**STUDENT SUPERVISION MANAGEMENT SYSTEM**

**(e-SSMS)**

**UMI AMIRA BINTI NORIZAN**

**UNIVERSITI TEKNIKAL MALAYSIA MELAKA**

**BORANG PENGESAHAN STATUS TESIS**

**STUDENT SUPERVISION MANAGEMENT SYSTEM**

**(e-SSMS)**

**UMI AMIRA BINTI NORIZAN**

**This report is submitted in partial fulfilment of the requirements for the Bachelor of Computer Science (Database Management)**

**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY UNIVERSITI TEKNIKAL MALAYSIA MELAKA**

**2017**

**DECLARATION**

**DEDICATION**

There are number of people without whom this thesis might not have been written. I dedicated this thesis for who I greatly appreciated.

To my beloved parents, Iswati and Norizan, who always support me through my upside down life as a student. People who has been an inspiration and source of encouragement to me throughout my life.

Last but not least, thanks to my family and friends, a person who see my potential as a great trustworthy and reliable student which develop me from nothing to something. I dedicated for you guys because you are one of the people who inspired me the most.

**ACKNOWLEDGEMENTS**

Alhamdulillah, all praises to Allah for the strengths and His blessing in completing this project. I really grateful because managed to complete this final year project although I slightly miss much of the time given.

I would like to express my gratitude and appreciation to all who gave me the responsibility to complete this report. A big thank you to my sypervisor,

Dr. Safiza Suhana Bt. Kamal Baharin

in giving suggestions and encouragement throughout my project especially in development of the system. She gives me a lot of encouragement and thank you.

Last but not least, thanks to people who help me directly or indirectly for the support and guidance given to improve the report produced and give more confidence in presentation skills by the comments and tips.

**ABSTRACT**

This report describes the activities that I have done throughout 1 semester doing “Project Sarjana Muda” during this one semester at the Universiti Teknikal Malaysia Melaka (UTeM). In this report it will describes in detail about how the e-SSMS is developed and what steps are being taken to ensure the system is running as planned.

In this report there is a chapter in which the Introduction of this chapter will explain about problems that occur and man objective of this system is developed. In addition, other chapters such as methods and planning, analysis, design, implementation and testing will be described in detail through reports provided. The last chapter is a conclusion where in this chapter, will be described on the overall conclusion when making other chpaters.

**ABSTRAK**

Laporan ini menerangkan tentang aktiviti yang saya lakukan sepanjang menjalani tempoh Projek Sarjana Muda selama 1 semester di Universiti Teknikal Malaysia Melaka (UTeM). Di dalam laporan ini menerangkan secara terperinci tentang bagaimana sistem e-SSMS ini dibangunkan serta Apa langkah-langkah yang diambil untuk memastikan sistem ini berjalan seperti yang dirancangkan.

Di dalam laporan ini terdapat bab pengenalan di mana bab ini akan menerankan tentang permasalahan yang berlaku dan objektif utama sistem ini dibangunkan. Selain itu, bab-bab yang lain seperti kaeah dan perancangan, analisis, reka bentuk, pelaksanaan dan ujian akan diterangkan dengan lebih mendalam menerusi laporan yang dsediakan. Bab yang terakhir ialah konklusi di mana dalam bab ini akan diterangkan tentang konklusi secara menyeluruh apabila membuat bab-bab yang lain.

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **SUBJECT** | **PAGE** |
|  | **DECLARATION**  **DEDICTION**  **ACKNOWLEDGEMENTS**  **ABSTRACT**  **ABSTRAK**  **TABLE OF CONTENT**  **LIST OF TABLES**  **LIST OF FIGURES**  **LIST OF ABBREVATIONS**  **LIST OF ATTACHMENTS** |  |
| **CHAPTER I** | **INTRODUCTION**   * 1. Introduction   2. Problem Statement(s)   3. Objective   4. Scope   1.4.1 Scope of user  1.4.2 Scope of module   * 1. Project Significance   2. Expected Output   3. Conclusion |  |
| **CHAPTER II** | **PROJECT METHODOLOGY AND PLANNING**  2.1 Introduction  2.2 Project Methodology  2.3 Project Schedule and Milestones  2.4 Conclusion |  |
| **CHAPTER III** | **ANALYSIS**  3.1 Introduction  3.2 Project Analysis  3.2.1 Background of current system  3.3 Requirement Analysis  3.3.1 Data requirement  3.3.2 Functional requirement  3.3.2.1 Context Diagram  3.3.2.2 Data Flow Diagram (DFD)  3.3.3 Functional Requirement (Process Model)  3.3.4 Non-functional requirement  3.3.5 Other requirement  3.3.5.1 Software requirement  3.3.5.2 Hardware requirement  3.3.5.3 Network requirement  3.4 Conclusion |  |
| **CHAPTER IV** | **DESIGN**  4.1 Introduction  4.2 Database design  4.2.1 Conceptual design  4.2.2 Logical design  4.2.3 Physical design  4.3 Graphical User Interface (GUI) Design  4.4 Conclusion |  |
| **CHAPTER V** | **IMPLEMENTATION**  5.1 Introduction  5.2 System Development Environment Setup  5.3 Database Implementation  5.4 Conclusion |  |
| **CHAPTER VI** | **TESTING**  6.1 Introduction  6.2 Test Plan  6.2.1 Test Organization  6.2.2 Test Environment  6.2.3 Test Schedule  6.3 Test Strategy  6.3.1 Classes of Tests  6.4 Test Design  6.4.1 Test Description  6.4.2 Test Data  6.5 Test Results and Analysis  6.6 Conclusion |  |
| **CHAPTER VII** | 7.1 Introduction  7.2 Observation on Weakness and Strengths  7.3 Propositions for Improvement  7.4 Contribution  7.5 Conclusion |  |
|  | **REFERENCES**  **BIBULIOGRAPHY**  **APPENDICES** |  |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE** | **TITLE** | **PAGE** |
| **Table 3.1** | **Position Table** |  |
| **Table 3.2** | **Description of Software Requirements** |  |
| **Table 3.3** | **Description of Hardware Requirements** |  |
| **Table 3.4** | **Description of Network Requirements** |  |
| **Table 4.1** | **Position Data Dictionary** |  |
| **Table 5.1** | **CREATE TABLE clause** |  |
| **Table 5.2** | **SELECT clause** |  |
| **Table 5.3** | **WHERE clause** |  |
| **Table 5.4** | **TRIGGER clause** |  |
| **Table 5.5** | **Stored Procedure** |  |
| **Table 6.1** | **Test Organization** |  |
| **Table 6.2** | **Environment Setup Specification** |  |
| **Table 6.3** | **Test Schedule Detail** |  |
| **Table 6.4** | **Details of testing type** |  |
| **Table 6.5** | **Test Case of Login for Admin** |  |
| **Table 6.6** | **Test Case of Login for Committee and Supervisor** |  |
| **Table 6.7** | **Test Case of Profile for Student** |  |
| **Table 6.8** | **Test Case of Profile for Committee and Supervisor** |  |
| **Table 6.9** | **Test Case of Appointment Details for Student** |  |
| **Table 6.10** | **Test Case of Appointment Details for Committee and Supervisor** |  |
| **Table 6.11** | **Test Case of Proposal Details for Student** |  |
| **Table 6.12** | **Test Case of Proposal Details for Committee and Supervisor** |  |
| **Table 6.13** | **Test Case of Final report Details for Student** |  |
| **Table 6.14** | **Test Case of Final report Details for Committee and Supervisor** |  |
| **Table 6.15** | **Test Case of Logbook Details for Student** |  |
| **Table 6.16** | **Test Case of Change password Details** |  |
| **Table 6.17** | **Details of Login Test Data** |  |
| **Table 6.18** | **Details of Profile Test Data** |  |
| **Table 6.19** | **Details of Appointment Test Data** |  |
| **Table 6.20** | **Details of Proposal Test Data** |  |
| **Table 6.21** | **Details of Final report Test Data** |  |
| **Table 6.22** | **Details of Logbook Test Data** |  |
| **Table 6.23** | **Details of Change password Test Data** |  |
| **Table 6.24** | **Test Result and Analysis of Valid Login Details** |  |
| **Table 6.25** | **Test Result and Analysis of Invalid Login Details** |  |
| **Table 6.26** | **Test Result and Analysis of Valid Profile Details** |  |
| **Table 6.27** | **Test Result and Analysis of Invalid Profile Details** |  |
| **Table 6.28** | **Test Result and Analysis of Valid Appointment Details** |  |
| **Table 6.29** | **Test Result and Analysis of Invalid Appointment Details** |  |
| **Table 6.30** | **Test Result and Analysis of Valid Proposal Details** |  |
| **Table 6.31** | **Test Result and Analysis of Invalid Proposal Details** |  |
| **Table 6.32** | **Test Result and Analysis of Valid Final report Details** |  |
| **Table 6.33** | **Test Result and Analysis of Invalid Final report Details** |  |
| **Table 6.34** | **Test Result and Analysis of Valid Logbook Details** |  |
| **Table 6.35** | **Test Result and Analysis of Invalid Logbook Details** |  |
| **Table 6.36** | **Test Result and Analysis of Valid Change password Details** |  |
| **Table 6.37** | **Test Result and Analysis of Invalid Change password Details** |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE** | **TITLE** | **PAGE** |
| **Figure 2.1** | **Database Methodology on DBLC** |  |
| **Figure 3.1** | **Context Diagram of e-SSMS** |  |
| **Figure 3.2** | **Data Flow Diagram (DFD) of e-SSMS** |  |
| **Figure 3.3** | **Login process level 1** |  |
| **Figure 3.4** | **Supervision process level 1** |  |
| **Figure 3.5** | **Report process level 1** |  |
| **Figure 3.6** | **Appointment process level 1** |  |
| **Figure 4.1** | **e-SSMS Entity Relationship Diagram (ERD)** |  |
| **Figure 4.3.1** | **Login interface** |  |
| **Figure 4.3.2** | **Login interface (Admin)** |  |
| **Figure 4.3.3** | **Main page interface** |  |
| **Figure 4.3.4** | **Edit profile interface (Student)** |  |
| **Figure 4.3.5** | **Edit profile interface (Committee and Supervisor)** |  |
| **Figure 4.3.6** | **Make appointment interface (Student)** |  |
| **Figure 4.3.7** | **View Appointment interface Student)** |  |
| **Figure 4.3.8** | **View appointment to need to be approve interface (Supervisor/Committee)** |  |
| **Figure 4.3.9** | **Appointment form interface that need to be approve (Supervisor/Committee)** |  |
| **Figure 4.3.10** | **View proposal interface** |  |
| **Figure 4.3.11** | **Upload proposal interface** |  |
| **Figure 4.3.12** | **View Final report interface** |  |
| **Figure 4.3.13** | **Upload Final report interface** |  |
| **Figure 4.3.14** | **Submit logbook interface** |  |
| **Figure 4.3.15** | **List of student’s proposal interface** |  |
| **Figure 4.3.16** | **Assign due submission (proposal) interface** |  |
| **Figure 4.3.17** | **List of student’s final report interface** |  |
| **Figure 4.3.18** | **Assign due submission (final report) interface** |  |
| **Figure 4.3.19** | **List of supervisor to be supervise interface** |  |
| **Figure 4.3.20** | **Assignation supervisor interface** |  |
| **Figure 4.3.21** | **Lecturer information interface** |  |
| **Figure 4.3.22** | **Reset password lecturer interface** |  |
| **Figure 4.3.23** | **Student information interface** |  |
| **Figure 4.3.24** | **Reset password student interface** |  |
| **Figure 4.3.25** | **List of supervisor to be change position interface** |  |
| **Figure 4.3.26** | **Change position interface** |  |
| **Figure 4.3.27** | **Change password interface** |  |
| **Figure 4.3.28** | **List student to be supervise and list of previous student** |  |
| **Figure 4.3.29** | **Detail report of student** |  |
| **Figure 4.3.30** | **Review Proposal** |  |
| **Figure 4.3.31** | **Assign mark for proposal workshop 1** |  |
| **Figure 4.3.32** | **Review Final Report** |  |
| **Figure 4.3.33** | **Assign mark for final report workshop 2** |  |
| **Figure 5.1** | **Tier System Architecture** |  |
| **Figure 5.2** | **Xampp Server installation** |  |
| **Figure 5.3** | **Directory interface** |  |
| **Figure 5.4** | **XAMPP Control Panel** |  |
| **Figure 5.5** | **Install Apache** |  |
| **Figure 5.6** | **Run as administrator** |  |
| **Figure 5.7** | **XAMPP Server Configuration Interface** |  |
| **Figure 5.8** | **phpMyAdmin database** |  |

**LIST OF ABBREVIATIONS**

|  |  |  |
| --- | --- | --- |
| DBMS | **-** | Database management system |
| DCL | **-** | Digital common language |
| DDL | **-** | Data definition language |
| DML | **-** | Data manipulation language |
| GUI | **-** | Graphical user interface |
| ID | **-** | Identity |
| PSM | **-** | Projek Sarjana Muda |
| SDLC | **-** | System development life cycle |
| DBLC | **-** | Database life cycle |
| SQL | **-** | Structured Query Language |

**LIST OF ATTACHEMENTS**

|  |  |  |
| --- | --- | --- |
| **ATTACHMENT** | **TITLE** | **PAGE** |
| **1.1** | **Appendix A** |  |
| **1.2** | **Appendix B** |  |
| **1.3** | **Appendix C** |  |
| **1.4** | **Appendix D** |  |
| **1.5** | **Appendix E** |  |

**CHAPTER I**

**INTRODUCTION**

* 1. **Project background**

In the Faculty of Information and Communication Technology (FTMK) of UTeM, it is compulsory for each of the student in second (2nd)and third (3rd) year semester to undergo Workshop 1 and Workshop 2 courses. All of this two courses required student to develop and present the project progress to the supervisor along the development process. Two of this courses is handle by a different committee which is consist of lecturers from different courses. The lecturers shall supervise the progress of the Workshop 1 and Workshop 2 by conducting activities such as series of discussion and presentation between student and lecturer.

The purpose of this research is to develop Student Supervision Management System (e-SSMS) for the student and lecturers that are involved in Workshop 1 and Workshop 2 courses. The system is consists of functions such as appointment scheduler, committee assignation, supervisor to student assignation,updating logbook activities and uploading workshop proposal and final report.

The system shall simplified and automate the current manual process currently practiced in Workshop 1 and Workshop 2 such as appointment scheduling between students and lecturers.By using the appointment module in e-SSMS, student can choose date and time based on the lecturer’s availability displayed in the system. Then, the respective lecturers can reject or approves all the requested appointment. Student can also view the status of their appointment upon approval of their supervisor. Thus this process and procedure will become much smoother and save times. Besides making an appointment, this system also provide the functionality of uploading the proposal, final report and log book. So the record of log book can be monitor week by week. Supervisor can also monitor the students by reviewing all activities that have been updated in the system.

* 1. **Problem Statement(s)**

A good planning and communication is one of the main factors the success of a system developement (J. L. Cuadrado-Garcia, 2011*)*. Lecturers and student who are involved in the courses of Workshop 1 and Workshop 2 should able to communicate efficiently in discussion and project progress tracking in order to achieve success of a system development. However, currently there are no system existed to assist student and lecturer to communicate efficiently and hence resulted difficulty in tracking workshop progress, lack of proper record of discussion and difficulty in scheduling appointment with supervisor.

From the student’s perspective, students find it difficult to make an appointment with their supervisor to show their development progress or to seek guidance for the project. Supervisor were unable to entertain all of their student due to their busy schedule. Student also having difficulty to keep track on their systematic log book which is used to record discussion and task given by their supervisor.

Furthermore, as a lecturer. They will be assigned to be the supervisor for several students who taking the Workshop 1 and Workshop 2 courses. Currently, supervisor is having difficulty to keep track filling and progress of the proposal and reports of previous and current students which is done manually.

In response to this problem, this research proposes to develop Student Supervision Management System (e-SSMS) for both student and lecturer of Workshop 1 and Wokshop 2 to assist them on the managing the development of their projects.

* 1. **Objective**

**Objective 1 :** To develop Student Supervision Management System that enables students and lecturer to manage their information effectively via web based platform.

**Objective 2 :** To provide a platform that enable students and lecturers to schedule their appointment for workshop project discussion.

**Objective 3 :** To create database of document and activities that enables student and lecturer to keep track their documents and task effectively.

* 1. **Project Scope**
     1. **Scope of user**

1. **Administrator**
2. Login to the system using secured password.
3. User shall able to access following functions:

* Update information lecturer and student.
* Assign supervisor for student workshop 1 and workshop 2.
* Assign lecturers to committee for workshop 1 and workshop 2.
* Change password.

1. **FTMK Committee**
2. Login to the system using secured password.
3. User shall able to access following functions:

* Update their own information.
* View all appointment that need to be approved.
* View information of student that need to be supervise.
* View all document that have been submit by student, such as proposal, final report and logbook.
* Assign mark and approve project of proposal and final report.
* View proposal and final report of student workshop 1 and workshop 2.
* Assign due submission of proposal and final report for workshop 1 and workshop 2.
* Assign supervisor for student workshop 1 and workshop 2.
* Change password.

1. **FTMK Lecturer**
2. Login to the system using secured password.
3. User shall able to access following functions:

* Update their own information.
* View all appointment that need to be approved.
* View information of student that need to be supervise.
* View all document that have been submit by student, such as proposal, final report and logbook.
* Assign mark and approve project of proposal and final report.
* Change password.

1. **FTMK Student**
2. Login to the system using secured password.
3. User shall able to access following functions:

* Update their information.
* Able to make an appointments.
* Able to view status of appointments.
* List of appointments history.
* Able to add, update and delete logbook activities.
* Able to uploading workshop proposal and final report.
* Able to view all uploaded documents.
* Change password.
  + 1. **Scope of module**

1. **Appointment module**

* Student shall able to make an appointment with their respective supervisor only.
* Supervisor shall able to approve or reject submitted appointment from their students only.
* Student can view status of their appointment between lecturers. Provide two ways communication between student and lecturers to setup an appointment.
* Student able to view all the history of previous appointments.
* Supervisor shall able to view all appointment that need to their approval.

1. **Supervision module**

* Committee shall able to assign supervisor for each students that register for workshop1 and workshop 2 courses.
* Admin shall able to assign supervisor for each students that register for workshop1 and workshop 2 courses.
* Admin shall able to assign committee for workshop 1 and workshop 2.
* Supervisor and committee can view information of student such as student detail, proposal, final report and logbook under their supervision.

1. **Report submission module**

* Report submission module is consist of two types of report submission which is Proposal and Final Report.
* Student shall able to upload their reports via the system.
* Supervisor shall able to review and approve/reject submitted document by their student.
* Committee shall able to set the submission due date for proposal and final report in workshop 1 and workshop 2.
* Committee shall able to view all the proposal and final report of student workshop 1 and workshop 2.
* Supervisor shall able to assign mark and approve submitted proposal and final report by their student.
  1. **Project Significance**

Communication and planning is one of the important factors in success of a system development. Communication is a fuel that keeps the system development running smoothly. A failure to communicate is often the greatest threats to the success of information technology projects.

By developing the Student Supervision Management System (e-SSMS), it will benefits all the student and lecturers that involves in the of Workshop 1 and Workshop 2 courses. The e-SSMS shall benefit lecturer by providing a platform to manage and monitor their students. It also benefit the Workshop 1 and Workshop 2 committee to oversee all the progress in the workshop courses.

In other hand, e-SSMS shall greatly benefit the student of workshop 1 and workshop 2 by providing a web based platform that able them to updates their logbook activities, upload proposal and final report in a more systematic manners and also able to schedule an appointment with their supervisor with ease.

* 1. **Expected Output**

**Output 1 :**  Student and lecturer able to monitor and manage their progress of workshop project with more visibility.

**Output 2 :** Student able to schedule their appointment with supervisor more effectively.

**Output 3 :** Committee able to manage and monitor overall process in Workshop 1 and Workshop 2.

* 1. **Conclusion**

In conclusions, the development of Student Supervision Management System (e-SSMS) shall eliminates all the problem and difficulty that lecturers and students is currently having. e-SSMS shall replace the manual tracking and monitioring mechanism in Workshop 1 and Workshop 2 with more systematic control and visible mechanism.

**CHAPTER II**

**PROJECT METHODOLOGY AND PLANNING**

1. **Introduction**

This chapter will cover the details explanation of methodology that is being used to make this project complete and working well. Many methodology or findings from this field mainly generated into journal for others to take advantages and improve as upcoming studies. The method is use to achieve the objective of the project that will accomplish a perfect result. In order to evaluate this project, the methodology based on System Development Life Cycle (SDLC).

1. **Database Development Methodology**

For this system, Agile development methodology is being used as a System Development Life Cycle (SDLC) approach. According to Essentials of Software Engineering Third Edition (2014), agile development methods hold the potential promise to develop software in smaller iterations, guaranteeing there is a finished product at all times, and demanding only normal effort from their developers. Agile methods are extremely good at dealing with change, which means requirements do not need to be completely specified from the beginning. While most of the success stories come from small-to medium-sized software projects, we believe that many parts of the Agile methodologies and processes hold high potential for large projects as well.



**Figure 2.1** Database Methodology on DBLC

**2.2.1 Database Planning**

The database planning includes the activities that allow the stages of the database system development lifecycle to be realized as efficiently and effectively as possible. This phase must be integrated with the overall Information System strategy of the organization. The very first step in database planning is to define the mission statement and objectives for the database system.

**2.2.2 System Definition**

In the systems definition phase, the scope and boundaries of the database application are described. The major user views are also described.

**2.2.3 Requirement Collection and Analysis**

During the requirements collection and analysis phase, the collection and analysis of the information about the part of the enterprise to be served by the database are completed.

**2.2.4 Database Design**

In the conceptual database design phase, the model of the data to be used independent of all physical considerations is to be constructed. The model is based on the requirements specification of the system. In the logical database design phase, the model of the data to be used is based on a specific data model, but independent of a particular database management system is constructed. This is based on the target data model for the database e.g relational data model. In the physical database design phase, the description of the implementation of the database on secondary stage is created. The base relations, indexes, integrity constraints, security, etc. are defined using SQL language.

**2.2.5 Database Management System Selection**

This is an optional phase, when there is a need for a new database management system (DBMS), this phase is done. DBMS means a database system like Access, SQL Server, MySQL, Oracle. In this phase the criteria for the new DBMS are defined. Then several products are evaluated according to the criteria. Finally. The recommendation for the selection is decided.

**2.2.6 Application Design**

In the application design phase, the design of the user interface and the application programs that use and process the database are defined and designed.

**2.2.7 Prototyping**

The purpose of a prototype is to allow the users to use the prototype to identify the features of the system using the computer.

**2.2.8 Implementation**

During the implementation phase, the physical realizations of the database and application designs are to be done. This is the programming phase of the systems development.

**2.2.9 Data Conversion and Loading**

This phase is needed when a new database is replacing an old system. During this phase the existing data will be transferred into the new database.

**2.2.10 Testing**

Before the new system is going to live, it should be thoroughly tested. The goal of testing is to find errors. The goal is not to prove the software is working well.

**2.2.11 Operational Maintenance**

The operational maintenance is the process of monitoring and maintaining the database system. Monitoring means that the performance of the system is observed. If the performance of the system falls below an acceptable level, tuning or reorganization of the database may be required. Maintaining and upgrading the database system means that, when new requirements arise, the new development lifecycle will be done.

1. **Project Schedule and Milestones**

The project schedule, milestone and Gantt chart are shown in Table 2.1 and Figure 2.2 respectively.

|  |  |  |
| --- | --- | --- |
| **Milestones** | **Expected Documents** | **Dates** |
| 1. Problems identification and analysis | 1. Flow chart of the current system  2. Flow chart of the proposed system  3. DFD of the proposed system  4. Requirement of the proposed system  (Functional, non-functional and devices) | 13 February 2017 |
| 2. Identify Objectives, scope and expected outputs | 1. Objective of proposed system  2. Scope of proposed system  3. Expected outputs of the proposed system. | 27 February 2017 |
| 3. Select a suitable database management system | 1. Database system methodology and planning. | 6 March 2017 |
| 4. Analyse the system | 1. Proposed improvement for current system.  2. Functional requirement  3. Non-functional requirement | 13 March 2017 |
| 5. Conceptual design  of the proposed system | 1. A complete ERD of proposed system  2. Logical design of proposed system  3. Physical design of proposed system  4. Graphical user interface design of proposed system | 20 March 2017 |
| 6. Implementation of proposed system | 1. Database setup  2. Database implementation | 27 March 2017 |
| 7. Testing project | 1. Developed system | 24 April 2017 |
| 8. Complete PSM I report | 1. Complete chapter I, II, III and IV | 1 May 2017 |
| 9. Project Demonstration | 1. Demonstrate complete system | 8 May 2017 |
| 10. Submit PSM I report | 1. Complete chapter I, II, III and IV of proposed system | 15 May 2017 |
| 11. PSM I showcase | 1. Completed system | 22 May 2017 |

1. **Conclusion**

As a conclusion, every project will have a different methodology that is being used to make the project successful and working well. Selecting System Development Life Cycle (SDLC) approach could be tricky if it is not suitable. Thus, this SDLC is the suitable for my project as it has to be test out several times.

With appropriate steps and methodology, any process of completing the project can be managed wisely and will be make a good result.

**CHAPTER III**

**ANALYSIS**

1. **Introduction**

In this chapter will discuss the analysis phase and requirement analysis. The analysis process is important step in order to create a good and efficient system. By doing the analysis to the current system and proposal system, we will get some information about the system so that the proposed system can be developed successfully and meet the user requirement.

Firstly, the analysis will discover the problems that are faced within the current system. This is important step because it will give the developer useful information to develop a good system hence overcome the problems faced by current system.

Then, the analysis of new system that will be developing soon will be implementing. This will be done by representing the diagrams which show the flow of the data within new system. Some explanation also will be provided to give brief information about the new system.

1. **Problem Analysis**

**3.2.1 Background of Current System**

Currently in the Faculty of Information and Communication Technology (FTMK) of UTEM, has no system existed to assist student and lecturer to communicate efficiently and hence resulted difficulty in tracking workshop progress, lack of proper record of discussion and difficulty in scheduling appointment with supervisor.

From the student’s perspective, students find it difficult to make an appointment with their supervisor to show their development progress or to seek guidance for the project. Supervisor were unable to entertain all of their student due to their busy schedule. Student also having difficulty to keep track on their systematic log book which is used to record discussion and task given by their supervisor.

Furthermore, as a lecturer. They will be assigned to be the supervisor for several students who taking the Workshop 1 and Workshop 2 courses. Currently, supervisor is having difficulty to keep track filling and progress of the proposal and reports of previous and current students which is done manually.

1. **Requirement Analysis**

Data requirement, functional requirement, non-functional requirement and other requirements will be cover in requirement analysis. Context diagram and data flow diagram of e-SSMS will be included in functional requirement. Software requirement, hardware requirement and network requirement will be state in other requirements.

**3.3.1 Data Requirement**

In this section, it will describe the data that requires to be used in the system. The attributes are the fields in the table. The data requirements of e-SSMS are described in more detail in the tables below. Other table refer to Appendix A.

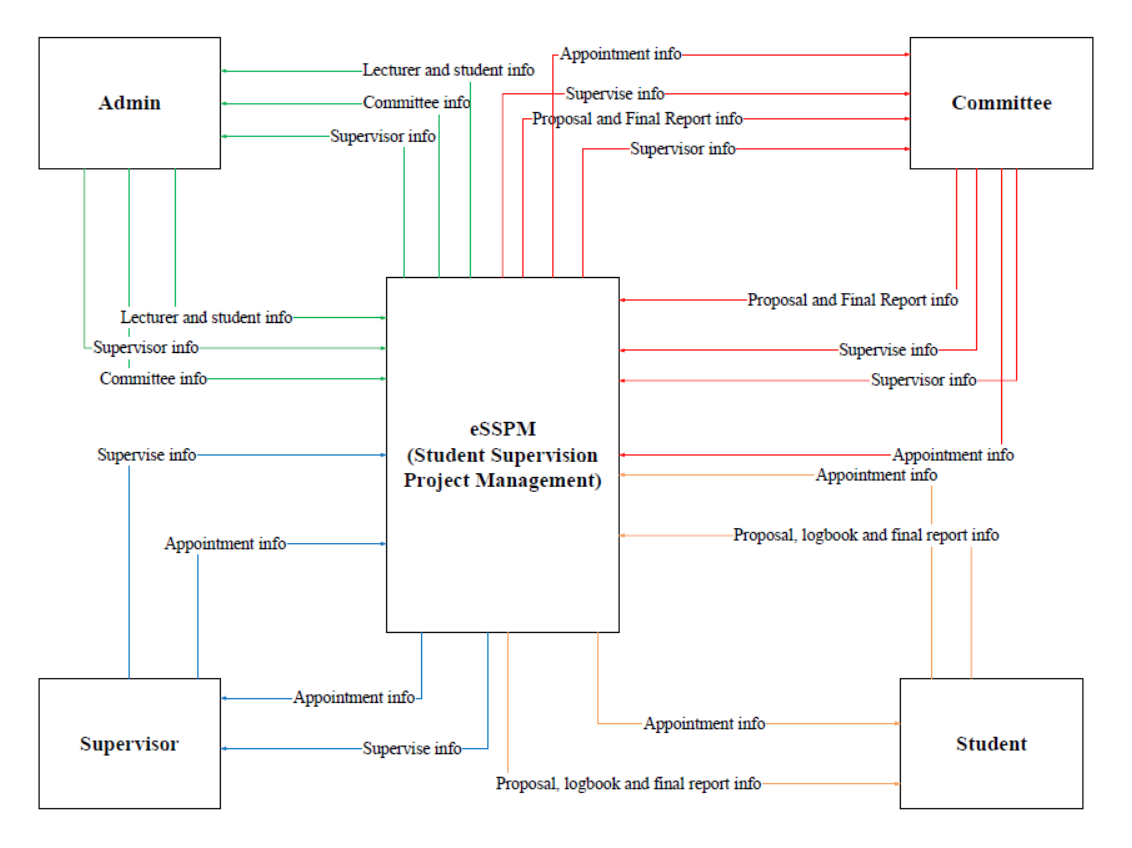
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute name** | **Data type** | **Field size** | **Content** | **Example** |
| Table : position | | | | |
| positionID | int | 10 | Position ID | 1 |
| positionName | varchar | 20 | Position Name | Admin |

**Table 3.1** Position Table

**3.3.2 Functional Requirement**

**3.3.2.1 Context Diagram**

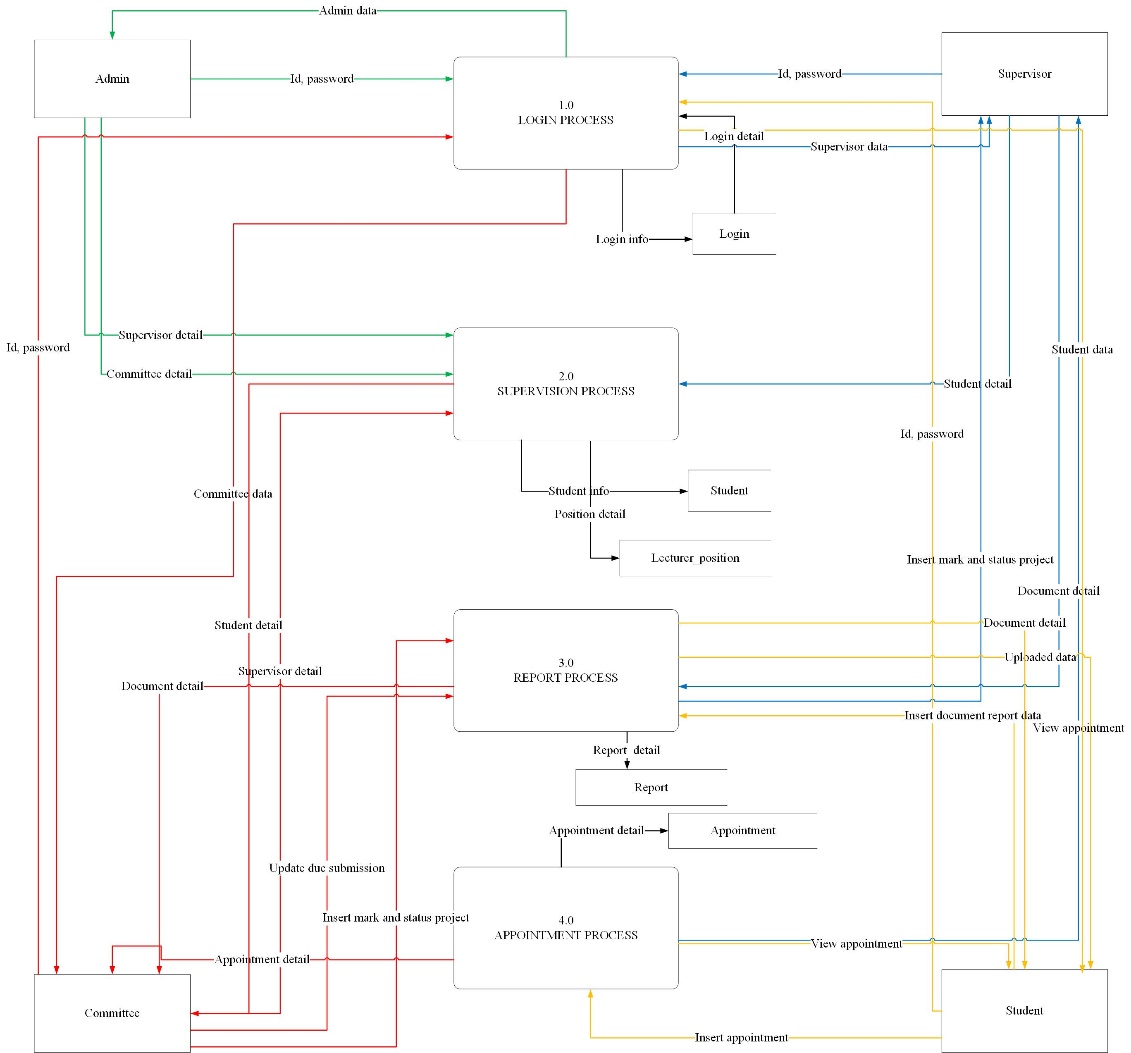
Context Diagram show the interaction between a system and process which the system is designed through interface. Figure 3.1 shows the Context Diagram for the Student Supervision Management System (e-SSMS).



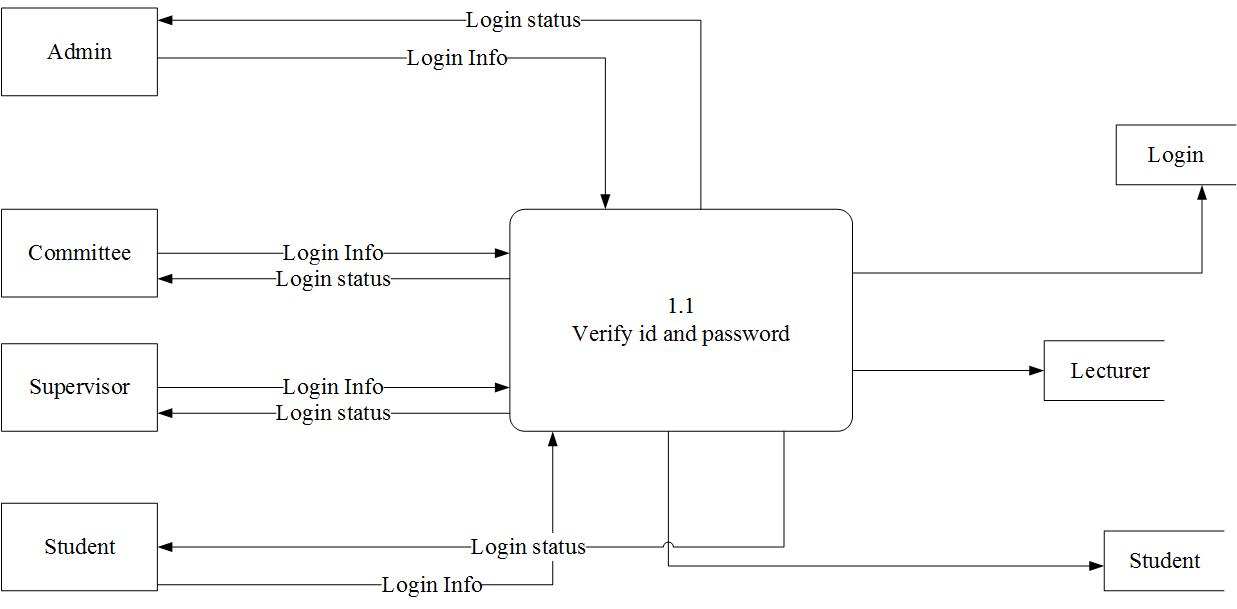
**Figure 3.1** Context Diagram of e-SSMS

**3.3.2.2 Data Flow Diagram (DFD)**

Figure 3.2 shows the DFD level 0 for an e-SSMS. This figure describes the data flow of each module in system.



**Figure 3.2** Data Flow Diagram (DFD) of e-SSMS

 **Figure 3.3** Login process level 1

**Process description/function**

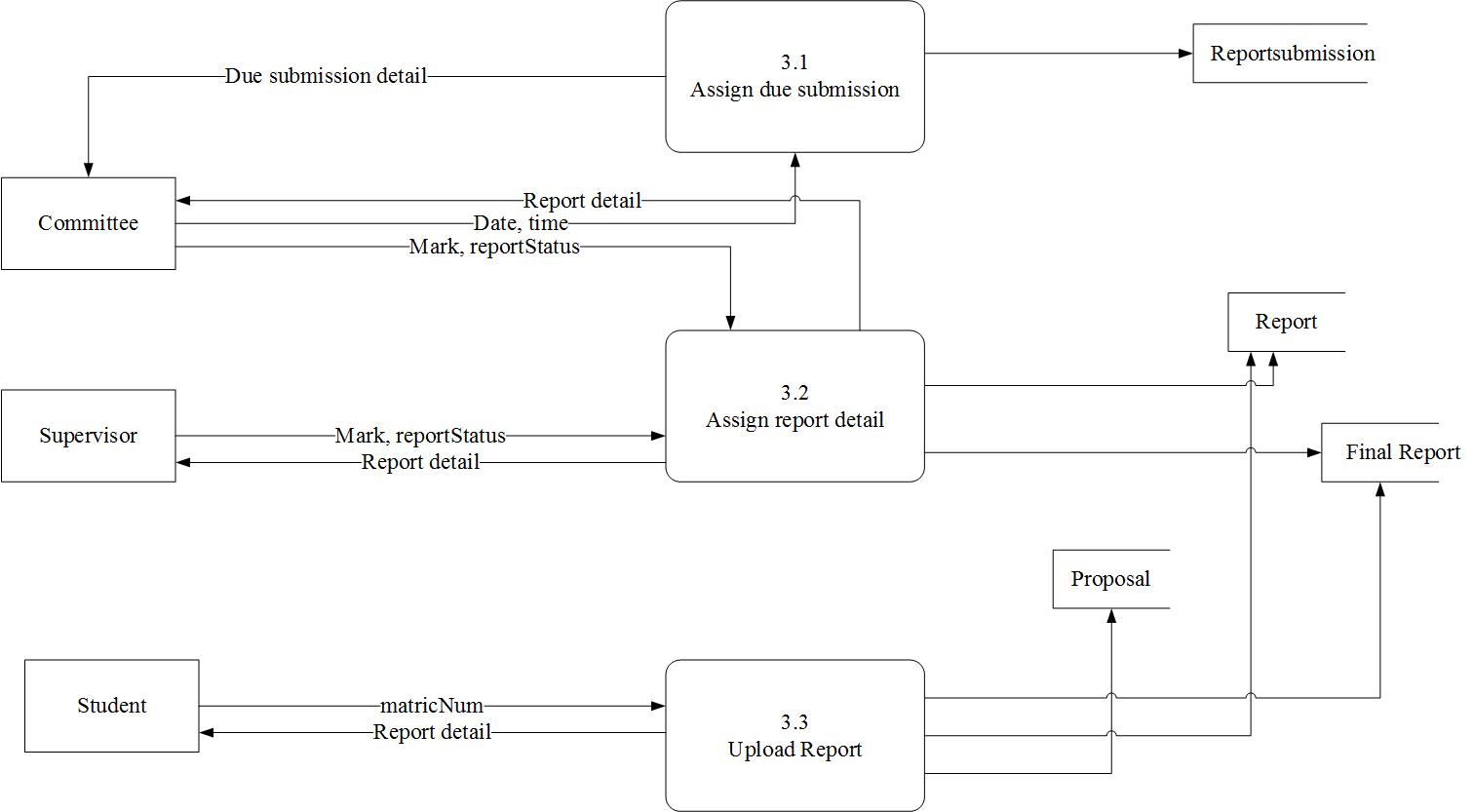
|  |  |
| --- | --- |
| Process | Login |
| Purpose | Process for login of this system |
| Definition | There are 1 sub process involved :-  1. Verify and password  Process involved inserting data of id and password and verified. |
| Responsibility | Admin, committee, supervisor and student |



**Figure 3.4** Supervision process level 1

**Process description/function**

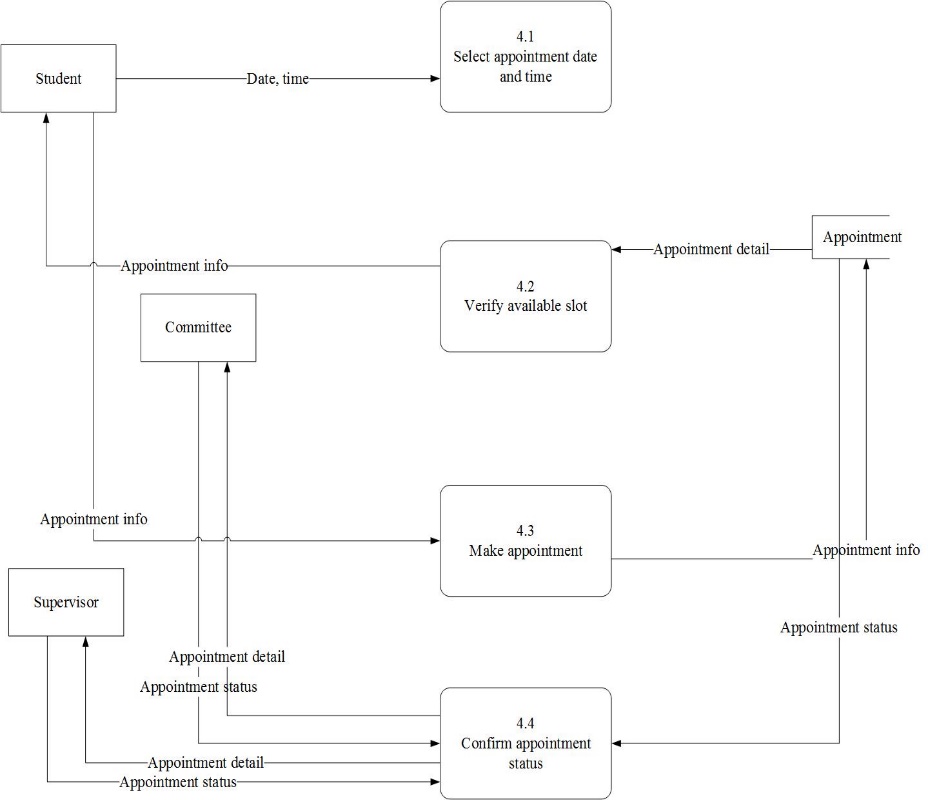
|  |  |
| --- | --- |
| Process | Supervision |
| Purpose | Process for supervision of this system |
| Definition | There are 4 sub process involved :-  1. Search data  - Process searching and view data of staff ID  - Process searching data of student that need to be assign  2. Update data  - Process of assignation lecturer position  - Process of assignation Assign supervisor for student that involve in workshop 1 and workshop 2 |
| Responsibility | Admin and committee |



**Figure 3.5** Report process level 1

**Process description/function**

|  |  |
| --- | --- |
| Process | Report |
| Purpose | Process for report of this system |
| Definition | There are 3 sub process involved :-  1. Insert due submission  Process involved inserting time and date.  2. Update data  Process that need to be assign mark and status of report  3. Insert report  Process involved inserting required report. |
| Responsibility | Committee, supervisor and student |



**Figure 3.6** Appointment process level 1

**Process description/function**

|  |  |
| --- | --- |
| Process | Appointment |
| Purpose | Process for appointment of this system |
| Definition | There are 4 sub process involved :-  1. Choose appointment  Process by selecting appointment date and time.  2. Verify slot  Process that can be view slot of appointment that have been made.  3. Make appointment  Process inserting requirement appointment.  4. Update data  Process update status of appointment. |
| Responsibility | Committee, supervisor and student |

**3.3.3** **Functional Requirement (Process Model)**

* 1. Can make an appointment.
  2. Can keep track all the report submitted by student.
  3. Can add, update certain things to make it easier.
  4. Can assign mark using rubric form application.
  5. Can search easier using radio button.

**3.3.4 Non-functional Requirement**

* 1. This system can functional using web-browser.

**3.3.5 Other Requirement**

**3.3.5.1 Software Requirement**

|  |  |  |  |
| --- | --- | --- | --- |
| **NFR No.** | **Type** | **Software** | **Description** |
| SR\_01 | Platform/Operating System | 1) Microsoft Windows | Operating system as a platform where the system will be uses. |
| SR\_02 | Database | 2) MySQL | MySQL databases are available in all major programming languages with language-specific API. |
| SR\_03 | Web Server Extension | 3) Apache | The Apache HTTP Server is a web server which is can support programming language originally of PHP and MySQL database. |
| SR\_04 | Client-Side Technology | 4) PHP | PHP is a reflective programming language originally designed for producing dynamic web pages. |
| SR\_05 | Web Browser | 5) Google Chrome | Used as web browser to use the website. |

**Table 3.2** Description of Software Requirements.

**3.3.5.2 Hardware Requirement**

|  |  |  |  |
| --- | --- | --- | --- |
| **NFR No.** | **Devices** | **Hardware** | **Description** |
| HR\_01 |  | Laptop | To be able to view the web based system. |
| HR\_02 |  | Printer | The printer is used to print documents, and reports. |
| HR\_03 | Memory | 512 | Need to make the system operate easily and smoothly. |
| HR\_04 | Hard Disk | 5gb | Need to large storage to use this e-learning. |

**Table 3.3** Description of Hardware Requirement.

**3.3.5.3 Network Requirement**

|  |  |  |
| --- | --- | --- |
| **NFR No.** | **Requirement** | **Description** |
| NR\_01 | Network Cable RJ-45 | As connector to cable |

**Table 3.4** Description of Network Requirement.

1. **Conclusion**

In conclusion, this chapter explains how is the current system operate and what method used in order to make sure the future system will overcome the problem occur. Besides that, in this chapter explains briefly about how the data flow for each of the process including requirement needed such as functional, non-functional and others requirement.

As for the next chapter which design, this chapter will explain briefly about the proposed design for the system.

**CHAPTER IV**

**DESIGN**

**4.1 Introduction**

Design is the most crucial phase of system development. These include conceptual design, logical design and physical design. Conceptual design will describe the relationships of Entity Relationship Diagram (ERD) in form of business rules. As for the logical design, data dictionary will be included and validate the conceptual design. Whereas for the physical design will describe briefly on the selection of DBMS.

This chapter is divided into three (3) designs which are system architecture design, database design and graphical user interface. Each of this design process is important in order to develop a system according to the user requirements. It will help in develop an interactive system.

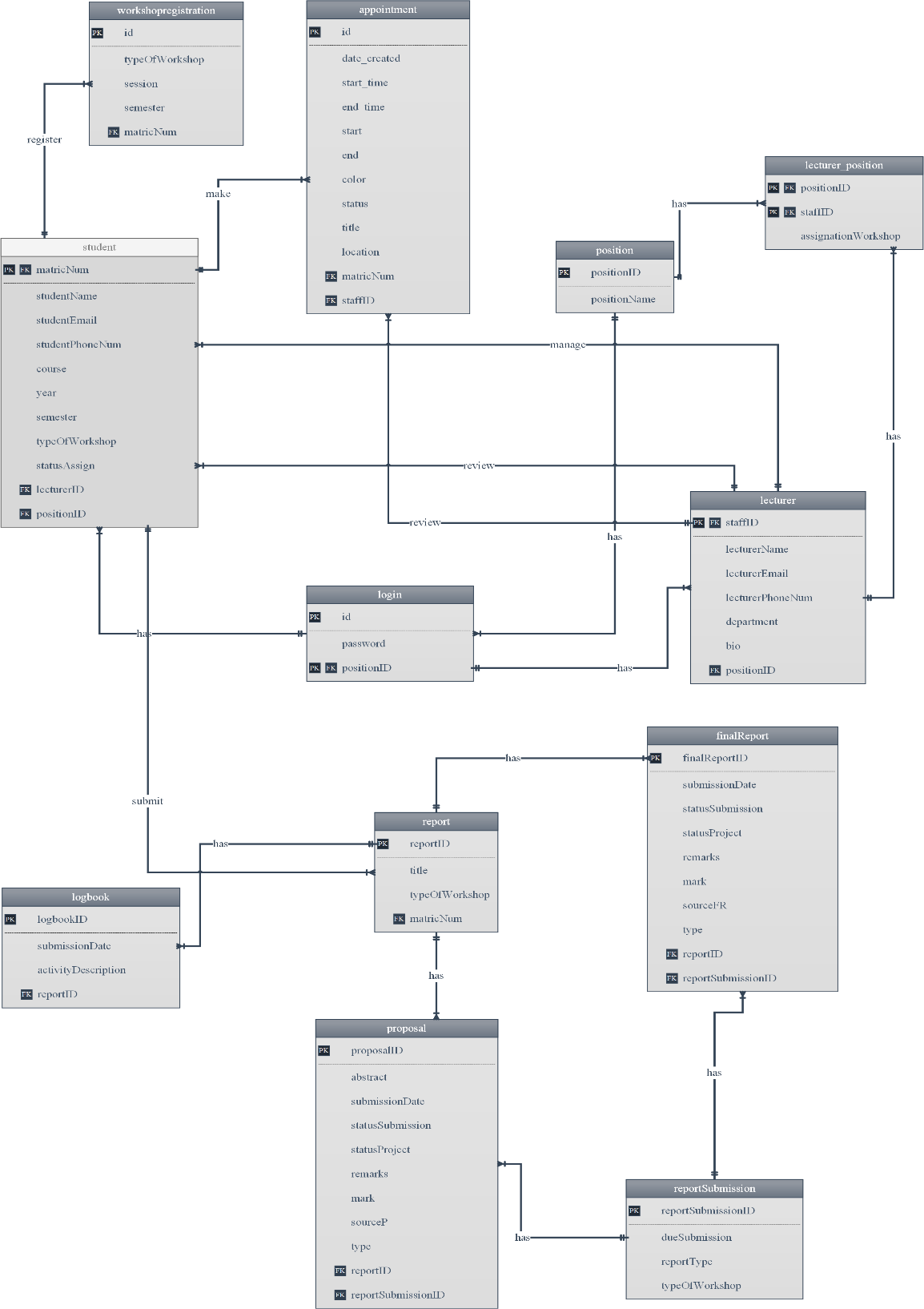
**4.2 Database design**

System architecture is the models that describe the structure of the system. Conceptual model, logical and physical model are the types of models. Entity Relationship Diagram is the example of conceptual model. Whereas, data dictionary is an example of logical model. Last but not least, coding and the interface is an example of physical model.

The architecture of the system will help in understanding the function of the system. The developer of a development system has to make user that the system architecture meets user requirement and provide an interactive interface for the system.

**4.2.1 Conceptual design**

The conceptual design will explain on how the system should work based on the requirements. It is important in order for the system to meet the user and performance requirements. Figure 4.1 is the Entity Relationship Diagram (ERD) for e-SSMS.



**Figure 4.1 e-SSMS Entity Relationship Diagram (ERD)**

**Business rule**

1. A student can register one or many workshop registration but only one

workshop registration registered by the student.

1. A student can make many appointment.
2. A lecturer can review many appointment.
3. A student can submit many report but one report can submit by one student.
4. A report can has many proposal to be submitted but one proposal can only has one report.
5. A report can has many final report to be submitted but one final report can only has one report.
6. A report can has many logbook to be submitted but one logbook can only has one report.
7. A due submission can be assign to many final report but one final report can be assign with one due submission.
8. A due submission can be assign to many proposal but one final report can be assign with one due submission.

**4.2.2 Logical design**

In this section, it will describe that data dictionary for the system. The data dictionaries of e-SSMS are described in more detail in the table below. Other tables refer to Appendix B.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attribute name** | **Content** | **Data type** | **Format** | **Required** | **PK/FK** | **Example** |
| Table : position | | | | | | |
| positionID | Position ID | int (10) | 999999 | Y | PK | 1 |
| positionName | Position Name | varchar (20) | Xxxxxx |  |  | Admin |

**Table 4.1** Position Data Dictionary

**4.2.3 Physical design**

The physical design is about the selection of Database Management System (DBMS). The DBMS choose for this system is MySQL. This DBMS is supported by the phpmyadmin. This MySQL has a graphical user interface which easy for developer to perform database tasks.

The DBMS used is to design a coding for database task as the planning system. The coding construct on the DBMS such as simple database coding (insert), trigger and stored procedure. The coding built in MySQL can show either have any error or problem at the interface. It also can show the success coding built by showing the output at the interface.

The function of stored procedure is to manage and maintain data easily. It has productivity and ease to use while it has a powerful database application. Stored procedure coding's built in phpmyadmin is connected with the selected language that will make the language easy to gain data from the database by using a simple query.

The trigger functions almost same as stored procedure but it will run as a unit. Moreover, trigger also function to atomicity generate derived column values.

**4.3 Graphical User Interface (GUI) Design**

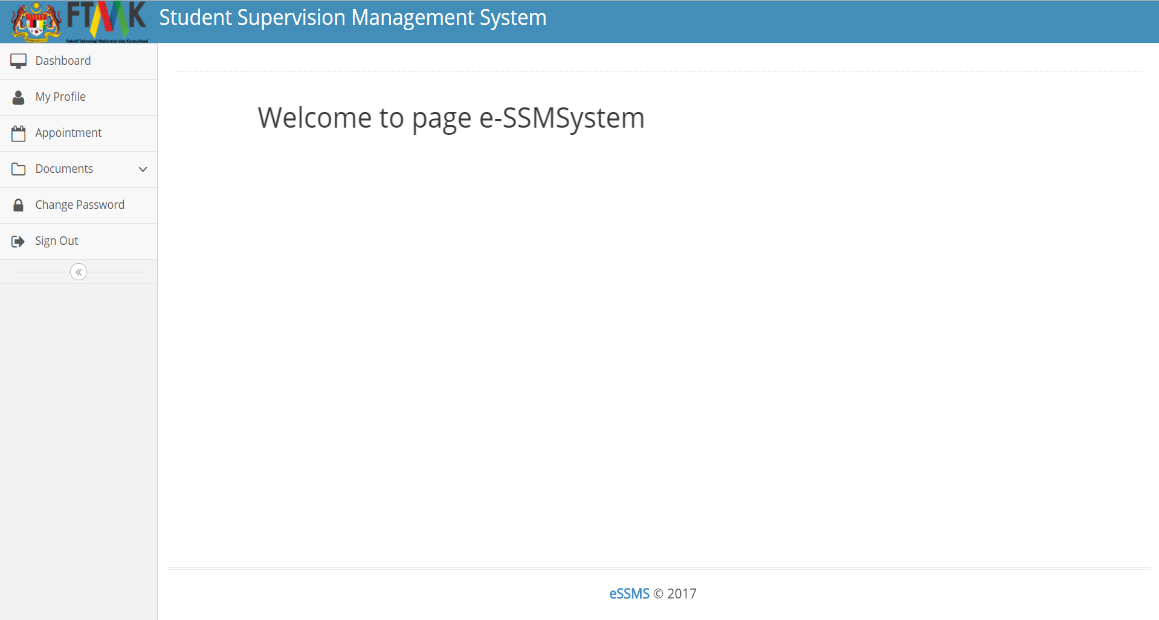
This module that will be used in the interface will explain detail in this section. Some of the function includes in the system are login and other will be explain.

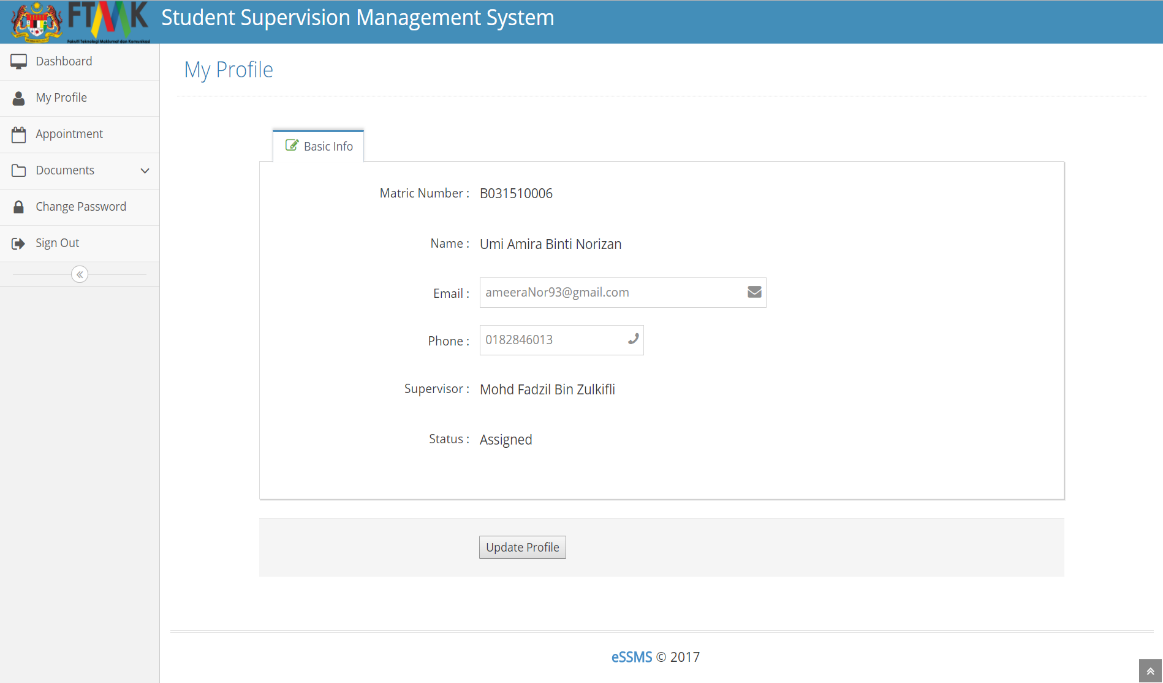


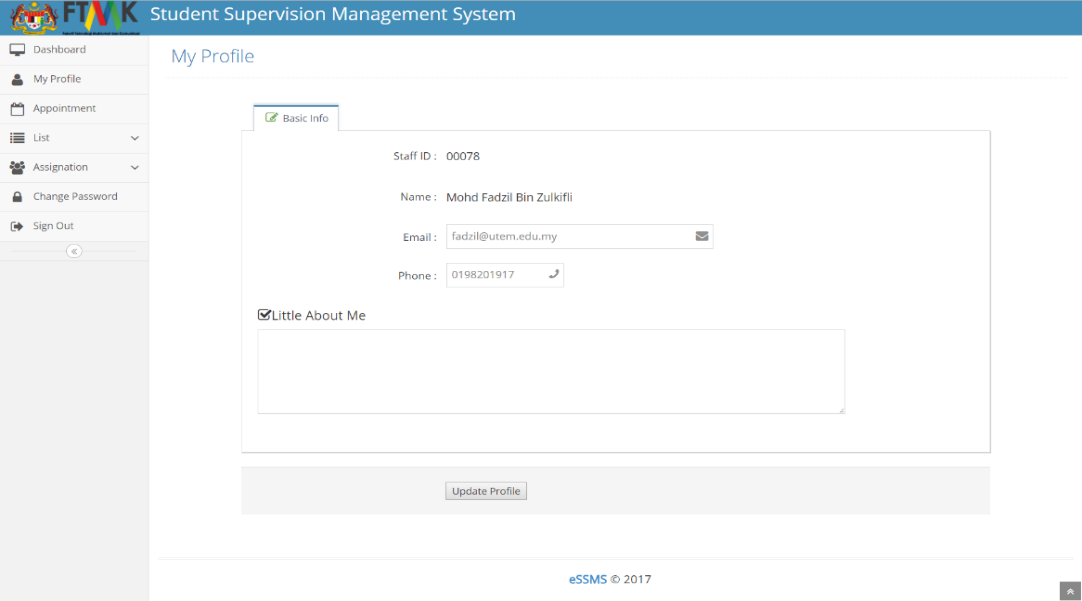
**Figure 4.3.1** Login interface



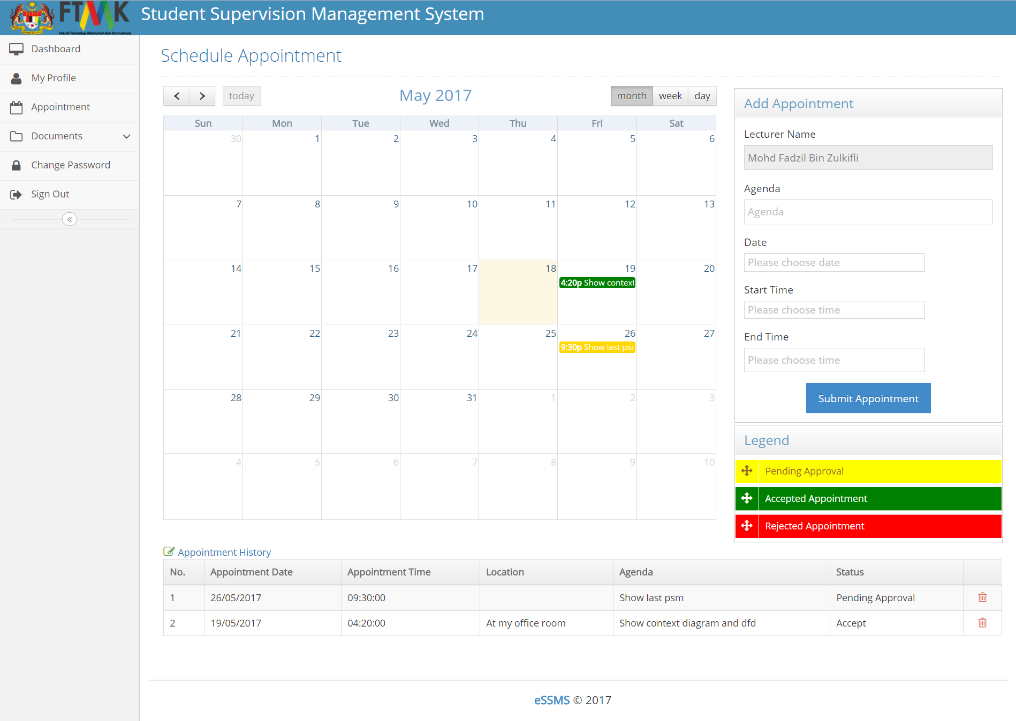
**Figure 4.3.2** Login interface (Admin)

**Figure 4.3.3** Main page interface

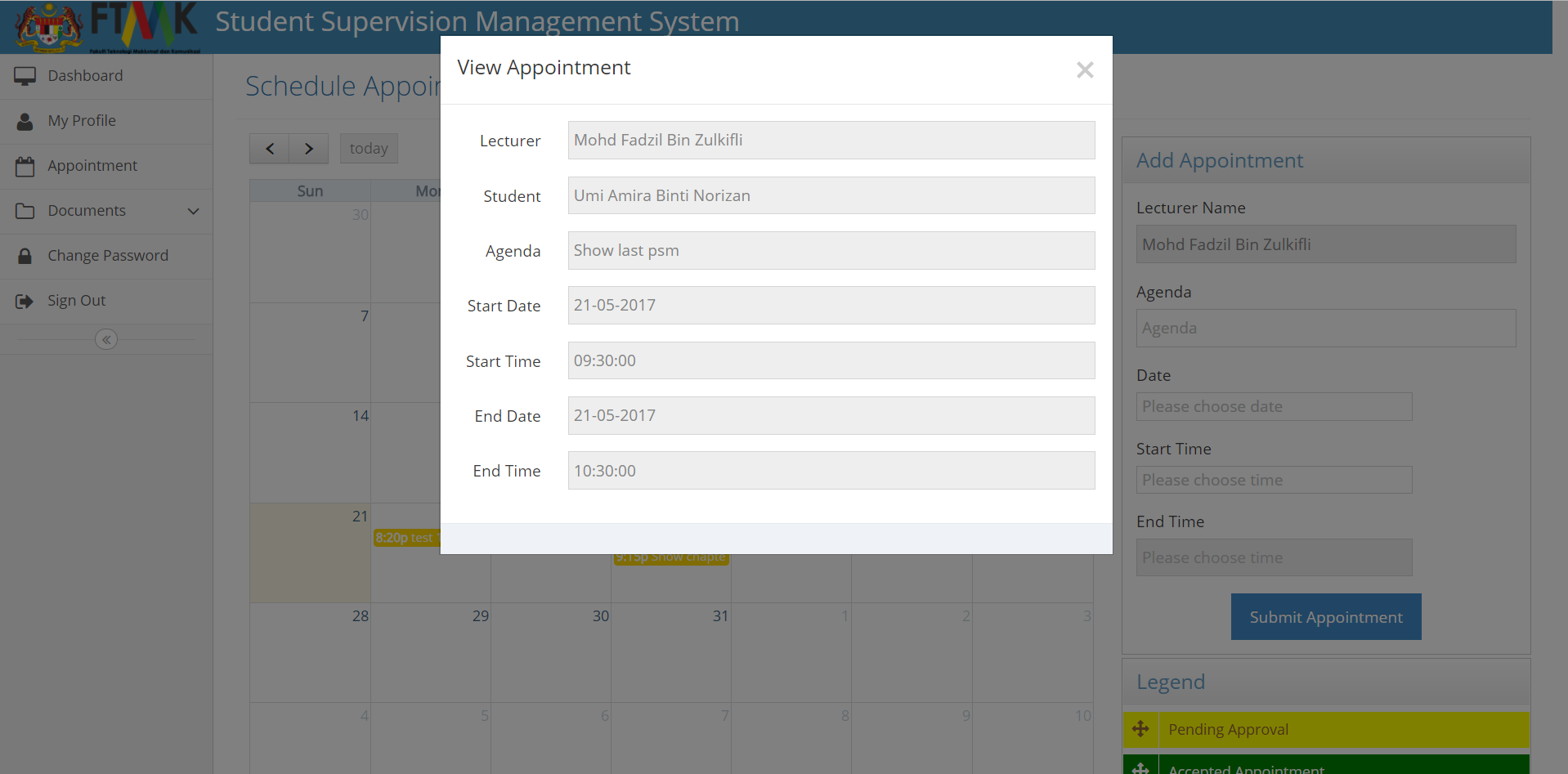
**Figure 4.3.4** Edit profile interface (Student)



**Figure 4.3.5** Edit profile interface (Committee and Supervisor)



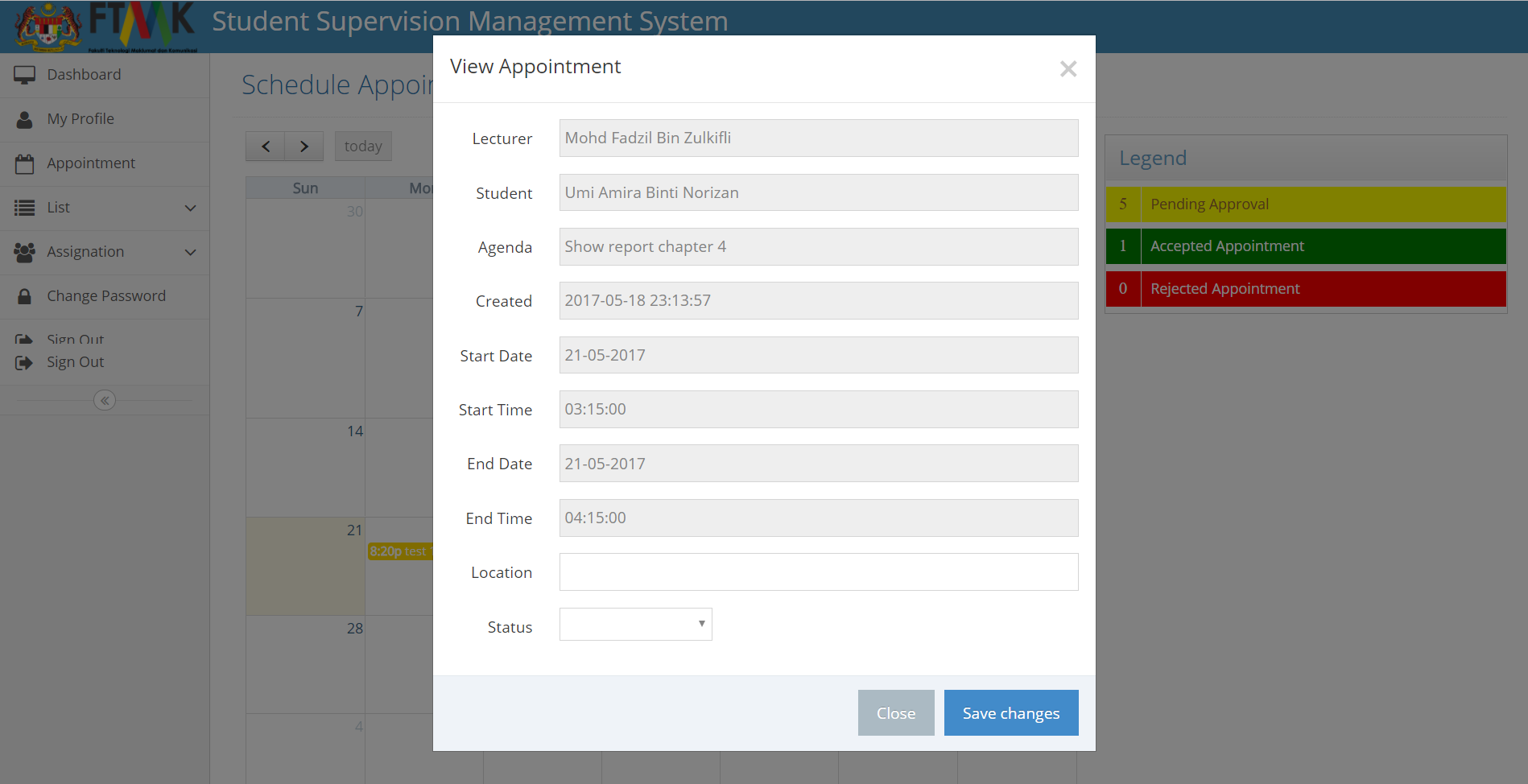
**Figure 4.3.6** Make appointment interface (Student)



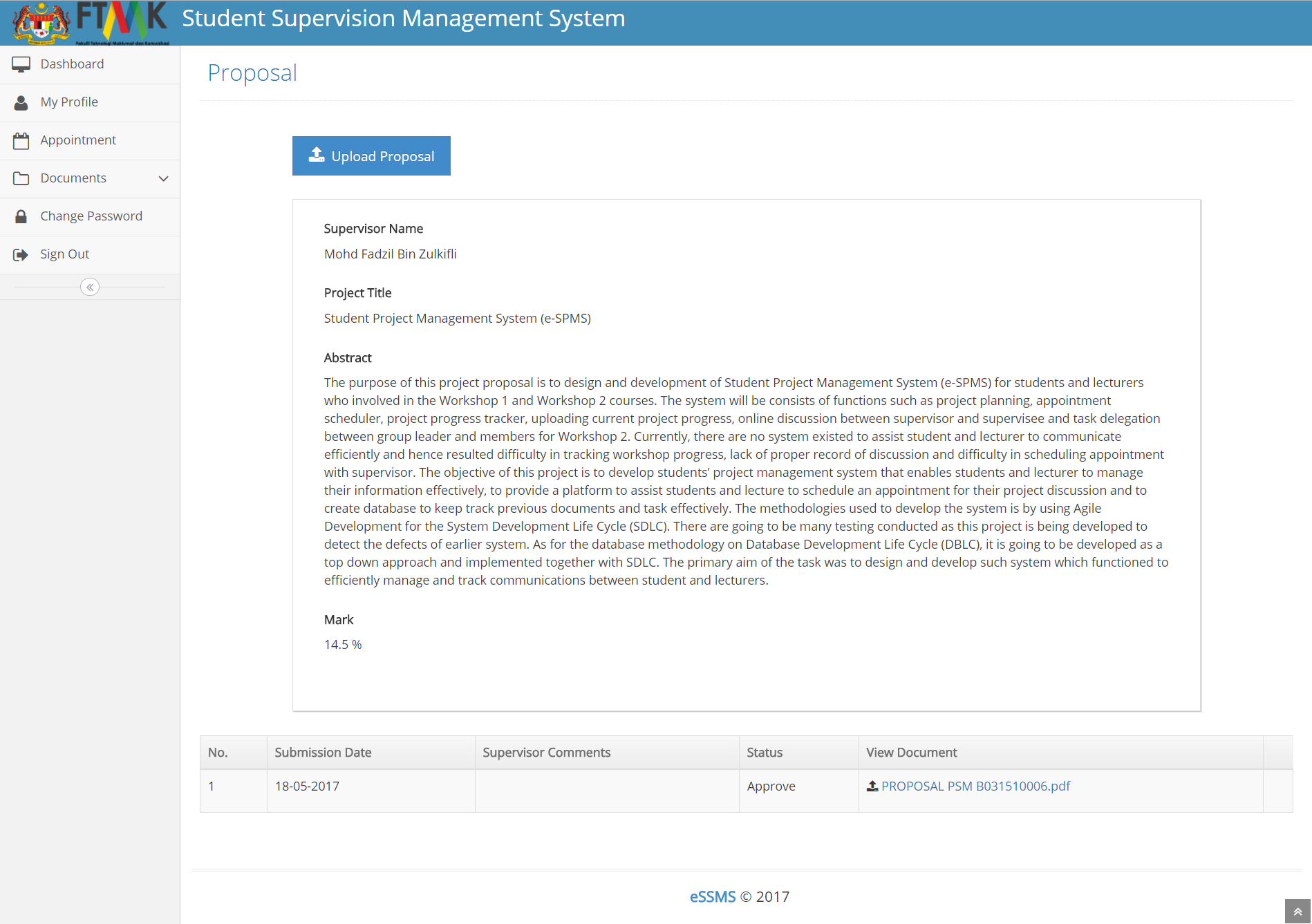
**Figure 4.3.7** View Appointment interface (Student)



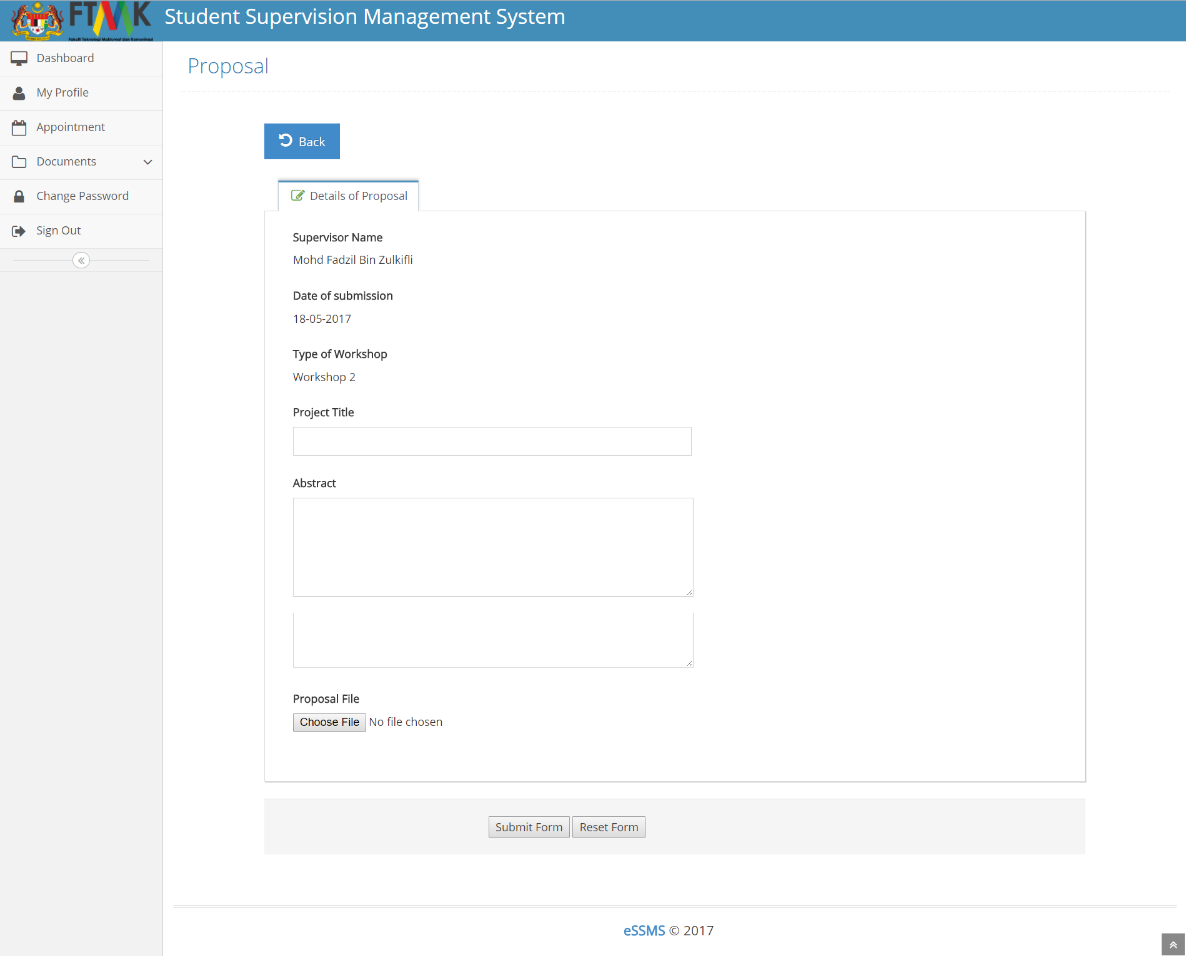
**Figure 4.3.8** View appointment that need to be approve interface (Supervisor/Committee)



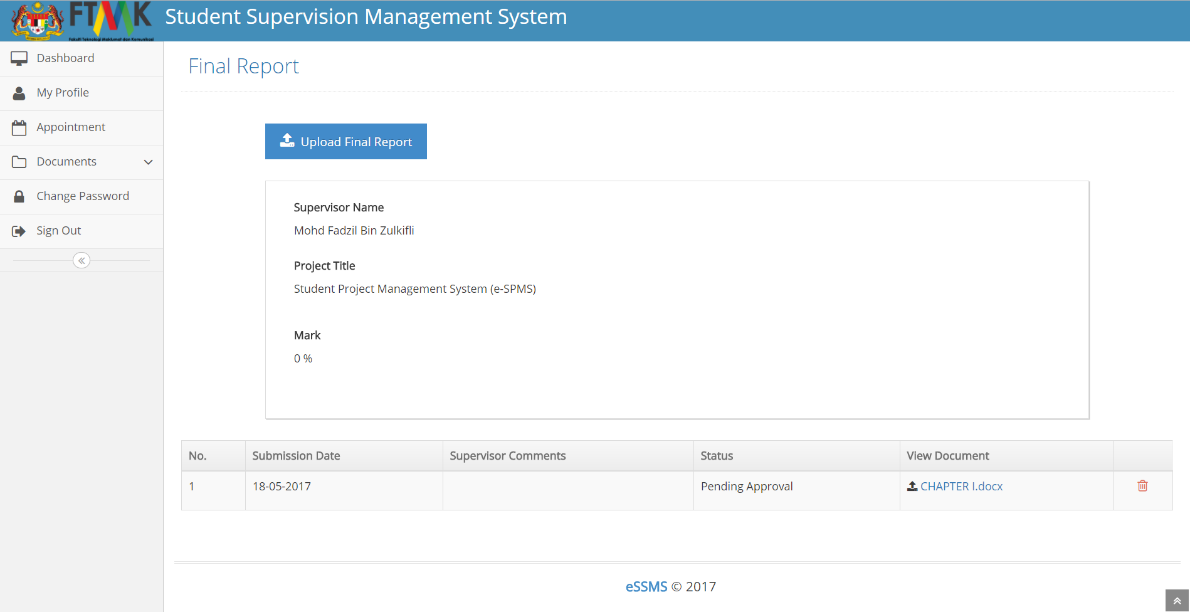
**Figure 4.3.9** Appointment form interface that need to approve (Supervisor/Committee)



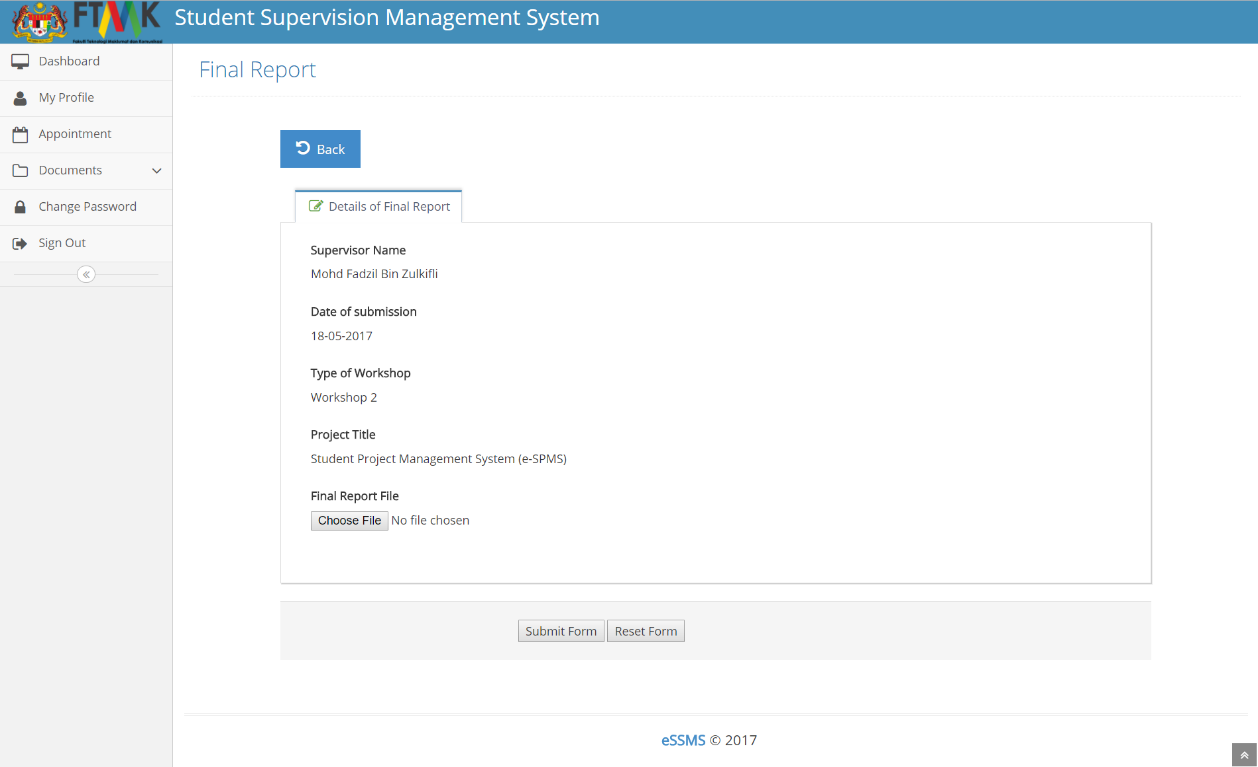
**Figure 4.3.10** View proposal interface



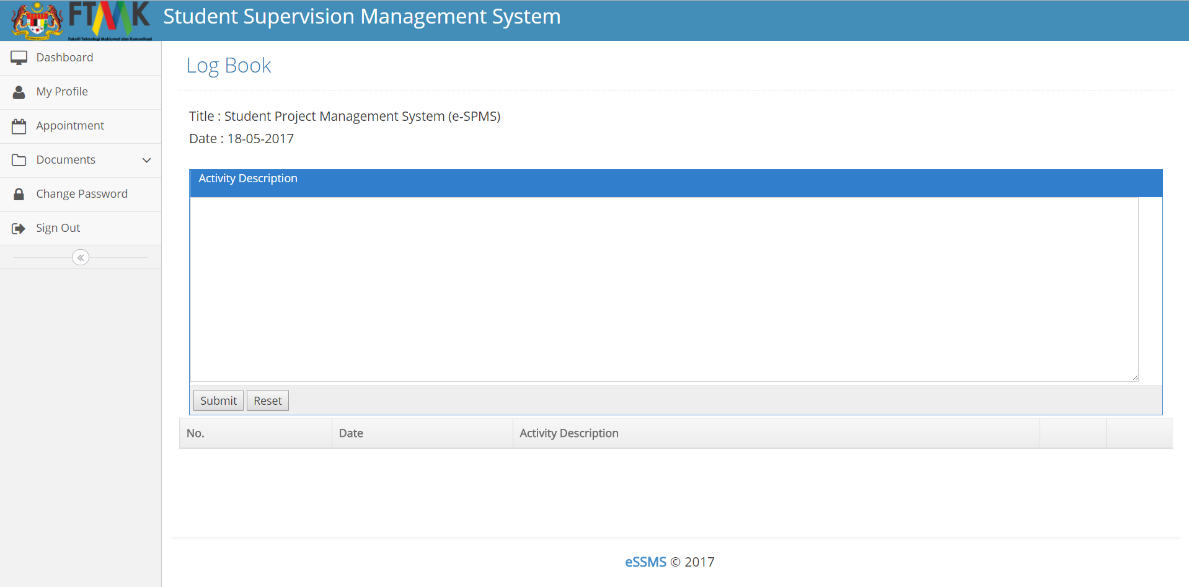
**Figure 4.3.11** Upload proposal interface



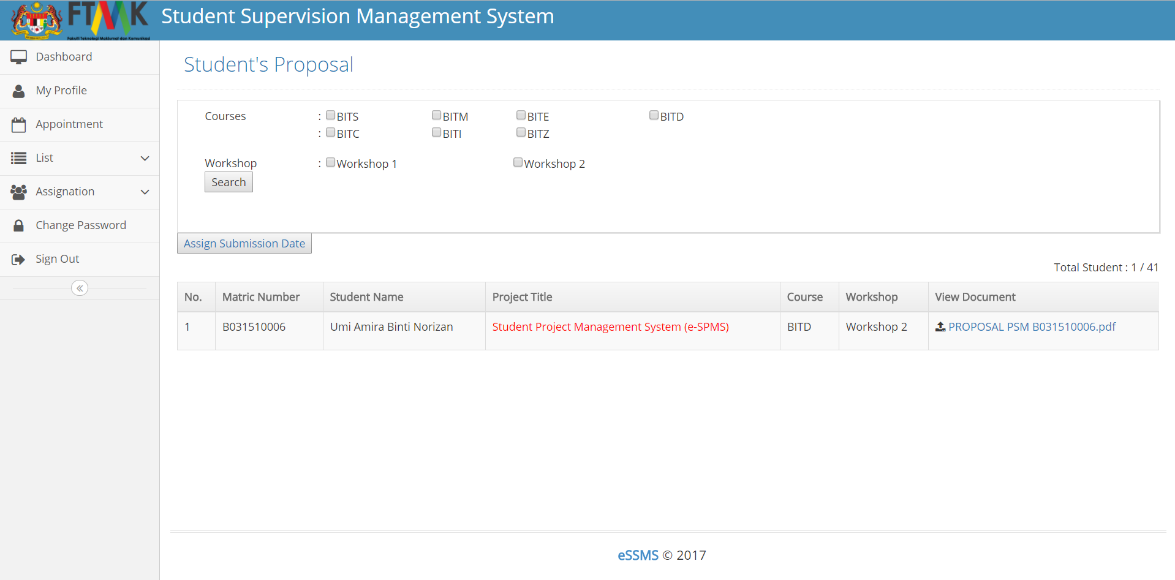
**Figure 4.3.12** View Final report interface



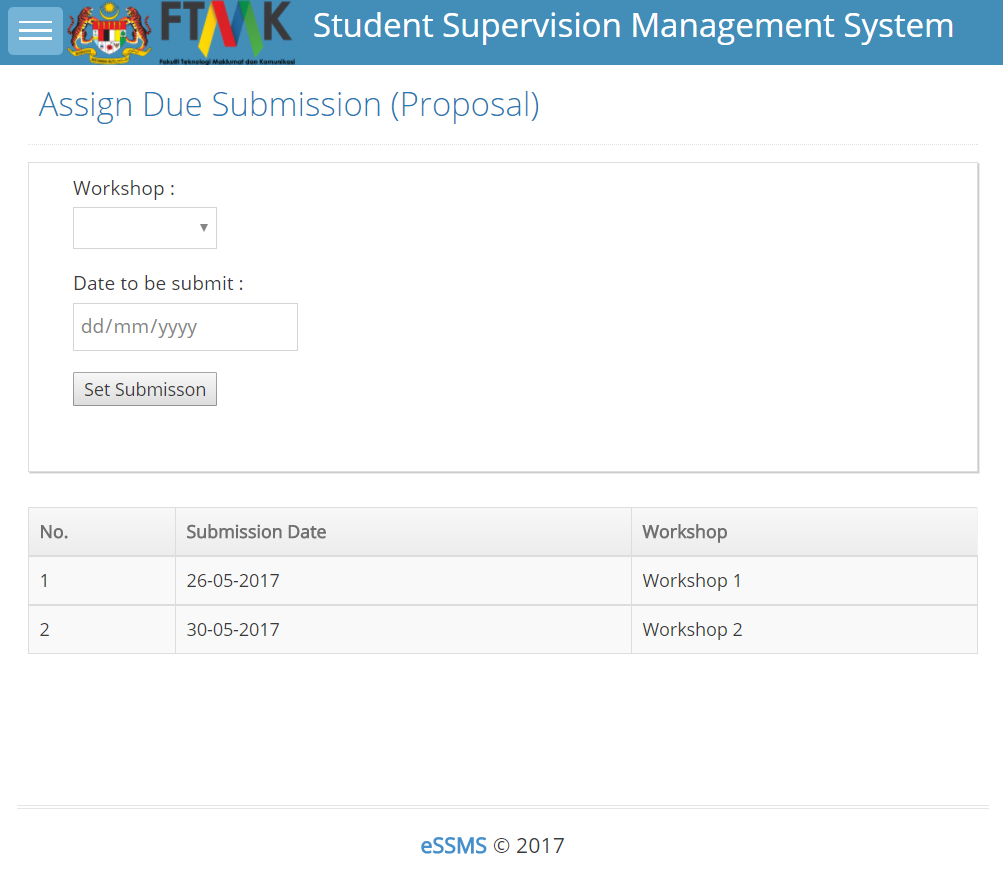
**Figure 4.3.13** Upload Final report interface



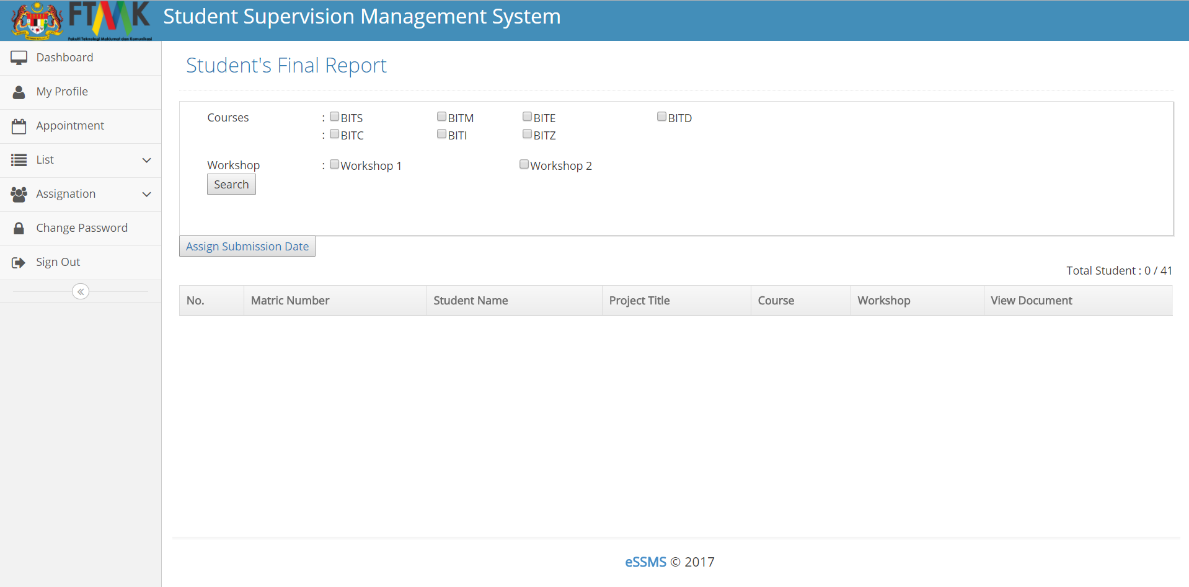
**Figure 4.3.14** Submit logbook interface



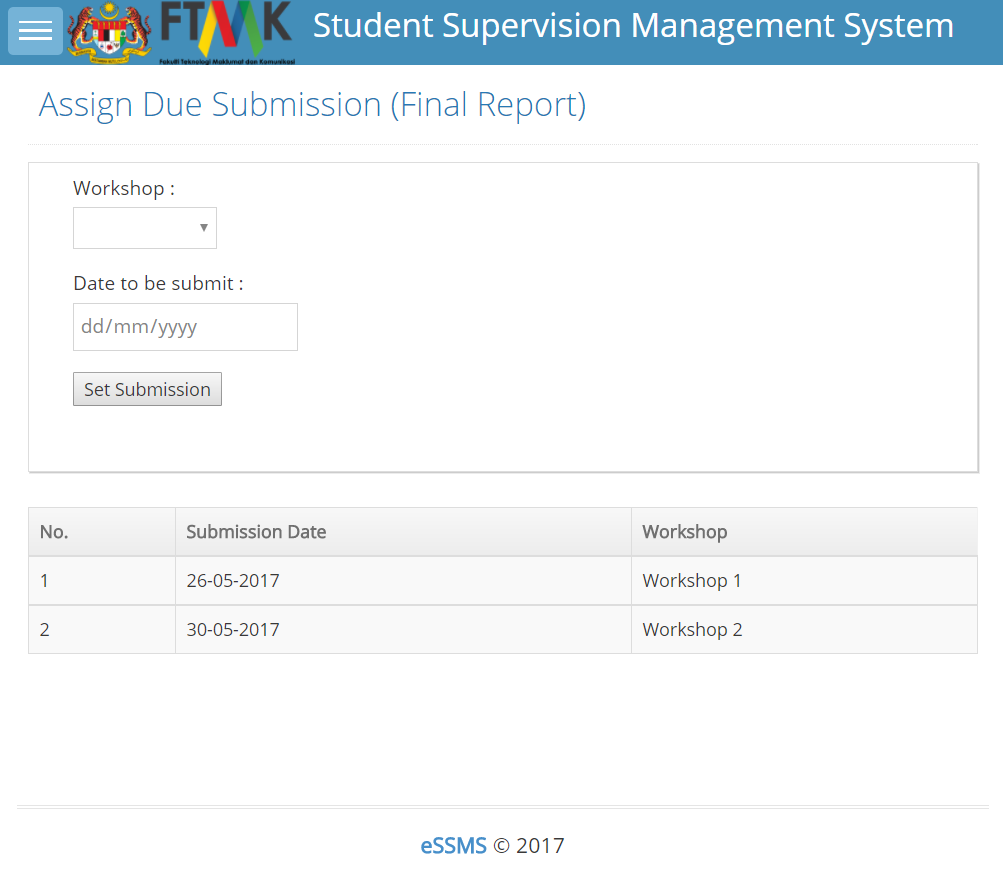
**Figure 4.3.15** List of student’s proposal interface



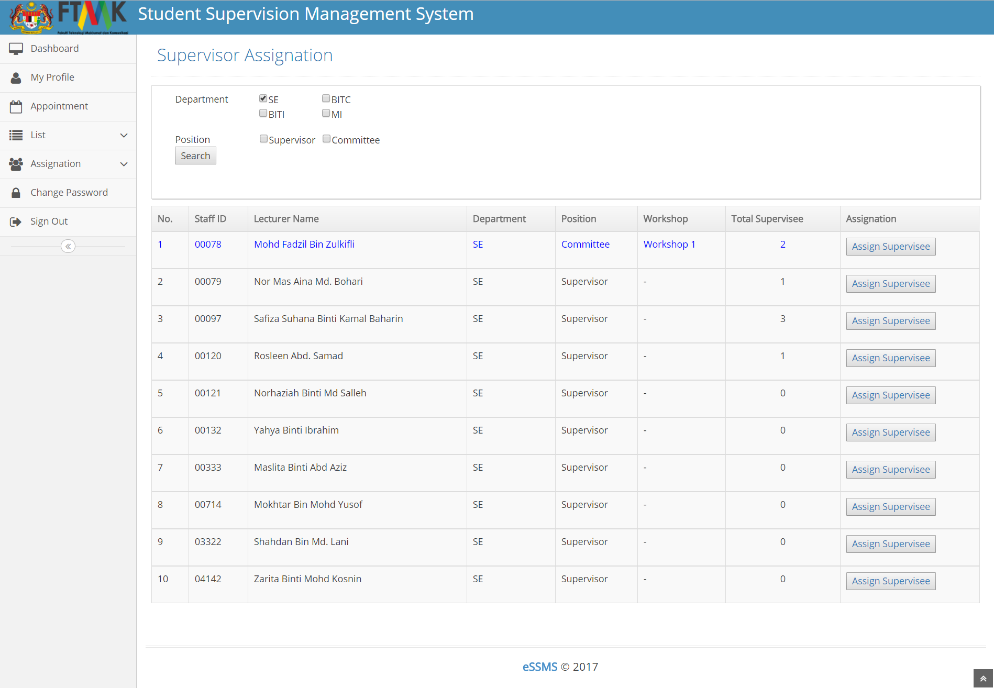
**Figure 4.3.16** Assign due submission (proposal) interface



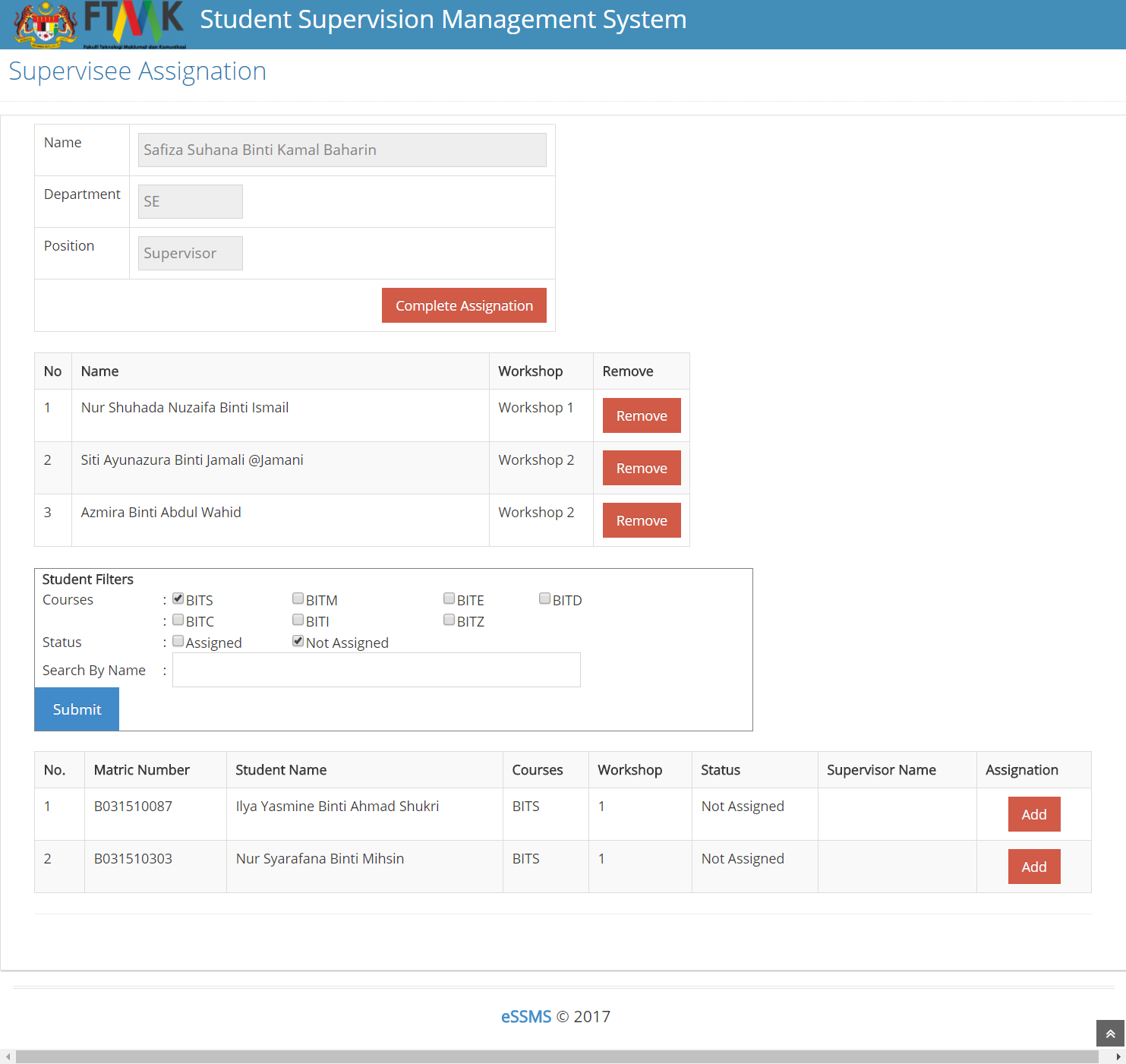
**Figure 4.3.17** List of student’s final report interface



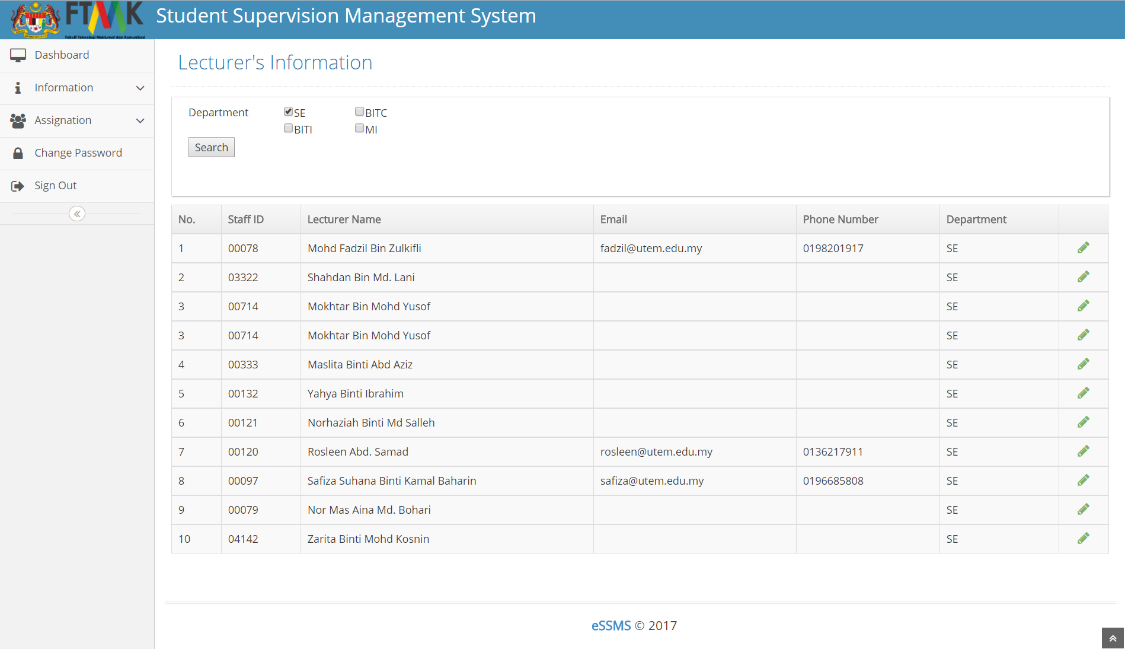
**Figure 4.3.18** Assign due submission (final report) interface



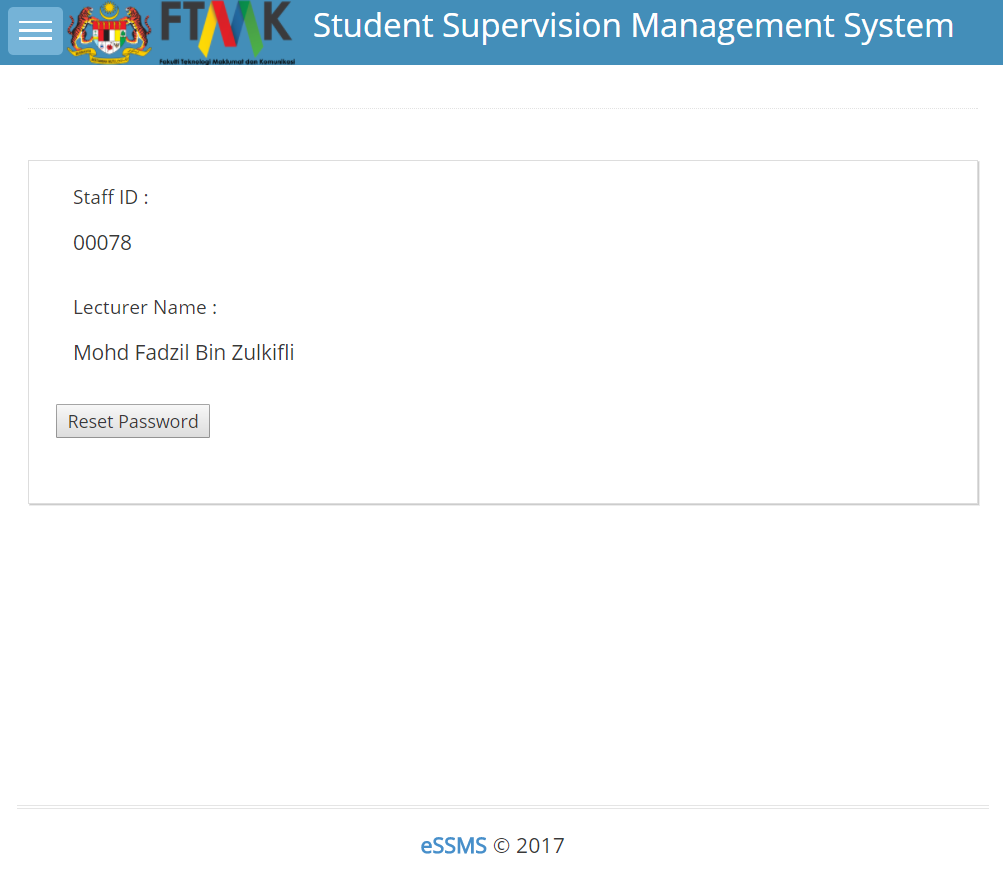
**Figure 4.3.19** List of supervisor to be supervise interface



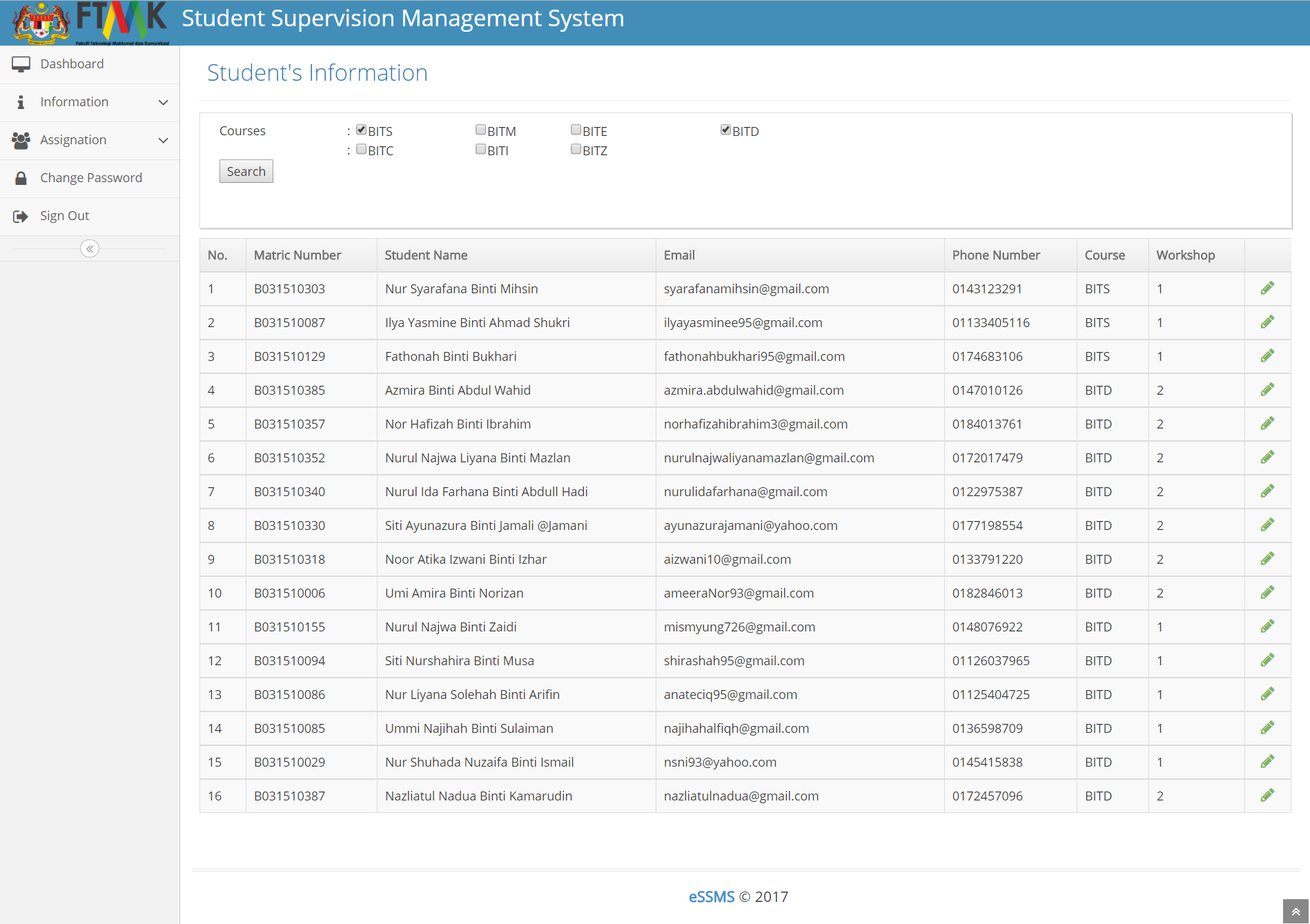
**Figure 4.3.20** Assignation supervisor interface



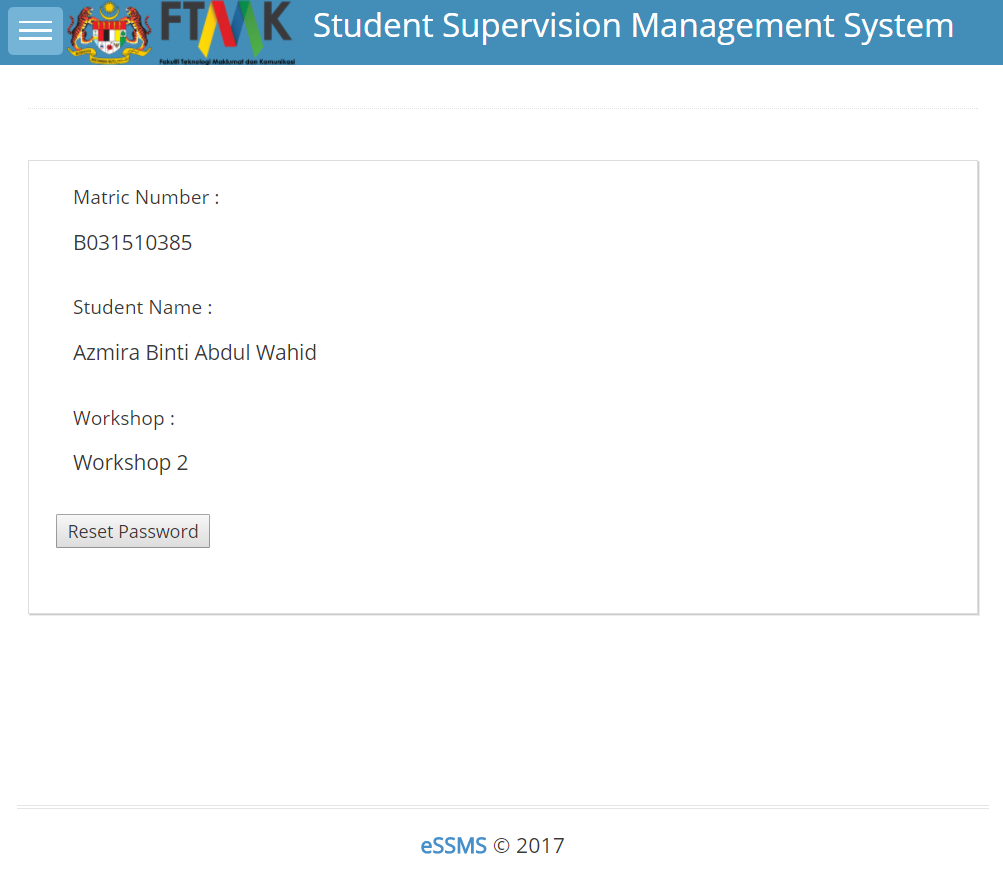
**Figure 4.3.21** Lecturer information interface



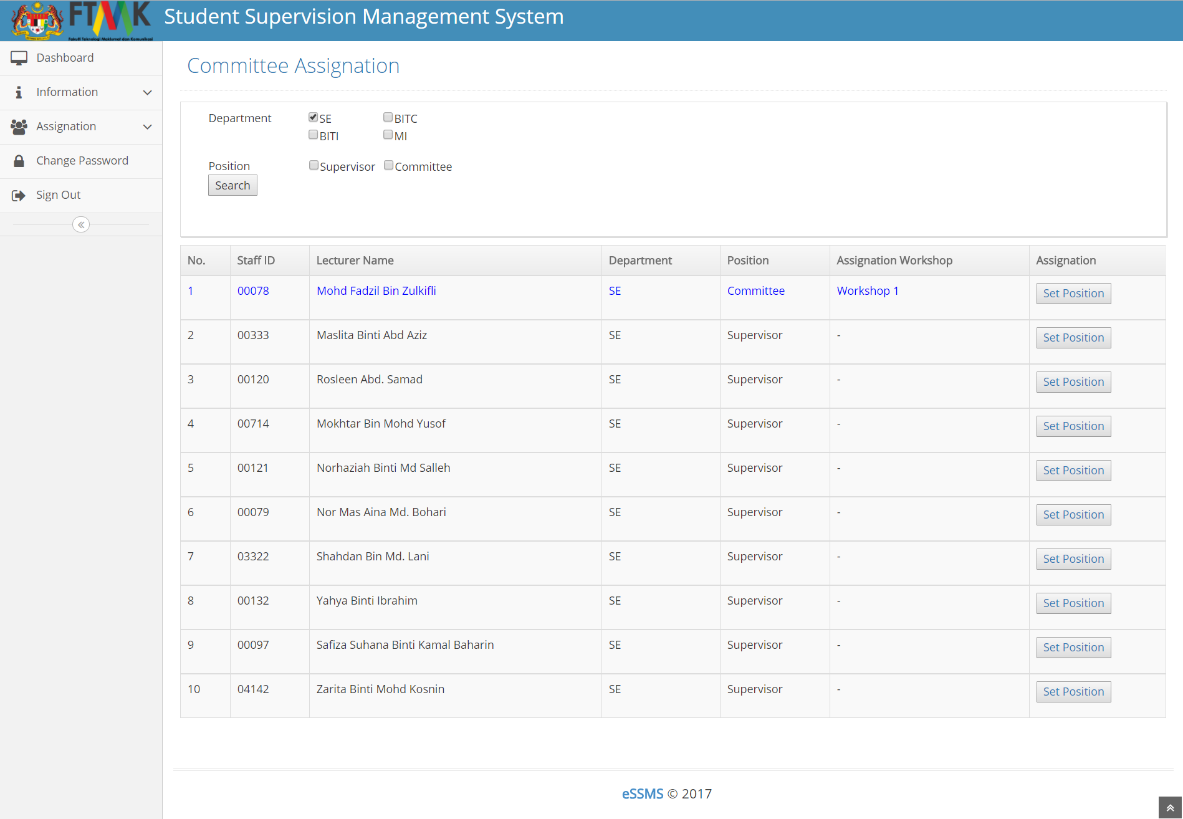
**Figure 4.3.22** Reset password lecturer interface



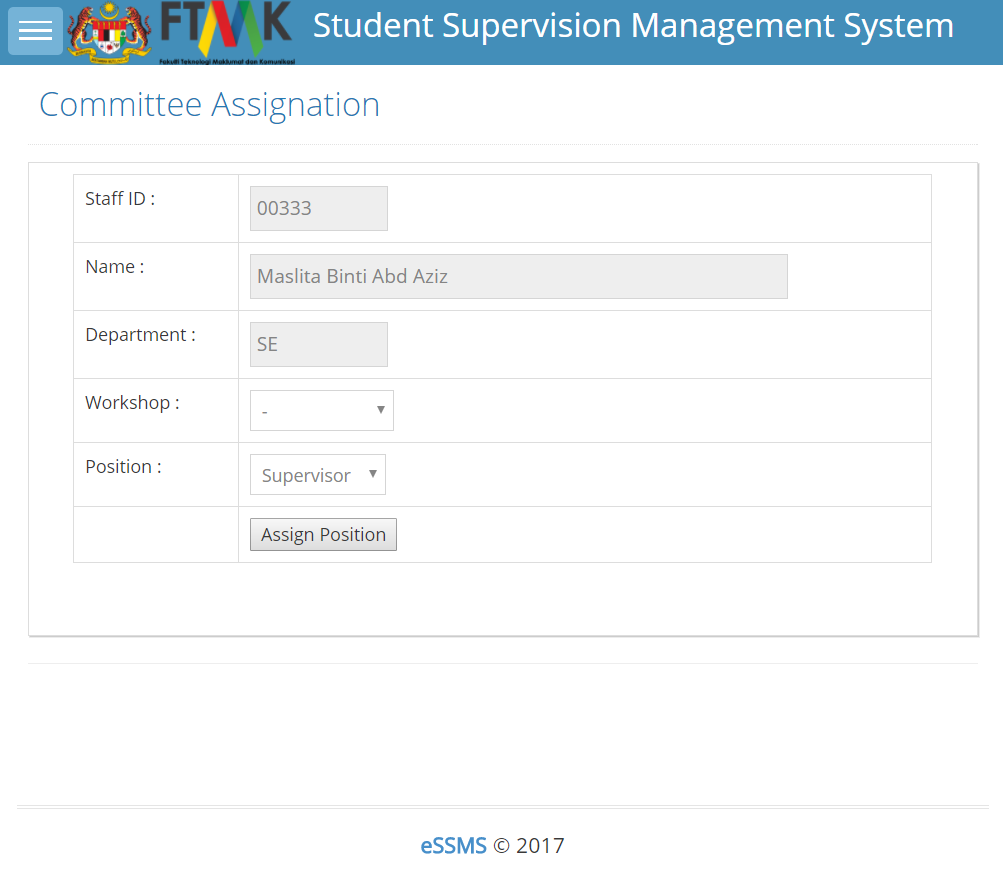
**Figure 4.3.23** Student information interface



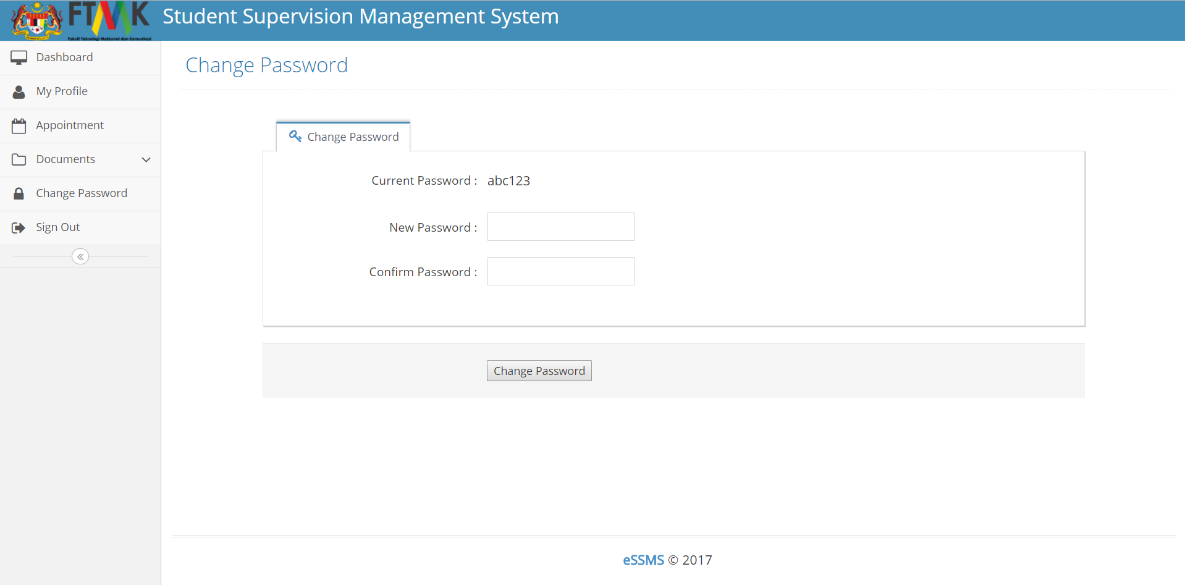
**Figure 4.3.24** Reset password student interface



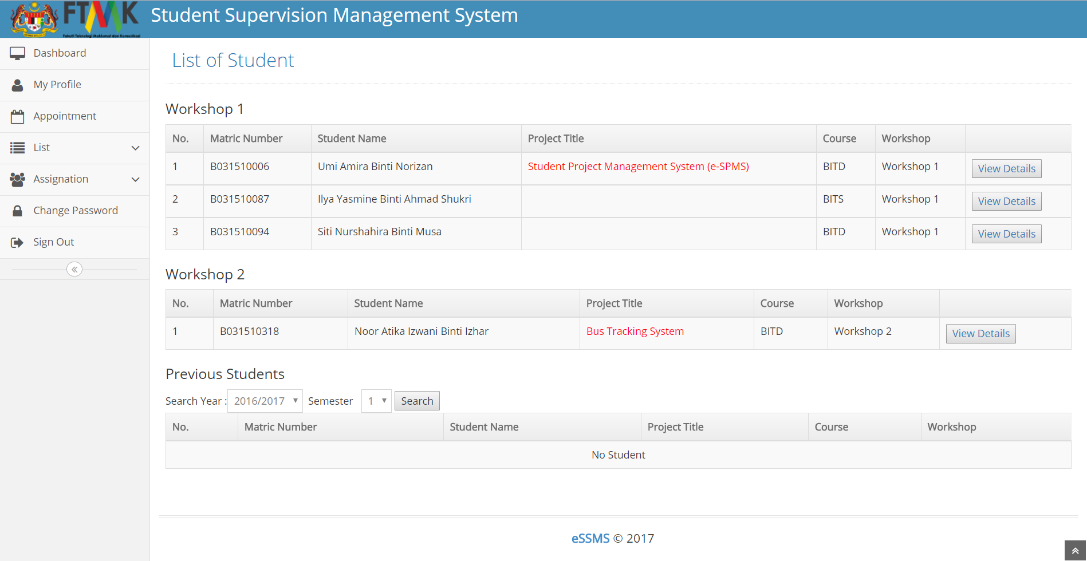
**Figure 4.3.25** List of supervisor to be change position interface



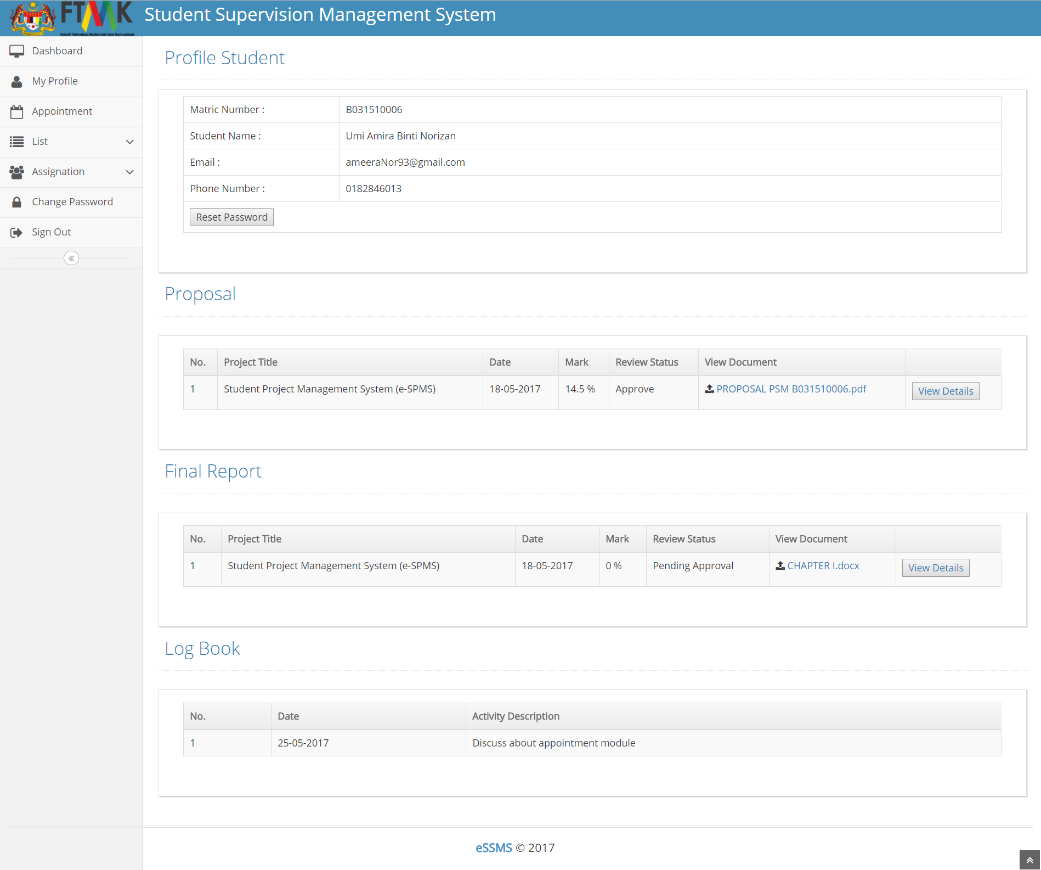
**Figure 4.3.26** Change position interface



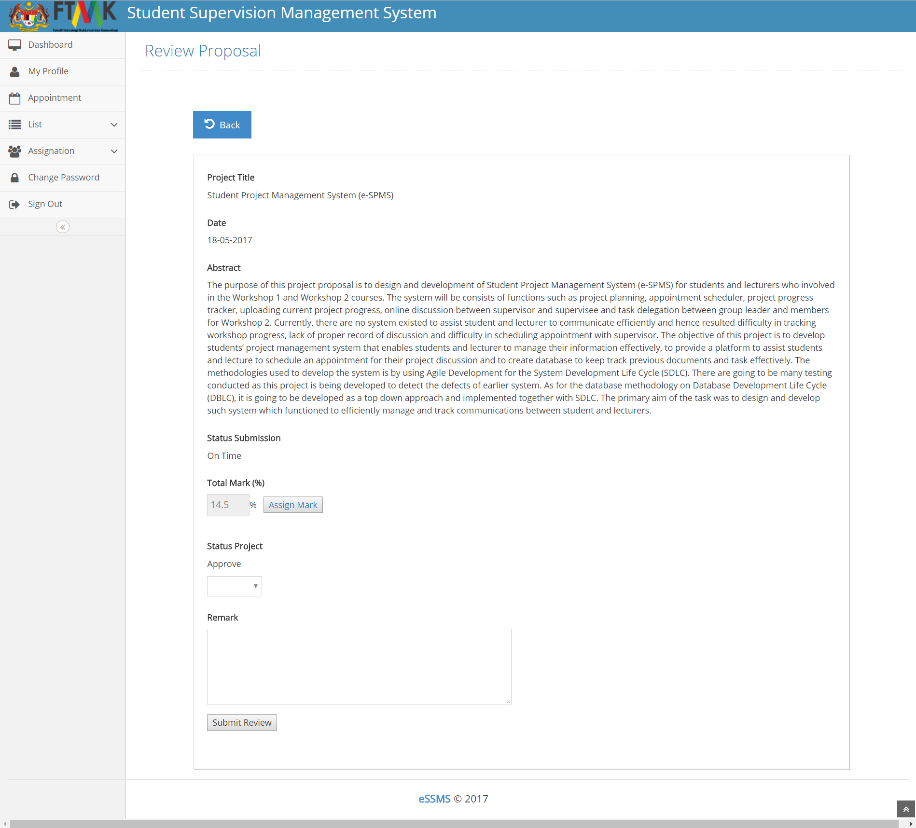
**Figure 4.3.27** Change password interface



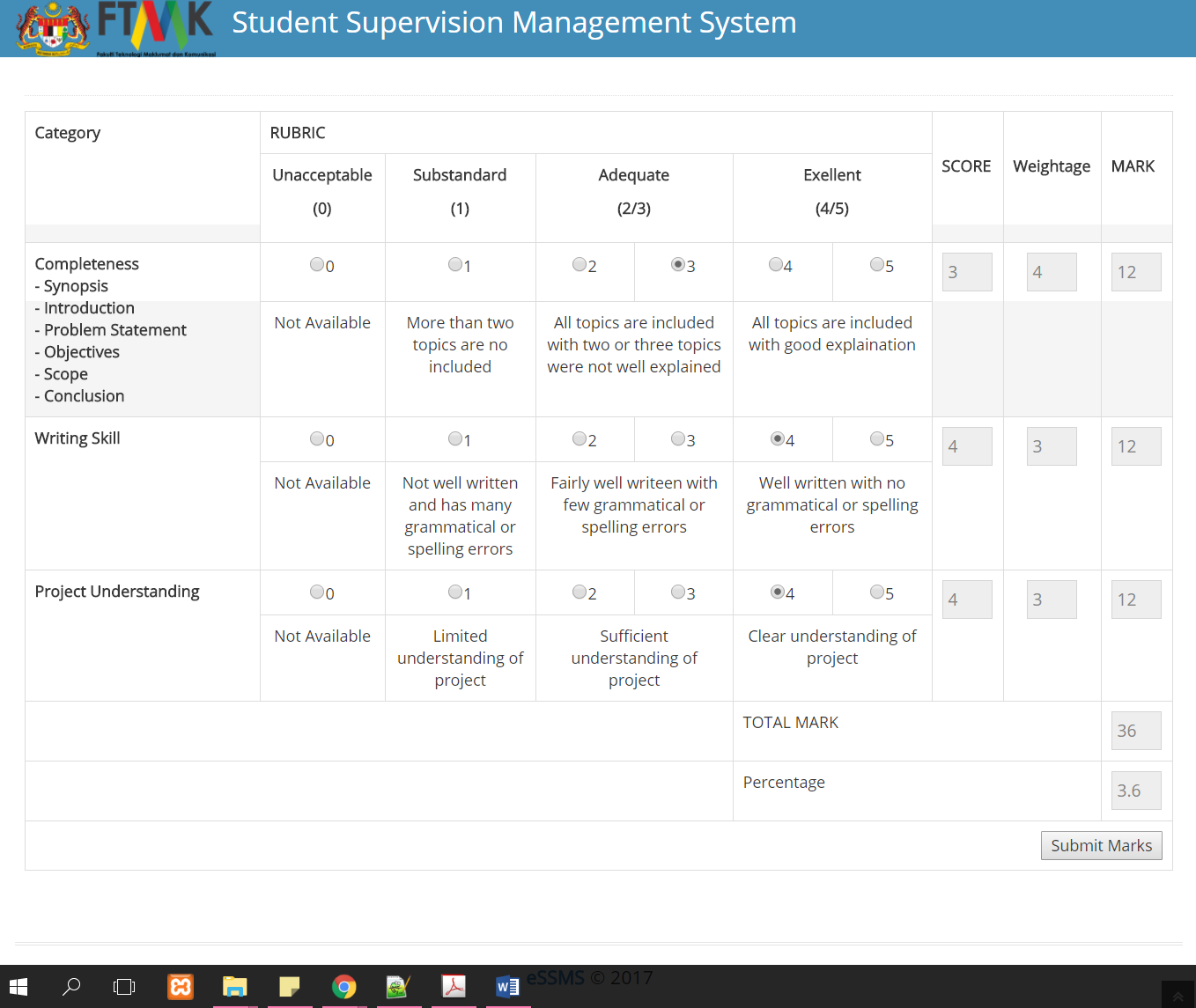
**Figure 4.3.28** List student to be supervise and list of previous student



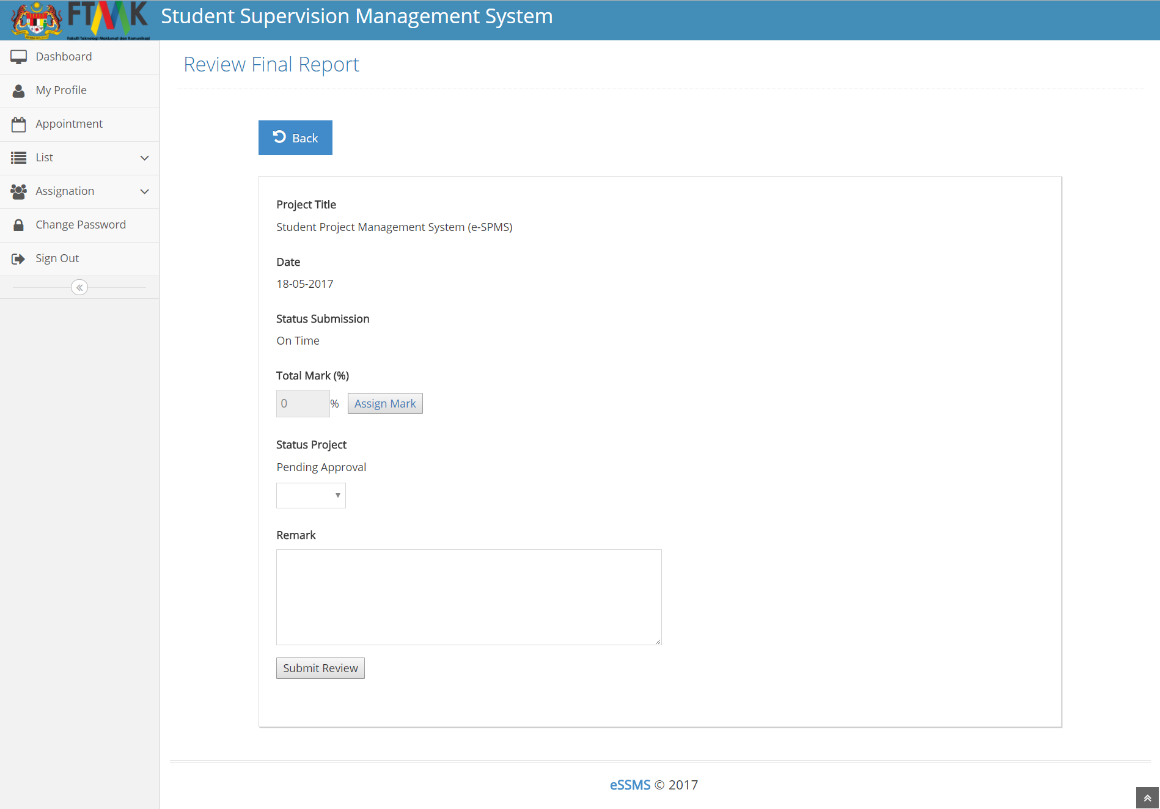
**Figure 4.3.29** Detail report of student



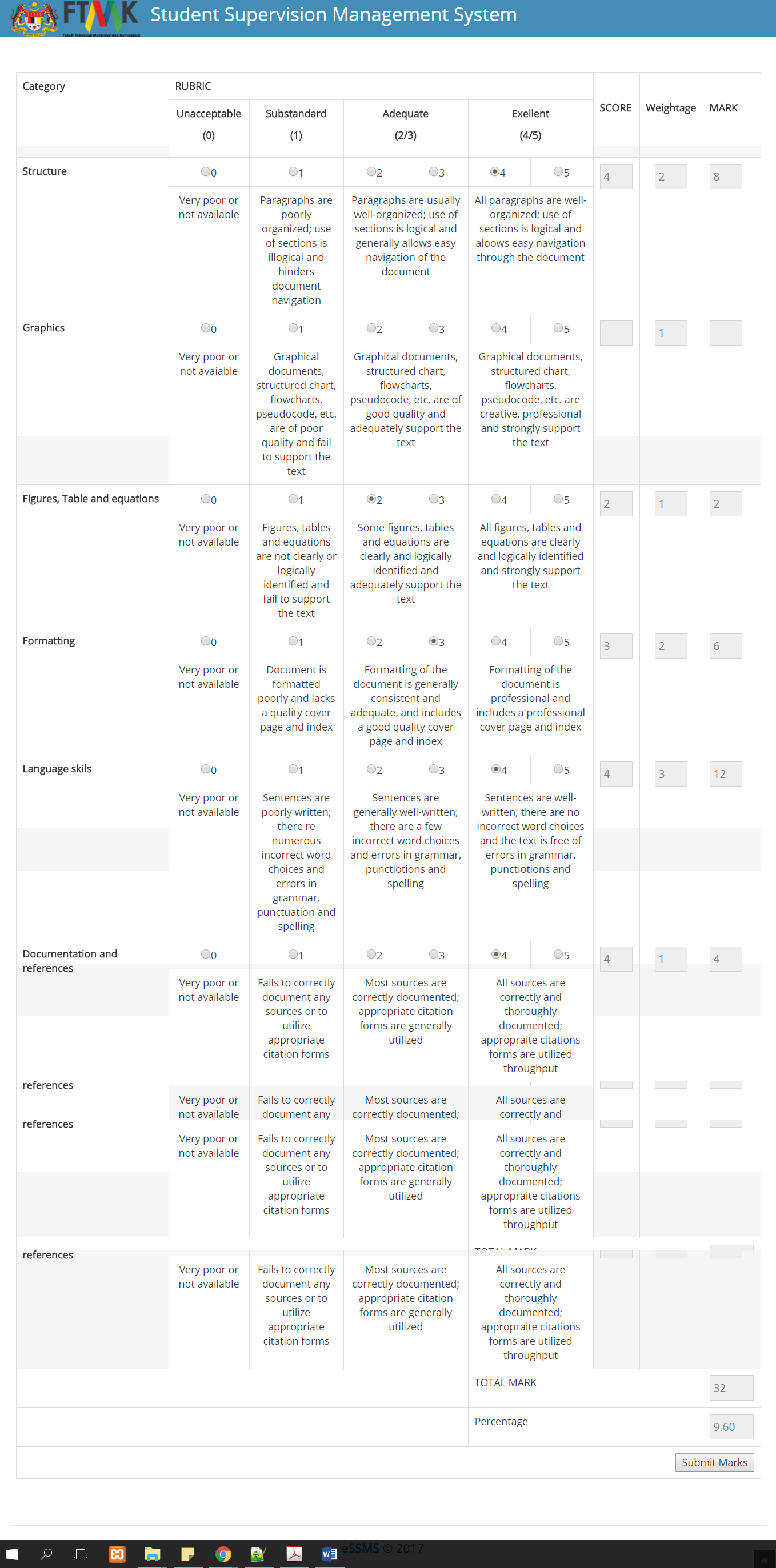
**Figure 4.3.30** Review Proposal



**Figure 4.3.31** Assign mark for proposal workshop 1



**Figure 4.3.32** Review Final Report



**Figure 4.3.33** Assign mark for final report workshop 1

**4.4 Conclusion**

This chapter discussed more about how design of the system is being built which include all type of design such as conceptual design, logical, physical and user interface. Each type provide different structure and function but with the same objectives. All the design type is related to each other to make the system work. The design is construct and being implement to make it real and can be used by targeted user.

**CHAPTER V**

**IMPLEMENTATION**

**5.1 Introduction**

This chapter discuss about the usage of the undertaking those two (2) sections which are the framework advancement and database execution. The system development environment will be explained on how the installation step, assign admin login and starting the database service. Besides that, I also consists about the database creation and database object. For the database implementation includes the DDL or DCL statements in the chosen DBMS which is mysql on Xampp Server. In this database includes main processes such as stored procedures and trigger by using this programming language.

**5.2 Software Development Environment Setup**

This section will explain about the initial setup of the project e-SSMS. All the components that are required for this project will be explained. The architecture used by the project is the three (3) – tier of system architecture.

Client Tier

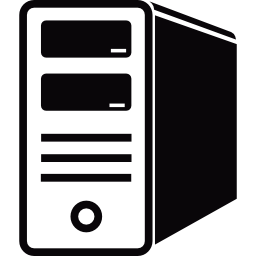
(Client Computers)

Business Logic Tier

(Application Server)

Database Tier

(Database Server)



**Xampp Server**

**MySQL**

**Figure 5.1** Three (3) – Tier System Architecture

**5.2.1 Software Development Setup**

The Software that will be used during development of e-SSMS is Adobe Dreamweaver CS5.5, in specific by using Hypertext Preprocessor (PHP) Language. To act a as platform for Database Management, MySQL Xampp Server will be used to store all the required data. Last but not least, Xampp Server version 3.0.12 will function as a server to integrate between database and interfaces of system.

**5.2.1.1 Software Development Setup – Server**

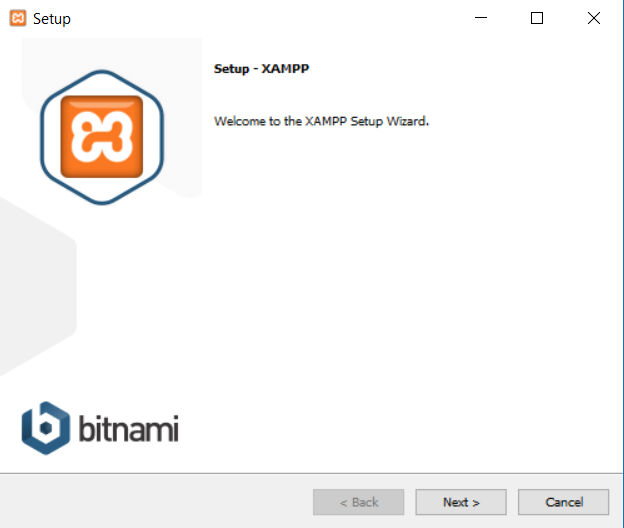
For the server parts, user need to type “http:\\localhost” in the web browser such as Google Chrome or Mozilla Firefox, the web browser will display the main page of the Xampp Server. User need to create a folder that will store the entire interface or the PHP language in the registry “C:\xampp\htdocs” so that user can view the interface in the web browser.

**Step 1 : Obtain a copy XAMPP by download**

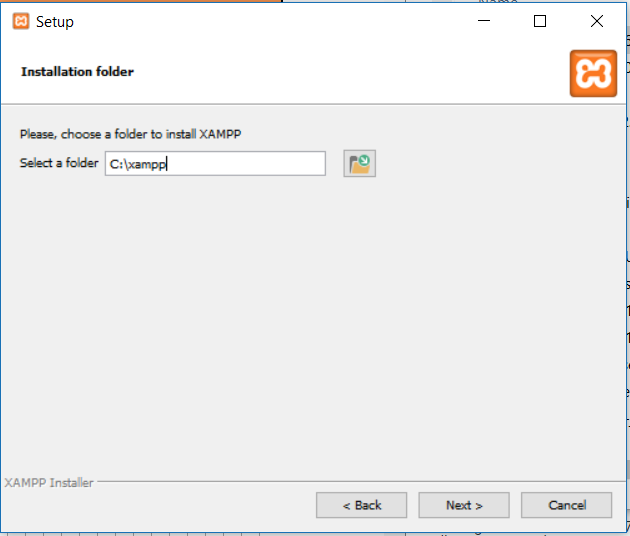
To obtain a copy of XAMPP, user need to download it from https://www.apachefriends.org/download\_success.html. It will provide an option whether 32 or 64-bit environment depending your computer’s configuration. In addition this download includes a phpMyAdmin to help you manage better on databases through web based GUI.

**Step 2 : Install the software**

XAMPP is downloaded and uncompressed; user will need to start the installation process.



**Figure 5.2** Xampp Server installation



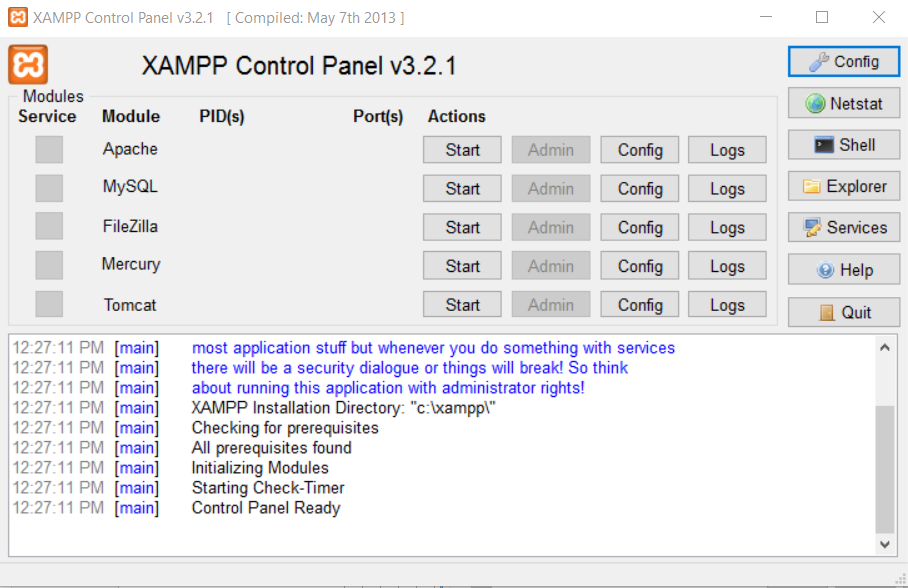
**Figure 5.3** Directory interface

Figure 5.3 shows that user need to accept the license agreement. User may accept since it is a GPL license. The next step is user is required to select the folder to place the install XAMPP Server. The “C:\xampp” will be default; however user can place the install folder into any directory they choose.

After selecting the directory, user will have option to setup icons. Click *Next* and then clicking *Install* to confirm the installation again. User will be asked to choose the default browser. Internet Explorer will act as default during the installation.

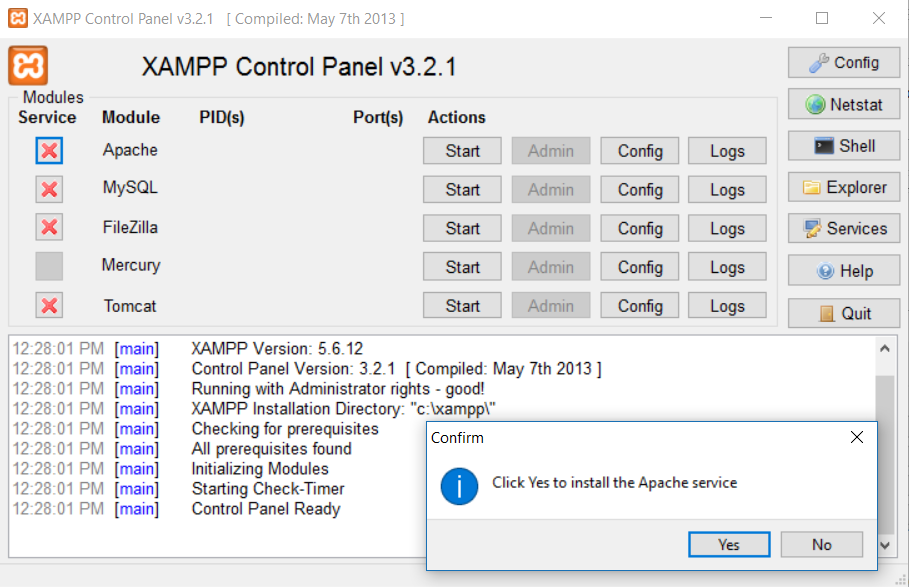
**Step 3 : Testing the installation**

Using the icon created or click *Start > All Programs > Apache Friends > XAMPP Control Panel*, the XAMPP Control Panel allow you to manually start and stop Apache and MySQL, or install them as services.



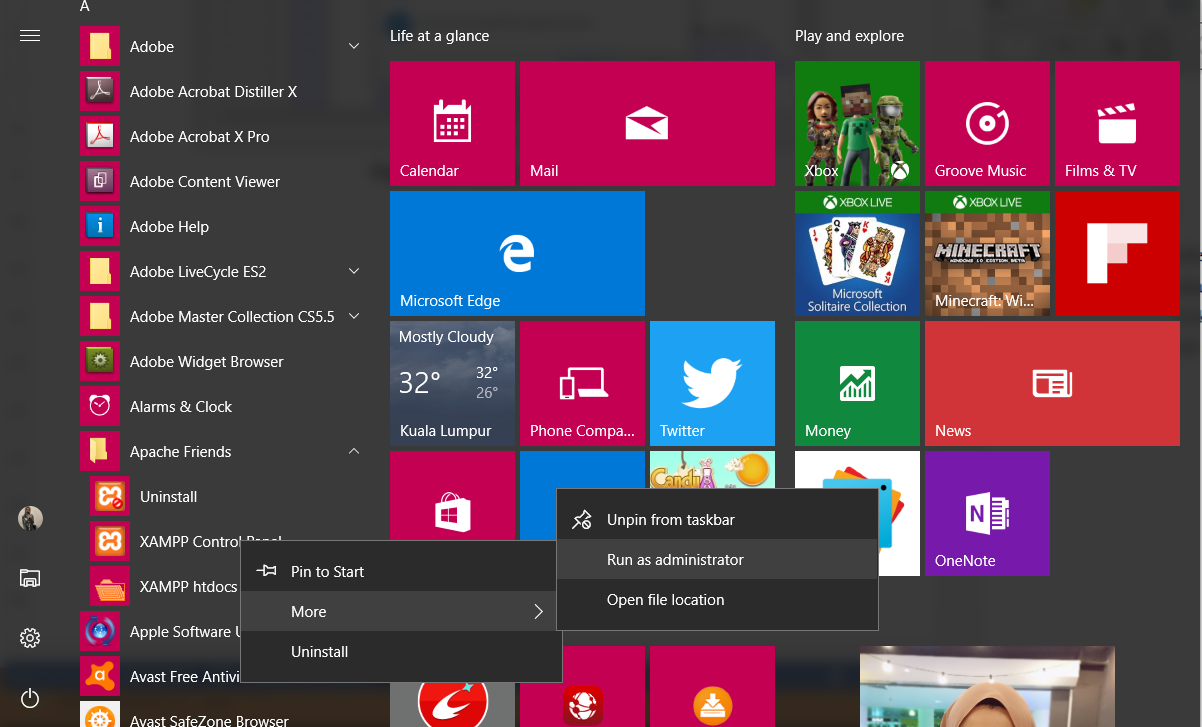
**Figure 5.4** XAMPP Control Panel

Figure 5.4 show how to start Apache or MySQL manually click the Start button under Actions next to the module. To install Apache and MySQL as services, click the red X to the left of each module under Service. You will see a small windows open, asking you to confirm that you want to install it as a service. Click Yes to complete the installation. Repeat for other module. The red X will change to a green check mark once the service has been installed successfully.



**Figure 5.5** Install Apache

After the initial setup is complete, you will need to run XAMPP as an administrator on Windows 10 when you start the program again. To do that, click Start find XAMPP Control Pnael in your Program list, the right-click and select Run as administrator show on Figure 5.6.



**Figure 5.6** Run as administrator

**5.2.1.2 Database Development Setup – Server**

Database server refers to back-end system in client/server architecture. Database server functions perform tasks such as data analysis, storage and manipulation. It related to database management system (DBMS). DBMS is collection of program that enables to extract information from database. DBMS used in this system is (PHPMyAdmin MySQL XAMPP Server). This DBMS has been chosen as it is the reliable way to perform a tasks such as retrieve, create, update and delete database objects. It also runs SQL statement and scripts, edit and debug SQL code and manipulate data.

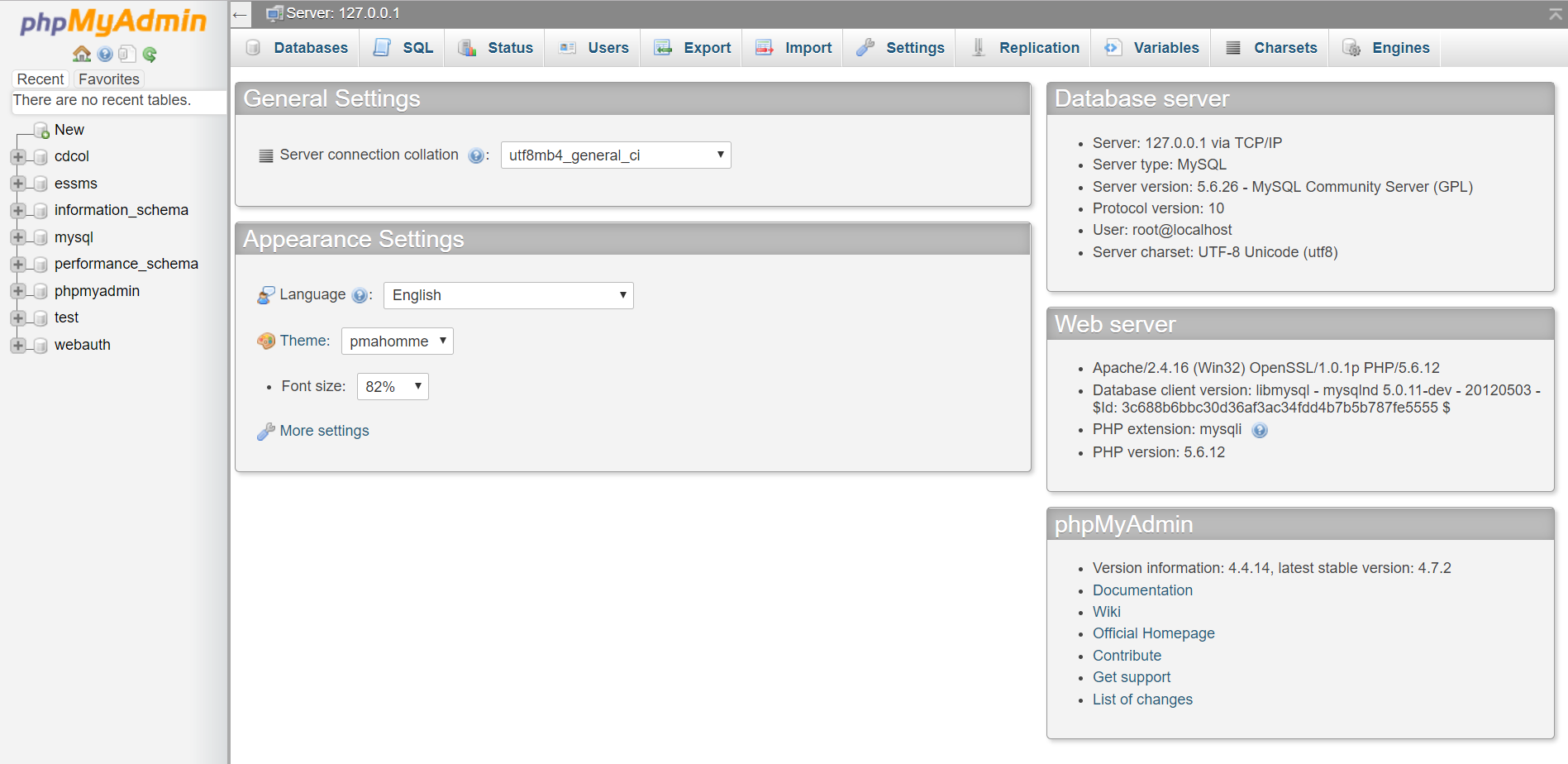
**Step 1 : Configuring MySQL database**

Since XAMPP Server has already provided MySQL, it is much easier to use as it only need to click phpMyAdmin in the XAMPP Server Configuration Interface.



**Figure 5.7** XAMPP Server Configuration Interface

After phpMyAdmin being clicked, this interface will be shown as Figure 5.8.



**Figure 5.8** phpMyAdmin database

If user wants to create a database, just click new on the left side bar. Then the new database will be crated. User need to insert the database name in provided field. If the database is created, the user can proceed on create a table, trigger or procedure in the database.

**5.3 Database Implementation**

This section will explained about how MySQL query is being used in accessing the database during the development of the system. There are a few ways to access the database data.

1. **CREATE TABLE clause**

Create table is create based on the ERD. The example of create table is in Figure 5.9 and the others will be include in Appendix D.

CREATE TABLE IF NOT EXISTS `appointment` (`id` int(255) NOT NULL AUTO\_INCREMENT, `date\_created` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,`start\_time` time NOT NULL, `end\_time` time NOT NULL,`start` datetime NOT NULL, `end` datetime NOT NULL, `color` varchar(7) NOT NULL, `status` varchar(30) NOT NULL, `title` varchar(50) NOT NULL,`location` varchar(50) NOT NULL, `matricNum` varchar(20) NOT NULL, `staffID` varchar(20) NOT NULL, PRIMARY KEY (`id`));

**Table 5.1** CREATE TABLE clause

1. **SELECT statement**

SELECT statement query to display data from database. The example of query is in Figure 5.10.

SELECT \* FROM STUDENT;

**Table 5.2** SELECT Statement

1. **WHERE clause statement**

WHERE clause statement is use to display an information with some condition and the example of query as shown in Figure 5.11.

SELECT \* FROM STUDENT WHERE studentName = 'Umi Amira Binti Norizan';

**Table 5.3** WHERE Clause

1. **TRIGGER clause**

TRIGGER clause will automatically execute when certain event occur in the table of database. Figure 5.12 show the example of trigger. The other example of the trigger can refer on the Appendix E.

DELIMITER $$

CREATE TRIGGER `before\_insert\_proposal` BEFORE INSERT ON `proposal`

FOR EACH ROW BEGIN

declare v\_typeworkshop varchar(50);

declare v\_dueDate date;

declare v\_subDate date;

declare v\_typeP varchar(50);

declare v\_reportID varchar(50);

declare v\_status varchar(50);

declare v\_reportSubmissionID int;

select typeOfWorkshop into v\_typeworkshop from report where reportID = NEW.reportID;

select dueSubmission,reportSubmissionID into v\_dueDate,v\_reportSubmissionID from reportSubmission where reportType = NEW.type and typeOfWorkshop = v\_typeworkshop;

if (NEW.submissionDate >= v\_dueDate)

then

set NEW.statusSubmission = 'Late';

set NEW.reportSubmissionID = v\_reportSubmissionID;

else

set NEW.statusSubmission = 'On Time';

set NEW.reportSubmissionID = v\_reportSubmissionID;

end if;

END

$$

DELIMITER ;

**Table 5.4** TRIGGER clause

1. **Stored Procedure**

Stored procedure is used to control the mechanism of the database. This stored procedure will return the result set of the information from database. Figure 5.13 shown the example of trigger. Other example of the procedure can refer on the Appendix F.

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert\_finalReport`(IN `v\_reportID` VARCHAR(35), IN `v\_finalreportID` VARCHAR(35), IN `v\_submissionDate` DATE, IN `v\_statusProject` VARCHAR(20), IN `v\_sourceFR` VARCHAR(100), IN `v\_type` VARCHAR(20))

BEGIN

insert into finalReport (finalreportID, submissionDate, statusProject, sourceFR, type, reportID)values

(v\_finalreportID, v\_submissionDate, v\_statusProject, v\_sourceFR, v\_type,v\_reportID);

END$$

**Table 5.5** Stored Procedure

**5.4 Conclusion**

A database can be thought of as a set of logically related files organized to facilitate access by one or more applications programs and to minimize data redundancy. Database, also called electronic database, any collection of data, or information that is specially organized for rapid search and retrieval by a computer. Databases are structured to facilitate the storage, retrieval, modification, and deletion of data in conjunction with various data-processing operations. A database management system (DBMS) extracts information from the database in response to queries.

**CHAPTER VI**

**TESTING**

**6.1 Introduction**

In this chapter will discuss about testing period and activity of the Student Supervision Management System (e-SSMS). Testing is a procedure with the objectives of identifying issues in a system or software and to know the quality level to flfill user needs. It has been conducted on the e-SSMS to validate and verify the system so that it can live up to its requirements of the organization. The system testing is important as it can avoid mistakes that are visible to users and also to avoid organization involved has a bad reputation.

Software testing is very challenging task. But, testing is not limited to the implementation of the system with the purpose of finding defects. It is also need to determine tes plan, test strategy, test design, test results and analysis during the testing phase. The software testing can be made effective and efficient by following the rules such as pick suitable set of test methods. This can gain confidence that the system will work smoothly and free from any errors.

The strategy for doing this system testing is for Faculty of Information and Communication Technology (FTMK) of UTeM as the client. After all the testing made, a questionnaire and an interview session being made in order to know whether this system meet user’s requirement or not.

**6.2 Test Plan**

Test plan is a document that contains of detailed procedures that determine the scope, approach, resources and schedule of all testing activities. Test plan is also used for help the testing activities; therefore it will achieve an efficient testing for the system.

**6.2.1 Test Organization**

On this topic will explain the person involve in the testing phases. This testing group will responsible in managing, executing and checking. The tester developer will lead the testing process and the main predominant in testing organization. The end users also aid to the testing group of system functionalities. The detailed about the person who involved in testing process is shown in Table 6.1 below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tester ID** | **Name** | **Roles** | **Responsibilities** |
|  | Umi Amira Binti Norizan | System Developer | Execute integration and component test. |
|  |  | Software Tester | - Analyze and review the functionality of the requirements.  - Monitoring the system performance. |
|  |  | Client | - Responsible in testing this system  - Analyze and review the functionality of the requirements. |

**Table 6.1** Test Organization

**6.2.2 Test Environment**

In this sub topic will explain details about the location of where this system is been tested. Table 6.2 shows the details of testing that has been made.

|  |  |
| --- | --- |
| **System Configuration** | **Specification** |
| Operating System | Windows 10 |
| Operating System | phpMyAdmin MySQL |
| Server | Xampp Server version 3.0.12 |
| Web browser | Google Chrome, Mozilla Firefox |
| System/ Programming Language | Hypertext Preprocessor (PHP), Hyper Text Markup Language (HTML) |

**Table 6.2** Environment Setup Specification

**6.2.3 Test Schedule**

Test schedule is a record of testing time table made for the system. All the testing made in based on the module state for the system. The testing schedule consist of the module, testing type, start date, end date and the duration take to complete the system. Table 6.3 describe the testing process in details.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Activities** | **Description** | **Start Date** | **End Date** | **Duration** |
| Unit Testing | Used to test functions and code module. | 10 July 2017 | 13 July 2017 | 3 days |
| Integration Testing | Used to test integrated module and verify combined functionality after integration. | 14 July 2017 | 17 July 2017 | 3 days |
| System Testing | Evaluate system compliance with its specific requirements. | 18 July 2017 | 21 July 2017 | 3 days |
| Acceptance Testing | Test completed system to end user. | 22 July 2017 | 25 July 2017 | 3 days |

**Table 6.3** Test Schedule Detail

**6.3 Test Strategy**

As for e-SSMS, Dynamic Testing will be used to evaluate this system. The execution of the software of a component or system is what involves in Dynamic Testing. Dynamic Testing method is divided into two different methods which are known as Black Box Testing and White Box Testing. Black Box Testing is known as behavioural testing which focus on the design of the system, while White Box Testing is a testing that permits user to examine the internal structure of the program. Table 6.4 describe in details about this approaches.

|  |  |
| --- | --- |
| **Approaches** | **Explanations** |
| White Box Testing | Testing that evaluate the internal structure of the program or evaluating system through developer using the program code. White box testing has three different technique, branch coverage technique and lastly path coverage technique. |
| Black Box Testing | Testing the system through their functional or non-functional without references to the internal structure of the component or system. Black box testing is focused solely on the output generated in respond to selected input and execution condition. |

**Table 6.4** Details of testing type

**6.4 Test Design**

**6.4.1 Test Description**

This sub topic will explain the test to be done every module stated in introduction section. Table 6.5 until table 6.16 will describe the test description in detail according to the system modules.

1. Login details

Login module is important for authorized user. User involve in this module is Admin, Committee, Supervisor and Student. User are required to enter ID and password in order to log in the system. Below are table 6.5 that will display the test details for Admin while Table 6.6 are for Committee, Supervisor and Student.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Input Test/Data** | **Action** | **Expected Output** |
| SSM A/01 | id : blank  password : blank | No input data | Error : Invalid id or password. Please try again. |
| SSM A/02 | id : admin  password : blank | Password left blank | Error : Invalid id or password. Please try again. |
| SSM A/03 | id : blank  password : \*\*\*\*\* | ID left blank | Error : Invalid id or password. Please try again. |
| SSM A/04 | id : admin  password : \*\*\*\*\* | All input inserted but misspelled | Error : Invalid id or password. Please try again. |
| SSM A/05 | id : admin  password : \*\*\*\*\* | All input inserted  and spell correctly | Error : Welcome to Admin Page. |

**Table 6.5** Test Case of Login for Admin

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Input Test/Data** | **Action** | **Expected Output** |
| SSM B/01 | id : blank  password : blank | No input data | Error : Invalid id or password. Please try again. |
| SSM B/02 | id : u1  password : blank | Password left blank | Error : Invalid id or password. Please try again. |
| SSM B/03 | id : blank  password : \*\*\*\*\* | ID left blank | Error : Invalid id or password. Please try again. |
| SSM B/04 | id : u1  password : \*\*\*\*\* | All input inserted but misspelled | Error : Invalid id or password. Please try again. |
| SSM B/05 | id : u1  password : \*\*\*\*\* | All input inserted  and spell correctly | Error : Welcome to eSSMS |

**Table 6.6** Test Case of Login for Committee, Supervisor and Student

1. Profile details

Profile module is for user to update their own profile and store into database. Below are table 6.7 that will display the test details for Committee and Supervisor while Table 6.8 are for Student.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Input Test/Data** | **Action** | **Expected Output** |
| SSM C/01 | email : blank  phone : blank | No input data | Error : Fill in required data |
| SSM C/02 | email : blank  phone : blank | If either one is blank | Error : Fill in required data |
| SSM C/03 | email : u1@gmail.com  phone : 0145637891 | All input is inserted | Your profile has been updated! |

**Table 6.7** Test Case of Profile for Student

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Input Test/Data** | **Action** | **Expected Output** |
| SSM D/01 | email : blank  phone : blank  interesting : blank | No input data | Error : Fill in required data |
| SSM D/02 | email : blank  phone : blank  interesting : blank | If either one is blank | Error : Fill in required data |
| SSM D/03 | email : u1@gmail.com  phone : 0145637891  interesting : Interesting with …. | All input is inserted | Your profile has been updated! |

**Table 6.8** Test Case of Profile for Committee and Supervisor

1. Appointment details

Table 6.9 shows the test case for appointment details. Student will insert the appointment details and it will store in database. Table 6.10 shows the test case for Committee and Supervisor to update their appointment details.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Input Test/Data** | **Action** | **Expected Output** |
| SSM E/01 | agenda : blank  date : blank  start time : blank  end time : blank | No input provided | Error : Fill in required data |
| SSM E/02 | agenda : blank  date : blank  start time : blank  end time : blank | Any required data is not inserted | Error : Fill in required data |
| SSM E/03 | agenda : Discuss about report  date : 20/7/2017  start time : 05.20 PM  end time : 06.20 PM | All input is inserted | Appointment Submitted |

**Table 6.9** Test Case of Appointment details for Student

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Input Test/Data** | **Action** | **Expected Output** |
| SSM F/01 | location : blank  status : blank | No input provided | Error : Fill in required data |
| SSM F/02 | location: blank  status: blank | Any required data is not inserted | Error : Fill in required data |
| SSM F/03 | location: At my office  status: Accept | All input is inserted |  |

**Table 6.10** Test Case of Appointment details for Committee and Supervisor

1. Proposal details

Table 6.11 shows the test case of proposal details. Student will insert all the details of upload proposal and it will keep it in database and all information is required. Table 6.12 shows the test case for Committee and Supervisor to update their student’s proposal details.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Input Test/Data** | **Action** | **Expected Output** |
| SSM G/01 | project title : blank  abstract : blank  proposal file: blank | No input provided | Error : Fill in required data |
| SSM G/02 | project title : blank  abstract : blank  proposal file: blank | Any required data is not inserted | Error : Fill in required data |
| SSM G/03 | project title : essms  abstract : This project ……  proposal file: B03151.docx | All required data is inserted | Upload Successful |

**Table 6.11** Test Case of Proposal details for Student

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Input Test/Data** | **Action** | **Expected Output** |
| SSM H/01 | mark : blank  status : blank  remarks : blank | No input provided | Error : Fill in required data |
| SSM H/02 | mark : blank  status : blank  remarks : blank | Any required data is not inserted | Error : Fill in required data |
| SSM H/03 | mark : blank  status : Accept  remarks : blank | Only required data is inserted | Review Submitted |
| SSM H/04 | mark : 4.7%  status : Accept  remarks : blank | All required data is inserted | Review Submitted |

**Table 6.12** Test Case of Proposal details for Committee and Supervisor

1. Final report details

Table 6.13 shows the test case of final report details. Student will insert all the details of upload final report and it will keep it in database and all information is required. Table 6.14 shows the test case for Committee and Supervisor to update their student’s final report details.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Input Test/Data** | **Action** | **Expected Output** |
| SSM I/01 | final report file: blank | No input provided | Error : Fill in required data |
| SSM I/02 | final report file: B03151.docx | All required data is inserted | Upload Successful |

**Table 6.13** Test Case of Final report details for Student

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Input Test/Data** | **Action** | **Expected Output** |
| SSM J/01 | mark : blank  status : blank  remarks : blank | No input provided | Error : Fill in required data |
| SSM J/02 | mark : blank  status : blank  remarks : blank | Any required data is not inserted | Error : Fill in required data |
| SSM J/03 | mark : blank  status : Accept  remarks : blank | Only required data is inserted | Review Submitted |
| SSM J/04 | mark : 10.7%  status : Accept  remarks : blank | All required data is inserted | Review Submitted |

**Table 6.14** Test Case of Final report details for Committee and Supervisor

1. Logbook details

Table 6.15 shows the test case of logbook details. Student will insert all the details of logbook and it will keep it in database and all information is required.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Input Test/Data** | **Action** | **Expected Output** |
| SSM K/01 | date : blank  activity description : blank | No input provided | Error : Fill in required data |
| SSM K/02 | abstract : blank  activity description : blank | Any required data is not inserted | Error : Fill in required data |
| SSM K/03 | date : 22/7/2017  activity description : Meeting for progress 2 | All required data is inserted | Upload Successful |

**Table 6.15** Test Case of Logbook details for Student

1. Change password details

Table 6.16 shows the test case of change password details. User will insert their password and kept it in database.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Input Test/Data** | **Action** | **Expected Output** |
| SSM L/01 | new password : blank  change password : blank | No input provided | Error : Fill in required data |
| SSM L/02 | new password : blank  change password : blank | Any required data is not inserted | Error : Fill in required data |
| SSM L/03 | new password : 123  change password : 123 | All required data is inserted | Your password has been change! |

**Table 6.16** Test Case of Change password details

**6.4.2 Test Data**

Test data is the part to test the actual data that will be used to ensure the system for correctness and system effectiveness. Table 6.17 and Table 6.23 below show the details to test data.

1. Login details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **User** | **Attribute** | **Data** |
| T/01 | Admin | id | admin |
|  |  | password | \*\*\*\*\* |
| T/02 | Committee | position | committee |
|  |  | id | 0078 |
|  |  | password | \*\*\*\*\*\* |
| T/03 | Supervisor | position | supervisor |
|  |  | id | 0097 |
|  |  | password | \*\*\*\*\*\* |
| T/04 | Student | position | student |
|  |  | id | B031510006 |
|  |  | password | \*\*\*\*\*\* |

**Table 6.17** Details of Login Test Data

1. Profile details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **User** | **Attribute** | **Data** |
| T/05 | Committee / Supervisor | lecturerEmail | fadzil@utem.my |
|  |  | lecturerPhoneNum | 0198985649 |
|  |  | bio | Interesting with php language |
| T/06 | Student | studentEmail | ameeraNor@gmail.com |
|  |  | studentPhoneNum | 0182846012 |

**Table 6.18** Details of Profile Test Data

1. Appointment details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **User** | **Attribute** | **Data** |
| T/07 | Committee / Supervisor | location | At my office |
|  |  | status | Accept |
| T/08 | Student | title | Discuss about report |
|  |  | start | 20/7/2017 |
|  |  | start\_time | 05.20 PM |
|  |  | end\_time | 06.20 PM |

**Table 6.19** Details of Appointment Test Data

1. Proposal details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **User** | **Attribute** | **Data** |
| T/09 | Committee / Supervisor | mark | 4.7% |
|  |  | statusProject | Accept |
|  |  | remarks | - |
| T/10 | Student | title | essms |
|  |  | abstract | This project …… |
|  |  | sourceP | B03151.docx |

**Table 6.20** Details of Proposal Test Data

1. Final report details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **User** | **Attribute** | **Data** |
| T/11 | Committee / Supervisor | mark | 17.7% |
|  |  | statusProject | Accept |
|  |  | remarks | - |
| T/12 | Student | sourceFR | final\_B03151.docx |

**Table 6.21** Details of Final report Test Data

1. Logbook details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **User** | **Attribute** | **Data** |
| T/13 | Student | submissionDate | 23/7/2017 |
|  |  | acitivityDescription | Meet lecturer for progress 2 |

**Table 6.22** Details of Logbook Test Data

1. Change password details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **User** | **Attribute** | **Data** |
| T/14 | Admin | password | \*\*\*\* |
| T/15 | Committee | password | \*\*\*\* |
| T/16 | Supervisor | password | \*\*\*\* |
| T/17 | Student | password | \*\*\*\* |

**Table 6.23** Details of Change password Test Data

**6.5 Test Result and Analysis**

This section will describe about the validation of the data over e-SSMS. This test document is to see whether the result in test design is the same with the test results and analysis document. Table below shows the details of the test document.

1. Login details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **Action** | **Result** | **Pass initial (pass/fail)** |
| SSM\_1000 | Valid input : All columns are fulfilling | System will prompt Welcome to eSSMS | Pass |

**Table 6.24** Test Result and Analysis of Valid Login Details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **Action** | **Result** | **Pass initial (pass/fail)** |
| SSM\_1001 | Invalid input : validate that the user id and password are incorrect | System will prompt invalid id or password. Please try again. | Pass |

**Table 6.25** Test Result and Analysis of Invalid Login Details

1. Profile details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **Action** | **Result** | **Pass initial (pass/fail)** |
| SSM\_1002 | Valid input : All required columns are fulfilling | System will prompt Your profile has been updated! | Pass |

**Table 6.26** Test Result and Analysis of Valid Profile Details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **Action** | **Result** | **Pass initial (pass/fail)** |
| SSM\_1003 | Invalid input : validate that all the required data are fulfilling. | System will prompt required! | Pass |

**Table 6.27** Test Result and Analysis of Invalid Profile Details

1. Appointment details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **Action** | **Result** | **Pass initial (pass/fail)** |
| SSM\_1004 | Valid input : All required columns are fulfilling | System will prompt Appointment Submited | Pass |

**Table 6.28** Test Result and Analysis of Valid Appointment Details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **Action** | **Result** | **Pass initial (pass/fail)** |
| SSM\_1005 | Invalid input : validate that all the required data are fulfilling. | System will prompt required! | Pass |

**Table 6.29**  Test Result and Analysis of Invalid Appointment Details

1. Proposal details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **Action** | **Result** | **Pass initial (pass/fail)** |
| SSM\_1006 | Valid input : All required columns are fulfilling | System will prompt Upload Successful | Pass |
| SSM\_1007 | Valid input : All required columns are fulfilling | System will prompt Review Submitted | Pass |

**Table 6.30** Test Result and Analysis of Valid Proposal Details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **Action** | **Result** | **Pass initial (pass/fail)** |
| SSM\_1008 | Invalid input : validate that all the required data are fulfilling. | System will prompt required! | Pass |
| SSM\_1009 | Invalid input : validate that all the required data are fulfilling. | System will prompt required! | Pass |

**Table 6.31** Test Result and Analysis of Invalid Proposal Details

1. Final report details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **Action** | **Result** | **Pass initial (pass/fail)** |
| SSM\_1010 | Valid input : All required columns are fulfilling | System will prompt Upload Successful | Pass |
| SSM\_1011 | Valid input : All required columns are fulfilling | System will prompt Review Submitted | Pass |

**Table 6.32** Test Result and Analysis of Valid Final report Details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **Action** | **Result** | **Pass initial (pass/fail)** |
| SSM\_1012 | Invalid input : validate that all the required data are fulfilling. | System will prompt required! | Pass |
| SSM\_1013 | Invalid input : validate that all the required data are fulfilling. | System will prompt required! | Pass |

**Table 6.33** Test Result and Analysis of Invalid Final report Details

1. Logbook details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **Action** | **Result** | **Pass initial (pass/fail)** |
| SSM\_1014 | Valid input : All required columns are fulfilling | System will prompt Upload Successful | Pass |

**Table 6.34** Test Result and Analysis of Valid Logbook Details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **Action** | **Result** | **Pass initial (pass/fail)** |
| SSM\_1015 | Invalid input : validate that all the required data are fulfilling. | System will prompt required! | Pass |

**Table 6.35** Test Result and Analysis of Invalid Logbook Details

1. Change password details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **Action** | **Result** | **Pass initial (pass/fail)** |
| SSM\_1014 | Valid input : All required columns are fulfilling | System will prompt Your password has been change! | Pass |

**Table 6.36** Test Result and Analysis of Valid Change password Details

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No** | **Action** | **Result** | **Pass initial (pass/fail)** |
| SSM\_1015 | Invalid input : validate that all the required data are fulfilling. | System will prompt required! | Pass |

**Table 6.37** Test Result and Analysis of Invalid Change password Details

**6.6 Conclusion**

As the conclusion, this chapter explains the method use to verify and validate the system to make sure the quality of the project have achieved through requirements. Testing must be planned thoroughly due to the cost of fixing one major defects can be very costly. Test plan consists of several test that examine varies aspects of the system. The next chapter will be discussed about project conclusion.

**CHAPTER VII**

**CONCLUSION**

**7.1 Introduction**

Student Supervision Management System (e-SSMS) is a system developed for the student and lecturers that are involved in Workshop 1 and Workshop 2 courses that make process and procedure task will become much smoother and save times and also e-SSMS will fulfil my qualifications to achieve Bachelor of Computer Science (Database Management).

However, this system has many flaws but there are some advantages and it is o upgrade in a long-term duration. During the development of this system, there are a lot of knowledge that I have gained n terms of drafting the final report, designing the system and flow data. This is the most challenging part throughout my three and a half years in Universiti Teknikal Malaysia Melaka (UTeM).

Hence, this system has the ability to be developed more sophisticated with better expertise than my lack knowledge and expertise in this area. Besides that, I think that this system has the value to go through market.

In conclusion, have developed a system that is 80% achieving the main objectives despite of many shortcomings and failures in work. I am quite proud of my ahievement as I was able to develop a system by myself.

**7.2 Observation on Weakness and Strengths**

1. **Strength**

* Can manage report via online (no more submitting on a piece of paper).
* Can control the daily appointments.
* Can assign supervisor and committee effectively.
* Reduce the use of time when user want to make an appointment.
* Can keep track their documents and task effectively.

1. **Weakness**

* There is no notification system.
* This system does not provide backup and recovery.
* There is no error handling if there are data entry errors.
* There is no report performance student.

**7.3 Propositions of improvement**

For the improvement of the system it should be better if backup and recovery are implemented in the system. If there is data corruption or problems with the system, all the data can be retrieved without have to think about loss of the important data.

Other than that, it should be better if others weaknesses also being implemented in this system as if all weaknesses covered then it will be a complete system without any error.

**7.4 Contribution**

Student Supervision Management System (e-SSMS) is a web based system to be develop for supervision management.This system provides user to make an appointment scheduling. User can choose date and time based on the availability displayed in the system, user can reject or approves all the requested appointment and also user can view the status of their appointment approval.

Moreover, this system also provide report module that can ease the user to keep track of the reports. It’s able to uploading the proposal, final report and log book. The record of log book can be monitor week by week and able to reviewing all activities that have been updated in the system.

Beside that, this system also provide module of supervision that able to assign supervisor and committe that involve with workshop1 and workshop 2 courses.

Thus, this system is very useful for student supervision management system (e-SSMS) as it has 80% fulfilment for the objectives. It will contribute in ease user management.

**7.5 Conclusion**

The conclusion that can be concluded after completing the system, it been developed to ease for the users to manage documentation management by implementing various functions in the system. Although the system is not met all the objectives, but it is still can function smoothly. However, if there is some improvements add to the system such as a user friendly system or backup and recovery is implemented the system will be more reliable and well developed.

**REFERENCES**

[1] Soler, N. G., Mast, T., & Kienzler, L. M. (1981). “A logbook system for monitoring student skills and experiences. *Academic Medicine”* [Online] Available f[rom: http://scholar.google.com/scholar](http://scholar.google.com/scholar)

[2] Universiti Teknikal Malaysia. “Panduan Penulisan Laporan PSM BITD 2015.” [Online] Available from: http:/www.ulearn.utem.edu.my/

**BIBLIOGRAPHY**

[1] Soler, N. G., Mast, T., & Kienzler, L. M. (1981). “A logbook system for

monitoring student skills and experiences. *Academic Medicine”* [Online] Available from: <http://scholar.google.com/scholar>

[2] Universiti Teknikal Malaysia Melaka . “Panduan Penulisan Laporan PSM BITD

2015.” [Online] Available from: http:[/www.ulearn.utem.edu.m](http://www.ulearn.utem.edu.my/)y[/](http://www.ulearn.utem.edu.my/)

**ATTACHMENT 1.1**

**APPENDIX A**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table name** | **Attribute name** | **Content** | **Type** | **Example** |
| **position** | positionID | Position ID | int (10) | 1 |
| positionName | Position Name | varchar (20) | Admin |
| **login** | id | Staff ID / Matric Number | int (10) | 00077 |
| password | Password | varchar (10) | abc123 |
| positionID | Position ID | int (10) | 3 |
| **student** | matricNum | Matric Number | varchar (20) | B031510006 |
| studentName | Student Name | varchar (50) | Umi Amira Binti Norizan |
| studentEmail | Student Email | varchar (50) | ameeraNor93@gmail.com |
| studentPhoneNum | Student Phone Number | varchar (15) | 0182846012 |
| course | Student Course | varchar (30) | BITD |
| year | Year | int (10) | 2 |
| semester | Semester | int (10) | 2 |
| typeOfWorkshop | Type of Workshop | int (10) | 2 |
| lecturerID | Staff ID | varchar (20) | 00097 |
| statusAssign | Status Assign | varchar (30) | Not Assigned |
| positionID | Position ID | int (10) | 4 |
| **lecturer** | staffID | Staff ID | varchar (20) | 00097 |
| lecturerName | Lecturer Name | varchar (50) | Safiza Suhana Binti Kamal Baharin |
| lecturerEmail | Lecturer Email | varchar (50) | safiza@utem.edu.my |
| lecturerPhoneNum | Lecturer Phone Number | varchar (15) | 0196685808 |
| department | Lecturer Department | varchar (30) | SE |
| bio | About Lecturer | varchar (100) |  |
| positionID | Position ID | int (10) | 3 |
| **lecturer\_position** | positionID | Position ID | int (10) | 3 |
| staffID | Staff ID | varchar (20) | 00097 |
| assignationWorkshop | Assignation Workshop | varchar (20) | 0 |
| **report** | reportID | Report ID | varchar (35) | R01\_2017\_2 |
| title | Report Title | varchar (60) | Car Rental System |
| typeOfWorkshop | Type of Workshop | int (10) | 2 |
| matricNum | Matric Number | varchar (20) | B031510352 |
| **finalreport** | finalreportID | Final Report ID | varchar (15) | FR01\_2017\_0 |
| submissionDate | Submission Date | date | 2017-05-03 |
| statusSubmission | Status Submission | varchar (20) | On Time |
| statusProject | Status Project | varchar (20) | Approve |
| remarks | Comment | text |  |
| mark | Mark | double | 13.2 |
| sourceFR | Upload Final Report file | varchar (100) | CHAPTER I.docx |
| type | Type of report | varchar (20) | Final Report |
| reportID | Report ID | varchar (35) | R01\_2017\_1 |
| reportSubmissionID | Report Submission ID | int (225) | 3 |
| **proposal** | proposalID | Proposal ID | varchar (35) | P01\_2017\_0 |
| abstract | Abstract | text | The purpose of this project proposal is to design ... |
| submissionDate | Submission Date | date | 2017-05-03 |
| statusSubmission | Status Submission | varchar (20) | On Time |
| statusProject | Status Project | varchar (20) | Approve |
| remarks | Comment | text |  |
| mark | Mark | double | 5.7 |
| sourceP | Upload Proposal file | varchar (100) | PROPOSAL PSM B031510006.pdf |
| type | Type of report | varchar (20) | Proposal |
| reportID | Report ID | varchar (35) | R01\_2017\_0 |
| reportSubmissionID | Report Submission ID | int (225) | 1 |
| **logbook** | logbookID | Logbook ID | varchar (35) | L01\_2017\_0 |
| submissionDate | Submission Date | date | 2017-05-03 |
| activityDescription | Activity Description | text | Decide the design of  poster project for psm1 |
| reportID | Report ID | varchar (35) | Decide the design of  poster project for psm1 |
| **reportsubmission** | reportSubmissionID | Report Submission ID | int (225) | 1 |
| dueSubmission | Due Submission Date | date | 2017-05-30 |
| reportType | Type of report | varchar (20) | Proposal |
| typeOfWorkshop | Type of Workshop | int (10) | 2 |

**ATTACHMENT 1.2**

**APPENDIX B**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Table name** | **Attribute name** | **Content** | **Type** | **Format** | **Required** | **PK / FK** | **Example** |
| **position** | positionID | Position ID | int (10) | 999999 | Y | PK | 1 |
| positionName | Position Name | varchar (20) | Xxxxxx |  |  | Admin |
| **login** | id | Staff ID / Matric Number | int (10) | 999999 | Y | PK | 00077 |
| password | Password | varchar (10) | Xxxxxx |  |  | abc123 |
| positionID | Position ID | int (10) | Xxxxxx | Y | PK/FK | 3 |
| **student** | matricNum | Matric Number | varchar (20) | Xxxxxx | Y | PK/FK | B031510006 |
| studentName | Student Name | varchar (50) | Xxxxxx |  |  | Umi Amira Binti Norizan |
| studentEmail | Student Email | varchar (50) | Xxxxxx |  |  | ameeraNor93@gmail.com |
| studentPhoneNum | Student Phone Number | varchar (15) | Xxxxxx |  |  | 0182846012 |
| course | Student Course | varchar (30) | Xxxxxx |  |  | BITD |
| year | Year | int (10) | 999999 |  |  | 2 |
| semester | Semester | int (10) | 999999 |  |  | 2 |
| typeOfWorkshop | Type of Workshop | int (10) | 999999 |  |  | 2 |
| lecturerID | Staff ID | varchar (20) | Xxxxxx |  |  | 00097 |
| statusAssign | Status Assign | varchar (30) | Xxxxxx |  | FK | Not Assigned |
| positionID | Position ID | int (10) | 999999 |  | FK | 4 |
| **lecturer** | staffID | Staff ID | varchar (20) | Xxxxxx | Y | PK/FK | 00097 |
| lecturerName | Lecturer Name | varchar (50) | Xxxxxx |  |  | Safiza Suhana Binti Kamal Baharin |
| lecturerEmail | Lecturer Email | varchar (50) | Xxxxxx |  |  | safiza@utem.edu.my |
| lecturerPhoneNum | Lecturer Phone Number | varchar (15) | Xxxxxx |  |  | 0196685808 |
| department | Lecturer Department | varchar (30) | Xxxxxx |  |  | SE |
| bio | About Lecturer | varchar (100) | Xxxxxx |  |  |  |
| positionID | Position ID | int (10) | 999999 |  | FK | 3 |
| **lecturer\_position** | positionID | Position ID | int (10) | 999999 | Y | PK/FK | 3 |
| staffID | Staff ID | varchar (20) | Xxxxxx | Y | PK/FK | 00097 |
| assignationWorkshop | Assignation Workshop | varchar (20) | Xxxxxx |  |  | 0 |
| **report** | reportID | Report ID | varchar (35) | Xxxxxx | Y | PK | R01\_2017\_2 |
| title | Report Title | varchar (60) | Xxxxxx |  |  | Car Rental System |
| typeOfWorkshop | Type of Workshop | int (10) | 999999 |  |  | 2 |
| matricNum | Matric Number | varchar (20) | Xxxxxx |  | FK | B031510352 |
| **finalreport** | finalreportID | Final Report ID | varchar (15) | Xxxxxx | Y | PK | FR01\_2017\_0 |
| submissionDate | Submission Date | date | yyyy-mm-dd |  |  | 2017-05-03 |
| statusSubmission | Status Submission | varchar (20) | Xxxxxx |  |  | On Time |
| statusProject | Status Project | varchar (20) | Xxxxxx |  |  | Approve |
| remarks | Comment | text | Xxxxxx |  |  |  |
| mark | Mark | double | 00.00 |  |  | 13.2 |
| sourceFR | Upload Final Report file | varchar (100) | Xxxxxx |  |  | CHAPTER I.docx |
| type | Type of report | varchar (20) | Xxxxxx |  |  | Final Report |
| reportID | Report ID | varchar (35) | Xxxxxx |  | FK | R01\_2017\_1 |
| reportSubmissionID | Report Submission ID | int (225) | 999999 |  | FK | 3 |
| **proposal** | proposalID | Proposal ID | varchar (35) | Xxxxxx | Y | PK | P01\_2017\_0 |
| abstract | Abstract | text | Xxxxxx |  |  | The purpose of this project proposal is to design ... |
| submissionDate | Submission Date | date | yyyy-mm-dd |  |  | 2017-05-03 |
| statusSubmission | Status Submission | varchar (20) | Xxxxxx |  |  | On Time |
| statusProject | Status Project | varchar (20) | Xxxxxx |  |  | Approve |
| remarks | Comment | text | Xxxxxx |  |  |  |
| mark | Mark | double | 00.00 |  |  | 5.7 |
| sourceP | Upload Proposal file | varchar (100) | Xxxxxx |  |  | PROPOSAL PSM B031510006.pdf |
| type | Type of report | varchar (20) | Xxxxxx |  |  | Proposal |
| reportID | Report ID | varchar (35) | Xxxxxx |  | FK | R01\_2017\_0 |
| reportSubmissionID | Report Submission ID | int (225) | 999999 |  | FK | 1 |
| **logbook** | logbookID | Logbook ID | varchar (35) | Xxxxxx | Y | PK | L01\_2017\_0 |
| submissionDate | Submission Date | date | yyyy-mm-dd |  |  | 2017-05-03 |
| activityDescription | Activity Description | text | Xxxxxx |  |  | Decide the design of  poster project for psm1 |
| reportID | Report ID | varchar (35) | Xxxxxx |  | FK | Decide the design of  poster project for psm1 |
| **reportsubmission** | reportSubmissionID | Report Submission ID | int (225) | 999999 | Y | PK | 1 |
| dueSubmission | Due Submission Date | date | yyyy-dd-mm |  |  | 2017-05-30 |
| reportType | Type of report | varchar (20) | Xxxxxx |  |  | Proposal |
| typeOfWorkshop | Type of Workshop | int (10) | 999999 |  |  | 2 |

**ATTACHMENT 1.3**

**APPENDIX C**

CREATE TABLE IF NOT EXISTS `appointment` (

`id` int(255) NOT NULL AUTO\_INCREMENT,

`date\_created` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,`start\_time` time NOT NULL,

`end\_time` time NOT NULL,`start` datetime NOT NULL,

`end` datetime NOT NULL, `color` varchar(7) NOT NULL,

`status` varchar(30) NOT NULL,`title` varchar(50) NOT NULL,

`location` varchar(50) NOT NULL,`matricNum` varchar(20) NOT NULL,

`staffID` varchar(20) NOT NULL,PRIMARY KEY (`id`))

ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=38 ;

**Appointment table**

CREATE TABLE IF NOT EXISTS `finalreport` (

`finalreportID` varchar(15) NOT NULL,`submissionDate` date DEFAULT NULL,

`statusSubmission` varchar(20) DEFAULT NULL,`statusProject` varchar(20) DEFAULT NULL,`remarks` text,`mark` double NOT NULL,

`sourceFR` varchar(100) DEFAULT NULL,`type` varchar(20) DEFAULT NULL,

`reportID` varchar(35) DEFAULT NULL,`reportSubmissionID` int(225) NOT NULL,

PRIMARY KEY (`finalreportID`))

ENGINE=InnoDB DEFAULT CHARSET=latin1;

**Final Report table**

CREATE TABLE IF NOT EXISTS `lecturer` (

`staffID` varchar(20) NOT NULL,`lecturerName` varchar(50) NOT NULL,

`lecturerEmail` varchar(50) NOT NULL,`lecturerPhoneNum` varchar(15) NOT NULL,

`department` varchar(30) NOT NULL,`bio` varchar(100) NOT NULL,

`positionID` int(10) NOT NULL,PRIMARY KEY (`staffID`))

ENGINE=InnoDB DEFAULT CHARSET=latin1;

**Lecturer table**

CREATE TABLE IF NOT EXISTS `lecturer\_position` (

`positionID` int(10) NOT NULL,

`staffID` varchar(20) NOT NULL,

`assignationWorkshop` varchar(10) NOT NULL,

PRIMARY KEY (`positionID`,`staffID`),

KEY `staffID` (`staffID`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

**Lecturer\_position table**

CREATE TABLE IF NOT EXISTS `logbook` (

`logbookID` varchar(35) NOT NULL,

`submissionDate` date NOT NULL,

`activityDescription` text NOT NULL,

`reportID` varchar(35) NOT NULL,

PRIMARY KEY (`logbookID`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

**Logbook table**

CREATE TABLE IF NOT EXISTS `login` (

`id` varchar(20) NOT NULL,

`password` varchar(10) NOT NULL,

`positionID` int(10) NOT NULL,

PRIMARY KEY (`id`,`positionID`),

KEY `positionID` (`positionID`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

**Login table**

CREATE TABLE IF NOT EXISTS `login` (

`id` varchar(20) NOT NULL,

`password` varchar(10) NOT NULL,

`positionID` int(10) NOT NULL,

PRIMARY KEY (`id`,`positionID`),

KEY `positionID` (`positionID`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

**Login table**

CREATE TABLE IF NOT EXISTS `proposal` (

`proposalID` varchar(35) NOT NULL,`abstract` text NOT NULL,

`submissionDate` date NOT NULL, `statusSubmission` varchar(20) NOT NULL,

`statusProject` varchar(20) NOT NULL,`remarks` text NOT NULL,

`mark` double NOT NULL,`sourceP` varchar(100) NOT NULL,

`type` varchar(20) NOT NULL,`reportID` varchar(35) NOT NULL,

`reportSubmissionID` int(225) NOT NULL,PRIMARY KEY (`proposalID`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;

**Proposal table**

CREATE TABLE IF NOT EXISTS `report` (

`reportID` varchar(35) NOT NULL,

`title` varchar(60) NOT NULL,

`typeOfWorkshop` int(10) NOT NULL,

`matricNum` varchar(20) NOT NULL,

PRIMARY KEY (`reportID`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

**Report table**

CREATE TABLE IF NOT EXISTS `reportsubmission` (

`reportSubmissionID` int(255) NOT NULL AUTO\_INCREMENT,

`dueSubmission` date NOT NULL,

`reportType` varchar(20) NOT NULL,

`typeOfWorkshop` int(10) NOT NULL,

PRIMARY KEY (`reportSubmissionID`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=5 ;

**Reportsubmission table**

CREATE TABLE IF NOT EXISTS `student` (

`matricNum` varchar(20) NOT NULL,`studentName` varchar(50) NOT NULL,

`studentEmail` varchar(50) NOT NULL,`studentPhoneNum` varchar(15) NOT NULL,

`course` varchar(30) NOT NULL,`year` int(10) NOT NULL,

`semester` int(10) NOT NULL,`typeOfWorkshop` int(10) NOT NULL,

`lecturerID` varchar(20) NOT NULL,`statusAssign` varchar(30) NOT NULL,

`positionID` int(10) NOT NULL,PRIMARY KEY (`matricNum`))

ENGINE=InnoDB DEFAULT CHARSET=latin1;

**Student table**

CREATE TABLE IF NOT EXISTS `workshopregistration` (

`id` int(225) NOT NULL AUTO\_INCREMENT,

`matricNum` varchar(20) NOT NULL,

`typeOfWorkshop` int(10) NOT NULL,

`session` varchar(12) NOT NULL,

`semester` int(10) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=44 ;

**Workshopregistration table**

**ATTACHMENT 1.4**

**APPENDIX D**

**ATTACHMENT 1.5**

**APPENDIX E**

CREATE DEFINER=`root`@`localhost` PROCEDURE `delete\_appointment`(IN `v\_logbookID` VARCHAR(35))

BEGIN

delete from appointment where id = v\_logbookID;

END$$

**delete\_appointment procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `delete\_finalreport`(IN `v\_finalreportID` VARCHAR(35))

BEGIN

delete from finalreport where finalreportID = v\_finalreportID ;

END$$

**delete\_finalreport procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `delete\_logbook`(IN `v\_logbookID` VARCHAR(35))

BEGIN

delete from logbook where logbookID = v\_logbookID;

END$$

**delete\_logbook procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `delete\_proposal`(IN `v\_proposalID` VARCHAR(35))

BEGIN

delete from proposal where proposalID = v\_proposalID ;

END$$

**delete\_proposal procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `importData\_student`(IN `v\_matricNum` VARCHAR(20), IN `v\_studentName` VARCHAR(50), IN `v\_studentEmail` VARCHAR(50), IN `v\_studentPhoneNum` VARCHAR(15), IN `v\_course` VARCHAR(30), IN `v\_year` INT(10), IN `v\_semester` INT(10), IN `v\_typeOfWorkshop` INT(10), `v\_lecturerID` VARCHAR(20), IN `v\_statusAssign` VARCHAR(30), IN `v\_positionID` INT(10), IN `v\_password` VARCHAR(10), IN `v\_session` VARCHAR(12))

BEGIN

INSERT INTO login (id, password, positionID)values(v\_matricNum, v\_password, v\_positionID);

INSERT INTO student (matricNum, studentName, studentEmail, studentPhoneNum, course, year, semester, typeOfWorkshop, lecturerID, statusAssign, positionID) values (v\_matricNum, v\_studentName, v\_studentEmail, v\_studentPhoneNum,v\_course,v\_year,v\_semester,v\_typeOfWorkshop, v\_lecturerID,v\_statusAssign,v\_positionID);

INSERT INTO workshopregistration (matricNum, typeOfWorkshop, session, semester)values(v\_matricNum, v\_typeOfWorkshop, v\_session, v\_semester);

END$$

**importData\_student procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert\_appointment`(IN `v\_title` VARCHAR(50), IN `v\_dateCreated` TIMESTAMP, IN `v\_startTime` TIME, IN `v\_endTime` TIME, IN `v\_start` DATETIME, IN `v\_end` DATETIME, IN `v\_color` VARCHAR(7), IN `v\_status` VARCHAR(50), IN `v\_matricNum` VARCHAR(20), IN `v\_staffID` VARCHAR(20))

NO SQL

INSERT INTO appointment(title, date\_created, start\_time,end\_time,start, end, color,status,staffID,matricNum)

values(v\_title,v\_dateCreated, v\_startTime, v\_endTime, v\_start, v\_end, v\_color, v\_status, v\_staffID, v\_matricNum)$$

**Insert\_appointment procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert\_finalReport`(IN `v\_reportID` VARCHAR(35), IN `v\_finalreportID` VARCHAR(35), IN `v\_submissionDate` DATE, IN `v\_statusProject` VARCHAR(20), IN `v\_sourceFR` VARCHAR(100), IN `v\_type` VARCHAR(20))

BEGIN

insert into finalReport (finalreportID, submissionDate, statusProject, sourceFR, type, reportID)values

(v\_finalreportID, v\_submissionDate, v\_statusProject, v\_sourceFR, v\_type,v\_reportID);

END$$

**Insert\_finalReport procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert\_logbook`(IN `v\_reportID` VARCHAR(35), IN `v\_logbookID` VARCHAR(35), IN `v\_submissionDate` DATE, IN `v\_activityDescription` TEXT)

BEGIN

insert into logbook (logbookID, submissionDate, activityDescription, reportID)values(v\_logbookID, v\_submissionDate, v\_activityDescription,v\_reportID);

END$$

**Insert\_logbook procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert\_proposal`(IN `v\_reportID` VARCHAR(35), IN `v\_title` VARCHAR(60), IN `v\_typeOfWorkshop` INT(10), IN `v\_matricNum` VARCHAR(20), IN `v\_proposalID` VARCHAR(35), IN `v\_abstract` TEXT, IN `v\_submissionDate` DATE, IN `v\_statusProject` VARCHAR(20), IN `v\_sourceP` VARCHAR(100), IN `v\_type` VARCHAR(20))

BEGIN

insert into report (reportID, title, typeOfWorkshop, matricNum)values (v\_reportID, v\_title, v\_typeOfWorkshop, v\_matricNum);

insert into proposal (proposalID, abstract,submissionDate, statusProject, sourceP, type, reportID)values(v\_proposalID, v\_abstract, v\_submissionDate, v\_statusProject, v\_sourceP, v\_type,v\_reportID);

END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert\_proposalAfterUpdate`(IN `v\_proposalID` VARCHAR(35), IN `v\_abstract` TEXT, IN `v\_submissionDate` DATE, IN `v\_statusProject` VARCHAR(20), IN `v\_sourceP` VARCHAR(100), IN `v\_type` VARCHAR(20), IN `v\_reportID` VARCHAR(35))

NO SQL

insert into proposal (proposalID, abstract,submissionDate, statusProject, sourceP, type, reportID)values(v\_proposalID, v\_abstract, v\_submissionDate, v\_statusProject, v\_sourceP, v\_type,v\_reportID)$$

**insert\_proposal procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert\_reportSubmission`(IN `v\_dueSubmission` DATE, IN `v\_reportType` VARCHAR(10), IN `v\_typeOfWorkshop` VARCHAR(20))

NO SQL

insert into reportSubmission

(dueSubmission, reportType, typeOfWorkshop)

values

(v\_dueSubmission, v\_reportType, v\_typeOfWorkshop)$$

**Insert\_reportSubmission procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_appointmentStatus`(IN `v\_color` VARCHAR(7), IN `v\_status` VARCHAR(30), IN `v\_location` VARCHAR(50), IN `v\_id` INT(225))

NO SQL

UPDATE appointment SET

color = v\_color,

status=v\_status,

location = v\_location

WHERE id = v\_id$$

**update\_appointmentStatus procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_assignMarkFR`(IN `v\_mark` DOUBLE, IN `v\_finalReportID` VARCHAR(15))

NO SQL

UPDATE finalreport SET

mark=v\_mark

WHERE finalReportID =v\_finalReportID$$

**update\_assignMarkFR procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_assignMarkP`(IN `v\_mark` DOUBLE, IN `v\_proposalID` VARCHAR(35))

NO SQL

UPDATE proposal SET

mark=v\_mark

WHERE proposalID =v\_proposalID$$

**update\_assignMarkP procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_AssignPositionC`(IN `v\_staffID` VARCHAR(20), IN `v\_positionID` INT(10), IN `assignWorkshop` VARCHAR(10))

BEGIN

update lecturer set positionID = v\_positionID where staffID = v\_staffID;

update lecturer\_position set positionID = v\_positionID , assignationWorkshop = assignWorkshop where staffID = v\_staffID;

update login set positionID = v\_positionID where id = v\_staffID;

END$$

**update\_AssignPositionC procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_assignSupervisor`(IN `v\_lecturerID` VARCHAR(20), IN `v\_statusAssign` VARCHAR(30), IN `v\_matricNum` VARCHAR(20))

NO SQL

update student

set lecturerID = v\_lecturerID ,

statusAssign=v\_statusAssign

where matricNum=v\_matricNum$$

**update\_assignSupervisor procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_changePass`(IN `v\_id` VARCHAR(20), IN `v\_pass` VARCHAR(10))

NO SQL

update login left join lecturer

ON login.id = lecturer.staffID

set password= v\_pass

where id= v\_id$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_changePassword`(IN `v\_id` VARCHAR(20), IN `v\_pass` VARCHAR(10))

NO SQL

update login

set password=v\_pass

where id= v\_id$$

**update\_changePassword procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_finalReport`(IN `v\_statusProject` VARCHAR(20), IN `v\_mark` DOUBLE, IN `v\_remarks` TEXT, IN `v\_finalReportID` VARCHAR(15))

NO SQL

UPDATE finalreport SET

statusProject=v\_statusProject,

mark = v\_mark,

remarks=v\_remarks

WHERE finalreportID = v\_finalReportID$$

**update\_finalReport procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_logbook`(IN `v\_logbookID` VARCHAR(35), IN `v\_submissionDate` DATE, IN `v\_activityDescription` TEXT)

NO SQL

update logbook

set

activityDescription=v\_activityDescription,

submissionDate=v\_submissionDate

where logbookID=v\_logbookID$$

**update\_logbook procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_profileLecturer`(IN `v\_id` VARCHAR(20), IN `v\_email` VARCHAR(50), IN `v\_phoneNum` VARCHAR(15), IN `v\_bio` VARCHAR(100))

BEGIN

update lecturer

set

lecturerEmail= v\_email,

lecturerPhoneNum= v\_phoneNum,

bio = v\_bio

where staffID= v\_id ;

END$$

**update\_profileLecturer procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_profileStudent`(IN `v\_id` VARCHAR(20), IN `v\_email` VARCHAR(50), IN `v\_phoneNum` VARCHAR(15))

BEGIN

update student set

studentEmail= v\_email,

studentPhoneNum= v\_phoneNum

where matricNum = v\_id;

END$$

**update\_profileStudent procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_proposal`(IN `v\_statusProject` VARCHAR(20), IN `v\_mark` DOUBLE, IN `v\_remarks` TEXT, IN `v\_proposalID` VARCHAR(35))

NO SQL

UPDATE proposal SET

statusProject=v\_statusProject,

mark = v\_mark,

remarks= v\_remarks

WHERE proposalID =v\_proposalID$$

**update\_proposal procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_report`(IN `v\_title` VARCHAR(60), IN `v\_matricNum` VARCHAR(20))

NO SQL

UPDATE report SET

title = v\_title

WHERE matricNum = v\_matricNum$$

**update\_report procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_reportSubmission`(IN `v\_dueSubmission` DATE, IN `v\_typeOfWorkshop` INT(10), IN `v\_reportType` VARCHAR(20))

NO SQL

UPDATE reportSubmission SET

dueSubmission = v\_dueSubmission,

typeOfWorkshop = v\_typeOfWorkshop

WHERE reportType = v\_reportType AND typeOfWorkshop = v\_typeOfWorkshop$$

**update\_reportSubmission procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_appoint`(IN `v\_matricNum` VARCHAR(20))

NO SQL

select \* from appointment where matricNum= v\_matricNum ORDER BY start DESC$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_appointment`(IN `v\_matricNum` VARCHAR(20), IN `v\_staffID` VARCHAR(20))

NO SQL

SELECT \* FROM appointment

LEFT JOIN student ON appointment.matricNum = student.matricNum

where appointment.matricNum=v\_matricNum

or appointment.staffid= v\_staffID$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_appointment2`(IN `v\_staffID` VARCHAR(20))

NO SQL

SELECT \* FROM appointment

where appointment.staffid= v\_staffID$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_appointmentSupervisor`(IN `v\_id` VARCHAR(20))

NO SQL

SELECT \* FROM appointment LEFT JOIN student ON appointment.matricNum = student.matricNum where appointment.matricNum=v\_id or appointment.staffid= v\_id$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_appointSupervisor`(IN `v\_staffID` VARCHAR(20))

NO SQL

select \* from appointment left join student on appointment.matricNum = student.matricNum where staffID= v\_staffID ORDER BY start DESC$$

**select\_appointment procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_assignStudent`(IN `v\_staffID` VARCHAR(20))

NO SQL

select \* from student where lecturerID=v\_staffID$$

**select\_assignStudent procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_changePass`(IN `v\_matricNum` VARCHAR(20))

NO SQL

select \* from login left join student ON login.id = student.matricNum where id= v\_matricNum$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_changePassL`(IN `v\_matricNum` VARCHAR(20))

NO SQL

select \* from login left join lecturer ON login.id = lecturer.staffID where id= v\_matricNum$$

**select\_changePassword procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_count`(IN `v\_matricNum` VARCHAR(20))

NO SQL

select count(\*) as totalAppointment from appointment where matricNum= v\_matricNum ORDER BY start DESC$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_count1`(IN `v\_staffID` VARCHAR(20))

NO SQL

select count(\*) as totalAppointment from appointment where staffID= v\_staffID ORDER BY start DESC$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_countFinalReport`()

NO SQL

select COUNT(\*) as totalRows from finalreport$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_countLogbook`()

NO SQL

select COUNT(\*) as totalRows from logbook$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_countProposal`()

NO SQL

select COUNT(\*) as totalRows from proposal$$

**select\_count procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_finalReport`(IN `v\_matricNum` VARCHAR(20))

NO SQL

select \* from report left join finalReport ON report.reportID = finalReport.reportID where matricNum= v\_matricNum ORDER BY finalreportID DESC$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_finalReport2`(IN `v\_matricNum` VARCHAR(20))

NO SQL

select \* from finalReport left join report ON finalReport.reportID = report.reportID where matricNum= v\_matricNum ORDER BY finalreportID desc$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_finalReport3`(IN `v\_matricNum` VARCHAR(20))

NO SQL

select \* from report left join finalReport ON report.reportID = finalReport.reportID where matricNum= v\_matricNum$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_finalReport4`(IN `v\_matricNum` VARCHAR(20))

NO SQL

select \* from report left join student ON report.matricNum = student.matricNum where report.matricNum =v\_matricNum$$

**select\_finalReport procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_lecturer`(IN `v\_staffID` VARCHAR(20))

NO SQL

select \* from lecturer inner join lecturer\_position ON

lecturer.staffID = lecturer\_position.staffID

inner join position ON lecturer\_position.positionID = position.positionID where lecturer.staffID=v\_staffID$$

**select\_lecturer procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_lecturerID`(IN `v\_id` VARCHAR(20))

NO SQL

select \* from login left join student ON login.id = student.matricNum

left join lecturer ON student.lecturerID = lecturer.staffID

where id= v\_id$$

**select\_lecturerID procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_lecturerID\_appointment`(IN `v\_staffID` VARCHAR(20))

NO SQL

select \* from login left join lecturer ON login.id = lecturer.staffID where id= v\_staffID$$

**select\_lecturerID\_appointment procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_logbook`(IN `v\_matricNum` VARCHAR(20))

NO SQL

select \* from report left join logbook ON report.reportID = logbook.reportID

where matricNum= v\_matricNum$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_logbook2`(IN `v\_matricNum` VARCHAR(20))

NO SQL

select \* from report left join student ON report.matricNum = student.matricNum where report.matricNum = v\_matricNum$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_logbook3`(IN `v\_matricNum` VARCHAR(20))

NO SQL

select \* from logbook left join report ON logbook.reportID = report.reportID

where matricNum= v\_matricNum ORDER BY logbookID desc$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_logbook4`(IN `v\_logbookID` VARCHAR(35))

NO SQL

select \* from report left join logbook ON report.reportID = logbook.reportID where logbookID=v\_logbookID$$

**select\_logbook procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_login`(IN `v\_id` VARCHAR(20), IN `v\_pass` VARCHAR(10), IN `v\_positionID` INT(10))

NO SQL

BEGIN

select \* from login where

id = v\_id and password =v\_pass and

positionID = v\_positionID;

END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_loginA`(IN `v\_id` VARCHAR(20), IN `v\_pass` VARCHAR(10), IN `v\_positionID` INT(10))

NO SQL

BEGIN

select \* from login where

id = v\_id and password =v\_pass and

positionID = v\_positionID;

END$$

**select\_login procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_maxCountFinalReport`()

NO SQL

select max(finalreportID) as maxNumber from finalreport$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_maxCountLogbook`()

NO SQL

select max(logbookID) as maxNumber from logbook$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_maxCountProposal`()

NO SQL

select max(proposalID) as maxNumber from proposal$$

**select\_maxCount procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_previousStudent`(IN `v\_sesi` VARCHAR(12), IN `v\_sem` INT(10), IN `v\_staffID` VARCHAR(20))

NO SQL

select \* from workshopregistration left join student on workshopregistration.matricNum = student.matricNum left join report on workshopregistration.matricNum=report.matricNum where workshopregistration.session=v\_sesi and workshopregistration.semester=v\_sem and student.lecturerID=v\_staffID and report.matricNum = student.matricNum$$

**select\_previousStudent procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_profileLecturer`(IN `v\_staffID` VARCHAR(20))

NO SQL

select \* from lecturer where staffID= v\_staffID$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_profileStudent`(IN `v\_matricNum` VARCHAR(20))

NO SQL

select \* from student left join lecturer ON student.lecturerID = lecturer.staffID where matricNum= v\_matricNum$$

**select\_profile procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_proposal`(IN `v\_matricNum` VARCHAR(20))

NO SQL

select \* from report left join proposal ON report.reportID = proposal.reportID

where matricNum= v\_matricNum ORDER BY proposalID DESC$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_proposal2`(IN `v\_matricNum` VARCHAR(20))

NO SQL

select \* from report left join student ON report.matricNum = student.matricNum where report.matricNum =v\_matricNum$$

**select\_proposal procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_reportSubmission`(IN `v\_type` VARCHAR(20), IN `v\_workshop` INT(10))

NO SQL

select \* from reportSubmission where reportType =v\_type and typeOfWorkshop =v\_workshop$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_reportSubmissionFR`()

NO SQL

select \* from reportSubmission where reportType = 'Final Report' ORDER BY typeOfWorKShop ASC$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_reportSubmissionP`()

NO SQL

select \* from reportSubmission where reportType = 'Proposal' ORDER BY typeOfWorKShop ASC$$

**select\_reportSubmission procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_resetPassL`(IN `v\_staffID` VARCHAR(20))

NO SQL

select \* from lecturer left join login ON lecturer.staffID = login.id where staffID =v\_staffID$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_resetPassS`(IN `v\_matricNum` VARCHAR(20))

NO SQL

select \* from student left join login ON student.matricNum = login.id where matricNum =v\_matricNum$$

**select\_resetPassword procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_session`()

NO SQL

SELECT session FROM workshopregistration group by session$$

**select\_session procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_student`(IN `v\_matricNum` VARCHAR(20))

NO SQL

select \* from student where matricNum =v\_matricNum$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_studentW1`(IN `v\_matricNum` VARCHAR(20))

NO SQL

select student.matricNum, studentName, title, course, student.typeOfWorkshop from student left join report ON student.matricNum = report.matricNum where lecturerID=v\_matricNum AND student.typeOfWorkshop=1$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_studentW2`(IN `v\_matricNum` VARCHAR(20))

NO SQL

select student.matricNum, studentName, title, course, student.typeOfWorkshop from student left join report ON student.matricNum = report.matricNum where lecturerID=v\_matricNum AND student.typeOfWorkshop=2$$

**select\_student procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_totalAppointment`(IN `v\_id` VARCHAR(20))

NO SQL

SELECT SUM(if(appointment.status = 'accept', 1, 0)) AS totalaccept,

SUM(if(appointment.status = 'Reject', 1, 0)) AS totalreject,

SUM(if(appointment.status = 'Pending Approval', 1, 0)) AS totalpending

FROM appointment

LEFT JOIN student ON appointment.matricNum = student.matricNum

where appointment.matricNum=v\_id or appointment.staffid= v\_id$$

**select\_totalAppointment procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_studentInfo`(IN `v\_c1` VARCHAR(30), IN `v\_c2` VARCHAR(30), IN `v\_c3` VARCHAR(30), IN `v\_c4` VARCHAR(30), IN `v\_c5` VARCHAR(30), IN `v\_c6` VARCHAR(30), IN `v\_c7` VARCHAR(30))

NO SQL

select \* from student where(course = v\_c1 OR course = v\_c2 OR course = v\_c3 OR course = v\_c4 OR course = v\_c5 OR course = v\_c6 OR course = v\_c7)ORDER BY course ASC, matricNum ASC$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_studentInfo2`()

NO SQL

select \* from student where (course = 'BITS' OR course = 'BITM' OR course = 'BITE' OR course = 'BITD' OR course = 'BITC' OR course = 'BITZ' OR course = 'BITI') ORDER BY course ASC, matricNum ASC$$

**select\_studentInfo procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_lecturerInfo`(IN `v\_dept1` VARCHAR(30), IN `v\_dept2` VARCHAR(30), IN `v\_dept3` VARCHAR(30), IN `v\_dept4` VARCHAR(30))

NO SQL

select \* from lecturer where(department = v\_dept1 OR department = v\_dept2 OR department = v\_dept3 OR department = v\_dept4 ) ORDER BY department DESC, staffID ASC $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_lecturerInfo2`()

NO SQL

select \* from lecturer where (department = 'SE' OR department = 'BITC' OR department = 'MI' OR department = 'BITI' ) ORDER BY department DESC, staffID ASC$$

**select\_lecturerInfo procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_searchSupervisor`(IN `v\_dept1` VARCHAR(30), IN `v\_dept2` VARCHAR(30), IN `v\_dept3` VARCHAR(30), IN `v\_dept4` VARCHAR(30))

NO SQL

select \*,count(student.lecturerID) as totalSupervisee from lecturer

left join lecturer\_position on lecturer.staffID = lecturer\_position.staffID

left join position on lecturer.positionID = position.positionID

left join student on student.lecturerID = lecturer.staffID where

(department = v\_dept1 OR department = v\_dept2 OR department = v\_dept3 OR department = v\_dept4 ) AND

(lecturer.positionID = '2' OR lecturer.positionID = '3' )

GROUP BY lecturer.staffID ORDER BY lecturer.department DESC, lecturer.staffID ASC$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_searchSupervisor2`(IN `v\_position1` INT(10), IN `v\_position2` INT(10))

NO SQL

select \*,count(student.lecturerID) as totalSupervisee from lecturer

left join lecturer\_position on lecturer.staffID = lecturer\_position.staffID

left join position on lecturer.positionID = position.positionID

left join student on student.lecturerID = lecturer.staffID

where

(department = 'SE' OR department = 'BITC' OR department = 'MI' OR department = 'BITI' ) AND

(lecturer.positionID = v\_position1 OR lecturer.positionID = v\_position2 )

GROUP BY lecturer.staffID ORDER BY lecturer.department DESC, lecturer.staffID ASC$$

**select\_searchSupervisor procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_searchSupervisor3`(IN `v\_dept1` VARCHAR(30), IN `v\_dept2` VARCHAR(30), IN `v\_dept3` VARCHAR(30), IN `v\_dept4` VARCHAR(30), IN `v\_position1` INT(10), IN `v\_position2` INT(10))

NO SQL

select \*,count(student.lecturerID) as totalSupervisee from lecturer

left join lecturer\_position on lecturer.staffID = lecturer\_position.staffID

left join position on lecturer.positionID = position.positionID

left join student on student.lecturerID = lecturer.staffID

where

(department = v\_dept1 OR department = v\_dept2 OR department = v\_dept3 OR department = v\_dept4 ) AND

(lecturer.positionID = v\_position1 OR lecturer.positionID = v\_position2 )

GROUP BY lecturer.staffID ORDER BY lecturer.department DESC, lecturer.staffID ASC$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_searchSupervisor4`()

NO SQL

select \*,count(student.lecturerID) as totalSupervisee from lecturer

left join lecturer\_position on lecturer.staffID = lecturer\_position.staffID

left join position on lecturer.positionID = position.positionID

left join student on student.lecturerID = lecturer.staffID

where (department = 'SE' OR department = 'BITC' OR department = 'BITI' OR department = 'MI' )

AND (lecturer.positionID = '2' OR lecturer.positionID = '3' )

GROUP BY lecturer.staffID ORDER BY lecturer.department DESC, lecturer.staffID ASC$$

**select\_searchSupervisor procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_searchCommittee`(IN `v\_dept1` VARCHAR(30), IN `v\_dept2` VARCHAR(30), IN `v\_dept3` VARCHAR(30), IN `v\_dept4` VARCHAR(30))

NO SQL

select \* from lecturer

left join lecturer\_position on lecturer.staffID = lecturer\_position.staffID

left join position on lecturer.positionID = position.positionID

where

(department = v\_dept1 OR department = v\_dept2 OR department = v\_dept3 OR department = v\_dept4 ) AND

(lecturer.positionID = '2' OR lecturer.positionID = '3' )ORDER BY positionName ASC,lecturer.department DESC, lecturer.staffID ASC$$

**select\_searchCommittee procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_searchCommittee2`(IN `v\_position1` INT(10), IN `v\_position2` INT(10))

NO SQL

select \* from lecturer

left join lecturer\_position on lecturer.staffID = lecturer\_position.staffID

left join position on lecturer.positionID = position.positionID

where

(department = 'SE' OR department = 'BITC' OR department = 'MI' OR department = 'BITI' ) AND

(lecturer.positionID = v\_position1 OR lecturer.positionID = v\_position2 )ORDER BY positionName ASC,lecturer.department DESC, lecturer.staffID ASC$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_searchCommittee3`(IN `v\_dept1` VARCHAR(30), IN `v\_dept2` VARCHAR(30), IN `v\_dept3` VARCHAR(30), IN `v\_dept4` VARCHAR(30), IN `v\_position1` INT(10), IN `v\_position2` INT(10))

NO SQL

select \* from lecturer

left join lecturer\_position on lecturer.staffID = lecturer\_position.staffID

left join position on lecturer.positionID = position.positionID

where

(department = v\_dept1 OR department = v\_dept2 OR department = v\_dept3 OR department = v\_dept4 ) AND

(lecturer.positionID = v\_position1 OR lecturer.positionID = v\_position2 )ORDER BY positionName ASC,lecturer.department DESC, lecturer.staffID ASC$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `select\_searchCommittee4`()

NO SQL

select \* from lecturer

left join lecturer\_position on lecturer.staffID = lecturer\_position.staffID

left join position on lecturer.positionID = position.positionID

where (department = 'SE' OR department = 'BITC' OR department = 'BITI' OR department = 'MI' )

AND (lecturer.positionID = '2' OR lecturer.positionID = '3' )ORDER BY positionName ASC,lecturer.department DESC, lecturer.staffID ASC$$

**select\_searchCommittee procedure**

**procedure**

**procedure**

**procedure**

**procedure**

**procedure**

**procedure**

**procedure**

**procedure**

**procedure**

**procedure**

**procedure**

**procedure**