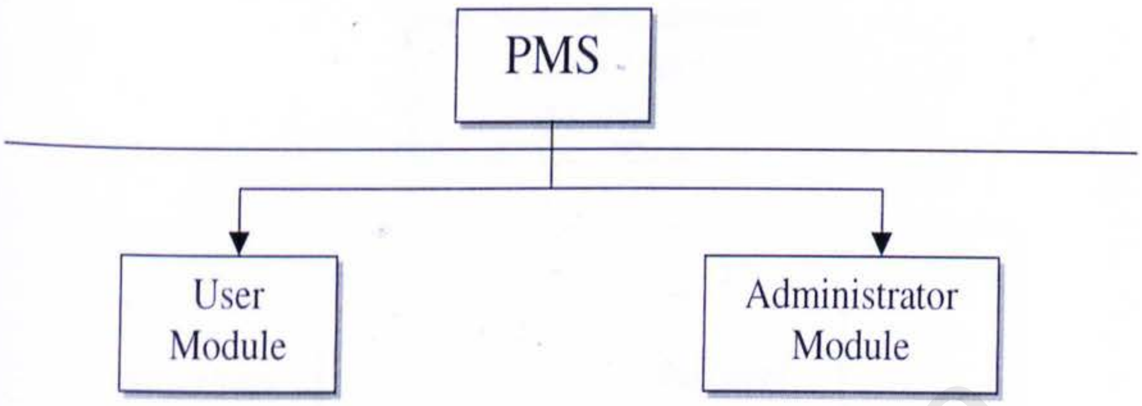


Figure 4.6 : Structure Chart of Payroll Management System

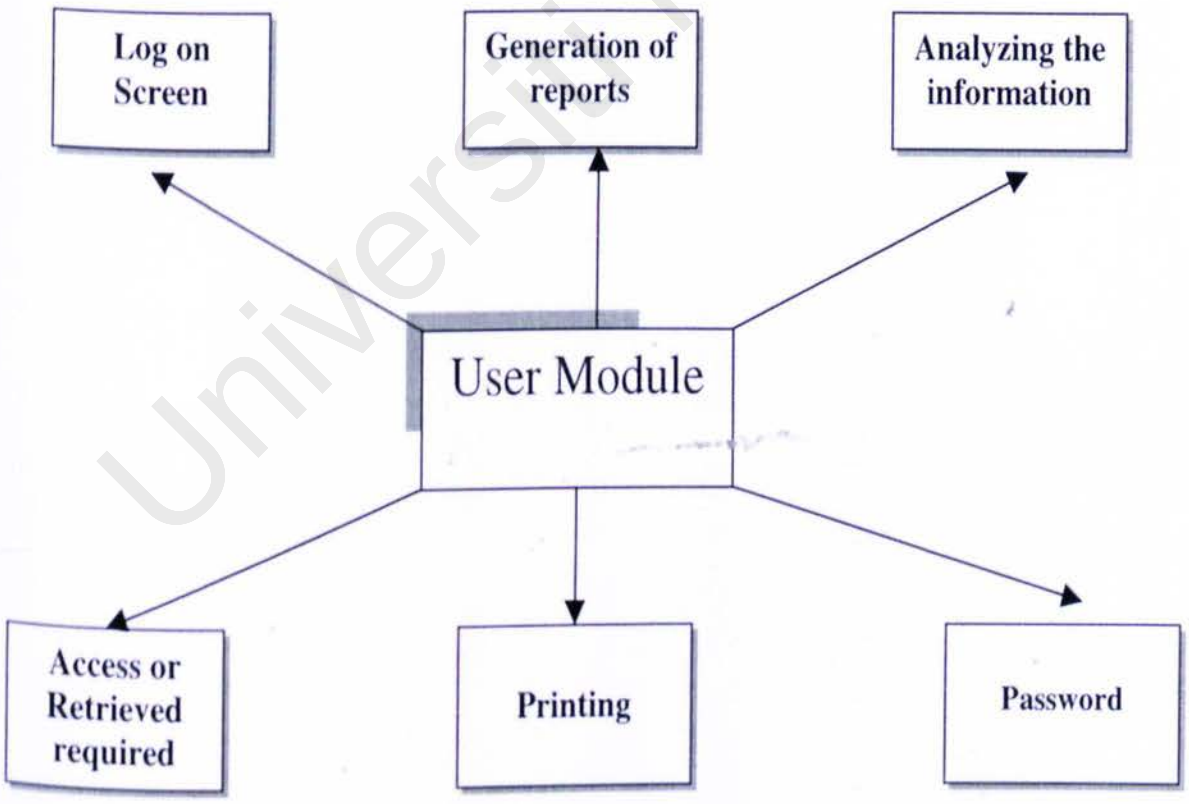
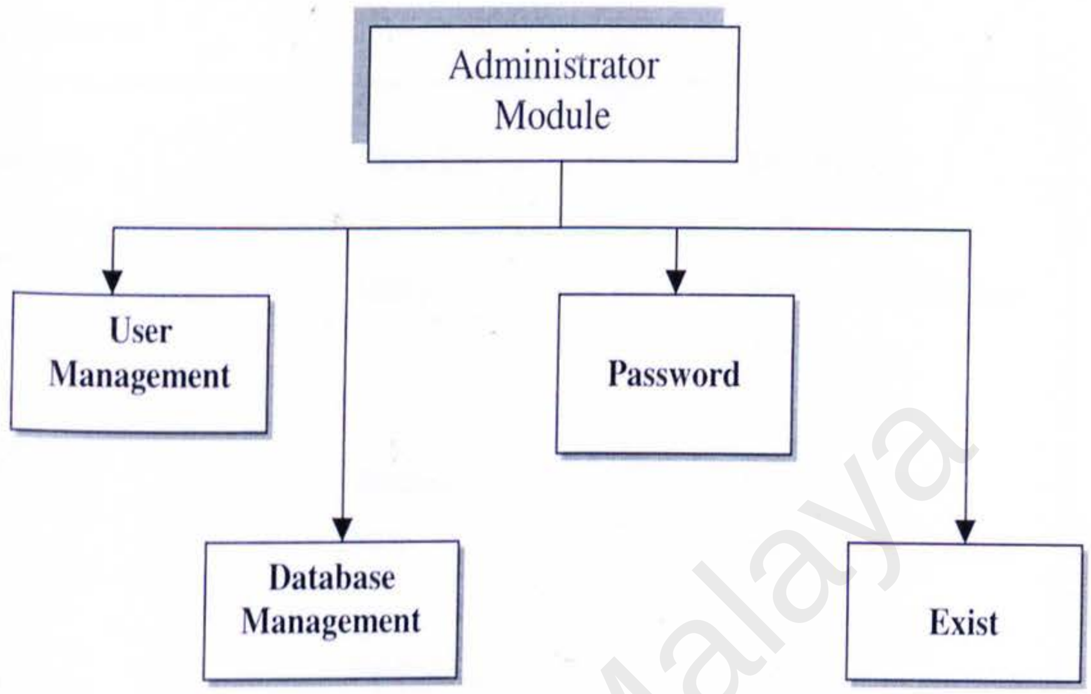


Figure 4.7 : PMS User Module

Figure 4.8 : PMS Administrator Module



4.9 Data Flow Diagram of PMS

DFD is a graphical representation of the data processed of the system. It uses the combination of 4 symbols to create a pictorial depiction. The four symbols are stated below:

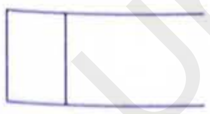
Symbols	Meaning	Description
▭	Entity	Source of destination of the data
○	Process	Occurrence of a transforming process
◇	Connector	something or anything created to join to entities
→	Flow of Data	Movement of data from one point to another
	Data Store	Represent the storage of the data

Figure 4.9 : Data Flow Diagram for the PMS User Module

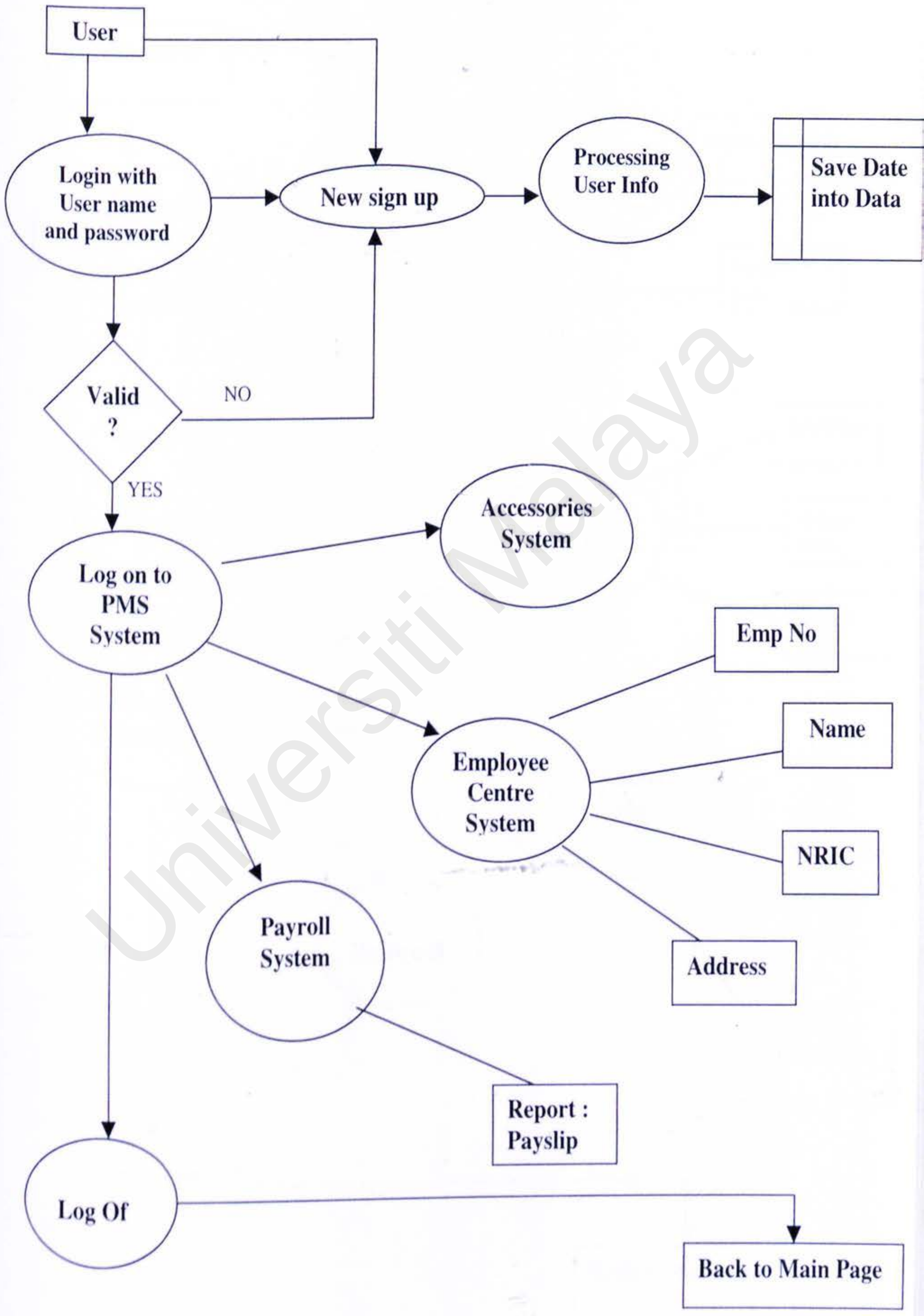
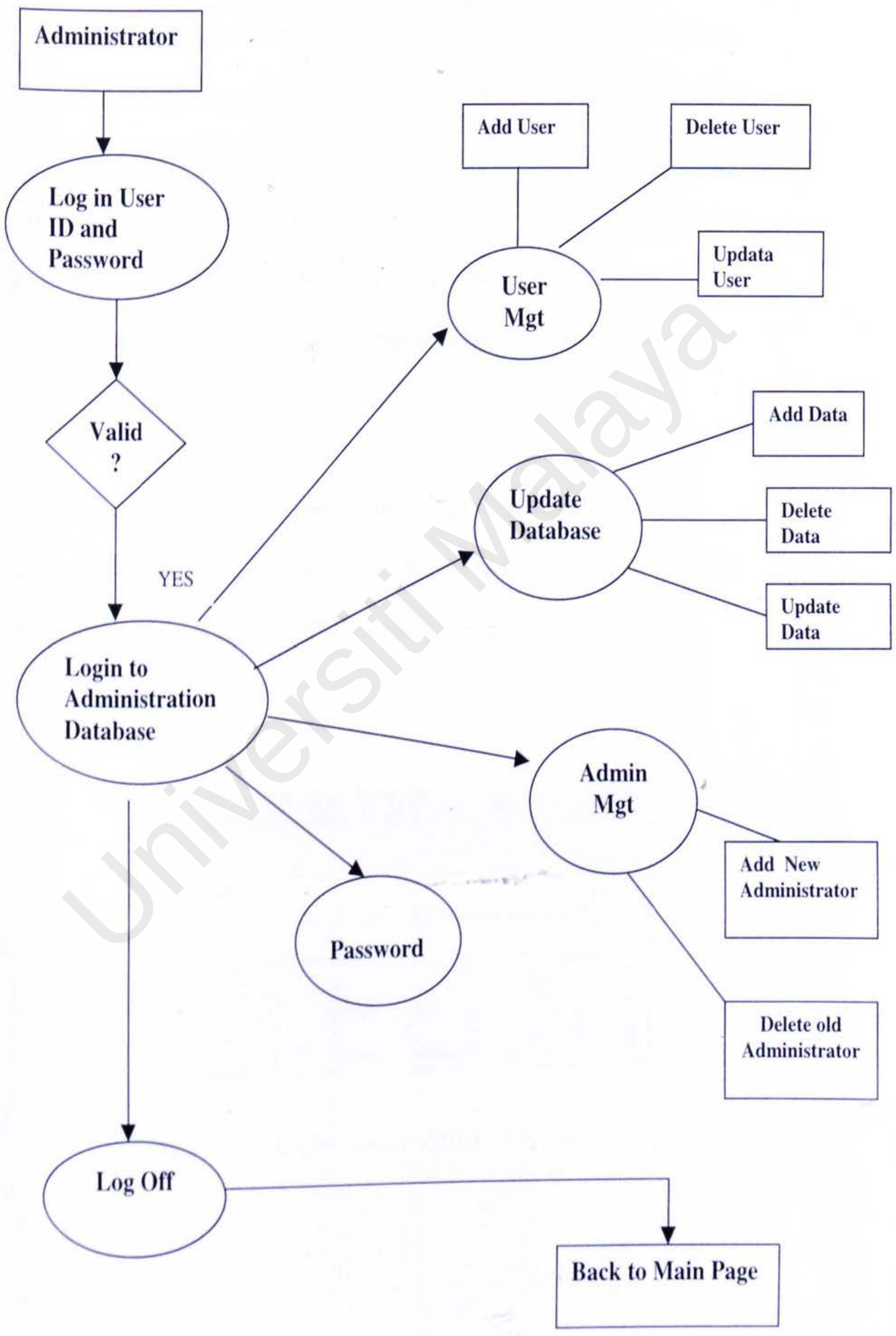


Figure 4.10 : Data Flow Diagram for the PMS Administrator Module



4.10 User Interface Design

User quality of system input determines the qualities of system output. It is vital that input forms and screens be designed with this critical relationship in mind. Well designed input forms and display terminal (VDT) screens should meet the objectiveness for effectiveness, accuracy, ease of use, consistency, simplicity and attractiveness. All of these objectives are attainable through use of basic design principles, knowledge of what is needed as input for the system and understanding of how the user should respond to differentiate elements in the forms and screens and forms.

Screen is another important element in the design stage. A badly designed interface will cause the system to be discarded, irrespective of the functionality that it offers. Knowing this fact PMS is placing more emphasis on he user interface design.

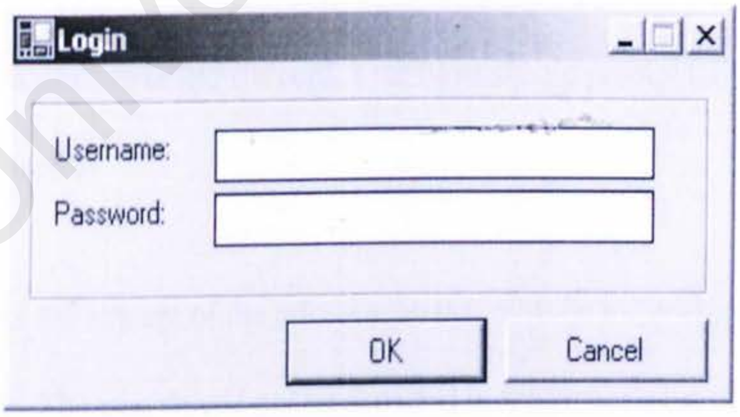


Figure 4.11 : Log on Screen of BOSS software

4.11 Guidelines for General Interactions

These guidelines cover the aspects of data entry, information display and overall system control of BOSS software, BOSS software is developed with these guidelines in mind for a good interface.

i) Be consistent

A consistent format for menu selection, command input, data display and myriad of other functions is typical in the Human Computer Interaction.

ii) Offer meaningful feedback

Provides user with visual auditory feedback to ensure that two-way communications (between user and interface) is effectively established

iii) Verification

Ask for any verification of any-trial but potentially destructive action. If user requests to delete a record, a message should prompt the user for confirmation

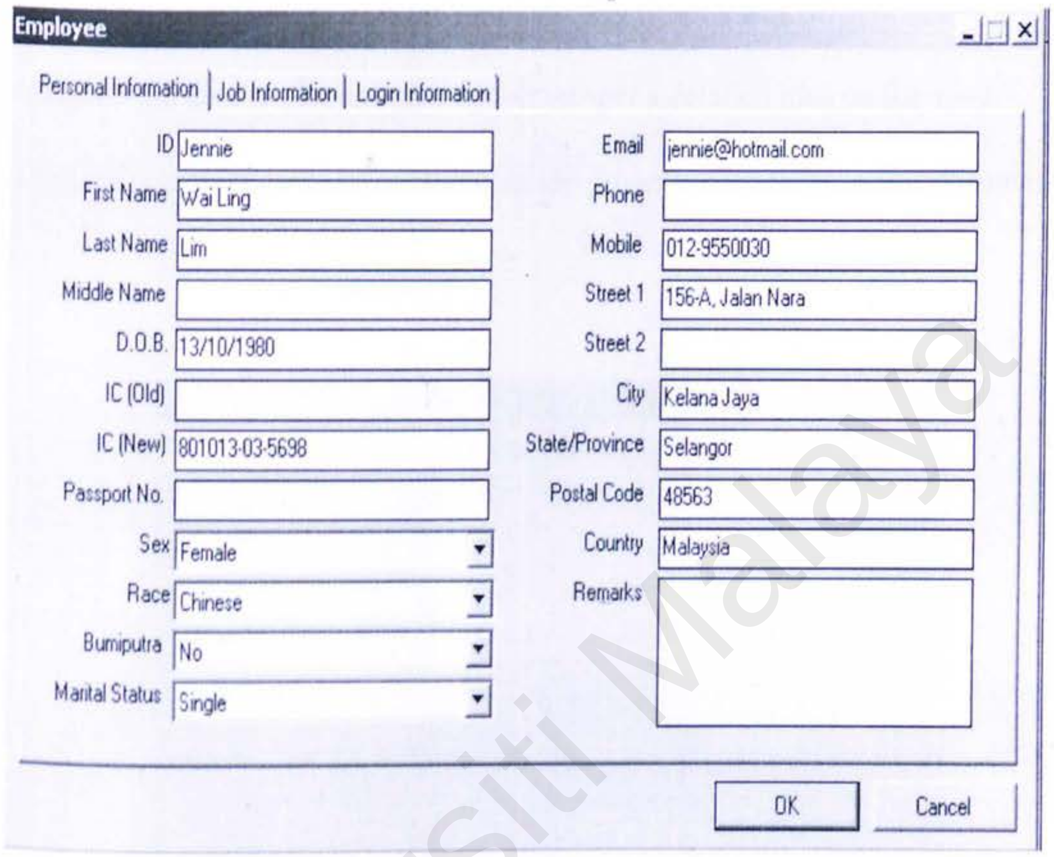
iv) Reduce the amount of the information that must be memorized in between actions. The user should not be expected to remember a list of members or names to be reused in consequent function.

4.12 Storing Data

To store data and the format of data type is often a vital decision making in the design of information and analyzing system. The structure of data has always been an important issue of software or application design, because of the architectural of data will have a profound influence on the architecture of the application that must process it.

Even though it is quite complicating, the design of a database is very important because it can give a huge affect on the performance of data retrieved, updating and query as well in run time period of the system.

Data are stored in the database in a table form. The tables in the employee date are as follow:



ID	Jennie	Email	jennie@hotmail.com
First Name	Wai Ling	Phone	
Last Name	Lim	Mobile	012-9550030
Middle Name		Street 1	156-A, Jalan Nara
D.O.B.	13/10/1980	Street 2	
IC (Old)		City	Kelana Jaya
IC (New)	801013-03-5698	State/Province	Selangor
Passport No.		Postal Code	48563
Sex	Female	Country	Malaysia
Race	Chinese	Remarks	
Bumiputra	No		
Marital Status	Single		

Figure 4.12 : Employee data Main Screen

4.13 Summary

This chapter concentrates on the system design of BOSS. The overview of BOSS, form design, interface design and database design were describes in detail. These designed are able to give the user and developer a detailed idea on the system. The DFD diagram gives an overview of the processes involved in BOSS software.

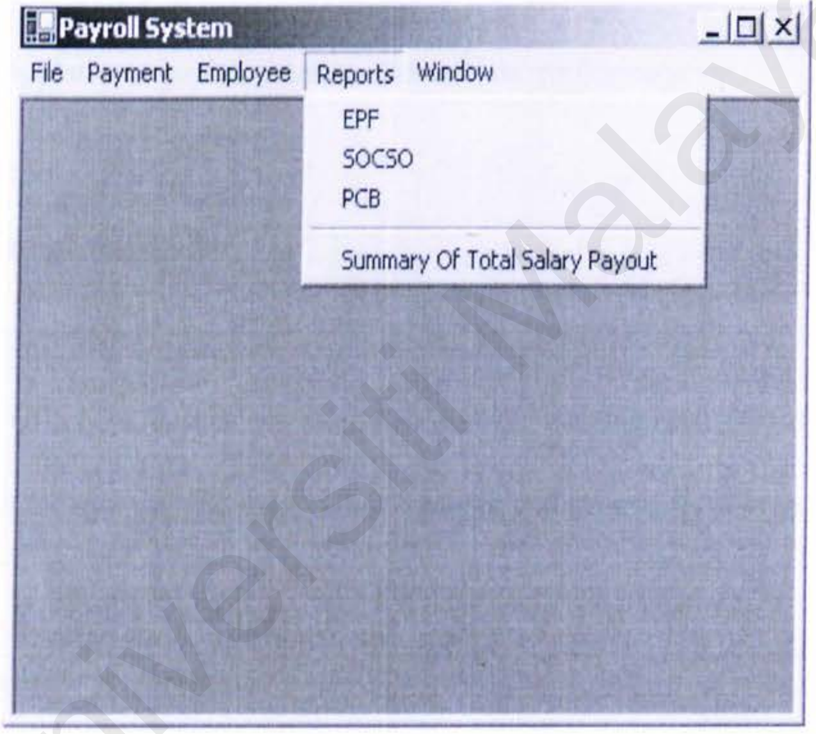


Figure 4.13 : Main Page of BOSS software

Chapter 5 : SYSTEM IMPLEMENTATION

5.1 Introduction

System implementation is a process that takes place after system design phase. It is a process to convert the system requirements into program codes. This phase describes how the initial and revised design was put into the real work.

Under this stage, we transform the design modes of the PMS into workable software. The system implementation of PMS will be divided into two component, platform implementation and module implementation.

5.2 Platform Implementation

The platform implementation includes setting up the operating system which is Microsoft Office 2000. It is very important to have suitable hardware and software in speeding up the system development and make a success to this project.

Software tools needed

Software	Usage	Description
Microsoft Visual Studio. Net	Minimum Requirement	Interface
Microsoft. Net Framework	Minimum Requirement	Interface
MDAC 2.7	Minimum Requirement	Interface
Microsoft SQL Server 2000	System Requirement	Database
Microsoft Office 2000	System Requirement	Operating System

Figure 5.1 : Software tool for Development

5.3 Functions Implementation

Add Function

Implement in the registration unit as Add New User function, employee particular as Add Employee Information function, result unit as Add Employee Result function, meeting unit as Add Meeting function, appointment unit Add appointment function, academic plan unit as Add Academic plan function and attendance unit as add attendance function.

Begin with getting values from the form and submit the form.

Records are added to the database

Notify message about the record added successfully.

Browse / Edit / Update / Delete Function

Implementation in the registration unit as Browse user function, student unit as Browse employee function, result unit as Browse Result Function, Appointment unit as View appointment function, meeting unit as View Meeting function, academic plan unit as View Academic plan function, and attendance unit as Browse attendance function.

Begin by getting the corresponding records from database by entering keyword and display it

Select which record is preferred to be edited or deleted.

Choose 'Edit' or 'Delete' function from the right hand side of the record.

Insert records to be updated in the updating form display and press 'Update'.

The update information will be displayed in the edit page.

Also can run the print function from here.

Search Function

Implementation in registration unit as Search user function, employee particular unit as Search appointment function, meeting unit as search meeting function, attendance unit as search attendance function.

Get the keyboard / keywords from the form as the condition to search for the record in the database.

If exist, then all the corresponding records with be displayed.

Otherwise the system notify the user that the 'record not found'

The function print also implemented here.

5.4 Module Implementation

Payroll management system has 3 main modules

i. Employee Center

ii. Government Module

iii. Salary Report

Each module is implemented using Enterprise Manager with the help of Microsoft Visual Studio. The function under each module was implemented by payroll management system main two users.

i. Administrator

ii. Active Employee

The implementation of each module will be explained according to the users

Administrator

Administrator as in charge in maintaining the data of user information, employee details and result. They can add, edit or delete and search for the information on these 3 modules.

Active Employee

Active employee can view their current month salary slip. The allowances and deduction from the transaction which has been made by Human Resource staff is correct. They may refer to the administrator if there is any error occurred in the their salary slip.

5.5 Steps of System Development

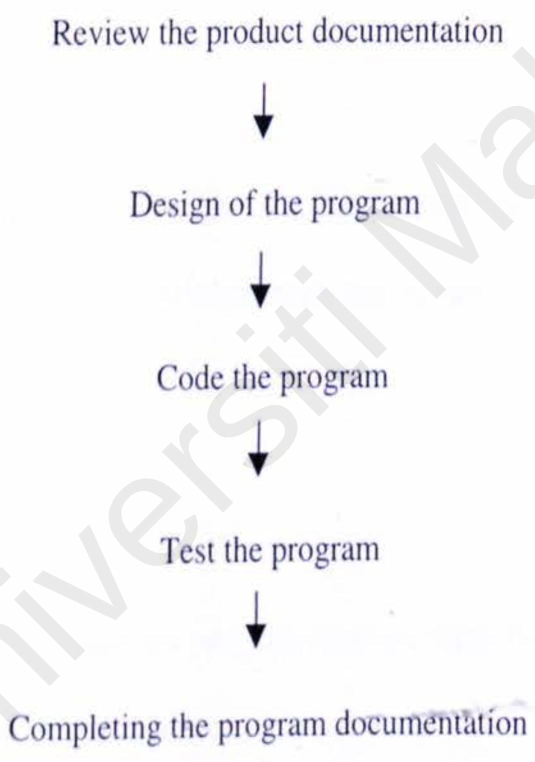

Review the product documentation
Design of the program
Code the program
Test the program
Completing the program documentation

Figure 5.2 : System Development

5.6 Review the product Documentation

Review the product documentation that was prepared during the previous phases. In addition to understand the work better.

5.7 Design of the program

This is the process of what it must do by developing logical solution to the programming problem. The logical solution is a step-by-step solution to a programming pattern.

5.8 Code the program

Process of writing the program instructions that implements the program design. If design is performed in detailed manner, coding can be accomplished mechanically.

5.9 Test the program

Program is tested to ensure the program function correctly before the program processes actual data and produces information on which user will be relying on.

5.10 Documentation of the program

Completing the program is essential for the successful operation and maintenance of the information system. Documentation includes the system's user manual that may be needed by customers.

5.11 Coding Approach

An easy to real source code makes the system easier to maintain and enhance. This coding process translates the system design information programming language, which is a machine-readable form.

5.12 Database Connection

Microsoft SQL Server 2000 is used in this for database connection. Microsoft Access provides the means by which program code access the database.

5.13 The adjustment for implementation

During the development of payroll management system. We have changed the tools and technology used. Among the changes that have been made are :-

- **Programming language**

  We choose Visual Basic 6.0 as our programming language to build the system instead of the previously proposed Java. It is because Visual Basic is easier to learn and the fact that we have learned it before. We realized that Visual Basic combines extraordinary ease of use and great power and flexibility. It can be used in many ways and at many levels from beginners to expert user. It is the easiest computer languages to work with and more understanding.

- **Database**

  We choose Microsoft SQL Server 2000 as our system database because it is already installed at my workstation. In addition, it saves our time in order to look for other database. In fact, SQL Server is used in order to retrieve input or data from the users and also the auditors. Therefore, the auditor will be able to have a view of the record that is transparent to the users.

- **Other adjustments**

  Previously we have proposed a web based application where the user could login to view their data from internet, we are unable to apply this function due to security purposes moreover every active employee would be given a user ID and user password where SQL Server is already installed to every workstation. Other than at their work place, the user might not need to view the data unless during working hours.

- **Database**

  We choose Microsoft SQL Server 2000 as our system database because it is already installed at my workstation. In addition, it saves our time in order to look for other database. In fact, SQL Server is used in order to retrieve input or data from the users and also the auditors. Therefore, the auditor will be able to have a view of the record that is transparent to the users.

- **Other adjustments**

  Previously we have proposed a web based application where the user could login to view their data from internet, we are unable to apply this function due to security purposes moreover every active employee would be given a user ID and user password where SQL Server is already installed to every workstation. Other than at their work place, the user might not need to view the data unless during working hours.

Chapter 6 : SYSTEM TESTING

Introduction

Testing is a process of executing a program with the intention of finding an error. Therefore changes and adjustment can be taken care of immediately.

6.1.1 Unit Testing

Unit testing was done where control paths and tested to uncover error. The first step is to examine the program code by reading through it. All of the coding is made sure there's no debug so that the paths and the flow of the application still fluently browsed. Finally, test cases are developed to show that the input is properly converted to the desired output.

6.1.2 Integrating Testing

This is an approached where the program structure was constructed at the same test are conducted to uncover errors associated with interfacing. Testing the interfaces explore how components interact with each other. Error with be corrected before processing to the next integration.

6.1.3 System Testing

It is the final phase. This process ensures that all units in the module will function accordingly when integrated and have fully satisfied its functional requirements. It reveals bugs that cannot be attributed to individual components or to the interaction among components and other objects.

6.2 System Evaluation

Evaluation was implemented to consider carefully before effectiveness can be concluded. Field test evaluation was carried out when the information system was believed to be of the final draft quality. If problems were identified, additional changes may be made. However the informal evaluation conducted at this point should ensure that the information system is completed or minimal changes will be required.

6.3 Coding the Program

Coding is the process whereby the physical design specifications created by the design team are turned into working computer code by the programming team. Depending on the size and complexity of the system

Open Login Menu

After connecting to the database, the data from the database table needs to be kept somewhere. Example of code to open table is shown at below:

FrmLogin

```
Private Sub btnOK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnOK.Click
  username = Trim(txtUsername.Text)
  Dim password As String = Trim(txtPassword.Text)

  Dim strSearch As String
  Dim dt As New DataTable()

  strSearch = "SELECT * " & _
          "FROM Login l, Employee e " & _
          "WHERE Username = '" & username & "' " & _
          "AND l.Username = e.ID " & _
          "AND PayStatus = 'Active'"

  dt = mObjDAL.ExecDatasetTx(strSearch).Tables(0)
  If dt.Rows.Count = 0 Then
     ErrorProvider.SetError(txtUsername, "Invalid username")
  Else
     If dt.Rows(0).Item("Password").ToString = password Then
        userlevel = dt.Rows(0).Item("Level").ToString

        Dim frmMenu As New frmMenu()
        Me.Close()
        If userlevel = "Normal User" Then
           frmMenu.mnuEmployee.Enabled = False
           frmMenu.mnuProcessPay.Enabled = False
        End If
        frmMenu.ShowDialog()
     Else
        ErrorProvider.SetError(txtPassword, "Invalid password")
     End If
     End If

End Sub
```

Database Connection

The database connection is important that must be done before we start the coding of Visual Basic programming appliaction file. Information such as database name must be specified correctly.

After configuration, the codes can directly connect to the server and communicate with the database. The coding on FrmMenu that shows the database connectivity.

```
FrmMenu

   Private mReportType As ReportTypeResult
   Private mPPreviewDlg As New PrintPreviewDialog()
   Private mPrintDlg As New PrintDialog()
   Private mObjPM As New PrintManagerImpl()
   Private miPageMarginLeft As Integer = 50
   Private mObjDAL As New DataObject()

   Private Enum ReportTypeResult As Integer
      EPF = 1
      SOCSO = 2
      PCB = 3
      Summary = 4
   End Enum

   Private Sub mnuCloseAll_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
mnuCloseAll.Click
      Dim MdiChild As Form

      For Each MdiChild In Me.MdiChildren
         MdiChild.Close()
      Next MdiChild
   End Sub
```

```
    Private Sub mnuCloseActive_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
mnuCloseActive.Click
        Dim MdiActiveChild As Form

        MdiActiveChild = Me.ActiveMdiChild

        If Not MdiActiveChild Is Nothing Then
            MdiActiveChild.Close()
        End If
    End Sub

    Private Sub mnuCascade_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
mnuCascade.Click
        Me.LayoutMdi(MdiLayout.Cascade)
    End Sub

    Private Sub mnuTileH_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles mnuTileH.Click
        Me.LayoutMdi(MdiLayout.TileHorizontal)
    End Sub

    Private Sub mnuTileV_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles mnuTileV.Click
        Me.LayoutMdi(MdiLayout.TileVertical)
    End Sub

    Private Sub mnuLogin_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
mnuLogin.Click
        Dim frmLogin As New frmLogin()

        Me.Dispose()
        frmLogin.ShowDialog()
    End Sub

    Private Sub mnuExit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles mnuExit.Click
        End
    End Sub

    Private Sub mnuEmployee_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
mnuEmployee.Click
        Dim frmEmployeeList As New frmEmployeeList()
        Dim MdiChild As Form
        Dim Found As Boolean

        For Each MdiChild In Me.MdiChildren
            If TypeOf MdiChild Is frmEmployeeList Then
                Found = True
                MdiChild.Activate()
                Exit For
            End If
        Next

        If Found = False Then
            frmEmployeeList.MdiParent = Me
            frmEmployeeList.Show()
        End If
    End Sub

    Private Sub mnuPaySlip_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
mnuPaySlip.Click
        If userlevel = "Administrator" Then
            Dim frmEmployeePaymentList As New frmEmployeePaymentList()
            Dim MdiChild As Form
            Dim Found As Boolean

            For Each MdiChild In Me.MdiChildren
                If TypeOf MdiChild Is frmEmployeePaymentList Then
                    Found = True
                    MdiChild.Activate()
                    Exit For
                End If
            Next

            If Found = False Then
                frmEmployeePaymentList.MdiParent = Me
                frmEmployeePaymentList.Show()
            End If
        Else
            Dim frmEmployeePayment As New frmEmployeePayment()
            Dim MdiChild As Form
            Dim Found As Boolean

            For Each MdiChild In Me.MdiChildren
                If TypeOf MdiChild Is frmEmployeePayment Then
                    Found = True
                    MdiChild.Activate()
                    Exit For
                End If
            Next

            If Found = False Then
                frmEmployeePayment.MdiParent = Me
                frmEmployeePayment.ID = username
                frmEmployeePayment.Show()
            End If
        End If
    End Sub

    Private Sub mnuProcessPay_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
mnuProcessPay.Click
        Dim frmProcessPay As New frmProcessPay()
        Dim MdiChild As Form
        Dim Found As Boolean

        For Each MdiChild In Me.MdiChildren
            If TypeOf MdiChild Is frmProcessPay Then
                Found = True
                MdiChild.Activate()
                Exit For
            End If
        Next

        If Found = False Then
            frmProcessPay.MdiParent = Me
            frmProcessPay.Show()
        End If
    End Sub

    Private Sub mnuEPF_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles mnuEPF.Click
        mReportType = ReportTypeResult.EPF
        Try
            mPPreviewDlg = New PrintPreviewDialog()
            mPPreviewDlg.Size = New System.Drawing.Size(700, 650)
            mPPreviewDlg.StartPosition = FormStartPosition.Manual
            mPPreviewDlg.Location = New System.Drawing.Point(30, 30)

            mPPreviewDlg.PrintPreviewControl.AutoZoom = True
            mPPreviewDlg.PrintPreviewControl.Zoom = 0.8
            mPPreviewDlg.Text &= " - " & "KWSP"
            mPPreviewDlg.Document = Me.PDocument
            mPPreviewDlg.MdiParent = Me.MdiParent
            mPPreviewDlg.ShowDialog()
        Catch Err As Exception
            MsgBox(Err.Message, MsgBoxStyle.Exclamation, )
        End Try
    End Sub

    Private Sub PDocument_PrintPage(ByVal sender As System.Object, ByVal ev As
System.Drawing.Printing.PrintPageEventArgs) Handles PDocument.PrintPage
        If mReportType = ReportTypeResult.EPF Then
    '
    ' CONSTRUCT DATASET
    '
    Dim ds As New DataSet()

    Dim dtHeader As DataTable = GetEPFReportHeader.Copy
    Dim dtDetailsHeader As DataTable = GetEPFDetailsHeader.Copy
    Dim dtDetails As DataTable = GetEPFDetails.Copy
    Dim dtTotal As DataTable = GetTotals.Copy
    Dim dtMainTotal As DataTable = GetMainTotals.Copy

    ds.Tables.Add(dtHeader)
    ds.Tables.Add(dtDetailsHeader)
    ds.Tables.Add(dtDetails)
    ds.Tables.Add(dtTotal)
    ds.Tables.Add(dtMainTotal)

    '
    ' ASSIGN DATASET TO PRINTMANAGER
    '

    mObjPM.LoadDataSource(ds, ev)

    FormatEPFReportHeader(dtHeader.TableName)
    FormatEPFHeaderDetails(dtDetailsHeader.TableName)
    FormatEPFDetails(dtDetails.TableName)
    FormatEPFTotal(dtTotal.TableName)
    FormatEPFMainTotal(dtMainTotal.TableName)
    '
    ' PRINT DOCUMENT
    '
    mObjPM.PrintDocument(ev, False)
ElseIf mReportType = ReportTypeResult.SOCSO Then
    '
    ' CONSTRUCT DATASET
    '
    Dim ds As New DataSet()

    Dim dtHeader As DataTable = GetSOCSOReportHeader.Copy
    Dim dtDetailsHeader As DataTable = GetSOCSODetailsHeader.Copy
    Dim dtDetails As DataTable = GetSOCSODetails.Copy
    Dim dtTotal As DataTable = GetSOCSOTotals.Copy

    ds.Tables.Add(dtHeader)
    ds.Tables.Add(dtDetailsHeader)
    ds.Tables.Add(dtDetails)
    ds.Tables.Add(dtTotal)

    '
    ' ASSIGN DATASET TO PRINTMANAGER
    '

    mObjPM.LoadDataSource(ds, ev)

    FormatSOCSOReportHeader(dtHeader.TableName)
    FormatSOCSOHeaderDetails(dtDetailsHeader.TableName)
    FormatSOCSODetails(dtDetails.TableName)
    FormatSOCSOTotal(dtTotal.TableName)
    '
    ' PRINT DOCUMENT
    '
    mObjPM.PrintDocument(ev, False)

ElseIf mReportType = ReportTypeResult.PCB Then
    '
    ' CONSTRUCT DATASET
    '
    Dim ds As New DataSet()

    Dim dtHeader As DataTable = GetPCBReportHeader.Copy
    Dim dtDetailsHeader As DataTable = GetPCBDetailsHeader.Copy
```

```
            Dim dtDetails As DataTable = GetPCBDetails.Copy
            Dim dtTotal As DataTable = GetPCBTotals.Copy

            ds.Tables.Add(dtHeader)
            ds.Tables.Add(dtDetailsHeader)
            ds.Tables.Add(dtDetails)
            ds.Tables.Add(dtTotal)

            '
            ' ASSIGN DATASET TO PRINTMANAGER
            '

            mObjPM.LoadDataSource(ds, ev)

            FormatPCBReportHeader(dtHeader.TableName)
            FormatPCBHeaderDetails(dtDetailsHeader.TableName)
            FormatPCBDetails(dtDetails.TableName)
            FormatPCBTotal(dtTotal.TableName)
            '
            ' PRINT DOCUMENT
            '
            mObjPM.PrintDocument(ev, False)

        ElseIf mReportType = ReportTypeResult.Summary Then
            '
            '  CONSTRUCT DATASET
            '
            Dim ds As New DataSet()

            Dim dtHeader As DataTable = GetSummaryReportHeader.Copy
            Dim dtDetailsHeader As DataTable = GetSummaryDetailsHeader.Copy
            Dim dtDetails As DataTable = GetSummaryDetails.Copy
            Dim dtTotal As DataTable = GetSummaryTotals.Copy

            ds.Tables.Add(dtHeader)
            ds.Tables.Add(dtDetailsHeader)
            ds.Tables.Add(dtDetails)
            ds.Tables.Add(dtTotal)

            '
            ' ASSIGN DATASET TO PRINTMANAGER
            '

            mObjPM.LoadDataSource(ds, ev)

            FormatSummaryReportHeader(dtHeader.TableName)
            FormatSummaryHeaderDetails(dtDetailsHeader.TableName)
            FormatSummaryDetails(dtDetails.TableName)
            FormatSummaryTotal(dtTotal.TableName)
            '
            ' PRINT DOCUMENT
            '
            mObjPM.PrintDocument(ev, False)

        End If
    End Sub

    EPF

#Region " Report Header "

    Private Function GetEPFReportHeader() As DataTable
        Dim dtHeader As New DataTable()

        Dim dr As DataRow

        dtHeader.TableName = "Report Header"
        dtHeader.Columns.Add("Captions")

        dr = dtHeader.NewRow
        dr.Item("Captions") = "KUMPULAN WANG SIMPANAN PEKERJA"
        dtHeader.Rows.Add(dr)

        dr = dtHeader.NewRow
        dr.Item("Captions") = "BORANG A"
        dtHeader.Rows.Add(dr)

        dr = dtHeader.NewRow
        dr.Item("Captions") = "JADUAL CARUMAN BULAN " & Now.Month & ", " & Now.Year
        dtHeader.Rows.Add(dr)
        dtHeader.TableName = "header"
        Return dtHeader
    End Function

    Private Sub FormatEPFReportHeader(ByVal _TableName As String)
        mObjPM.SetTableAttribute(_TableName, AttributeType.LINEBREAK, "2")
        mObjPM.SetTableAttribute(_TableName, AttributeType.ALIGN, "center")
        mObjPM.SetTableAttribute(_TableName, AttributeType.FONT, ",16,bold")
        mObjPM.SetTableAttribute(_TableName, AttributeType.POSX, miPageMarginLeft.ToString)
        mObjPM.SetColumnAttribute(_TableName, 0, AttributeType.WIDTH, "700")
    End Sub
#End Region

#Region " Report Details Header "
    Private Function GetEPFDetailsHeader() As DataTable
        '
        ' Create Report Header
        '
        Dim dr As DataRow
        Dim dtHeader As New DataTable()

        dtHeader.TableName = "Details Header"
        dtHeader.Columns.Add("NoAhli")
        dtHeader.Columns.Add("NoKadPengenalan")
        dtHeader.Columns.Add("Name")
        dtHeader.Columns.Add("Majikan")
        dtHeader.Columns.Add("Pekerja")

        dr = dtHeader.NewRow
        dr.Item("NoAhli") = "NO. AHLI"
        dr.Item("NoKadPengenalan") = "NO KAD PENGENALAN"
        dr.Item("Name") = "NAME"
        dr.Item("Majikan") = "MAJIKAN"
        dr.Item("Pekerja") = "PEKERJA"
        dtHeader.Rows.Add(dr)
        dtHeader.TableName = "header details"
        Return dtHeader
    End Function

    Private Sub FormatEPFHeaderDetails(ByVal _TableName As String)
        mObjPM.SetTableAttribute(_TableName, AttributeType.LINEBREAK, "1")
        mObjPM.SetTableAttribute(_TableName, AttributeType.ALIGN, "LEFT")
        mObjPM.SetTableAttribute(_TableName, AttributeType.FONT, ",12,bold")
        mObjPM.SetTableAttribute(_TableName, AttributeType.POSX, miPageMarginLeft.ToString)
        mObjPM.SetColumnAttribute(_TableName, 0, AttributeType.WIDTH, "100")
        mObjPM.SetColumnAttribute(_TableName, 1, AttributeType.WIDTH, "200")
        mObjPM.SetColumnAttribute(_TableName, 2, AttributeType.WIDTH, "200")
        mObjPM.SetColumnAttribute(_TableName, 3, AttributeType.ALIGN, "right")
        mObjPM.SetColumnAttribute(_TableName, 4, AttributeType.ALIGN, "right")
        mObjPM.SetColumnAttribute(_TableName, 3, AttributeType.WIDTH, "100")
        mObjPM.SetColumnAttribute(_TableName, 4, AttributeType.WIDTH, "100")
    End Sub
#End Region

#Region " Details Section "

    Private Function GetEPFDetails() As DataTable
        Dim str As String
        Dim dt As New DataTable()
        str = "SELECT e.EPFNo, NewIC, LastName + ' ' + FirstName AS Name, h.EmployeeAmount, h.EmployerAmount " 
& _
            "FROM Employee e, History h " & _
            "WHERE e.ID = h.EmployeeID " & _
            "AND h.Type = 'Deduction' " & _
            "AND h.Name = 'EPF' " & _
            "AND h.Month = '" & Now.Month & "' " & _
            "AND h.Year = '" & Now.Year & "'"

        dt = mObjDAL.ExecDatasetTx(str).Tables(0)
        dt.TableName = "details"

        Return dt
    End Function

    Private Sub FormatEPFDetails(ByVal _TableName As String)
        'mObjPM.SetTableAttribute(_TableName, AttributeType.LINEBREAK, "1")
        mObjPM.SetTableAttribute(_TableName, AttributeType.ALIGN, "LEFT")
        mObjPM.SetTableAttribute(_TableName, AttributeType.FONT, ",12,")
        mObjPM.SetTableAttribute(_TableName, AttributeType.POSX, miPageMarginLeft.ToString)
        mObjPM.SetColumnAttribute(_TableName, 0, AttributeType.WIDTH, "100")
        mObjPM.SetColumnAttribute(_TableName, 1, AttributeType.WIDTH, "200")
        mObjPM.SetColumnAttribute(_TableName, 2, AttributeType.WIDTH, "200")
        mObjPM.SetColumnAttribute(_TableName, 3, AttributeType.ALIGN, "right")
        mObjPM.SetColumnAttribute(_TableName, 4, AttributeType.ALIGN, "right")
        mObjPM.SetColumnAttribute(_TableName, 3, AttributeType.WIDTH, "100")
        mObjPM.SetColumnAttribute(_TableName, 4, AttributeType.WIDTH, "100")
    End Sub
#End Region

#Region " TOTALS "
    Private Function GetTotals() As DataTable
        Dim str As String
        Dim dt As New DataTable()

        str = "SELECT '' AS Nothing, SUM(h.EmployeeAmount) AS Employee, SUM(h.EmployerAmount) AS Employer " 
& _
            "FROM Employee e, History h " & _
            "WHERE e.ID = h.EmployeeID " & _
            "AND h.Type = 'Deduction' " & _
            "AND h.Name = 'EPF' " & _
            "AND h.Month = '" & Now.Month & "' " & _
            "AND h.Year = '" & Now.Year & "'"

        dt = mObjDAL.ExecDatasetTx(str).Tables(0)
        dt.Rows(0).Item("Nothing") = "JUMLAH"
        dt.TableName = "total"

        Return dt
    End Function

    Private Sub FormatEPFTotal(ByVal _TableName As String)
        mObjPM.SetTableAttribute(_TableName, AttributeType.LINEBREAK, "1")
        mObjPM.SetTableAttribute(_TableName, AttributeType.ALIGN, "Right")
        mObjPM.SetTableAttribute(_TableName, AttributeType.FONT, ",12,Bold")
        mObjPM.SetTableAttribute(_TableName, AttributeType.POSX, miPageMarginLeft.ToString)
        mObjPM.SetColumnAttribute(_TableName, 0, AttributeType.WIDTH, "500")
        mObjPM.SetColumnAttribute(_TableName, 1, AttributeType.WIDTH, "100")
        mObjPM.SetColumnAttribute(_TableName, 2, AttributeType.WIDTH, "100")
    End Sub
#End Region

#Region " MAIN TOTALS "
    Private Function GetMainTotals() As DataTable
        Dim str As String
        Dim dt As New DataTable()

        str = "SELECT '' AS Nothing, SUM(h.EmployeeAmount) + SUM(h.EmployerAmount) AS Main " & _
            "FROM Employee e, History h " & _
            "WHERE e.ID = h.EmployeeID " & _
            "AND h.Type = 'Deduction' " & _
            "AND h.Name = 'EPF' " & _
            "AND h.Month = '" & Now.Month & "' " & _
            "AND h.Year = '" & Now.Year & "'"

        dt = mObjDAL.ExecDatasetTx(str).Tables(0)
        dt.Rows(0).Item("Nothing") = "JUMLAH BESAR"
        dt.TableName = "main total"

        Return dt
    End Function

    Private Sub FormatEPFMainTotal(ByVal _TableName As String)
        mObjPM.SetTableAttribute(_TableName, AttributeType.LINEBREAK, "1")
        mObjPM.SetTableAttribute(_TableName, AttributeType.ALIGN, "Right")
        mObjPM.SetTableAttribute(_TableName, AttributeType.FONT, ",12,Bold")
        mObjPM.SetTableAttribute(_TableName, AttributeType.POSX, miPageMarginLeft.ToString)
        mObjPM.SetColumnAttribute(_TableName, 0, AttributeType.WIDTH, "500")
        mObjPM.SetColumnAttribute(_TableName, 1, AttributeType.ALIGN, "center")
        mObjPM.SetColumnAttribute(_TableName, 1, AttributeType.WIDTH, "200")
    End Sub
#End Region

    'SOCSO

#Region " Report Header "

    Private Function GetSOCSOReportHeader() As DataTable
        Dim dtHeader As New DataTable()

        Dim dr As DataRow

        dtHeader.TableName = "Report Header"
        dtHeader.Columns.Add("Captions")

        dr = dtHeader.NewRow
        dr.Item("Captions") = "PERTUBUHAN KESELAMATAN SOSIAL"
        dtHeader.Rows.Add(dr)

        dr = dtHeader.NewRow
        dr.Item("Captions") = "JADUAL CARUMAN BULANAN - BORANG 8A"
        dtHeader.Rows.Add(dr)

        dr = dtHeader.NewRow
        dr.Item("Captions") = "UNTUK CARUMAN BULAN " & Now.Month & ", " & Now.Year
        dtHeader.Rows.Add(dr)
        dtHeader.TableName = "header"
        Return dtHeader
    End Function

    Private Sub FormatSOCSOReportHeader(ByVal _TableName As String)
        mObjPM.SetTableAttribute(_TableName, AttributeType.LINEBREAK, "2")
        mObjPM.SetTableAttribute(_TableName, AttributeType.ALIGN, "center")
        mObjPM.SetTableAttribute(_TableName, AttributeType.FONT, ",16,bold")
        mObjPM.SetTableAttribute(_TableName, AttributeType.POSX, miPageMarginLeft.ToString)
        mObjPM.SetColumnAttribute(_TableName, 0, AttributeType.WIDTH, "700")
    End Sub
#End Region

#Region " Report Details Header "
    Private Function GetSOCSODetailsHeader() As DataTable
        '
        ' Create Report Header
        '
        Dim dr As DataRow
        Dim dtHeader As New DataTable()

        dtHeader.TableName = "Details Header"
        dtHeader.Columns.Add("NoAhli")
        dtHeader.Columns.Add("NoKadPengenalan")
        dtHeader.Columns.Add("Name")
        dtHeader.Columns.Add("Caruman")

        dr = dtHeader.NewRow
        dr.Item("NoAhli") = "NO. AHLI"
        dr.Item("NoKadPengenalan") = "NO KAD PENGENALAN"
        dr.Item("Name") = "NAME"
        dr.Item("Caruman") = "CARUMAN"
        dtHeader.Rows.Add(dr)
        dtHeader.TableName = "header details"
        Return dtHeader
    End Function

    Private Sub FormatSOCSOHeaderDetails(ByVal _TableName As String)
        mObjPM.SetTableAttribute(_TableName, AttributeType.LINEBREAK, "1")
        mObjPM.SetTableAttribute(_TableName, AttributeType.ALIGN, "LEFT")
        mObjPM.SetTableAttribute(_TableName, AttributeType.FONT, ",12,bold")
        mObjPM.SetTableAttribute(_TableName, AttributeType.POSX, miPageMarginLeft.ToString)
        mObjPM.SetColumnAttribute(_TableName, 0, AttributeType.WIDTH, "200")
        mObjPM.SetColumnAttribute(_TableName, 1, AttributeType.WIDTH, "200")
        mObjPM.SetColumnAttribute(_TableName, 2, AttributeType.WIDTH, "200")
        mObjPM.SetColumnAttribute(_TableName, 3, AttributeType.ALIGN, "right")
        mObjPM.SetColumnAttribute(_TableName, 3, AttributeType.WIDTH, "100")
    End Sub
#End Region

#Region " Details Section "

    Private Function GetSOCSODetails() As DataTable
        Dim str As String
        Dim dt As New DataTable()

        str = "SELECT e.SOCSONo, NewIC, LastName + ' ' + FirstName AS Name, h.EmployeeAmount " & _
            "FROM Employee e, History h " & _
            "WHERE e.ID = h.EmployeeID " & _
            "AND h.Type = 'Deduction' " & _
            "AND h.Name = 'SOCSO' " & _
            "AND h.Month = '" & Now.Month & "' " & _
            "AND h.Year = '" & Now.Year & "'"

        dt = mObjDAL.ExecDatasetTx(str).Tables(0)
        dt.TableName = "details"

        Return dt
    End Function

    Private Sub FormatSOCSODetails(ByVal _TableName As String)
        'mObjPM.SetTableAttribute(_TableName, AttributeType.LINEBREAK, "1")
        mObjPM.SetTableAttribute(_TableName, AttributeType.ALIGN, "LEFT")
        mObjPM.SetTableAttribute(_TableName, AttributeType.FONT, ",12,")
        mObjPM.SetTableAttribute(_TableName, AttributeType.POSX, miPageMarginLeft.ToString)
        mObjPM.SetColumnAttribute(_TableName, 0, AttributeType.WIDTH, "200")
        mObjPM.SetColumnAttribute(_TableName, 1, AttributeType.WIDTH, "200")
        mObjPM.SetColumnAttribute(_TableName, 2, AttributeType.WIDTH, "200")
        mObjPM.SetColumnAttribute(_TableName, 3, AttributeType.ALIGN, "right")
        mObjPM.SetColumnAttribute(_TableName, 3, AttributeType.WIDTH, "100")
    End Sub
#End Region

#Region " TOTALS "
    Private Function GetSOCSOTotals() As DataTable
        Dim str As String
        Dim dt As New DataTable()

        str = "SELECT '' AS Nothing, SUM(h.EmployeeAmount) AS Employee " & _
            "FROM Employee e, History h " & _
            "WHERE e.ID = h.EmployeeID " & _
            "AND h.Type = 'Deduction' " & _
            "AND h.Name = 'SOCSO' " & _
            "AND h.Month = '" & Now.Month & "' " & _
        "AND h.Year = '" & Now.Year & "'"

      dt = mObjDAL.ExecDatasetTx(str).Tables(0)
      dt.Rows(0).Item("Nothing") = "JUMLAH BESAR"
      dt.TableName = "total"

      Return dt
    End Function

    Private Sub FormatSOCSOTotal(ByVal _TableName As String)
      mObjPM.SetTableAttribute(_TableName, AttributeType.LINEBREAK, "1")
      mObjPM.SetTableAttribute(_TableName, AttributeType.ALIGN, "Right")
      mObjPM.SetTableAttribute(_TableName, AttributeType.FONT, ",12,Bold")
      mObjPM.SetTableAttribute(_TableName, AttributeType.POSX, miPageMarginLeft.ToString)
      mObjPM.SetColumnAttribute(_TableName, 0, AttributeType.WIDTH, "600")
      mObjPM.SetColumnAttribute(_TableName, 1, AttributeType.WIDTH, "100")
    End Sub
#End Region

    'PCB

#Region " Report Header "

    Private Function GetPCBReportHeader() As DataTable
      Dim dtHeader As New DataTable()

      Dim dr As DataRow

      dtHeader.TableName = "Report Header"
      dtHeader.Columns.Add("Captions")

      dr = dtHeader.NewRow
      dr.Item("Captions") = "CUKAI PENDAPATAN MALAYSIA"
      dtHeader.Rows.Add(dr)

      dr = dtHeader.NewRow
      dr.Item("Captions") = "PENYATA POTONGAN CUKAI OLEH MAJIKAN"
      dtHeader.Rows.Add(dr)

      dr = dtHeader.NewRow
      dr.Item("Captions") = "POTONGAN BAGI BULAN " & Now.Month & " TAHUN " & Now.Year
      dtHeader.Rows.Add(dr)
      dtHeader.TableName = "header"
      Return dtHeader
    End Function

    Private Sub FormatPCBReportHeader(ByVal _TableName As String)
      mObjPM.SetTableAttribute(_TableName, AttributeType.LINEBREAK, "2")
      mObjPM.SetTableAttribute(_TableName, AttributeType.ALIGN, "center")
      mObjPM.SetTableAttribute(_TableName, AttributeType.FONT, ",16,bold")
      mObjPM.SetTableAttribute(_TableName, AttributeType.POSX, miPageMarginLeft.ToString)
      mObjPM.SetColumnAttribute(_TableName, 0, AttributeType.WIDTH, "700")
    End Sub
#End Region

#Region " Report Details Header "
    Private Function GetPCBDetailsHeader() As DataTable
      '
      ' Create Report Header
      '
      Dim dr As DataRow
      Dim dtHeader As New DataTable()

      dtHeader.TableName = "Details Header"
      dtHeader.Columns.Add("NoAhli")
      dtHeader.Columns.Add("NoKadPengenalan")
      dtHeader.Columns.Add("Name")
      dtHeader.Columns.Add("Caruman")

      dr = dtHeader.NewRow
```

```
        dr.Item("NoAhli") = "NO. FAIL CUKAI"
        dr.Item("NoKadPengenalan") = "NO KAD PENGENALAN"
        dr.Item("Name") = "NAME PEKERJA"
        dr.Item("Caruman") = "PCB (RM)"
        dtHeader.Rows.Add(dr)
        dtHeader.TableName = "header details"
        Return dtHeader
    End Function

    Private Sub FormatPCBHeaderDetails(ByVal _TableName As String)
        mObjPM.SetTableAttribute(_TableName, AttributeType.LINEBREAK, "1")
        mObjPM.SetTableAttribute(_TableName, AttributeType.ALIGN, "LEFT")
        mObjPM.SetTableAttribute(_TableName, AttributeType.FONT, ",12,bold")
        mObjPM.SetTableAttribute(_TableName, AttributeType.POSX, miPageMarginLeft.ToString)
        mObjPM.SetColumnAttribute(_TableName, 0, AttributeType.WIDTH, "200")
        mObjPM.SetColumnAttribute(_TableName, 1, AttributeType.WIDTH, "200")
        mObjPM.SetColumnAttribute(_TableName, 2, AttributeType.WIDTH, "200")
        mObjPM.SetColumnAttribute(_TableName, 3, AttributeType.ALIGN, "right")
        mObjPM.SetColumnAttribute(_TableName, 3, AttributeType.WIDTH, "100")
    End Sub
#End Region

#Region " Details Section "

    Private Function GetPCBDetails() As DataTable
        Dim str As String
        Dim dt As New DataTable()

        str = "SELECT e.taxNo, NewIC, LastName + ' ' + FirstName AS Name, h.EmployeeAmount " & _
            "FROM Employee e, History h " & _
            "WHERE e.ID = h.EmployeeID " & _
            "AND h.Type = 'Deduction' " & _
            "AND h.Name = 'PCB' " & _
            "AND h.Month = '" & Now.Month & "' " & _
            "AND h.Year = '" & Now.Year & "'"

        dt = mObjDAL.ExecDatasetTx(str).Tables(0)
        dt.TableName = "details"

        Return dt
    End Function

    Private Sub FormatPCBDetails(ByVal _TableName As String)
        mObjPM.SetTableAttribute(_TableName, AttributeType.LINEBREAK, "1")
        mObjPM.SetTableAttribute(_TableName, AttributeType.ALIGN, "LEFT")
        mObjPM.SetTableAttribute(_TableName, AttributeType.FONT, ",12,")
        mObjPM.SetTableAttribute(_TableName, AttributeType.POSX, miPageMarginLeft.ToString)
        mObjPM.SetColumnAttribute(_TableName, 0, AttributeType.WIDTH, "200")
        mObjPM.SetColumnAttribute(_TableName, 1, AttributeType.WIDTH, "200")
        mObjPM.SetColumnAttribute(_TableName, 2, AttributeType.WIDTH, "200")
        mObjPM.SetColumnAttribute(_TableName, 3, AttributeType.ALIGN, "right")
        mObjPM.SetColumnAttribute(_TableName, 3, AttributeType.WIDTH, "100")
    End Sub
#End Region

#Region " TOTALS "
    Private Function GetPCBTotals() As DataTable
        Dim str As String
        Dim dt As New DataTable()

        str = "SELECT '' AS Nothing, SUM(h.EmployeeAmount) AS Employee " & _
            "FROM Employee e, History h " & _
            "WHERE e.ID = h.EmployeeID " & _
            "AND h.Type = 'Deduction' " & _
            "AND h.Name = 'PCB' " & _
            "AND h.Month = '" & Now.Month & "' " & _
            "AND h.Year = '" & Now.Year & "'"

        dt = mObjDAL.ExecDatasetTx(str).Tables(0)
        dt.Rows(0).Item("Nothing") = "JUMLAH BESAR"
        dt.TableName = "total"

        Return dt
    End Function

    Private Sub FormatPCBTotal(ByVal _TableName As String)
        mObjPM.SetTableAttribute(_TableName, AttributeType.LINEBREAK, "1")
        mObjPM.SetTableAttribute(_TableName, AttributeType.ALIGN, "Right")
        mObjPM.SetTableAttribute(_TableName, AttributeType.FONT, ",12,Bold")
        mObjPM.SetTableAttribute(_TableName, AttributeType.POSX, miPageMarginLeft.ToString)
        mObjPM.SetColumnAttribute(_TableName, 0, AttributeType.WIDTH, "600")
        mObjPM.SetColumnAttribute(_TableName, 1, AttributeType.WIDTH, "100")
    End Sub
#End Region


    'Summary

#Region " Report Header "

    Private Function GetSummaryReportHeader() As DataTable
        Dim dtHeader As New DataTable()

        Dim dr As DataRow

        dtHeader.TableName = "Report Header"
        dtHeader.Columns.Add("Captions")

        dr = dtHeader.NewRow
        dr.Item("Captions") = "TOTAL SALARY PAYOUT"
        dtHeader.Rows.Add(dr)

        dr = dtHeader.NewRow
        dr.Item("Captions") = "FOR THE MONTH OF " & Now.Month & " YEAR " & Now.Year
        dtHeader.Rows.Add(dr)

        dtHeader.TableName = "header"
        Return dtHeader
    End Function

    Private Sub FormatSummaryReportHeader(ByVal _TableName As String)
        mObjPM.SetTableAttribute(_TableName, AttributeType.LINEBREAK, "2")
        mObjPM.SetTableAttribute(_TableName, AttributeType.ALIGN, "center")
        mObjPM.SetTableAttribute(_TableName, AttributeType.FONT, ",16,bold")
        mObjPM.SetTableAttribute(_TableName, AttributeType.POSX, miPageMarginLeft.ToString)
        mObjPM.SetColumnAttribute(_TableName, 0, AttributeType.WIDTH, "700")
    End Sub
#End Region

#Region " Report Details Header "
    Private Function GetSummaryDetailsHeader() As DataTable

        'Create Report Header

        Dim dr As DataRow
        Dim dtHeader As New DataTable()

        dtHeader.TableName = "Details Header"
        dtHeader.Columns.Add("ID")
        dtHeader.Columns.Add("Name")
        dtHeader.Columns.Add("Basic")
        dtHeader.Columns.Add("EPFE")
        dtHeader.Columns.Add("EPFY")
        dtHeader.Columns.Add("SOCSOE")
        dtHeader.Columns.Add("PCBE")

        dr = dtHeader.NewRow
        dr.Item("ID") = "ID"
        dr.Item("Name") = "Name"
        dr.Item("Basic") = "Basic"
        dr.Item("EPFE") = "EPF-E"
        dr.Item("EPFY") = "EPF-Y"
        dr.Item("SOCSOE") = "SOCSO-E"
        dr.Item("PCBE") = "PCB-E"

        dtHeader.Rows.Add(dr)
        dtHeader.TableName = "header details"
        Return dtHeader
    End Function

    Private Sub FormatSummaryHeaderDetails(ByVal _TableName As String)
        mObjPM.SetTableAttribute(_TableName, AttributeType.LINEBREAK, "1")
        mObjPM.SetTableAttribute(_TableName, AttributeType.ALIGN, "LEFT")
        mObjPM.SetTableAttribute(_TableName, AttributeType.FONT, ",12,bold")
        mObjPM.SetTableAttribute(_TableName, AttributeType.POSX, miPageMarginLeft.ToString)
        mObjPM.SetColumnAttribute(_TableName, 0, AttributeType.WIDTH, "100")
        mObjPM.SetColumnAttribute(_TableName, 1, AttributeType.WIDTH, "200")
        mObjPM.SetColumnAttribute(_TableName, 2, AttributeType.ALIGN, "right")
        mObjPM.SetColumnAttribute(_TableName, 2, AttributeType.WIDTH, "80")
        mObjPM.SetColumnAttribute(_TableName, 3, AttributeType.ALIGN, "right")
        mObjPM.SetColumnAttribute(_TableName, 3, AttributeType.WIDTH, "80")
        mObjPM.SetColumnAttribute(_TableName, 4, AttributeType.ALIGN, "right")
        mObjPM.SetColumnAttribute(_TableName, 4, AttributeType.WIDTH, "80")
        mObjPM.SetColumnAttribute(_TableName, 5, AttributeType.ALIGN, "right")
        mObjPM.SetColumnAttribute(_TableName, 5, AttributeType.WIDTH, "80")
        mObjPM.SetColumnAttribute(_TableName, 6, AttributeType.ALIGN, "right")
        mObjPM.SetColumnAttribute(_TableName, 6, AttributeType.WIDTH, "80")
    End Sub
#End Region

#Region " Details Section "

    Private Function GetSummaryDetails() As DataTable
        Dim str As String
        Dim dt As New DataTable()

        str = "SELECT ID, LastName + ' ' + FirstName AS Name, BasicPayment, " & _
                "(SELECT EmployeeAmount FROM History WHERE Month = '" & Now.Month & "' AND " & "Year = '" &
Now.Year & "' AND" & " Type = 'Deduction' AND Name = 'EPF' AND EmployeeID = e.ID) AS EPFE, " & _
                "(SELECT EmployerAmount FROM History WHERE Month = '" & Now.Month & "' AND " & "Year = '" &
Now.Year & "' AND" & " Type = 'Deduction' AND Name = 'EPF' AND EmployeeID = e.ID) AS EPFY, " & _
                "(SELECT EmployeeAmount FROM History WHERE Month = '" & Now.Month & "' AND " & "Year = '" &
Now.Year & "' AND" & " Type = 'Deduction' AND Name = 'SOCSO' AND EmployeeID = e.ID) AS SOCSOE, " & _
                "(SELECT EmployeeAmount FROM History WHERE Month = '" & Now.Month & "' AND" & "Year = '" &
Now.Year & "' AND" & " Type = 'Deduction' AND Name = 'PCB' AND EmployeeID = e.ID) AS PCBE " & _
            "FROM Employee e "

        dt = mObjDAL.ExecDatasetTx(str).Tables(0)
        dt.TableName = "details"

        Return dt
    End Function

    Private Sub FormatSummaryDetails(ByVal _TableName As String)
        'mObjPM.SetTableAttribute(_TableName, AttributeType.LINEBREAK, "1")
        mObjPM.SetTableAttribute(_TableName, AttributeType.ALIGN, "LEFT")
        mObjPM.SetTableAttribute(_TableName, AttributeType.FONT, ",12,")
        mObjPM.SetTableAttribute(_TableName, AttributeType.POSX, miPageMarginLeft.ToString)
        mObjPM.SetColumnAttribute(_TableName, 0, AttributeType.WIDTH, "100")
        mObjPM.SetColumnAttribute(_TableName, 1, AttributeType.WIDTH, "200")
        mObjPM.SetColumnAttribute(_TableName, 2, AttributeType.ALIGN, "right")
        mObjPM.SetColumnAttribute(_TableName, 2, AttributeType.WIDTH, "80")
        mObjPM.SetColumnAttribute(_TableName, 3, AttributeType.ALIGN, "right")
        mObjPM.SetColumnAttribute(_TableName, 3, AttributeType.WIDTH, "80")
        mObjPM.SetColumnAttribute(_TableName, 4, AttributeType.ALIGN, "right")
        mObjPM.SetColumnAttribute(_TableName, 4, AttributeType.WIDTH, "80")
        mObjPM.SetColumnAttribute(_TableName, 5, AttributeType.ALIGN, "right")
        mObjPM.SetColumnAttribute(_TableName, 5, AttributeType.WIDTH, "80")
        mObjPM.SetColumnAttribute(_TableName, 6, AttributeType.ALIGN, "right")
        mObjPM.SetColumnAttribute(_TableName, 6, AttributeType.WIDTH, "80")
    End Sub
#End Region

#Region " TOTALS "
    Private Function GetSummaryTotals() As DataTable
        Dim str As String
        Dim dt As New DataTable()

        str = "SELECT '' AS Nothing, SUM(BasicPayment) AS Basic, " & _
                "(SELECT SUM(EmployeeAmount) FROM History WHERE Month = '" & Now.Month & "' AND " &
"Year = '" & Now.Year & "' AND" & " Type = 'Deduction' AND Name = 'EPF') AS EPFE, " & _
                "(SELECT SUM(EmployerAmount) FROM History WHERE Month = '" & Now.Month & "' AND " &
"Year = '" & Now.Year & "' AND" & " Type = 'Deduction' AND Name = 'EPF') AS EPFY, " & _
                "(SELECT SUM(EmployeeAmount) FROM History WHERE Month = '" & Now.Month & "' AND " &
"Year = '" & Now.Year & "' AND" & " Type = 'Deduction' AND Name = 'SOCSO') AS SOCSOE, " & _
                "(SELECT SUM(EmployeeAmount) FROM History WHERE Month = '" & Now.Month & "' AND " &
"Year = '" & Now.Year & "' AND" & " Type = 'Deduction' AND Name = 'PCB') AS PCBE " & _
            "FROM Employee e "

        dt = mObjDAL.ExecDatasetTx(str).Tables(0)
        dt.Rows(0).Item("Nothing") = "TOTAL"
        dt.TableName = "total"

        Return dt
    End Function

    Private Sub FormatSummaryTotal(ByVal _TableName As String)
        mObjPM.SetTableAttribute(_TableName, AttributeType.ALIGN, "right")
        mObjPM.SetTableAttribute(_TableName, AttributeType.FONT, ",12,bold")
        mObjPM.SetTableAttribute(_TableName, AttributeType.POSX, miPageMarginLeft.ToString)
        mObjPM.SetColumnAttribute(_TableName, 0, AttributeType.WIDTH, "300")
        mObjPM.SetColumnAttribute(_TableName, 1, AttributeType.WIDTH, "80")
        mObjPM.SetColumnAttribute(_TableName, 2, AttributeType.WIDTH, "80")
        mObjPM.SetColumnAttribute(_TableName, 3, AttributeType.WIDTH, "80")
        mObjPM.SetColumnAttribute(_TableName, 4, AttributeType.WIDTH, "80")
        mObjPM.SetColumnAttribute(_TableName, 5, AttributeType.WIDTH, "80")
    End Sub
#End Region

    Private Sub mnuSOCSO_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
mnuSOCSO.Click
        mReportType = ReportTypeResult.SOCSO
        Try
            mPPreviewDlg = New PrintPreviewDialog()
            mPPreviewDlg.Size = New System.Drawing.Size(700, 650)
            mPPreviewDlg.StartPosition = FormStartPosition.Manual
            mPPreviewDlg.Location = New System.Drawing.Point(30, 30)

            mPPreviewDlg.PrintPreviewControl.AutoZoom = True
            mPPreviewDlg.PrintPreviewControl.Zoom = 0.8
            mPPreviewDlg.Text &= " - " & "SOCSO"
            mPPreviewDlg.Document = Me.PDocument
            mPPreviewDlg.MdiParent = Me.MdiParent
            mPPreviewDlg.ShowDialog()
        Catch Err As Exception
            MsgBox(Err.Message, MsgBoxStyle.Exclamation, )
        End Try
    End Sub

    Private Sub mnuPCB_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles mnuPCB.Click
        mReportType = ReportTypeResult.PCB
        Try
            mPPreviewDlg = New PrintPreviewDialog()
            mPPreviewDlg.Size = New System.Drawing.Size(700, 650)
            mPPreviewDlg.StartPosition = FormStartPosition.Manual
            mPPreviewDlg.Location = New System.Drawing.Point(30, 30)

            mPPreviewDlg.PrintPreviewControl.AutoZoom = True
            mPPreviewDlg.PrintPreviewControl.Zoom = 0.8
            mPPreviewDlg.Text &= " - " & "PCB"
            mPPreviewDlg.Document = Me.PDocument
            mPPreviewDlg.MdiParent = Me.MdiParent
            mPPreviewDlg.ShowDialog()
        Catch Err As Exception
            MsgBox(Err.Message, MsgBoxStyle.Exclamation, )
        End Try
    End Sub

    Private Sub mnuAbout_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
mnuAbout.Click
        Dim frmAbout As New frmAbout()
        Dim MdiChild As Form
        Dim Found As Boolean

        For Each MdiChild In Me.MdiChildren
            If TypeOf MdiChild Is frmAbout Then
                Found = True
                MdiChild.Activate()
                Exit For
            End If
        Next

        If Found = False Then
            frmAbout.MdiParent = Me
            frmAbout.Show()
        End If
    End Sub

    Private Sub mnuSummary_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
mnuSummary.Click
        mReportType = ReportTypeResult.Summary
        Try
            mPPreviewDlg = New PrintPreviewDialog()
            mPPreviewDlg.Size = New System.Drawing.Size(700, 650)
            mPPreviewDlg.StartPosition = FormStartPosition.Manual
            mPPreviewDlg.Location = New System.Drawing.Point(30, 30)

            mPPreviewDlg.PrintPreviewControl.AutoZoom = True
            mPPreviewDlg.PrintPreviewControl.Zoom = 0.8
            mPPreviewDlg.Text &= " - " & "Summary of Total Salary Payout"
            mPPreviewDlg.Document = Me.PDocument
            mPPreviewDlg.MdiParent = Me.MdiParent
            mPPreviewDlg.ShowDialog()
        Catch Err As Exception
            MsgBox(Err.Message, MsgBoxStyle.Exclamation, )
        End Try
    End Sub
```

FrmEmployeeList

```
    Private Sub frmEmployeeList_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        Dim strSearch As String
        Dim dt As New DataTable()
        Dim gridStyle As New DataGridTableStyle()
        Dim colID As New DataGridNoActiveCellColumn()
        Dim colName As New DataGridNoActiveCellColumn()
        Dim colPhone As New DataGridNoActiveCellColumn()

        Try
            strSearch = "SELECT *, FirstName + ' ' + LastName AS Name " & _
                        "FROM Employee " & _
                        "WHERE PayStatus = 'Active' "

            dt = mObjDAL.ExecDatasetTx(strSearch).Tables(0)

            DGEmployee.RowHeaderWidth = 0

            gridStyle.MappingName = dt.TableName

            gridStyle.GridColumnStyles.Add(colID)
            colID.Alignment = HorizontalAlignment.Left
            colID.HeaderText = "ID"
            colID.MappingName = "ID"
            colID.Width = 100

            gridStyle.GridColumnStyles.Add(colName)
            colName.Alignment = HorizontalAlignment.Left
            colName.HeaderText = "Name"
            colName.MappingName = "Name"
            colName.Width = 200

            gridStyle.GridColumnStyles.Add(colPhone)
            colPhone.Alignment = HorizontalAlignment.Left
            colPhone.HeaderText = "Phone"
            colPhone.MappingName = "Phone"
            colPhone.Width = DGEmployee.Width - _
                        DGEmployee.RowHeaderWidth - _
                        colID.Width - colName.Width - 40

            DGEmployee.TableStyles.Add(gridStyle)
            DGEmployeeRefresh()
        Catch Err As Exception
            MsgBox(Err.Message, MsgBoxStyle.Exclamation, )
        End Try

    End Sub

    Private Sub DGEmployeeRefresh()
        Dim strSearch As String
        Dim dt As New DataTable()

        strSearch = "SELECT *, FirstName + ' ' + LastName AS Name " & _
                    "FROM Employee " & _
                    "WHERE PayStatus = 'Active'"

        dt = mObjDAL.ExecDatasetTx(strSearch).Tables(0)

        DGEmployee.DataSource = dt
    End Sub

    Private Sub DGEmployee_DoubleClick(ByVal sender As Object, ByVal e As System.EventArgs) Handles
DGEmployee.DoubleClick
        Dim frmEmployeePayment As New frmEmployeePayment()

        frmEmployeePayment.ID = DGEmployee.Item(DGEmployee.CurrentRowIndex, 0).ToString
        frmEmployeePayment.ShowDialog()
        DGEmployeeRefresh()
    SetDataGridPointer(Me, DGEmployee, 0, frmEmployeePayment.ID)
End Sub
```

FrmEmployee

```
   Private Sub frmEmployee_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
      mAAction = AActionResult.Add
      mDAction = DActionResult.Add

      If mFormAction = FormActionResult.Add Then
         TCEmployee.TabPages.Remove(TPAllowance)
         TCEmployee.TabPages.Remove(TPDeduction)
         TCEmployee.SelectedTab = TPPersonalInfo
      ElseIf mFormAction = FormActionResult.Edit Then
         Dim strGet As String
         Dim dt As New DataTable()

         strGet = "SELECT * " & _
               "FROM Employee " & _
               "WHERE ID = '" & mID & "'"

         dt = mObjDAL.ExecDatasetTx(strGet).Tables(0)

         With dt.Rows(0)
            txtID.Text = .Item("ID").ToString
            txtFirstName.Text = .Item("FirstName").ToString
            txtLastName.Text = .Item("LastName").ToString
            txtMiddleName.Text = .Item("MiddleName").ToString
            txtDOB.Text = .Item("DOB").ToString
            txtICOld.Text = .Item("OldIC").ToString
            txtICNew.Text = .Item("NewIC").ToString
            txtPassport.Text = .Item("Passport").ToString
            cboSex.Text = .Item("Sex").ToString
            cboRace.Text = .Item("Race").ToString
            cboBumiputra.Text = .Item("Bumiputra").ToString
            cboMaritalStatus.Text = .Item("MaritalStatus").ToString
            txtEmail.Text = .Item("Email").ToString
            txtPhone.Text = .Item("Phone").ToString
            txtMobile.Text = .Item("Mobile").ToString
            txtStreet1.Text = .Item("Street1").ToString
            txtStreet2.Text = .Item("Street2").ToString
            txtCity.Text = .Item("City").ToString
            txtState.Text = .Item("State").ToString
            txtPostalCode.Text = .Item("PostalCode").ToString
            txtCountry.Text = .Item("Country").ToString
            txtRemarks.Text = .Item("Remarks").ToString
            txtJobTitle.Text = .Item("JobTitle").ToString
            txtEPFNo.Text = .Item("EPFNo").ToString
            txtSOCSONo.Text = .Item("SOCSONo").ToString
            txtTaxNo.Text = .Item("TaxNo").ToString
            txtBasicPayment.Text = Format(CDbl(.Item("BasicPayment")), "0.00")
            dtCommerce.Value = CDate(.Item("DateCommerce"))
            dtResign.Value = CDate(.Item("DateResign"))
            cboPayStatus.Text = .Item("PayStatus").ToString
         End With
         txtID.ReadOnly = True


         strGet = "SELECT * " & _
               "FROM Login " & _
               "WHERE Username = '" & mID & "'"

         dt = mObjDAL.ExecDatasetTx(strGet).Tables(0)

         With dt.Rows(0)
            txtUsername.Text = .Item("Username").ToString
            txtPassword.Text = .Item("Password").ToString
            cboLevel.Text = .Item("Level").ToString
         End With

         Dim AGridStyle As New DataGridTableStyle()
         Dim colAUID As New DataGridNoActiveCellColumn()
```

```
        Dim colAName As New DataGridNoActiveCellColumn()
        Dim colAAmount As New DataGridNoActiveCellColumn()
        Dim Astr As String

        DGAllowance.RowHeaderWidth = 0

        Astr = "SELECT * FROM Allowance WHERE EmployeeID = '" & TrimSqlVariable(txtID.Text) & "' "

        AGridStyle.MappingName = mObjDAL.ExecDatasetTx(Astr).Tables(0).TableName

        AGridStyle.GridColumnStyles.Add(colAUID)
        colAUID.Alignment = HorizontalAlignment.Left
        colAUID.MappingName = "UID"
        colAUID.Width = 0

        AGridStyle.GridColumnStyles.Add(colAName)
        colAName.Alignment = HorizontalAlignment.Left
        colAName.HeaderText = "Name"
        colAName.MappingName = "Name"
        colAName.Width = 400

        AGridStyle.GridColumnStyles.Add(colAAmount)
        colAAmount.Alignment = HorizontalAlignment.Left
        colAAmount.HeaderText = "Amount"
        colAAmount.MappingName = "Amount"
        colAAmount.Format = "0.00"
        colAAmount.Width = DGAllowance.Width - _
                    DGAllowance.RowHeaderWidth - _
                    colAName.Width - 20

        DGAllowance.TableStyles.Add(AGridStyle)
        DGAllowance.DataSource = mObjDAL.ExecDatasetTx(Astr).Tables(0)

        Dim DGridStyle As New DataGridTableStyle()
        Dim colDUID As New DataGridNoActiveCellColumn()
        Dim colDName As New DataGridNoActiveCellColumn()
        Dim colDAmount As New DataGridNoActiveCellColumn()
        Dim Dstr As String

        DGDeduction.RowHeaderWidth = 0

        Dstr = "SELECT * FROM Deduction WHERE EmployeeID = '" & TrimSqlVariable(txtID.Text) & "' "

        DGridStyle.MappingName = mObjDAL.ExecDatasetTx(Dstr).Tables(0).TableName

        DGridStyle.GridColumnStyles.Add(colDUID)
        colDUID.Alignment = HorizontalAlignment.Left
        colDUID.MappingName = "UID"
        colDUID.Width = 0

        DGridStyle.GridColumnStyles.Add(colDName)
        colDName.Alignment = HorizontalAlignment.Left
        colDName.HeaderText = "Name"
        colDName.MappingName = "Name"
        colDName.Width = 400

        DGridStyle.GridColumnStyles.Add(colDAmount)
        colDAmount.Alignment = HorizontalAlignment.Left
        colDAmount.HeaderText = "Amount"
        colDAmount.MappingName = "Amount"
        colDAmount.Format = "0.00"
        colDAmount.Width = DGDeduction.Width - _
                    DGDeduction.RowHeaderWidth - _
                    colDName.Width - 20

        DGDeduction.TableStyles.Add(DGridStyle)
        DGDeduction.DataSource = mObjDAL.ExecDatasetTx(Dstr).Tables(0)
    End If
End Sub
```

```
Private Sub btnCancel_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCancel.Click
  Me.Close()
End Sub

Private Sub btnOK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnOK.Click
  Try
    If mFormAction = FormActionResult.Add Then
      Dim strAdd As String

      Dim strCheck As String

      strCheck = "SELECT ID " & _
            "FROM Employee " & _
            "WHERE ID = '" & TrimSqlVariable(txtID.Text) & "'"

      If Trim(txtID.Text) = "" Then
        TCEmployee.SelectedTab = TPPersonalInfo
        txtID.Focus()
        ErrorProvider.SetError(txtID, "ID cannot be empty")
      ElseIf Trim(txtFirstName.Text) = "" Then
        TCEmployee.SelectedTab = TPPersonalInfo
        txtFirstName.Focus()
        ErrorProvider.SetError(txtFirstName, "First name cannot be empty")
      ElseIf Trim(txtLastName.Text) = "" Then
        TCEmployee.SelectedTab = TPPersonalInfo
        txtLastName.Focus()
        ErrorProvider.SetError(txtLastName, "Last name cannot be empty")
      ElseIf Trim(txtICNew.Text) = "" Then
        TCEmployee.SelectedTab = TPPersonalInfo
        txtICNew.Focus()
        ErrorProvider.SetError(txtICNew, "New IC cannot be empty")
      ElseIf Trim(txtEPFNo.Text) = "" Then
        TCEmployee.SelectedTab = TPJobInfo
        txtEPFNo.Focus()
        ErrorProvider.SetError(txtEPFNo, "EPF No cannot be empty")
      ElseIf Trim(txtSOCSONo.Text) = "" Then
        TCEmployee.SelectedTab = TPJobInfo
        txtSOCSONo.Focus()
        ErrorProvider.SetError(txtSOCSONo, "SOCSO No cannot be empty")
      ElseIf Trim(txtTaxNo.Text) = "" Then
        TCEmployee.SelectedTab = TPJobInfo
        txtTaxNo.Focus()
        ErrorProvider.SetError(txtTaxNo, "Tax No cannot be empty")
      ElseIf Trim(txtBasicPayment.Text) = "" Then
        TCEmployee.SelectedTab = TPJobInfo
        txtBasicPayment.Focus()
        ErrorProvider.SetError(txtBasicPayment, "Payment cannot be empty")
      ElseIf Not IsNumeric(Trim(txtBasicPayment.Text)) Then
        TCEmployee.SelectedTab = TPJobInfo
        txtBasicPayment.Focus()
        ErrorProvider.SetError(txtBasicPayment, "Payment can contain only number")
      ElseIf Trim(cboPayStatus.Text) = "" Then
        TCEmployee.SelectedTab = TPJobInfo
        cboPayStatus.Focus()
        ErrorProvider.SetError(cboPayStatus, "Pay status cannot be empty")
      ElseIf Trim(cboLevel.Text) = "" Then
        TCEmployee.SelectedTab = TPLogin
        cboLevel.Focus()
        ErrorProvider.SetError(cboLevel, "Level cannot be empty")
      ElseIf Not mObjDAL.ExecDatasetTx(strCheck).Tables(0).Rows.Count = 0 Then
        TCEmployee.SelectedTab = TPPersonalInfo
        txtID.Focus()
        ErrorProvider.SetError(txtID, "ID Already Exist")
      Else
        ErrorProvider.SetError(txtID, "")
        ErrorProvider.SetError(txtFirstName, "")
        ErrorProvider.SetError(txtLastName, "")
        ErrorProvider.SetError(txtEPFNo, "")
        ErrorProvider.SetError(txtSOCSONo, "")
        ErrorProvider.SetError(txtBasicPayment, "")
```

```
            ErrorProvider.SetError(cboPayStatus, "")
            ErrorProvider.SetError(cboLevel, "")

            strAdd = "INSERT INTO Employee " & _
                     "(ID, FirstName, LastName, MiddleName, DOB, OldIC, NewIC, " & _
                     "Passport, Sex, Race, Bumiputra, MaritalStatus, Email, " & _
                     "Phone, Mobile, Street1, Street2, City, State, PostalCode, " & _
                     "Country, Remarks, JobTitle, EPFNo, SOCSONo, TaxNo, BasicPayment, " & _
                     "DateCommerce, DateResign, PayStatus) " & _
                 "VALUES " & _
                     "( '" & TrimSqlVariable(txtID.Text) & "', " & _
                      "'" & TrimSqlVariable(txtFirstName.Text) & "', " & _
                      "'" & TrimSqlVariable(txtLastName.Text) & "', " & _
                      "'" & TrimSqlVariable(txtMiddleName.Text) & "', " & _
                      "'" & TrimSqlVariable(txtDOB.Text) & "', " & _
                      "'" & TrimSqlVariable(txtICOld.Text) & "', " & _
                      "'" & TrimSqlVariable(txtICNew.Text) & "', " & _
                      "'" & TrimSqlVariable(txtPassport.Text) & "', " & _
                      "'" & TrimSqlVariable(cboSex.Text) & "', " & _
                      "'" & TrimSqlVariable(cboRace.Text) & "', " & _
                      "'" & TrimSqlVariable(cboBumiputra.Text) & "', " & _
                      "'" & TrimSqlVariable(cboMaritalStatus.Text) & "', " & _
                      "'" & TrimSqlVariable(txtEmail.Text) & "', " & _
                      "'" & TrimSqlVariable(txtPhone.Text) & "', " & _
                      "'" & TrimSqlVariable(txtMobile.Text) & "', " & _
                      "'" & TrimSqlVariable(txtStreet1.Text) & "', " & _
                      "'" & TrimSqlVariable(txtStreet2.Text) & "', " & _
                      "'" & TrimSqlVariable(txtCity.Text) & "', " & _
                      "'" & TrimSqlVariable(txtState.Text) & "', " & _
                      "'" & TrimSqlVariable(txtPostalCode.Text) & "', " & _
                      "'" & TrimSqlVariable(txtCountry.Text) & "', " & _
                      "'" & TrimSqlVariable(txtRemarks.Text) & "', " & _
                      "'" & TrimSqlVariable(txtJobTitle.Text) & "', " & _
                      "'" & TrimSqlVariable(txtEPFNo.Text) & "', " & _
                      "'" & TrimSqlVariable(txtSOCSONo.Text) & "', " & _
                      "'" & TrimSqlVariable(txtTaxNo.Text) & "', " & _
                      "'" & CDbl(txtBasicPayment.Text) & "', " & _
                      "'" & GetFormattedDate(dtCommerce.Value) & "', " & _
                      "'" & GetFormattedDate(dtResign.Value) & "', " & _
                      "'" & TrimSqlVariable(cboPayStatus.Text) & "')"

            mObjDAL.ExecNonQueryTx(strAdd)

            Dim level As String = cboLevel.Text

            If Trim(level) = "" Then
                level = "Normal User"
            End If

            strAdd = "INSERT INTO Login " & _
                     "(Username, Password, Level) " & _
                 "VALUES " & _
                     "( '" & TrimSqlVariable(txtUsername.Text) & "', " & _
                      "'" & TrimSqlVariable(txtPassword.Text) & "', " & _
                      "'" & TrimSqlVariable(level) & "')"

            mObjDAL.ExecNonQueryTx(strAdd)

            TCEmployee.TabPages.Add(TPAllowance)
            TCEmployee.TabPages.Add(TPDeduction)
            TCEmployee.SelectedTab = TPAllowance
            mFormAction = FormActionResult.Edit
        End If

    ElseIf mFormAction = FormActionResult.Edit Then
        If Trim(txtFirstName.Text) = "" Then
            TCEmployee.SelectedTab = TPPersonalInfo
            txtFirstName.Focus()
            ErrorProvider.SetError(txtFirstName, "First name cannot be empty")
        ElseIf Trim(txtLastName.Text) = "" Then
        TCEmployee.SelectedTab = TPPersonalInfo
        txtLastName.Focus()
        ErrorProvider.SetError(txtLastName, "Last name cannot be empty")
    ElseIf Trim(txtICNew.Text) = "" Then
        TCEmployee.SelectedTab = TPPersonalInfo
        txtICNew.Focus()
        ErrorProvider.SetError(txtICNew, "New IC cannot be empty")
    ElseIf Trim(txtEPFNo.Text) = "" Then
        TCEmployee.SelectedTab = TPJobInfo
        txtEPFNo.Focus()
        ErrorProvider.SetError(txtEPFNo, "EPF No cannot be empty")
    ElseIf Trim(txtSOCSONo.Text) = "" Then
        TCEmployee.SelectedTab = TPJobInfo
        txtSOCSONo.Focus()
        ErrorProvider.SetError(txtSOCSONo, "SOCSO No cannot be empty")
    ElseIf Trim(txtTaxNo.Text) = "" Then
        TCEmployee.SelectedTab = TPJobInfo
        txtTaxNo.Focus()
        ErrorProvider.SetError(txtTaxNo, "Tax No cannot be empty")
    ElseIf Trim(txtBasicPayment.Text) = "" Then
        TCEmployee.SelectedTab = TPJobInfo
        txtBasicPayment.Focus()
        ErrorProvider.SetError(txtBasicPayment, "Payment cannot be empty")
    ElseIf Not IsNumeric(Trim(txtBasicPayment.Text)) Then
        TCEmployee.SelectedTab = TPJobInfo
        txtBasicPayment.Focus()
        ErrorProvider.SetError(txtBasicPayment, "Payment can contain only number")
    ElseIf Trim(cboPayStatus.Text) = "" Then
        TCEmployee.SelectedTab = TPJobInfo
        cboPayStatus.Focus()
        ErrorProvider.SetError(cboPayStatus, "Pay status cannot be empty")
    ElseIf Trim(cboLevel.Text) = "" Then
        TCEmployee.SelectedTab = TPLogin
        cboLevel.Focus()
        ErrorProvider.SetError(cboLevel, "Level cannot be empty")
    Else
        ErrorProvider.SetError(txtID, "")
        ErrorProvider.SetError(txtFirstName, "")
        ErrorProvider.SetError(txtLastName, "")
        ErrorProvider.SetError(txtEPFNo, "")
        ErrorProvider.SetError(txtSOCSONo, "")
        ErrorProvider.SetError(txtBasicPayment, "")
        ErrorProvider.SetError(cboPayStatus, "")
        ErrorProvider.SetError(cboLevel, "")

        Dim strUpdate As String

        strUpdate = "UPDATE Employee " & _
                "SET FirstName = '" & TrimSqlVariable(txtFirstName.Text) & "', " & _
                  "LastName = '" & TrimSqlVariable(txtLastName.Text) & "', " & _
                  "MiddleName =  '" & TrimSqlVariable(txtMiddleName.Text) & "', " & _
                  "DOB = '" & TrimSqlVariable(txtDOB.Text) & "', " & _
                  "OldIC = '" & TrimSqlVariable(txtICOld.Text) & "', " & _
                  "NewIC= '" & TrimSqlVariable(txtICNew.Text) & "', " & _
                  "Passport = '" & TrimSqlVariable(txtPassport.Text) & "', " & _
                  "Sex= '" & TrimSqlVariable(cboSex.Text) & "', " & _
                  "Race= '" & TrimSqlVariable(cboRace.Text) & "', " & _
                  "Bumiputra= '" & TrimSqlVariable(cboBumiputra.Text) & "', " & _
                  "MaritalStatus = '" & TrimSqlVariable(cboMaritalStatus.Text) & "', " & _
                  "Email = '" & TrimSqlVariable(txtEmail.Text) & "', " & _
                  "Phone = '" & TrimSqlVariable(txtPhone.Text) & "', " & _
                  "Mobile= '" & TrimSqlVariable(txtMobile.Text) & "', " & _
                  "Street1= '" & TrimSqlVariable(txtStreet1.Text) & "', " & _
                  "Street2 = '" & TrimSqlVariable(txtStreet2.Text) & "', " & _
                  "City = '" & TrimSqlVariable(txtCity.Text) & "', " & _
                  "State = '" & TrimSqlVariable(txtState.Text) & "', " & _
                  "PostalCode= '" & TrimSqlVariable(txtPostalCode.Text) & "', " & _
                  "Country = '" & TrimSqlVariable(txtCountry.Text) & "', " & _
                  "Remarks = '" & TrimSqlVariable(txtRemarks.Text) & "', " & _
                  "JobTitle = '" & TrimSqlVariable(txtJobTitle.Text) & "', " & _
```

```
                        "EPFNo = '" & TrimSqlVariable(txtEPFNo.Text) & "', " & _
                        "SOCSONo = '" & TrimSqlVariable(txtSOCSONo.Text) & "', " & _
                        "TaxNo = '" & TrimSqlVariable(txtTaxNo.Text) & "', " & _
                        "BasicPayment = '" & TrimSqlVariable(txtBasicPayment.Text) & "', " & _
                        "DateCommerce = '" & GetFormattedDate(dtResign.Value) & "', " & _
                        "DateResign = '" & GetFormattedDate(dtResign.Value) & "', " & _
                        "PayStatus = '" & TrimSqlVariable(cboPayStatus.Text) & "' " & _
                      "WHERE ID = '" & TrimSqlVariable(txtID.Text) & "'"

            mObjDAL.ExecNonQueryTx(strUpdate)

            Dim level As String = cboLevel.Text

            If Trim(level) = "" Then
               level = "Normal User"
            End If

            strUpdate = "UPDATE Login " & _
                     "SET Password = '" & TrimSqlVariable(txtPassword.Text) & "', " & _
                        "Level = '" & TrimSqlVariable(level) & "' " & _
                     "WHERE Username = '" & TrimSqlVariable(txtID.Text) & "'"

            mObjDAL.ExecNonQueryTx(strUpdate)
            Me.Close()
         End If
      End If
   Catch Err As Exception
      MsgBox(Err.Message, MsgBoxStyle.Information, )
   End Try
End Sub

Private Sub btnAOK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnAOK.Click
   If Trim(txtAName.Text) = "" Then
      TCEmployee.SelectedTab = TPAllowance
      txtAName.Focus()
      ErrorProvider.SetError(txtAName, "Name cannot be empty")
   ElseIf Trim(txtAAmount.Text) = "" Then
      TCEmployee.SelectedTab = TPAllowance
      txtAAmount.Focus()
      ErrorProvider.SetError(txtAAmount, "Amount cannot be empty")
   ElseIf Not IsNumeric(Trim(txtAAmount.Text)) Then
      TCEmployee.SelectedTab = TPAllowance
      txtAAmount.Focus()
      ErrorProvider.SetError(txtAAmount, "Amount can contain only number")
   Else
      If mAAction = AActionResult.Add Then
         Dim strAdd As String

         strAdd = "INSERT INTO Allowance " & _
                  "(UID, Name, Amount, EmployeeID) " & _
               "VALUES " & _
                  "( '" & Guid.NewGuid.ToString & "', " & _
                  "'" & TrimSqlVariable(txtAName.Text) & "', " & _
                  "" & CDbl(txtAAmount.Text) & ", " & _
                  "'" & TrimSqlVariable(txtID.Text) & "')"

         mObjDAL.ExecNonQueryTx(strAdd)
      ElseIf mAAction = AActionResult.Edit Then
         Dim strUpdate As String

         strUpdate = "UPDATE Allowance " & _
                  "SET Name = '" & TrimSqlVariable(txtAName.Text) & "', " & _
                     "Amount = '" & TrimSqlVariable(txtAAmount.Text) & "' " & _
                  "WHERE UID = '" & mUID.ToString & "'"

         mObjDAL.ExecNonQueryTx(strUpdate)
      End If
      mAAction = AActionResult.Add

      Dim str As String
```

```
            str = "SELECT * FROM Allowance WHERE EmployeeID = '" & TrimSqlVariable(txtID.Text) & "' "
            DGAllowance.DataSource = mObjDAL.ExecDatasetTx(str).Tables(0)
        End If
    End Sub

    Private Sub btnDOK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnDOK.Click
        If Trim(txtDName.Text) = "" Then
            TCEmployee.SelectedTab = TPDeduction
            txtDName.Focus()
            ErrorProvider.SetError(txtDName, "Name cannot be empty")
        ElseIf Trim(txtDAmount.Text) = "" Then
            TCEmployee.SelectedTab = TPDeduction
            txtDAmount.Focus()
            ErrorProvider.SetError(txtDAmount, "Amount cannot be empty")
        ElseIf Not IsNumeric(Trim(txtDAmount.Text)) Then
            TCEmployee.SelectedTab = TPDeduction
            txtDAmount.Focus()
            ErrorProvider.SetError(txtDAmount, "Amount can contain only number")
        Else
            If mDAction = DActionResult.Add Then
                Dim strAdd As String

                strAdd = "INSERT INTO Deduction " & _
                        "(UID, Name, Amount, EmployeeID) " & _
                    "VALUES " & _
                        "( '" & Guid.NewGuid.ToString & "', " & _
                        "'" & TrimSqlVariable(txtDName.Text) & "', " & _
                        "" & CDbl(txtDAmount.Text) & ", " & _
                        "'" & TrimSqlVariable(txtID.Text) & "')"

                mObjDAL.ExecNonQueryTx(strAdd)
            ElseIf mDAction = DActionResult.Edit Then
                Dim strUpdate As String

                strUpdate = "UPDATE Deduction " & _
                        "SET Name = '" & TrimSqlVariable(txtDName.Text) & "', " & _
                          "Amount = '" & TrimSqlVariable(txtDAmount.Text) & "' " & _
                        "WHERE UID = '" & mUID.ToString & "'"

                mObjDAL.ExecNonQueryTx(strUpdate)
            End If
            mDAction = DActionResult.Add

            Dim str As String
            str = "SELECT * FROM Deduction WHERE EmployeeID = '" & TrimSqlVariable(txtID.Text) & "' "
            DGDeduction.DataSource = mObjDAL.ExecDatasetTx(str).Tables(0)

        End If
    End Sub

    Private Sub DGAllowance_DoubleClick(ByVal sender As Object, ByVal e As System.EventArgs) Handles
DGAllowance.DoubleClick
        If IsDataGridRowClicked(DGAllowance) = True Then
            mAAction = AActionResult.Edit
            mUID = New Guid(DGAllowance.Item(DGAllowance.CurrentRowIndex, 0).ToString)
            txtAName.Text = DGAllowance.Item(DGAllowance.CurrentRowIndex, 1).ToString
            txtAAmount.Text = DGAllowance.Item(DGAllowance.CurrentRowIndex, 2).ToString
        End If
    End Sub

    Private Sub DGDeduction_DoubleClick(ByVal sender As Object, ByVal e As System.EventArgs) Handles
DGDeduction.DoubleClick
        If IsDataGridRowClicked(DGAllowance) = True Then
            mDAction = DActionResult.Edit
            mUID = New Guid(DGDeduction.Item(DGDeduction.CurrentRowIndex, 0).ToString)
            txtAName.Text = DGDeduction.Item(DGDeduction.CurrentRowIndex, 1).ToString
            txtAAmount.Text = DGDeduction.Item(DGDeduction.CurrentRowIndex, 2).ToString
        End If
    End Sub
    Private Sub btnAClear_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnAClear.Click
        mAAction = AActionResult.Add
        txtAName.Clear()
        txtAAmount.Clear()
    End Sub

    Private Sub btnDClear_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnDClear.Click
        mDAction = DActionResult.Add
        txtDName.Clear()
        txtDAmount.Clear()
    End Sub

    Private Sub txtID_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
txtID.TextChanged
        txtUsername.Text = txtID.Text
    End Sub
```

FrmProcessPay

```
  Private Sub btnOK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnOK.Click
    Dim strAdd As String

    'Delete and add history of the month

    'delete all the existing first (current month and current year)
    Dim strDelete As String

    strDelete = "DELETE FROM History " & _
            "WHERE [Month] = '" & Now.Month.ToString & "' " & _
            "AND [Year] = '" & Now.Year.ToString & "'"

    mObjDAL.ExecNonQueryTx(strDelete)

    PB.Value = 10

    Dim strGet As String
    Dim dt As New DataTable()
    Dim i As Integer

    strGet = "SELECT * FROM Employee WHERE PayStatus = 'Active'"

    dt = mObjDAL.ExecDatasetTx(strGet).Tables(0)

    For i = 0 To dt.Rows.Count - 1
      Dim EmployeeID As String = dt.Rows(i).Item("ID").ToString

      'insert allowance
      Dim strA As String
      Dim dtA As New DataTable()
      Dim A As Integer

      strA = "SELECT * FROM Allowance WHERE EmployeeID = '" & EmployeeID & "'"

      dtA = mObjDAL.ExecDatasetTx(strA).Tables(0)

      For A = 0 To dtA.Rows.Count - 1
        With dtA.Rows(A)
          strAdd = "INSERT INTO History " & _
                  "(Name, EmployeeAmount, EmployeeID, [Month], [Year], Type) " & _
              "VALUES " & _
                  "( '" & TrimSqlVariable(.Item("Name").ToString) & "', " & _
                  "'" & CDbl(.Item("Amount")) & "', " & _
                  "'" & TrimSqlVariable(EmployeeID) & "', " & _
                  "'" & Now.Month.ToString & "', " & _
                  "'" & Now.Year.ToString & "', " & _
                  "'Allowance')"

          mObjDAL.ExecNonQueryTx(strAdd)
        End With
      Next

      'insert deduction
      Dim strD As String
      Dim dtD As New DataTable()
      Dim D As Integer

      strD = "SELECT * FROM Deduction WHERE EmployeeID = '" & EmployeeID & "'"

      dtD = mObjDAL.ExecDatasetTx(strD).Tables(0)

      For D = 0 To dtD.Rows.Count - 1
        With dtD.Rows(D)
          strAdd = "INSERT INTO History " & _
                  "(Name, EmployeeAmount, EmployeeID, [Month], [Year], Type) " & _
              "VALUES " & _
                  "( '" & TrimSqlVariable(.Item("Name").ToString) & "', " & _
                  "'" & CDbl(.Item("Amount")) & "', " & _
```

```
                        "'" & TrimSqlVariable(EmployeeID) & "', " & _
                        "'" & Now.Month.ToString & "', " & _
                        "'" & Now.Year.ToString & "', " & _
                        "'Deduction')"

            mObjDAL.ExecNonQueryTx(strAdd)
          End With
        Next

        Dim TAllowance As Double
        Dim TDeduction As Double
        Dim Basic As Double

        strA = "SELECT SUM(Amount) AS TAllowance FROM Allowance WHERE EmployeeID = '" &
TrimSqlVariable(EmployeeID) & "'"
        strD = "SELECT SUM(Amount) AS TDeduction FROM Deduction WHERE EmployeeID = '" &
TrimSqlVariable(EmployeeID) & "'"

        If Not IsDBNull(mObjDAL.ExecDatasetTx(strA).Tables(0).Rows(0).Item("TAllowance")) Then
          TAllowance = CDbl(mObjDAL.ExecDatasetTx(strA).Tables(0).Rows(0).Item("TAllowance"))
        Else
          TAllowance = 0
        End If

        If Not IsDBNull(mObjDAL.ExecDatasetTx(strD).Tables(0).Rows(0).Item("TDeduction")) Then
          TDeduction = CDbl(mObjDAL.ExecDatasetTx(strD).Tables(0).Rows(0).Item("TDeduction"))
        Else
          TDeduction = 0
        End If


        Basic = CDbl(dt.Rows(i).Item("BasicPayment"))

        'insert EPF
        Dim EmployeeEPF As Double
        Dim EmployerEPF As Double
        EmployeeEPF = (Basic + TAllowance - TDeduction) * (9 / 100)
        EmployerEPF = (Basic + TAllowance - TDeduction) * (12 / 100)

        strAdd = "INSERT INTO History " & _
                  "(Name, EmployeeAmount, EmployerAmount, EmployeeID, [Month], [Year], Type) " & _
              "VALUES " & _
                  "('EPF', " & _
                   "'" & EmployeeEPF & "', " & _
                   "'" & EmployerEPF & "', " & _
                   "'" & TrimSqlVariable(EmployeeID) & "', " & _
                   "'" & Now.Month.ToString & "', " & _
                   "'" & Now.Year.ToString & "', " & _
                   "'Deduction')"

        mObjDAL.ExecNonQueryTx(strAdd)

        'insert SOCSO
        Dim SOCSO As Double = 0

        If Basic <= 1500 Then
          SOCSO = 5.75
        ElseIf Basic <= 1800 Then
          SOCSO = 7.75
        ElseIf Basic > 1800 Then
          SOCSO = 9.75
        End If

        If SOCSO <> 0 Then
          strAdd = "INSERT INTO History " & _
                  "(Name, EmployeeAmount, EmployeeID, [Month], [Year], Type) " & _
              "VALUES " & _
                  "('SOCSO', " & _
                   "'" & SOCSO & "', " & _
                   "'" & TrimSqlVariable(EmployeeID) & "', " & _
```

```
                    "'" & Now.Month.ToString & "', " & _
                    "'" & Now.Year.ToString & "', " & _
                    "'Deduction')"

            mObjDAL.ExecNonQueryTx(strAdd)
        End If

        'insert PCB
        Dim PCB As Double = 0
        If Basic >= 2501 And Basic <= 3000 Then
            PCB = (Basic + TAllowance - TDeduction) * (0.8 / 100)
        ElseIf Basic >= 3001 And Basic <= 4000 Then
            PCB = (Basic + TAllowance - TDeduction) * (0.9 / 100)
        ElseIf Basic >= 4001 And Basic <= 5000 Then
            PCB = (Basic + TAllowance - TDeduction) * (1 / 100)
        ElseIf Basic > 5000 Then
            PCB = (Basic + TAllowance - TDeduction) * (1.2 / 100)
        End If

        If PCB <> 0 Then
            strAdd = "INSERT INTO History " & _
                    "(Name, EmployeeAmount, EmployeeID, [Month], [Year], Type) " & _
                "VALUES " & _
                    "( 'PCB', " & _
                    "" & PCB & ", " & _
                    "'" & TrimSqlVariable(EmployeeID) & "', " & _
                    "'" & Now.Month.ToString & "', " & _
                    "'" & Now.Year.ToString & "', " & _
                    "'Deduction')"

            mObjDAL.ExecNonQueryTx(strAdd)
        End If

        'insert Basic
        strAdd = "INSERT INTO History " & _
                    "(Name, EmployeeAmount, EmployeeID, [Month], [Year], Type) " & _
                "VALUES " & _
                    "( 'Basic', " & _
                    "" & Basic & ", " & _
                    "'" & TrimSqlVariable(EmployeeID) & "', " & _
                    "'" & Now.Month.ToString & "', " & _
                    "'" & Now.Year.ToString & "', " & _
                    "'Basic')"

        mObjDAL.ExecNonQueryTx(strAdd)
    Next
    PB.Value = 100
End Sub
```

FrmEmployeePayment

```
   Private Sub frmEmployee_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
      Dim strGet As String
      Dim dt As New DataTable()

      strGet = "SELECT * " & _
            "FROM Employee " & _
            "WHERE ID = '" & mID & "'"

      dt = mObjDAL.ExecDatasetTx(strGet).Tables(0)

      With dt.Rows(0)
         txtID.Text = .Item("ID").ToString
         txtName.Text = .Item("FirstName").ToString & " " & .Item("LastName").ToString
         txtIC.Text = .Item("NewIC").ToString
         txtBasicPayment.Text = Format(.Item("BasicPayment"), "0.00")
      End With

      Dim AGridStyle As New DataGridTableStyle()
      Dim colAUID As New DataGridNoActiveCellColumn()
      Dim colAName As New DataGridNoActiveCellColumn()
      Dim colAAmount As New DataGridNoActiveCellColumn()
      Dim Astr As String

      DGAllowance.RowHeaderWidth = 0

      Astr = "SELECT * FROM History " & _
         "WHERE EmployeeID = '" & TrimSqlVariable(txtID.Text) & "' " & _
         "AND [Month] = '" & Now.Month.ToString & "' " & _
         "AND [Year] = '" & Now.Year.ToString & "' " & _
         "AND Type = 'Allowance'"

      AGridStyle.MappingName = mObjDAL.ExecDatasetTx(Astr).Tables(0).TableName

      AGridStyle.GridColumnStyles.Add(colAUID)
      colAUID.Alignment = HorizontalAlignment.Left
      colAUID.MappingName = "UID"
      colAUID.Width = 0

      AGridStyle.GridColumnStyles.Add(colAName)
      colAName.Alignment = HorizontalAlignment.Left
      colAName.HeaderText = "Name"
      colAName.MappingName = "Name"
      colAName.Width = 400

      AGridStyle.GridColumnStyles.Add(colAAmount)
      colAAmount.Alignment = HorizontalAlignment.Left
      colAAmount.HeaderText = "Amount"
      colAAmount.MappingName = "EmployeeAmount"
      colAAmount.Format = "0.00"
      colAAmount.Width = DGAllowance.Width - _
               DGAllowance.RowHeaderWidth - _
               colAName.Width - 40

      DGAllowance.TableStyles.Add(AGridStyle)
      DGAllowance.DataSource = mObjDAL.ExecDatasetTx(Astr).Tables(0)

      Dim DGridStyle As New DataGridTableStyle()
      Dim colDUID As New DataGridNoActiveCellColumn()
      Dim colDName As New DataGridNoActiveCellColumn()
      Dim colDAmount As New DataGridNoActiveCellColumn()
      Dim Dstr As String

      DGDeduction.RowHeaderWidth = 0

      Dstr = "SELECT * FROM History " & _
         "WHERE EmployeeID = '" & TrimSqlVariable(txtID.Text) & "' " & _
         "AND [Month] = '" & Now.Month.ToString & "' " & _
```

```
        "AND [Year] = '" & Now.Year.ToString & "' " & _
        "AND Type = 'Deduction'"

    DGridStyle.MappingName = mObjDAL.ExecDatasetTx(Dstr).Tables(0).TableName

    DGridStyle.GridColumnStyles.Add(colDUID)
    colDUID.Alignment = HorizontalAlignment.Left
    colDUID.MappingName = "UID"
    colDUID.Width = 0

    DGridStyle.GridColumnStyles.Add(colDName)
    colDName.Alignment = HorizontalAlignment.Left
    colDName.HeaderText = "Name"
    colDName.MappingName = "Name"
    colDName.Width = 400

    DGridStyle.GridColumnStyles.Add(colDAmount)
    colDAmount.Alignment = HorizontalAlignment.Left
    colDAmount.HeaderText = "Amount"
    colDAmount.MappingName = "EmployeeAmount"
    colDAmount.Format = "0.00"
    colDAmount.Width = DGDeduction.Width - _
                DGDeduction.RowHeaderWidth - _
                colDName.Width - 40

    DGDeduction.TableStyles.Add(DGridStyle)
    DGDeduction.DataSource = mObjDAL.ExecDatasetTx(Dstr).Tables(0)

    Astr = "SELECT SUM(EmployeeAmount) AS ATotal " & _
        "FROM History WHERE EmployeeID = '" & TrimSqlVariable(txtID.Text) & "' " & _
        "AND [Month] = '" & Now.Month.ToString & "' " & _
        "AND [Year] = '" & Now.Year.ToString & "' " & _
        "AND Type = 'Allowance'"

    Dstr = "SELECT SUM(EmployeeAmount) AS DTotal " & _
        "FROM History WHERE EmployeeID = '" & TrimSqlVariable(txtID.Text) & "' " & _
        "AND [Month] = '" & Now.Month.ToString & "' " & _
        "AND [Year] = '" & Now.Year.ToString & "' " & _
        "AND Type = 'Deduction'"

    Dim Bstr As String
    Bstr = "SELECT SUM(EmployeeAmount) AS Basic " & _
        "FROM History WHERE EmployeeID = '" & TrimSqlVariable(txtID.Text) & "' " & _
        "AND [Month] = '" & Now.Month.ToString & "' " & _
        "AND [Year] = '" & Now.Year.ToString & "' " & _
        "AND Type = 'Basic'"

    Dim allowance, deduction As Double

    If Not IsDBNull(mObjDAL.ExecDatasetTx(Astr).Tables(0).Rows(0).Item("ATotal")) Then
      allowance = mObjDAL.ExecDatasetTx(Astr).Tables(0).Rows(0).Item("ATotal")
    Else
      allowance = 0
    End If

    If Not IsDBNull(mObjDAL.ExecDatasetTx(Dstr).Tables(0).Rows(0).Item("DTotal")) Then
      deduction = mObjDAL.ExecDatasetTx(Dstr).Tables(0).Rows(0).Item("DTotal")
    Else
      deduction = 0
    End If

    lblTotal.Text = CDbl(mObjDAL.ExecDatasetTx(Bstr).Tables(0).Rows(0).Item("Basic")) + _
              allowance - deduction

    lblTotal.Text = Format(CDbl(lblTotal.Text), "0.00")
    If CDbl(lblTotal.Text) < 0 Then
      lblTotal.Text = "0.00"
    End If
    End Sub
```

Chapter 7 : SYSTEM EVALUATION AND CONCLUSION

7.1 Introduction

A lot of system analysis needs to be done on technologies and programming concepts before starting to develop PMS system. The basic knowledge needed as a foundation in building an application of this nature involves studies in fields such as Visual Basic, information system and others. Overall for the development of PMS, a few problems were encountered. However, most of them were resolved eventually. Some of the problem encountered was:

7.1.1 Lack of knowledge In the Programming Language

Due to time constraints the learning and developing process was dine in parallel. Without a strong base of the language, we need to spend a lot of time looking for solutions to solve problems encountered that occurred during the development. This happen mostly in situations related to the concept of programming language that is new.

7.1.2 Slow Response Time

Some of the modules need to be able to response on minimum amount of time. If all the information input of each user is stored in database, the response time will be very slow and thus favorable. In order to speed up response time, each form's information to be filled by the administrator in updating the latest employee data process was stored in a separate table rather than having to store all information of 1 user into 1 table. It will be much easier this way and the administrator could view information much more conveniently as well.

7.1.3 Difficulty in Choosing An Appropriate Operating System

There were some difficulties in choosing the appropriate OS to host. Because of limited facilities in the faculty and problems as well as lack of resources, therefore windows 2000 professional was used for it is considered a stable and robust OS available.

7.2 System Strength

7.2.1 User Friendly

In overall, PMS could be evaluated as a simple to use application. PNS provides simple, user friendly and graphical based interface for user to deal with it. Besides, sufficient instructions and guidance are provided to guide and assist in saving their time to learn on using the system. For example, error messages will be displayed to guide users whenever invalid user inputs are encountered by the system.

7.2.2 Transparency

The system is transparency to the users, as they do not need to know where the database resides, how the system is structured. For example, users do not need to know how to retrieve and insert records into the database. All they need to do is submit data and then view necessary information required.

7.2.3 Error Messaging

The error messages are immediately displayed after a button is clicked if the user has input in the wrong information required. Error pages or messages box will be displayed to allow users to identify their errors effectively and make appropriate corrections.

7.2.4 System Weaknesses

There are some limitations due to time constraints, facilities, limitations and constraints of the programming language itself including:

7.2.5 Mailing Capabilities

There is no function available to mail the employee data or inquires to the administrator.

7.2.6 Online System

There is no online transaction that available for user interface and PMS system to be connected.

7.2.7 Help Module

There is no help module available here in PMS system for new user.

7.3 Problem face during system development

During the early development of our PMS system, we face problems in integrating our system because it consists of three different auditing steps. Besides that, since payroll system is a very specific subject so there's not much reference can be found in the library or the Internet. In addition to that, we also face problems when using visual basic to connect Microsoft access database since the visual basic software that we use is not compatible with the Microsoft access that is available in our computer. Also, we face some problem with adapting the SQL server database algorithm with our system. However, we manage to solve our problem through guidance from the book and En. Mustaffa, our supervisor.

7.4 Future Enhancements

Some of future enhancements that should be considered to be included:

7.4.1 Software Update

Database development tools used is Microsoft SQL Server. In future, a higher performance and more stable database platform such as ORACLE. In addition, we have use Visual Basic 6.0 as our programming language and to improve it, the usage of latest programming language which is Java would be implemented.

7.4.2 Security Enforcement

To enforce Secure Socket Layer (SSL) so that all employee identification will not be viewed by anyone else. The possibility of frauds will be reduced.

7.4.3 Online Connection

In the coming future, we will make this system web-based because most of the system nowadays can be found online. By putting it online, users can access it 24/7 and they can also check their application themselves to see if it is approved or not. And fraud can be detected anytime during the transaction. Also, we will be able to provide the email function by putting it online so that auditors can inform the users with related matters.

7.4.4 User Interface

To make the interface more realistic and professional, we would create it in way that it is applicable in the real working environment.

7.5 Achievement of Objectives

The primary goal of this project is to evaluate the usefulness of PMS system in supporting auditing works. We have build intelligent system that can help business discover hidden patterns in their data. Identified high-risk and low-risk of user application. Help business to understand the purchasing behavior of their key customers, detect likely commercial fraud, and predict credit application approval and detect any unauthorized activity through intrusion detection method. Besides that, we also provide a computer system protection against intrusion from intruder and hence prevent fraudulent activity.

7.6 Conclusion

Overall, the requirements of this project as determined during the system analysis phase were done eagerly. PMS uses the database management system to do maintenance. Database is setup to record all the records of the employee and the administrator. For example the employee data, database will record the updating processes like edited administrator password or user and the status of the employee data whether it is active or not. The aim of this project is to develop a system for the use of auditors to audit all data and identify the unusual pattern and also to check frequently occur episode.