

UTeM BUS SCHEDULING MANAGEMENT SYSTEM



HO JIA QING

UTeM

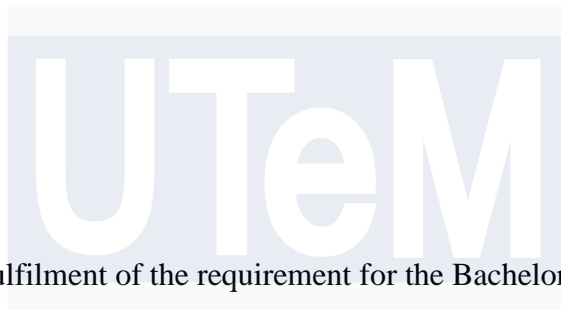
اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

UTeM BUS SCHEDULING MANAGEMENT SYSTEM

HO JIA QING



This report is submitted in partial fulfilment of the requirement for the Bachelor of Computer Science (Database Management) with Honours.

اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITY TEKNIKAL MALAYSIA MELAKA

2024

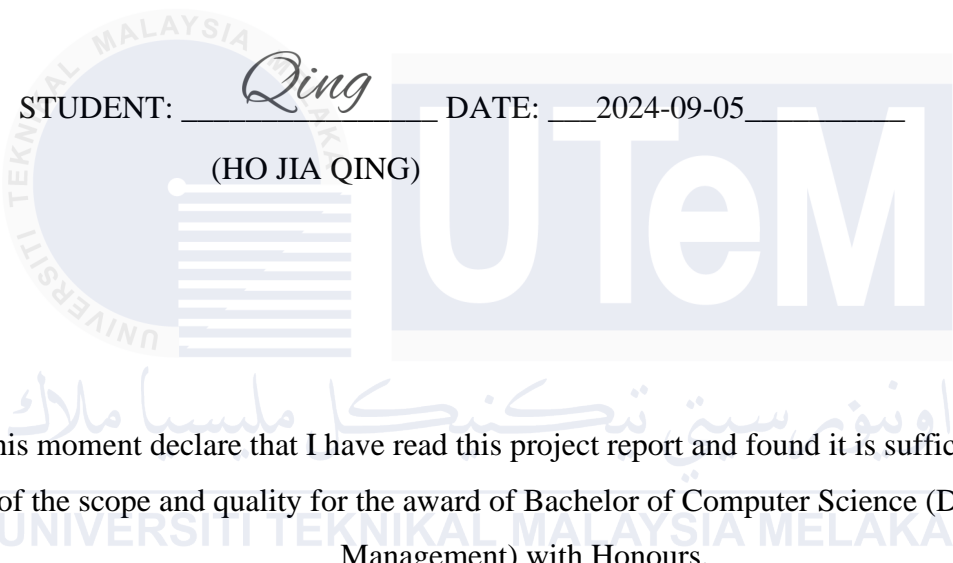
DECLARATION

I at this moment declare that this project report entitled.

UTeM BUS SCHEDULING MANAGEMENT SYSTEM

is written by me and is my effort and no part has been plagiarized without citations.

STUDENT: Qing DATE: 2024-09-05
(HO JIA QING)



I at this moment declare that I have read this project report and found it is sufficient in terms of the scope and quality for the award of Bachelor of Computer Science (Database Management) with Honours.

SUPERVISOR: Namrah DATE: 2024-09-05
(DR. NURUL AKMAR BINTI EMRAN)

DEDICATION

I would like to convey my special appreciation to my dear parents and supervisor who have been giving me direction and encouragement throughout my project. I would especially like to express my sincere gratitude to the following important mentors and collaborators.



ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to Dr. Nurul Akmar binti Emran for her invaluable assistance in successfully completing this project. Her expertise, advice, guidance, and encouragement were instrumental throughout this journey. Dr. Nurul Akmar consistently fostered a spirit of adventure in research and provided invaluable support and motivation.

Additionally, I extend my deepest thanks to my beloved parents and friends who have supported and motivated me every step of the way. Their unwavering encouragement has been a source of strength throughout this endeavour.

I am also deeply grateful to the following significant advisors and contributors for their invaluable insights and support.

Furthermore, I extend sincere appreciation to my institution, the Faculty of Information and Communications Technology at Universiti Teknikal Malaysia Melaka, lecturers, and friends for their guidance and teachings throughout this report. Without their assistance, this report would have remained a distant reality.

Thank you to everyone who has played a role in the success of this project.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

ABSTRACT

UTeM Bus Scheduling Management System is a web-based system that provides a platform to manage the bus scheduling process by viewing the bus schedule on the website. In the current system, all data is only represented by social media officials. There is no available system to manage the data. Moreover, the admin may face difficulty in updating the schedule. Updating the schedule is challenging because all the data is recorded manually and shared through social media. Additionally, manual recording makes it harder to keep the schedule up to date. Admins need to upload to social media again if there are some changes in the bus schedule. It makes students miss the updated bus schedule from social media. Therefore, this system is developed to increase the efficiency of the bus scheduling process by replacing the existing manual paper-based system. The user are able to view the bus schedules at the university website. It also allows users to keep the bus schedules as pictures to easy their life. In the same time, the system can minimize human errors by insert the bus schedule through website and reduce data redundancy by enforcing key constraints in database. This system convenience admin and driver to manage the bus schedule and user details. Furthermore, this system is using Database Life Cycle (DBLC) as methodology to develop because easy to manage due to the rigidity of the model each phase has specific deliverables and a review process. DBLC contains six important phases that is database initial study, database design, implementation & loading, training & Evaluation, operation, and maintenance & Evaluation. As a result, the management of bus scheduling process is more efficient and systematic with the development of UTeM Bus Scheduling Management System.

ABSTRACT

Sistem Pengurusan Penjadualan Bas UTeM adalah sistem berasaskan web yang menyediakan platform untuk menguruskan proses penjadualan bas dengan melihat jadual bas di laman web. Dalam sistem semasa, semua data hanya dikumpul oleh Admin . Tiada sistem yang tersedia untuk menguruskan data. Tambahan pula, mungkin menghadapi kesulitan dalam mengemaskini jadual. Mengemaskini jadual adalah mencabar kerana semua data direkod secara manual dan dikongsi melalui media sosial. Selain itu, pencatatan manual menjadikannya lebih sukar untuk mengekalkan jadual yang terkini. Pentadbir perlu mengemaskini semula ke media sosial jika terdapat perubahan dalam jadual bas. Ini menyebabkan pelajar terlepas jadual bas yang dikemaskini dari media sosial. Oleh itu, sistem ini dibangunkan untuk meningkatkan kecekapan proses penjadualan bas dengan menggantikan sistem manual berdasarkan kertas yang sedia ada. Pengguna boleh melihat jadual bas di laman web universiti. Ia juga membenarkan pengguna menyimpan jadual bas sebagai gambar untuk memudahkan kehidupan mereka. Pada masa yang sama, sistem ini dapat mengurangkan kesilapan manusia dengan memasukkan jadual bas melalui laman web dan mengurangkan kebarangkalian data dengan menguatkuasakan kekangan kunci dalam pangkalan data. Sistem ini memudahkan pentadbir dan pemandu menguruskan jadual bas dan butiran pengguna. Selain itu, sistem ini menggunakan Kitaran Hidup Pangkalan Data (DBLC) sebagai metodologi untuk pembangunan kerana mudah diuruskan kerana kekerasan model setiap fasa mempunyai hasil khusus dan proses ulasan. DBLC mengandungi enam fasa penting iaitu kajian awal pangkalan data, reka bentuk pangkalan data, pelaksanaan & pemuatan, latihan & Penilaian, operasi, dan penyelenggaraan & Penilaian. Sebagai hasilnya, pengurusan proses penjadualan bas menjadi lebih efisien dan sistematik dengan pembangunan Sistem Pengurusan Penjadualan Bas UTeM.

Table of Contents

1.INTRODUCTION	1
1.1Project Background.....	1
1.2 Problem Statements	2
1.3 Objective.....	2
1.4 Scope.....	3
1.4.1 Target User.....	3
4.1.3Modules.....	3
1.5 Project Significant.....	4
1.6 Expected Outcomes	4
1.7 Conclusion	4
2.PROJECT METHODOLOGY AND PLANNING	5
2.1 Introduction.....	5
2.2 Project Methodology.....	5
2.2.1 Database Initial Study	5
2.2.3 Implementation and Loading	6
2.2.4 Testing and Evaluation.....	6
2.2.5 Operation.....	7
2.2.6 Maintenance and Evolution	7
2.3 Project Schedule and Milestones	9
2.4 Conclusion	11
3.ANALYSIS.....	12
3.1 Introduction.....	12
3.2 Problem Analysis	13
3.3 The Proposed Improvement/Solutions.....	16
3.4 Requirement Analysis of the to-be-System	18
3.4.1 Functional Requirement.....	18
3.4.2 Non- Functional Requirements	23
3.4.3 Other Requirements	23

3.5 Conclusion	25
4.DESIGN.....	26
4.1 Introduction.....	26
4.2 System Architecture Design.....	26
4.3 Database Design.....	27
4.3.1 Conceptual Design	28
4.3.2 Logical Design	29
4.4 Graphical User Interface (GUI) Design	45
4.4.1 Navigation Design.....	45
4.4.2 Input Design.....	47
4.4.3 Output Design	47
4.5 Conclusion	47
5.IMPLEMENTATION.....	48
5.1 Introduction.....	48
5.2 System Development Environment Setup	48
5.2.1 Steps of Installation Setup.....	49
5.3 Database Implementation.....	56
5.3.1 Data Definition Language (DDL)	56
5.3.2 Data Manipulation Language.....	64
5.3.3 Stored Procedures	64
5.3.4 Triggers	67
5.3.5 Data Loading Process.....	68
5.4 Implementation Status	68
5.5 Conclusion	69
6. Testing.....	70
6.1 Introduction.....	70
6.2 Test Plan.....	70
6.2.1 Test Organization.....	71
6.2.2 Test Environment.....	72

6.2.3 Test Schedule	73
6.3 Test Strategy	75
6.3.1 Classes of Tests.....	78
6.4 Test Design	78
6.4.1 Test Description	78
6.4.3 Test Data and Test Result	90
6.5 Test Result and Analysis.....	93
6.6 Conclusion	99
7. CONCLUSION.....	100
7.1 Introduction.....	100
7.2 Observation Weakness and Strengths	100
7.2.1 Strengths	101
7.2.2 Weakness	101
7.3 Proposition of Improvement	102
7.4 Contribution	103
7.5 Conclusion	103
References.....	104
Appendix.....	105

LIST OF TABLES

Table 2.1 : Project Milestones	10
Table 3.1: Non-Functional Requirements.....	23
Table 3.2 : Software Requirements.....	24
Table 3.3 : Hardware Requirements	25
Table 4.1 : Table User.....	30
Table 4.2 : Table Driver.....	30
Table 4.3 : Table External Bus	31
Table 4.4 : Table Checkin.....	31
Table 4.5 : Table Bus	32
Table 4.6 : Table Semester.....	32
Table 4.7 : Table Schedule.....	33
Table 4.8 : Table Route.....	33
Table 4.9 : Table Announcement.....	34
Table 4.10 : Table Stop.....	34
Table 4.11: Table Stop_Schedule	35
Table 4.12 : Table Seat	35
Table 4.13 : Departure Date Table.....	36
Table 4.14 : Booking Table	36
Table 4.15 : Query Design of UTeM Bus Scheduling Management System	41
Table 4.16: Triggers Relates Database Object Detail.....	42
Table 4.17 : Stored Procedure Relates Database Object Detail.....	43
Table 5.1: Stored Procedures Query	64
Table 5.2: Triggers Query.....	67
Table 6.1 : User Responsibilities List.....	71
Table 6.2 : Test Environment.....	72
Table 6.3 : Test Environment Software List.....	73
Table 6.4 : Test Schedule.....	74
Table 6.5 : Type of test and test design techniques for white box and black box testing.....	76
Table 6.6 : Test Description for User Login (student).....	79
Table 6.7 : Test Description of Registration for Driver Account	80
Table 6.8 : Test Description of Add Bus for Driver	81
Table 6.9 : Test Description of User Content	82
Table 6.10 : Test Description of Bus Route.....	83

Table 6.11 : Test Description of Bus Schedule.....	84
Table 6.12 : Test Description of Manage Stop	85
Table 6.13 : Test Description of Announcement	86
Table 6.14 : Test Description of External Buses Registration.....	87
Table 6.15 : Test Description of Searching Schedule.....	87
Table 6.16 : Test Description of Booking Seats	88
Table 6.17 : Test Description for Driver Check-in	89
Table 6.18 : Test Description of Driver Check-out	89
Table 6.19: Test Data of User Login	90
Table 6.20 : Test Data of Registration for Driver Account	90
Table 6.21 : Test Data of Add Bus for Driver	90
Table 6.22 : Test Data of User Content	90
Table 6.23 : Test Data of Bus Route.....	91
Table 6.24 : Test Data of Bus Schedule.....	91
Table 6.25 : Test Data of Manage Stop	91
Table 6.26 : Test Data of Announcement	91
Table 6.27 : Test Data of External Buses Registration.....	92
Table 6.28 : Test Data of Searching Schedule.....	92
Table 6.29 : Test Data of Booking Seats	92
Table 6.30 : Test Data of Driver Check-in	92
Table 6.31 : Test Data of Driver Check-out	92
Table 6.32 : Test Result of User Login.....	93
Table 6.33 : Test Result of Registration for Driver Account.....	93
Table 6.34 : Test Result of Add Bus for Driver.....	94
Table 6.35 : Test Result of User Content.....	94
Table 6.36 : Test Result of Bus Route	95
Table 6.37 : Test Result of Bus Schedule	95
Table 6.38 : Test Result of Manage Stop.....	96
Table 6.39 : Test Result of Announcement	96
Table 6.40 : Test Result of External Buses Registration	97
Table 6.41 : Test Result of Searching Schedule	97
Table 6.42 : Test Result of Booking Seats.....	98
Table 6.43 : Test Result of Driver Check-in.....	98
Table 6.44 : Test Result of Driver Check-out.....	99

LIST OF FIGURES

Figure 2.1 Database life cycle illustration	8
Figure 2.2 : Gantt Chart of UTeM Bus Scheduling Management System	9
Figure 3.1: Current UTeM Bus Scheduling Process Flow Chart.....	14
Figure 3.2: UTeM Bus Scheduling Management System Flow Chart	16
Figure 3.3: Context Diagram	19
Figure 3.4: Data Flow Diagram (DFD) Level 1	20
Figure 3.5 : Data Flow Diagram (DFD) Level 2 (Check Bus Schedule).....	21
Figure 3.6 : Data Flow Diagram (DFD) Level 2 (Check Bus Tracking).....	21
Figure 3.7 : Data Flow Diagram (DFD) Level 2 (Manage Bus Ticket)	22
Figure 3.8 : Data Flow Diagram (DFD) Level 2 (Handle Announcement).....	22
Figure 4.1 : 3 Tier Architecture	27
Figure 4.2: ERD of UTeM Bus Scheduling Management System	28
Figure 4.3 : 3 NF of User Table.....	37
Figure 4.4 : 3 NF of Driver Table	37
Figure 4.5 : 3 NF of External Bus Table.....	37
Figure 4.6: 3 NF of Checkin Table	38
Figure 4.7 : 3 NF of Bus Table	38
Figure 4.8: 3 NF of Semester Table.....	38
Figure 4.9 : 3 NF of Schedule Table.....	39
Figure 4.10: 3 NF of Route table	39
Figure 4.11: 3 NF of Announcement Table	39
Figure 4.12 : 3 NF of Stop Table	40
Figure 4.13: 3 NF of Stop_Schedule Table	40
Figure 4.14: Navigation Path of UTeM BUS SCHEDULING MANAGEMENT SYSTEM .	46
Figure 5.1 : Download MySQL installer for Windows	49
Figure 5.2 : MySQL installer	49
Figure 5.3 : Select Custom Setup Type	50
Figure 5.4 : Select MySQL Products	50
Figure 5.5 : Server Configuration Type.....	51
Figure 5.6 : MySQL Settings	52
Figure 5.7 : Apply Configuration.....	52

Figure 5.8 : MySQL workbench interface	53
Figure 5.9 : Enter the root password.....	53
Figure 5.10 : Enter MySQL workbench main page.....	54
Figure 5.11 : Create database schema.....	54
Figure 5.12 : Add New Table into Database.....	55
Figure 5.13 : Add Attribute into Table	55
Figure 5.14 : Create Table Announcement.....	56
Figure 5.15 : Create Table Booking.....	57
Figure 5.16 : Create Table Bus	57
Figure 5.17 : Create Table Checkin	58
Figure 5.18 : Create Table Departure_Date.....	58
Figure 5.19 : Create Table Driver	59
Figure 5.20 : Create Table External_Bus.....	59
Figure 5.21 : Create Table GPS_Data.....	60
Figure 5.22 : Create Table Route.....	60
Figure 5.23 : Create Table Schedule.....	61
Figure 5.24 : Create Table Seat.....	61
Figure 5.25 : Create Table Semester.....	62
Figure 5.26 : Create Table Stop_Schedule	62
Figure 5.27 : Create Table Stop.....	63
Figure 5.28 : Create Table User.....	63
Figure 5.29 : Insert Statement – insert data into table semester	64
Figure 5.30 : Update Statement – update table bus	64
Figure 5.31 : Delete Statement – delete table driver.....	64
Figure 5.32 : Select Statement – retrieve table driver.....	64
Figure 6.1 : Level of Testing.....	77
Figure 8.1 : Login Page.....	105
Figure 8.2 : Add Driver.....	105
Figure 8.3 : Add Bus.....	106
Figure 8.4 : Add Schedule	106
Figure 8.5 : Add Route.....	107
Figure 8.6 : Add Stop.....	107
Figure 8.7 : Assign Schedule	108
Figure 8.8 : Add Announcement.....	108

Figure 8.9 : Admin Dashboard.....	109
Figure 8.10 : Driver Detail.....	109
Figure 8.11 : Driver Report.....	110
Figure 8.12 : View User.....	110
Figure 8.13 : View Schedule.....	111
Figure 8.14 : View Announcement.....	111
Figure 8.15 : View External Bus.....	112
Figure 8.16 : User View Schedule	112
Figure 8.17 : List All Stops.....	113
Figure 8.18 : Bus Seat Booking Page	113
Figure 8.19 : User View Ticket.....	114
Figure 8.20 : Driver Check-in / Check-out Page	114
Figure 8.21 : Driver Tracking	115
Figure 8.22 : Driver Attendance	115

اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

CHAPTER 1

INTRODUCTION

1.1 Project Background

In today's fast-paced world, where efficient access to education is paramount, many students heavily rely on public transportation to commute to and from their educational institutions. Among these transportation services, bus systems stand out as a cornerstone, providing students with a reliable and affordable means of travel.

Students at Universiti Teknikal Malaysia Melaka (UTeM) frequently rely on social media sites like Facebook and Instagram to find out bus timings. While these platforms serve as convenient channels for disseminating information, relying solely on them poses challenges for students who may not have access to or actively use social media applications. Consequently, students without such access risk missing important updates and announcements regarding bus schedules and routes provided by their educational institution.

To address this issue and ensure equitable access to essential transportation information, the introduction of a modern UTeM bus scheduling management system becomes imperative. This system is designed to centralize and streamline the distribution of bus schedule information through the official university website. By using this platform, students can conveniently access up-to-date bus schedules, route details, and any relevant announcements directly from a reliable and accessible source.

Implementing such a system not only enhances the accessibility of transportation information but also fosters a more inclusive environment within the university community.

Students, regardless of their social media usage, can stay informed and plan their journeys effectively, thereby reducing uncertainties and enhancing their overall educational experience.

In conclusion, the development of a modern UTeM bus scheduling management system aligns with the university's commitment to ensuring equal access to essential services for all students. By embracing technology and using these digital platforms, UTeM tries to optimise transportation solutions and support the diverse needs of its student population to enrich their educational journeys for all.

1.2 Problem Statements

Several issues have been identified with the current system:

i. Difficulty updating the schedule because data are recorded in manuals.

Updating the schedule is challenging because all the data is recorded manually and shared through social media. Additionally, manual recording makes it harder to keep the schedule up to date.

ii. Human errors make it difficult to keep schedules up-to-date and accurate.

Administrator may have human errors in inserting the time into bus schedules, the wrong schedule will cause the students to follow the wrong schedules for their bus.

iii. Difficulty in searching the timetable based on the route and time.

Students will face challenges when they want to search for timetable information based on specific routes and times because of the lack of a user-friendly search function. With this web-based platform, students can manage to search for the relevant information they need. Students can generate files to store their schedules.

1.3 Objective

The objectives of this system are as follows:

i. To improve the efficiency of the bus timetable management system by replacing the existing manual paper-based system.

Administrator can use this system to update the bus schedules to provide real-time updates for students. This system can help the admin to manage the bus schedules efficiently.

ii. To develop and implement automated data management systems for bus schedules, aimed at reducing human errors and ensuring the accuracy of schedule updates.

The system will help admins manage the data stored to reduce human errors.

- iii. **To enhance user experience and streamline timetable accessibility by developing a digital search functionality that enables users to efficiently search for bus schedules based on route and time criteria.**

Students can search their bus schedules based on route and time criteria and save them in pdf format.

1.4 Scope

1.4.1 Target User

The intended users and functional modules of the system can be used to classify its scope.

- i. **Users**

Students are given options to log in with an account They can search the bus schedules from this system and save them as pdf.

- ii. **Administrator**

The administrator oversees overseeing the information kept on this system. The administrator will update the schedules and enter the data supplied by the institution into the system. When some emergencies cause the bus cannot arrive destination, the admin should update this information in the system and give notification to the students that will take this bus to the university.

- iii. **Driver**

Driver is given a page to record their job working time and show in table.

4.1.3 Modules

- i. **User Authentication Module**

In this module, user can create an account to ensure the security of their accounts. Users are also given the option to log in without creating an account, but they may have less functionality compared to the students who have accounts.

- ii. **Searching and Reporting Module**

This module allows students to search for their bus schedules based on route and time and generate pdf to save their bus schedules.

- iii. **Notification Module**

Send notifications to students when schedules are updated or cancelled.

iv. Admin Dashboard Module

This module allows the admin to manage the bus schedules and make changes to the bus schedules when needed to adjust the time in the schedule.

v. Tracking Module

This module able users to track the bus in the real time.

vi. Booking Module

This module able users to book seats in the system.

1.5 Project Significant

UTeM bus scheduling management system is developed to manage the bus schedules for students. This system enables students to search their bus schedules based on route and time and save them as pdf. This system also gives students notification when there are some changes in the bus schedules. Admin can efficiently update the bus schedules in real time, ensuring accuracy and reliability. This system facilitates the storage and management of student data. By implementing this system, bus schedule management will become more organized and efficient.

1.6 Expected Outcomes

UTeM bus scheduling management system is developed to manage the bus schedules for students. First, the admin can manage the bus schedules and insert them into the system. Next, students can view the bus schedules from this system and search based on route and time. Admin can update the bus schedules when they are informed that there are some changes in the bus schedules. Students also can get notifications from this system to know which bus schedules get affected. Using this bus scheduling system, students do not need to search the bus schedules from social media. They can log in to this system arrange their preferred schedules and save them as pdf or pictures.

1.7 Conclusion

This chapter is about UTeM bus scheduling management system. For UTeM, this system is incredibly practical and helpful in managing the bus scheduling procedure. This system also improves data consistency and is user-friendly. The development of the UTeM bus scheduling management system has made the bus scheduling process more organized and efficient.

CHAPTER II

PROJECT METHODOLOGY AND PLANNING

2.1 Introduction

This chapter outlines the chosen development methodology for the project. Many types of System Development Life Cycle (SDLC) methodologies can be used in the development of projects. For example, the waterfall model, agile model, and spiral mode can be used to design the project. The projects use the Database Life Cycle (DBLC) methodology for development. It provides a framework to follow identifying the need for a database to its ongoing use and upkeep. In conclusion, the Database Life Cycle (DBLC) encompasses the entire lifespan of the database.

2.2 Project Methodology

The system will be constructed on the Windows 11 operating system utilising a MySQL database. For system development, the Database Life Cycle (DBLC) approach has been selected. The initial database research, database design, implementation and data loading, testing and evaluation, operation, maintenance, and evolution are the six stages of the Database Life Cycle (DBLC).

2.2.1 Database Initial Study

In the context of the UTeM Bus Scheduling Management System project, this phase involves a thorough analysis of the existing process for managing and communicating bus schedules. Currently, bus schedule information is disseminated exclusively through social media platforms like Facebook, Instagram, and Telegram. This method has proven to be difficult to manage due to the dispersed and inconsistent nature of the posts, leading to

communication gaps, missed updates, and confusion among students. By conducting detailed observations, the project team identified the key issues and constraints of the current system, such as the lack of real-time updates, the inefficiency of managing multiple platforms, and the difficulty students face in accessing accurate schedule information. Based on these findings, the objectives for the new system were established to address these challenges, focusing on creating a centralized, real-time platform that improves accessibility and communication. The project's scope was defined to ensure that the new system meets the operational requirements, while considering the constraints imposed by existing hardware and software within the university's infrastructure.

2.2.2 Database Design

The second phase of the database design process is critical to ensuring that the final system aligns with the specified requirements and objectives. The four sub-phases of this phase are conceptual design, logical design, physical design, and DBMS selection. In the conceptual design phase, an entity-relationship diagram is developed based on system requirements, and normalization is applied to ensure data integrity and efficiency. MySQL has been selected as the DBMS for this system. In the logical design phase, a relational data model is developed to define data structures and queries according to system needs. Finally, in the physical design phase, the logical model is translated into database objects such as tables, columns, indexes, sequences, and constraints, grouping attributes to represent core business rules and data relationships.

2.2.3 Implementation and Loading

In this phase, MySQL is installed and configured on the computer. The databases are created by using Data Definition Language (DDL) and data is added using Data Manipulation Language (DML). The programming language or the software used to develop the interface are VS Code with a PHP server extension, MySQL workbench to manage the database and MySQL server to connect the local host. Furthermore, optimization, security measures and backup procedures are implemented to ensure functionality and reliability.

2.2.4 Testing and Evaluation

Three sub-phases comprise the testing and assessment phase: database testing, database tuning, and database and application program evaluation. The purpose of this step is to guarantee the database's backup, recovery, performance, security, and integrity.

a. Testing the Database

During the testing and evaluation phase, comprehensive testing is performed to ensure the system's integrity, security, and performance. Data integrity is enforced by the DBMS through primary keys (PK), foreign keys (FK), unique constraints, and other rules. Data security is evaluated through tests that involve password protection, data encryption, privileges, and access control. This phase encompasses both unit testing and system testing. Unit testing involves testing individual functions, such as login, admin, and student features, by inputting sample data to validate database connections, error handling, and correct data types. System testing evaluates the overall business processes to ensure the system operates smoothly without errors.

b. Fine-Tune the Database

The database will be refined if errors are detected or if data requirements are not fully met, to enhance its performance.

c. Evaluating the Database and Its Application Programs

Once the database has been tested and optimized, the system undergoes a comprehensive evaluation. Multiple testing processes are performed to ensure the system fulfils all data requirements.

2.2.5 Operation

The system is put into place as a fully working information system following the conclusion of the review procedure. To enable the required information flow, users may start using the system to perform operations like importing data, maintaining data, and creating reports.

2.2.6 Maintenance and Evolution

In this phase, many tasks are performed including index maintenance, table optimisation and user management such as adding or removing users and updating passwords. Backup and recovery can be done when failure happens. If there are new requirements needs or changes, adaptive maintenance may be necessary to improve performance and meet the goal.

The Figure 2.1 show the Database Life Cycle Illustration that used in this system.

The Database Life Cycle (DBLC)

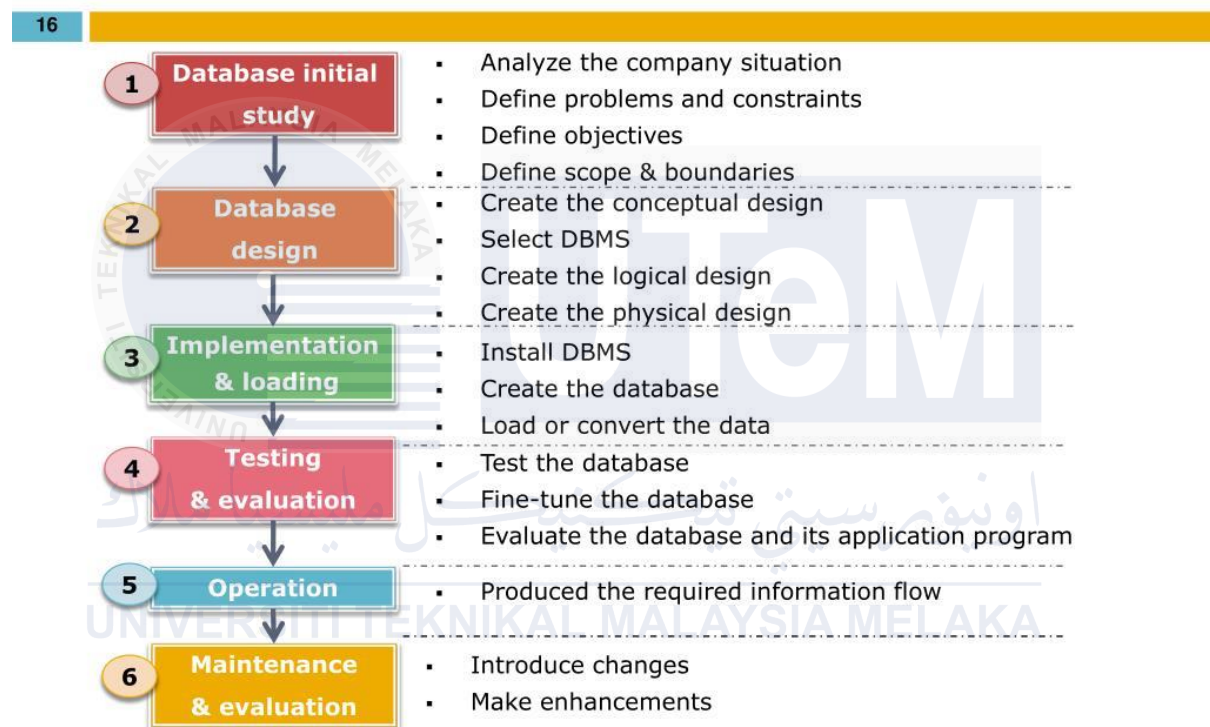


Figure 2.1 Database life cycle illustration

Marlon. (n.d.). Figure 2.1: Database life cycle illustration. In *Database design*. Retrieved from <https://www.slideserve.com/marlon/database-design>

2.3 Project Schedule and Milestones

Figure 2.2 illustrates the complete development timeline for the progress in the year 2024.

Task	Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Planning the Project		■	■													
Problem Identification and Analysis				■	■	■										
Database Design						■	■	■								
Development And Implementation of the Project							■	■	■	■	■	■	■	■		
Testing and Evaluation													■	■	■	
Presentation and Demonstration															■	
Operation															■	
Maintenance and evaluation																■

Figure 2.2 : Gantt Chart of UTeM Bus Scheduling Management System

Table 2.1 shows the milestones of the development of the UTeM Bus Scheduling Management System. In each milestone, some outcomes are expected to be produced to ensure all processes can done within the expected time and meet the goals.

Table 2.1 : Project Milestones

Milestones	Expected Outcomes	Date
Observation	<ol style="list-style-type: none"> 1. Decide what title want to do 2. Do draft 	12 March 2024
Problems identification and analysis	<ol style="list-style-type: none"> 1. Problem statement, objective 2. Flow chart of the current system and the proposed system 3. Requirement and module of proposed system. 	26 March 2024
Conceptual design of the proposed system	<ol style="list-style-type: none"> 1. DFD and ERD of the proposed system 2. Business rules 3. Data Definition Language 4. Data Manipulation Language 5. Normalization and query 	26 April 2024
Implementation of the proposed system	<ol style="list-style-type: none"> 1. Database environment set up 2. Graphical User Interface (GUI) of the proposed system 	25 May 2024

	<ol style="list-style-type: none"> 3. Source code of the proposed system each function 4. Create procedure, package and trigger. 	
Testing of the proposed system functionality	<ol style="list-style-type: none"> 1. Test plan 2. Test case 3. Test result and analysis 4. Solution solving of error message from testing process 	15 July 2024
Completed documentation and logbook	<ol style="list-style-type: none"> 1. PSM 1 final report and logbook 2. Project Demonstration 	30 August 2024
Project Demonstration	<ol style="list-style-type: none"> 1. Final Presentation 	5 September 2024

2.4 Conclusion

In conclusion, this chapter outlines the selected methodology for guiding the system development process, detailing the approach and the stages of the Database Life Cycle (DBLC). DBLC was chosen due to its structured six-stage process: initial database study, database design, implementation and loading, testing and evaluation, operations, and maintenance and evolution. The maintenance phase continues throughout the system's operation to ensure smooth performance. The next chapter will delve deeper into the technical analysis of the overall system.

CHAPTER III

ANALYSIS



3.1 Introduction

This chapter will detail the project's analysis phase. Analysis involves breaking down a system into sub-modules, gathering data, identifying problems, and recommending solutions to improve it. This includes studying the insurance company's current recruitment process, gathering operational data, understanding data flow, identifying bottlenecks, and easing user workloads. This chapter focuses on studying the present recruiting process, identifying its weaknesses, and improving the new system to meet user expectations.

3.2 Problem Analysis

An observation was carried out to assess and analyze the weaknesses of the university's current bus scheduling process. In the manual process, all bus scheduling tasks are still performed without the use of a system. The users can refer to the bus schedule when the university organisation release the bus schedule through media sources only. This situation makes it difficult to manage the bus schedule. For example, the admin hard to update in real-time if there are some changes in the bus schedule. They need to replace the changes in the bus schedule and post again on social media. This will make some users miss the updated bus schedule.

Before the bus scheduling process, the admin needs to prepare the bus details and collect where locations should become bus stops. All manual actions will cause some problems like data entry mistakes and communication issues. For example, a scheduling staff member inputs the wrong departure time for a bus route, indicating it leaves at 9:00 AM instead of the correct time, 8:00 AM. The scheduling team updates a bus route due to road construction but fails to effectively communicate the change to drivers. As a result, several drivers continue following the old route, unaware of the update. These situations will cause human error that makes the current manual system difficult to keep accurate.

Besides that, some users felt difficulty when looking for new bus schedules. For new students or new staff, they don't have to join any related group and follow some pages on social media. For the manual system, the bus schedules will be posted when any changes happen. If not any changes, the organisation post once only.

Moreover, the current process requires the admin to update the schedule details manually and summarise the schedules and posts to social media. This is time-consuming and less effective because the admin needs to insert the time and location manually. Users only can follow what data is inserted by the admin. They cannot filter what they want and save as their bus schedules.

Figure 3.1 depicts the overall structure of the current manual bus scheduling management system at UTeM.

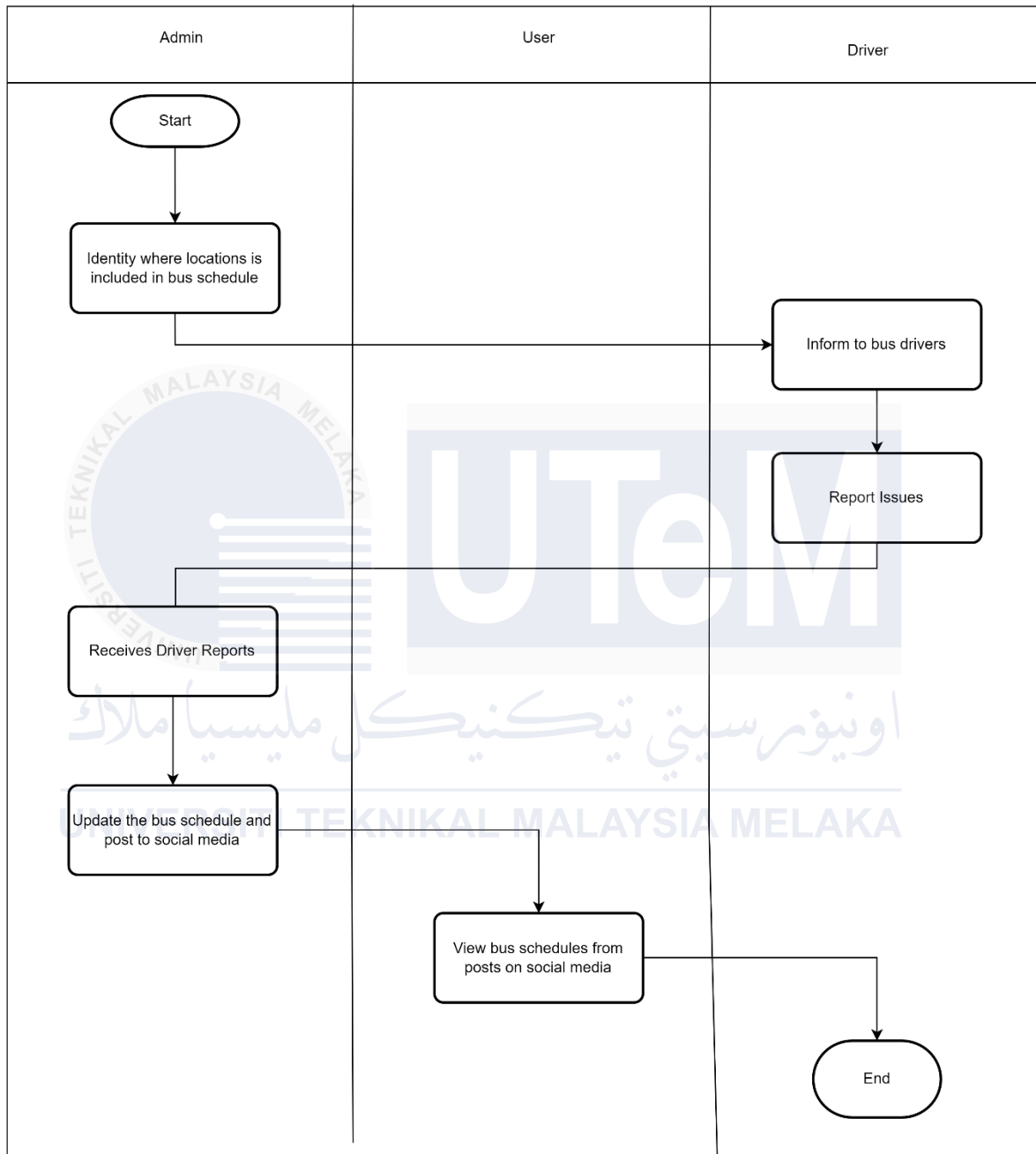


Figure 3.1: Current UTeM Bus Scheduling Process Flow Chart

Figure 3.1 illustrates the current flow of the bus scheduling management system at UTeM. The current flow of the UTeM bus scheduling management system reveals several issues, primarily due to its reliance on manual processes and centralized communication. The identification and updating of bus stops are time-consuming, requiring discussions that can delay the scheduling process. Additionally, the system depends heavily on the admin to communicate changes and address driver issues, creating potential bottlenecks if the admin is unavailable or overwhelmed. This dependency on manual updates and communication can lead to inefficiencies, delays, and potential miscommunication, ultimately affecting the accuracy and timeliness of bus schedules for users.



3.3 The Proposed Improvement/Solutions

Figure 3.2 illustrates the overall proposed online bus scheduling management system in UTeM.

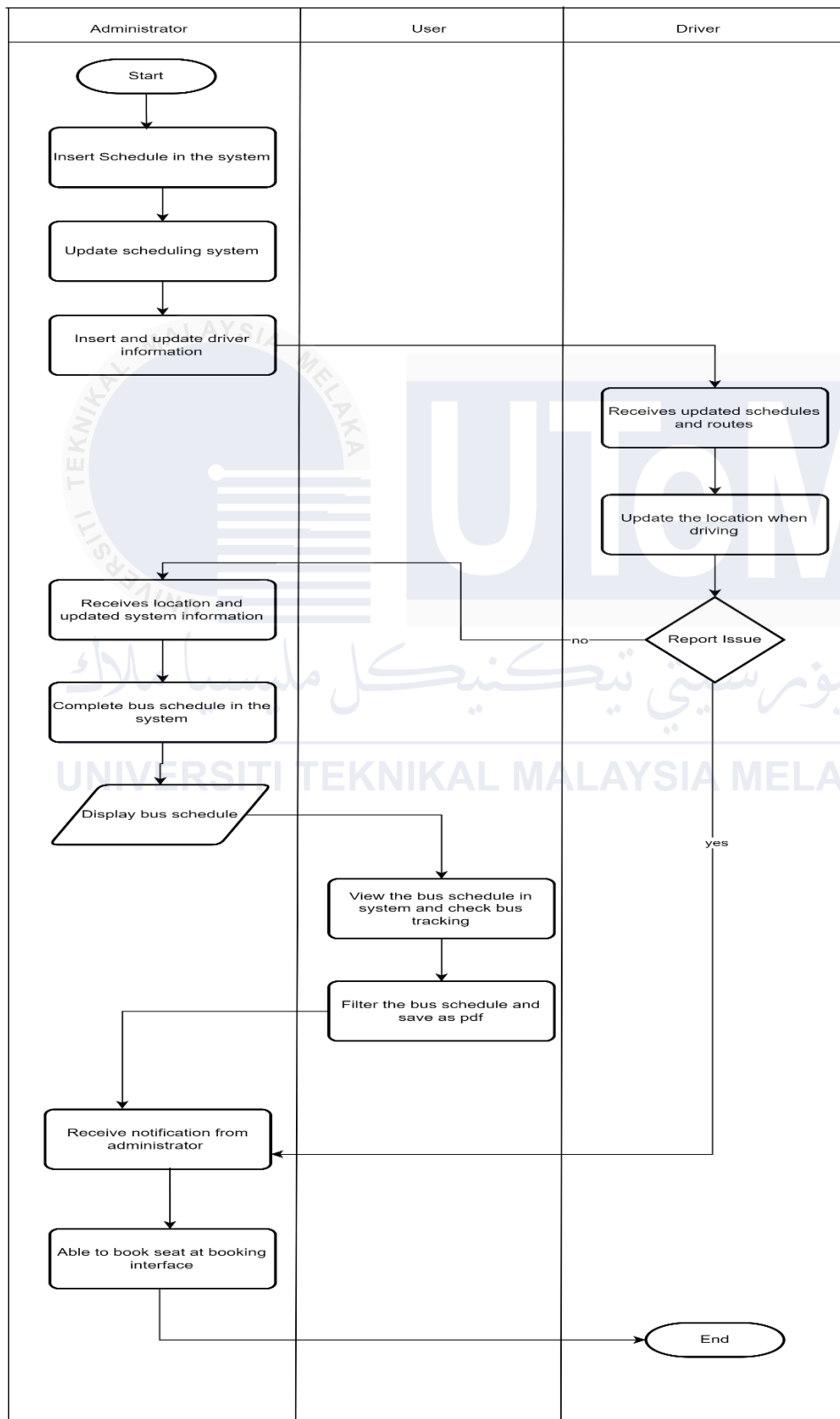
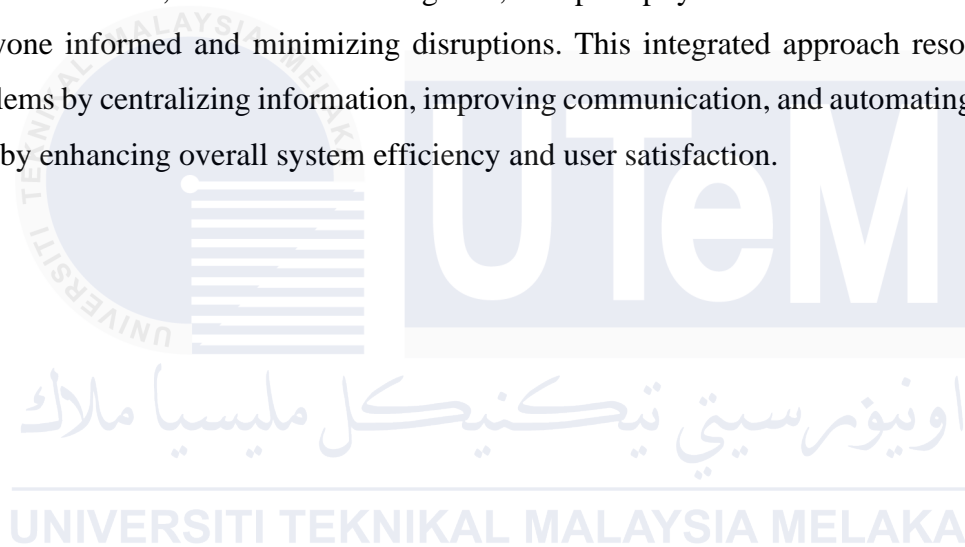


Figure 3.2: UTeM Bus Scheduling Management System Flow Chart

The new process for managing the UTeM bus scheduling system addresses several issues through improved functionality and efficiency. By allowing the administrator to update bus schedules and driver information directly in the system, it eliminates delays and potential errors associated with manual updates. Real-time driver updates and issue reporting streamline communication, enabling quicker responses to problems such as delays or route changes. The system's capability to monitor driver locations in real time enhances route safety and ensures timely adjustments if needed. For users, the interface provides convenient access to bus schedules, filtering options, and the ability to save schedules as PDFs, improving user experience and accessibility. Additionally, the automated notification system ensures that any significant issues, like a driver being late, are promptly communicated to users, keeping everyone informed and minimizing disruptions. This integrated approach resolves previous problems by centralizing information, improving communication, and automating notifications, thereby enhancing overall system efficiency and user satisfaction.



3.4 Requirement Analysis of the to-be-System

Each user's system requirements will be outlined and the functional requirements will be specified by the requirements analysis.

3.4.1 Functional Requirement

- i. Administrator
 - Administer the bus schedule on the website.
 - Administer the user account and view their user details.
 - Administer the driver's details.
 - Administer the bus stop location
 - Administer the announcement
- ii. User
 - View bus schedules from the website.
 - Can check driver tracking on the map.
 - View the available bus stop.
- iii. Driver.
 - Update their location in real time.
 - Able to check-in and check-out

3.4.1.1 Data Flow Diagram (DFD)

The context diagram (Figure 3.3) illustrates the high-level interactions within the UTeM Bus Scheduling Management System. It shows how external entities, such as Admins, Drivers, and Users, interact with the system. Admins manage bus schedules, stops, and driver assignments, while Drivers update their locations via GPS. Users interact with the system to book seats on buses, view schedules, and track buses in real-time. The data flow between these entities and the system is depicted in the diagram, which also emphasises how the system combines several features to provide effective bus scheduling and administration.

i. Context Diagram

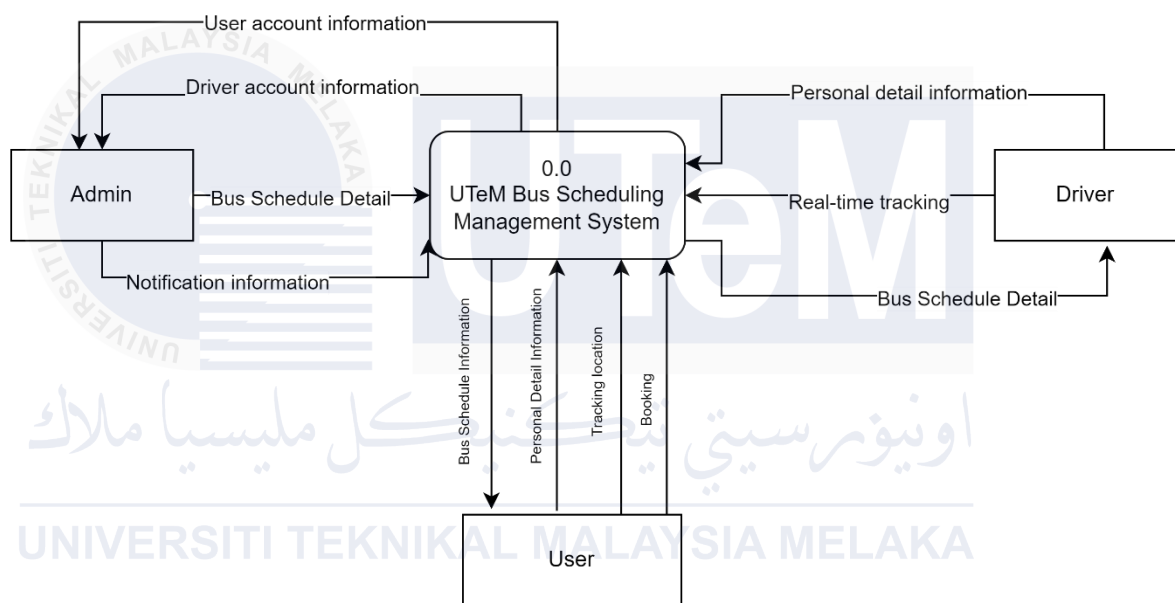


Figure 3.3: Context Diagram

ii. Data Flow Diagram (DFD) Level 1

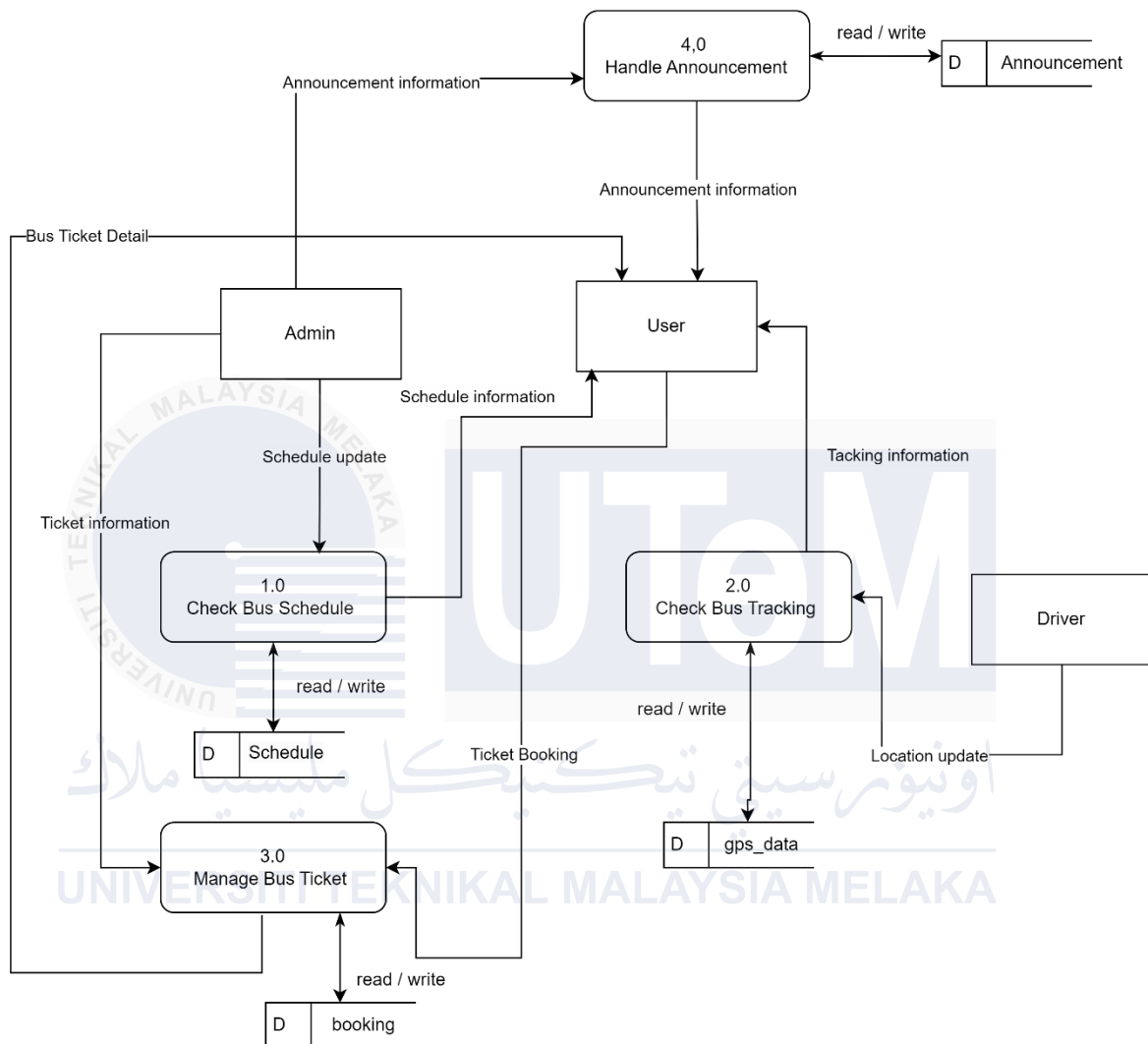


Figure 3.4: Data Flow Diagram (DFD) Level 1

A thorough explanation of the elements shown in the Context Level Diagram may be found in Figure 3.4. It breaks down the high-level process depicted in the Context Diagram into its smaller processes and highlights the primary roles of the system.

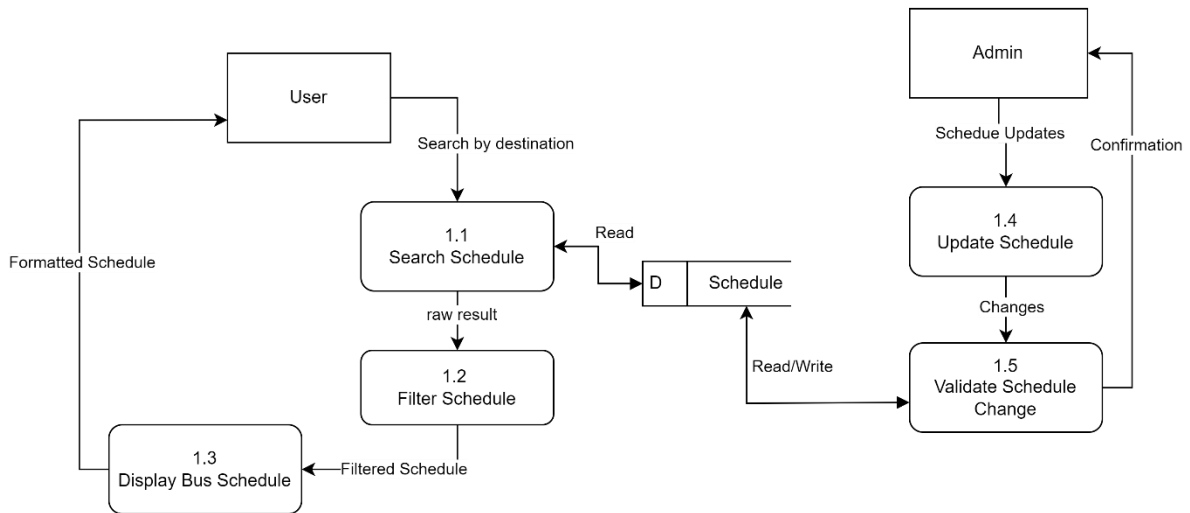


Figure 3.5 : Data Flow Diagram (DFD) Level 2 (Check Bus Schedule)

Figure 3.5 illustrates the bus schedule management process. Administrators can manage schedule information, which will be posted on the company website. Users can view the details of the bus schedules. The schedule table stores all bus schedule information, and when users perform a search on the website, the schedule details are retrieved from the database.

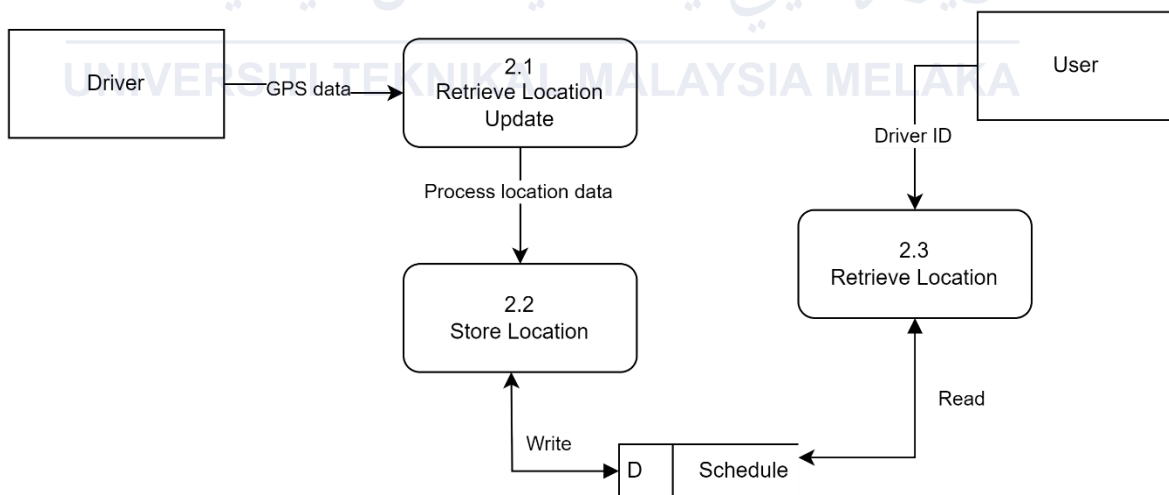


Figure 3.6 : Data Flow Diagram (DFD) Level 2 (Check Bus Tracking)

Figure 3.6 user view bus tracking process. User can view the bus tracking in website. Driver updates the location and stores the location in the database.

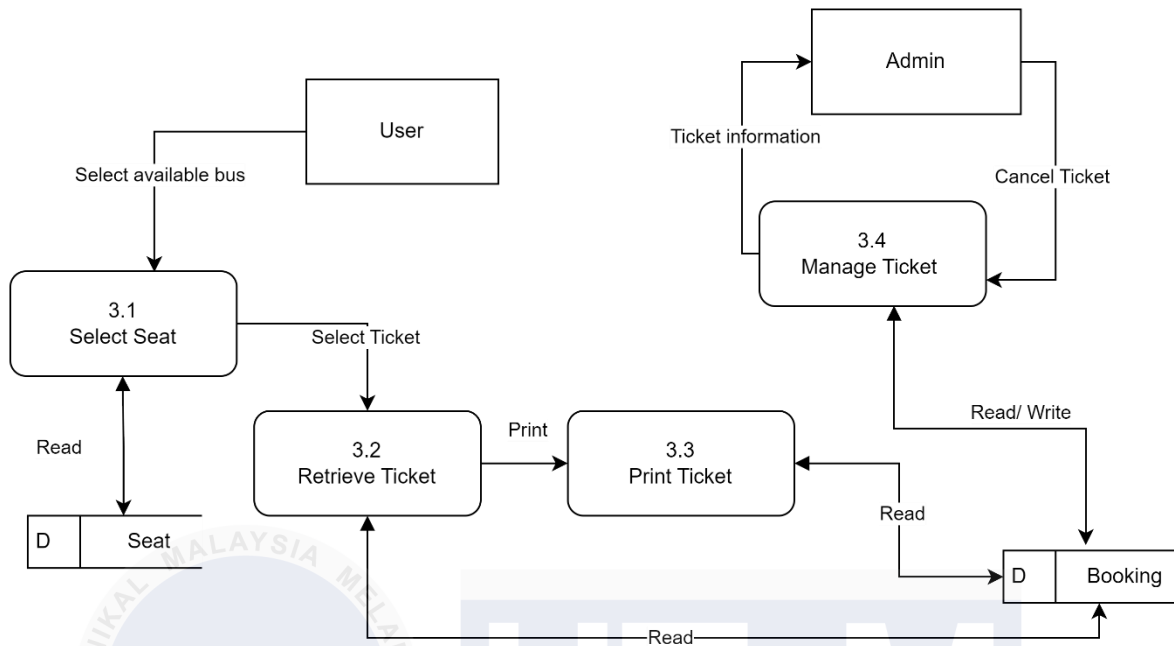


Figure 3.7 : Data Flow Diagram (DFD) Level 2 (Manage Bus Ticket)

Figure 3.7 show user selects the available bus and choose a seat to get a ticket. After retrieving a ticket, user can print the ticket. Administration able to manage the ticket information.

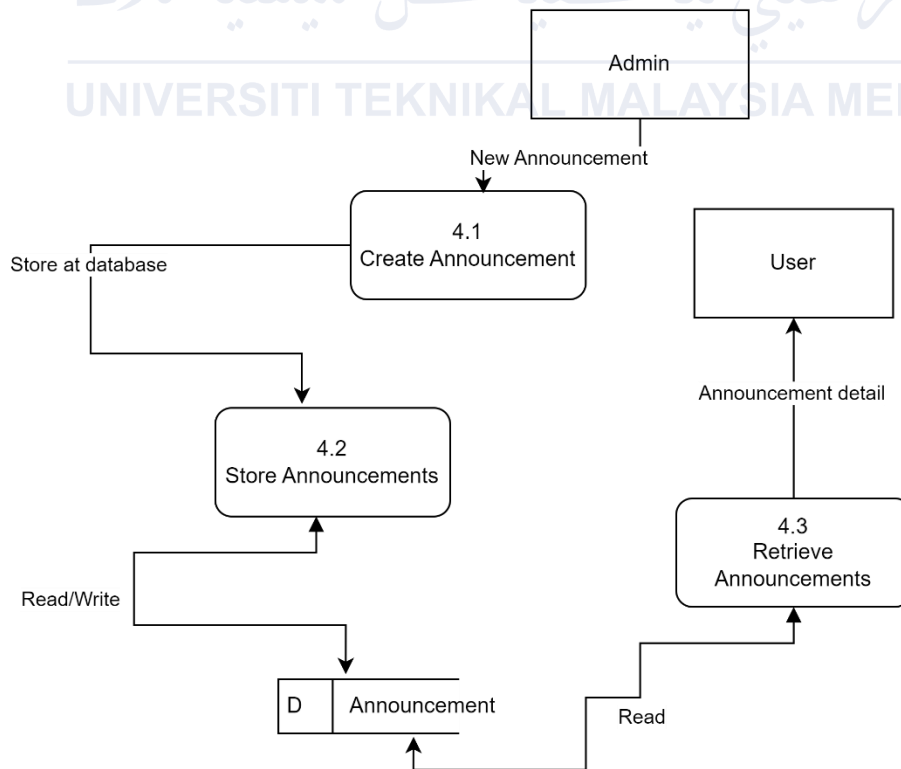


Figure 3.8 : Data Flow Diagram (DFD) Level 2 (Handle Announcement)

Figure 3.8 show the process of notification. Administrator will insert a new notification and stored in announcement table and store at database. When database store the announcement data, it will show the real-time notification at the user side website.

3.4.2 Non- Functional Requirements

The specifications that outline the capabilities and limitations of the system's functioning are known as non-functional requirements, or NFRs. (AltexSoft, n.d.). Table 3.1 shows the non-functional requirements and their description for UTeM Bus Scheduling Management System.

No	Non-Functional	Description
1	Performance	<ul style="list-style-type: none"> -The system should efficiently process and display real-time bus tracking information to users. -Have a notification come out when the bus schedules have been delayed
2	Security	<ul style="list-style-type: none"> - The system should authenticate users using a username and password and determine their access level accordingly. - User passwords must be securely encrypted to ensure data protection.
3	Usability	<ul style="list-style-type: none"> -The system features a user-friendly interface, making it easy to learn and operate. -It provides guidance to assist users in navigating and using the system effectively
4	Reliability	<ul style="list-style-type: none"> -Data integrity should be maintained, ensuring that bus schedules and driver availability information are accurate and consistent.

Table 3.1: Non-Functional Requirements

3.4.3 Other Requirements

The prerequisites for developing a database system fall into two categories: software and hardware. Software requirements involve the applications and platforms used for system design,



while hardware requirements denote the physical equipment necessary for system support and operation.

3.4.3.1 Software Requirement

There have listed the requirements and description of software components, which have been used in the system. There are:

Table 3.2 : Software Requirements

Software	Description
DrawIO 	A cross-platform diagramming tool called Diagrams.net was created using JavaScript and HTML5. With its user-friendly interface, a variety of diagrams, such as flowcharts, wireframes, UML diagrams, organisational charts, and network diagrams, may be constructed.
Microsoft Word 	Microsoft Word is a word processing application used for creating and writing reports.
MySQL (My Structured Query Language) 	Because of its scalability and dependability, MySQL is a prominent open-source relational database management system (RDBMS) that is used for creating databases for a variety of purposes.
Laravel 	Laravel is a web application framework known for its expressive and elegant syntax. It provides a solid foundation that allows developers to focus on creating their applications without getting bogged down by the details.
Visual Studio Code	Microsoft created Visual Studio Code (VS Code), a portable yet potent source code editor. It is popular among developers for its

	<p>rich feature set, extensibility, and cross-platform compatibility, making it suitable for a wide range of programming languages and platforms.</p>
<p>MySQL workbench</p> 	<p>MySQL workbench is used to run database and store.</p>

3.4.3.2 Hardware Requirement

The list of hardware components required for the system is shown in Table 3.3.

Table 3.3 : Hardware Requirements

No	Hardware	Description
1	Processor	Intel Core i5
2	Memory	16 GB DDR4
3	SSD	1TB
4	Telephone	Redmi note 9s

3.5 Conclusion

The examination of the present system is compiled in Chapter III, along with suggestions for enhancements and the creation of a new system's idea. Flowcharts are used to illustrate the current (as-is) and proposed (to-be) system processes, providing a visual representation of how information flows within the system. The Context Diagram offers a high-level perspective of the system's components and their interconnections, giving a general picture of the complete system that has to be constructed. The Data Flow Diagram, on the other hand, is primarily concerned with depicting the data flow inside the system or process.

It illustrates the inputs, outputs, and interactions between entities and processes. Chapter III serves to guide the development process towards creating an improved and more efficient system.

CHAPTER IV

DESIGN

4.1 Introduction

In the design stage of the project, the system architecture is meticulously established through three key design levels: **conceptual**, **logical**, and **physical**. The **conceptual design** provides a high-level overview of the system's major components and their relationships, setting the framework for further development. The **logical design** translates this overview into detailed, technology-agnostic models that outline how data will be organized and interact within the system. Finally, the **physical design** specifies the actual technologies, hardware, and database schemas needed for implementation. This structured approach ensures that the system design not only meets the initial application requirements but also effectively addresses all specified functions, paving the way for successful development and deployment.

4.2 System Architecture Design

UTeM Bus Scheduling Management System is designed on a 3-tier design architecture. There are 3 types of functionalities for this 3-Tier architecture. These are the presentation tier, logical tier and data tier.

The presentation tier is the topmost layer of this application and is responsible for the user interface. This tier is built using HTML, CSS, JavaScript and Laravel's Blade templating engine and users can display bus schedules, routes and locations of bus stops. The login tier is the layer between the presentation and data tiers. This login tier is implemented using Laravel controllers, in which each controller handles specific requests related to bus scheduling,

administrative side and driver check-in, and check-out. In addition, the data tier is stored and retrieved from a database and implemented using Eloquent ORM which provides a simple implementation for working with the database.

The Figure 4.1 provided represents a multi-tier architecture for a Laravel-based web application.

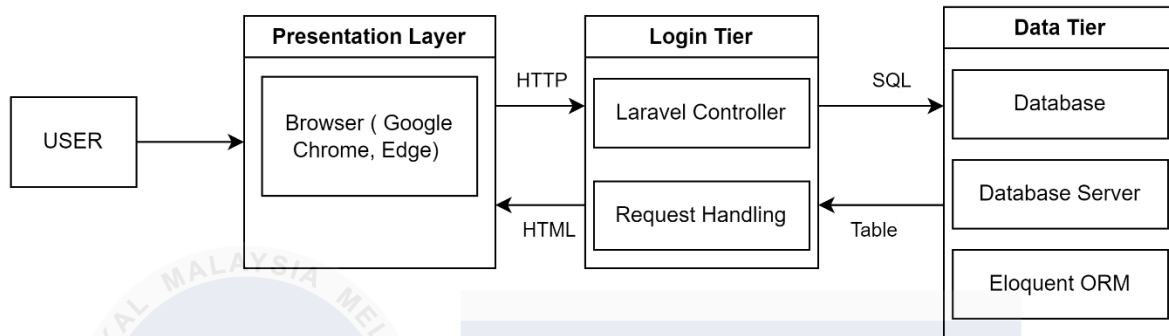


Figure 4.1 : 3 Tier Architecture

4.3 Database Design

The systematic process of creating a comprehensive data model for a database is known as database design. This data model includes all the logical and physical design choices that are required, together with the actual storage. Each entity's specific properties are included in a fully attributed data model.

Many distinct aspects of the overall design of a database system can be referred to by a term "database design." It makes sense to think of it mostly as the logical arrangement of the basic data structures that house the data. In an object database, relationships and entities precisely translate to named relationships and object classes. These are the tables and views that make up the relational model. In Entity-Relationship Design (ERD), three primary types of designs guide the creation of a database system. **Conceptual Design** focuses on defining the high-level structure by identifying main entities, their attributes, and relationships without delving into implementation details. This stage aims to capture the essential business requirements and data interactions in an abstract form. **Logical Design** then refines this abstract model into a detailed schema that specifies primary keys, foreign keys, and normalization rules to ensure data integrity and organization. This design serves as a blueprint for the database structure. **Physical Design** translates the logical schema into the actual database implementation, including the creation of tables, indexes, and optimization strategies for

performance. This final stage addresses the practical aspects of data storage and access, ensuring the database operates efficiently within a specific DBMS environment. Each design phase builds on the previous one to create a comprehensive and effective database system.

4.3.1 Conceptual Design

Figure 4.2 illustrates the Entity Relationship Diagram for the UTem Bus Scheduling Management System.

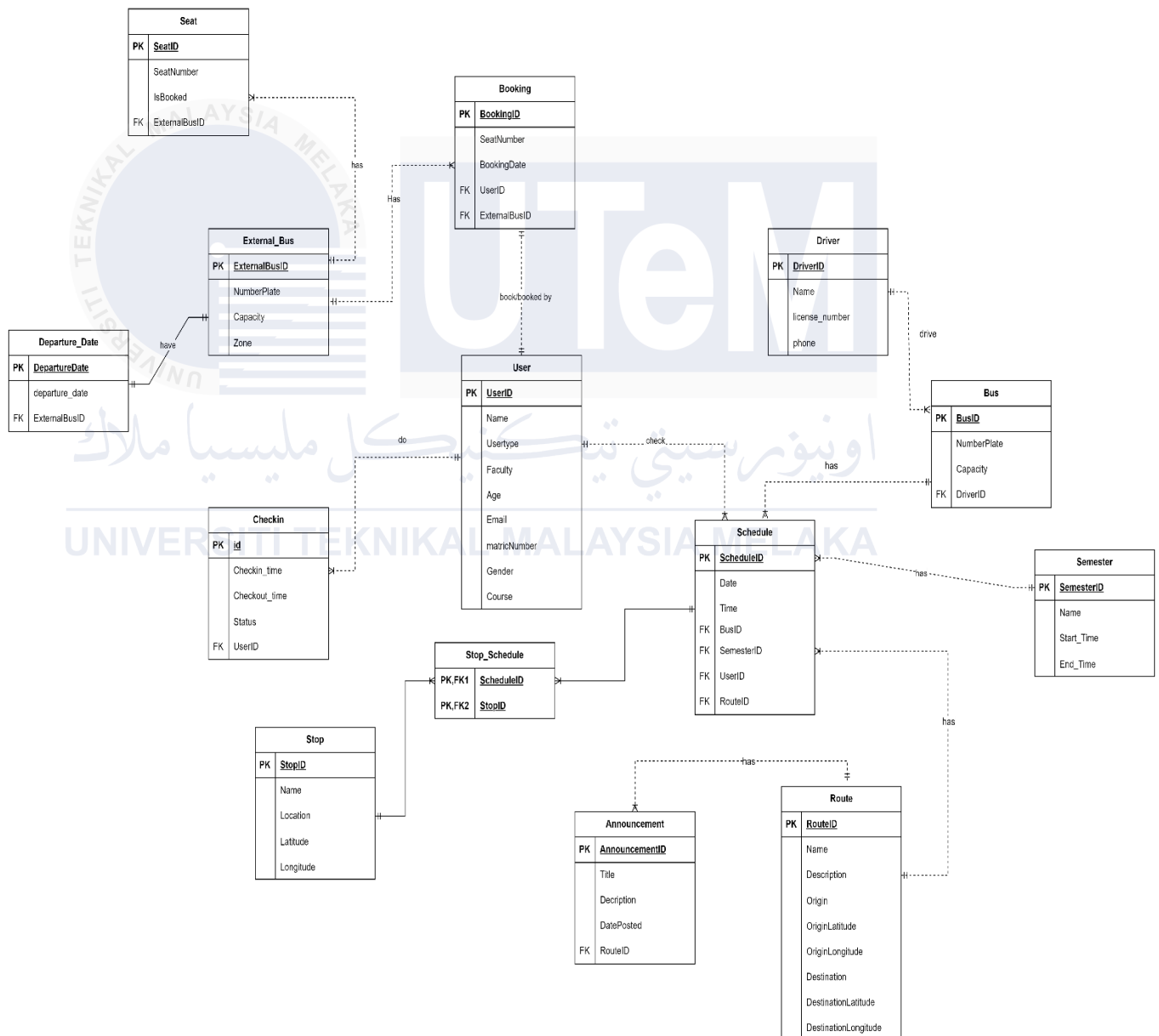


Figure 4.2: ERD of UTem Bus Scheduling Management System

Business rule of UTeM Bus Scheduling Management System

1. Each user can give one or many feedback while each feedback only can be given from one and only one user.
2. Each user can do one or many checkin while each checkin only can be done by one and only one user.
3. Each user can check one or many schedules while each schedule only can be checked by one and only one user.
4. Each schedule has one and only one bus while each bus belongs to one or many schedules.
5. Each driver can driver one or many busses while each bus driven by one driver only.
6. Each semester has one or many schedules while each schedule belongs to one semester only.
7. Each route has one or many schedules while each schedule belongs to one route only.
8. Each routes has one or many announcements while each announcement belongs to one route only.
9. Each stop has many schedules while each schedule has many stops.
10. A user allowed book one seat only while each seat can be booked by one user only.
11. An external bus has one or many seats while each seat has belonged to one bus only.
12. Each external bus has one departure date while each departure date has belonged to one bus only
13. An external bus has one or many booking while each booking is belonging to one bus only

4.3.2 Logical Design

The second stage of database design is called logical design. Based on user transactions, the linkages between the logical data are defined, normalised, and validated in this stage. Every logical data element's restrictions are also evaluated.

4.3.2.1 Data Dictionary for Entity Relational Diagram

a) User table

This table keeps each user's detailed data.

Table 4.1 : Table User

Attribute	Description	Data Type	Required	PK/FK	Reference Table
UserID	User ID	INT	Yes	PK	
Name	Name	varchar(255)	Yes		
Usertype	User Type	varchar(255)	Yes		
Faculty	User Faculty	varchar(255)	Yes		
MatricNumber	User Matric Number	varchar(255)	Yes		
Age	User Age	INT	Yes		
Gender	User Gender	varchar(255)	Yes		
Course	User Course	varchar(255)	Yes		
Phone	User Phone	varchar(255)	Yes		
Email	User Email	varchar(255)	Yes		
Password	User Password	varchar(255)	Yes		

b) Driver table

This table keeps each driver's detailed data.

Table 4.2 : Table Driver

Attribute	Description	Data Type	Required	PK/FK	Reference Table
DriverID	Driver ID	INT	Yes	PK	
Name	Driver Name	varchar(255)	Yes		
Licence Number	Driver Licence Number	varchar(255)	Yes		
Phone	Driver Phone	varchar(255)	Yes		

c) External Bus table

This table keeps each feedback detailed data.

Table 4.3 : Table External Bus

Attribute	Description	Data Type	Required	PK/FK	Reference Table
ExternalBusID	External Bus ID	INT	Yes	PK	
NumberPlate	Bus Number Plate	varchar(255)	Yes		
Capacity	Bus Capacity	varchar(255)	Yes		
Zone	Zone	varchar(255)	Yes		

d) Checkin table

This table keeps each checkin and checkout detailed data.

Table 4.4 : Table Checkin

Attribute	Description	Data Type	Required	PK/FK	Reference Table
Id	Check ID	INT	Yes	PK	
Checkin_time	Check-in time	Timestamp	Yes		
Checkout_time	Check-out time	Timestamp	Yes		
Status	Driver Status	ENUM('work', 'rest')	Yes		
UserID	User ID	INT	Yes	FK	User

e) Bus table

This table keeps each bus's detailed data.

Table 4.5 : Table Bus

Attribute	Description	Data Type	Required	PK/FK	Reference Table
BusID	Bus ID	INT	Yes	PK	
NumberPlate	Bus Number Plate	varchar(255)	Yes		
Capacity	Bus Capacity	INT	Yes		
DriverID	Driiver ID	INT	Yes	FK	Driver

f) Semester table

This table keeps each semester's detailed data.

Table 4.6 : Table Semester

Attribute	Description	Data Type	Required	PK/FK	Reference Table
Semester ID	Semester ID	INT	Yes	PK	
Name	Semester Name	varchar(50)	Yes		
Start_Date	Semester Start Date	Timestamp	Yes		
End_Date	Semester End Date	Timestamp	Yes		

g) Schedule table

This table keeps each schedule detailed data.

Table 4.7 : Table Schedule

Attribute	Description	Data Type	Required	PK/FK	Reference Table
ScheduleID	Schedule ID	INT	Yes	PK	
Date	Schedule Date	Date	Yes		
Time	Schedule Time	Time	Yes		
BusID	Bus ID	INT	Yes	FK	Bus
SemesterID	Semester ID	INT	Yes	FK	Semester
RouteID	Route ID	INT	Yes	FK	Route

h) Route table

This table keeps each route detailed data.

Table 4.8 : Table Route

Attribute	Description	Data Type	Required	PK/FK	Reference Table
RouteID	Route ID	INT	Yes	PK	
Description	Route Description	varchar(100)	Yes		
Origin	Route Origin	varchar(50)	Yes		
OriginLatitude	Origin Latitude	double	Yes		
OriginLongitude	Origin Longitude	double	Yes		
Destination	Route Destination	varchar(50)	Yes		
DestinationLatitude	Destination Latitude	double	Yes		

DestinationLongitude	Destination Longitude	double	Yes		
----------------------	-----------------------	--------	-----	--	--

i) Announcement table

This table keeps each announcement's detailed data.

Table 4.9 : Table Announcement

Attribute	Description	Data Type	Required	PK/FK	Reference Table
AnnouncementID	Announcement ID	INT	Yes	PK	
Title	Announcement Title	varchar(50)	Yes		
Description	Announcement Description	varchar(100)	Yes		
DatePosted	Date Posted	date	Yes		
RouteID	Route ID	INT	Yes	FK	Route

j) Stop table

This table keeps each stop with detailed data.

Table 4.10 : Table Stop

Attribute	Description	Data Type	Required	PK/FK	Reference Table
StopID	Stop ID	INT	Yes	PK	
Name	Stop Name	varchar(50)	Yes		
Location	Stop Location	varchar(50)	Yes		
Latitude	Stop Latitude	decimal(10,7)	Yes		
Longitude	Stop Longitude	decimal(10,7)	Yes		
Picture	Stop Picture	varchar(255)	Yes		

k) Stop_Schedule table

This table keeps a pivot table of schedule and stop detailed data.

Table 4.11: Table Stop_Schedule

Attribute	Description	Data Type	Required	PK/FK	Reference Table
StopID	Stop ID	INT	Yes	FK	Stop
ScheduleID	Schedule ID	INT	Yes	FK	Schedule

l) Seat Table

This table keeps each seat with detailed data.

Table 4.12 : Table Seat

Attribute	Description	Data Type	Required	PK/FK	Reference Table
SeatID	Seat ID	INT	Yes	PK	
SeatNumber	Seat Number	varchar(50)	Yes		
IsBooked	Is Booked	Boolean	Yes		
ExternalBusID	External Bus ID	INT	Yes	FK	External Bus

m) Departure_Date Table

This table keeps each departure date with detailed data.

Table 4.13 : Departure Date Table

Attribute	Description	Data Type	Required	PK/FK	Reference Table
DepartureDateID	Departure Date ID	INT	Yes	PK	
Departure_date	Departure Date	date	Yes		
ExternalBusID	External Bus ID	INT	Yes	FK	External Bus

n) Booking Table

This table keeps each booking with detailed data.

Table 4.14 : Booking Table

Attribute	Description	Data Type	Required	PK/FK	Reference Table
BookingID	Booking ID	INT	Yes	PK	
SeatNumber	Departure Date	varchar(50)	Yes		
BookingDate	Booking Date	DATE	Yes		
ExternalBusID	External Bus ID	INT	Yes	FK	External Bus
UserID	User ID	INT	Yes	FK	User

4.3.2.2 Normalization

The conceptual design employs normalization, which is illustrated in the tables showing attributes, primary keys, and foreign keys. Figures 4.3 to 4.14 present the Third Normal Form (3NF) of the UTeM Bus Scheduling Management System. Tables are in 3NF when they eliminate transitive dependencies, ensuring that all non-key attributes are functionally dependent solely on the primary key. This process reduces redundancy and enhances data integrity.

Figure 4.3 shows that the User table is in Third Normal Form (3NF).

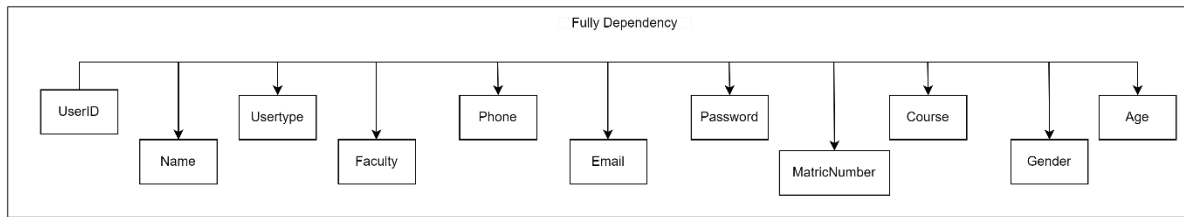


Figure 4.3 : 3 NF of User Table

Figure 4.4 shows that Driver table is in Third Normal Form (3NF).

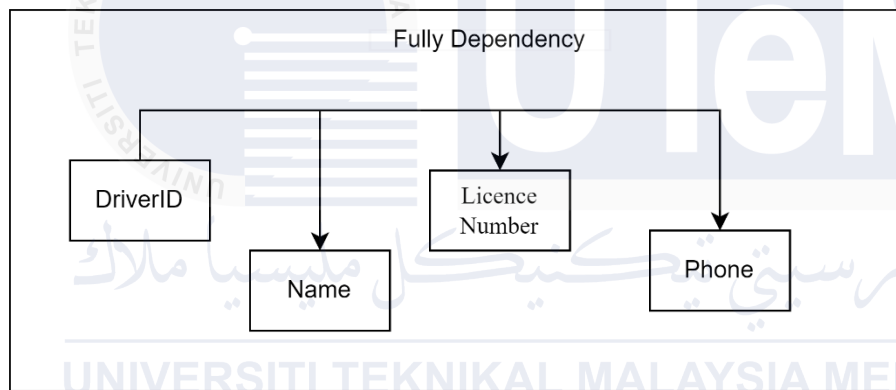


Figure 4.4 : 3 NF of Driver Table

Figure 4.5 shows that External Bus table is in Third Normal Form (3NF).

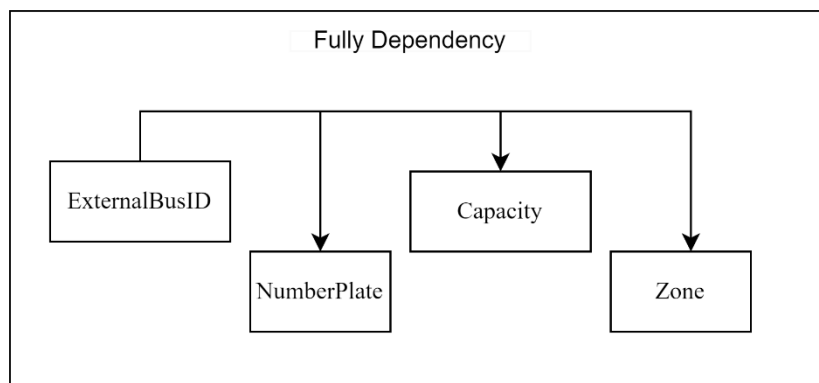


Figure 4.5 : 3 NF of External Bus Table

Figure 4.6 shows that Checkin table is in Third Normal Form (3NF).

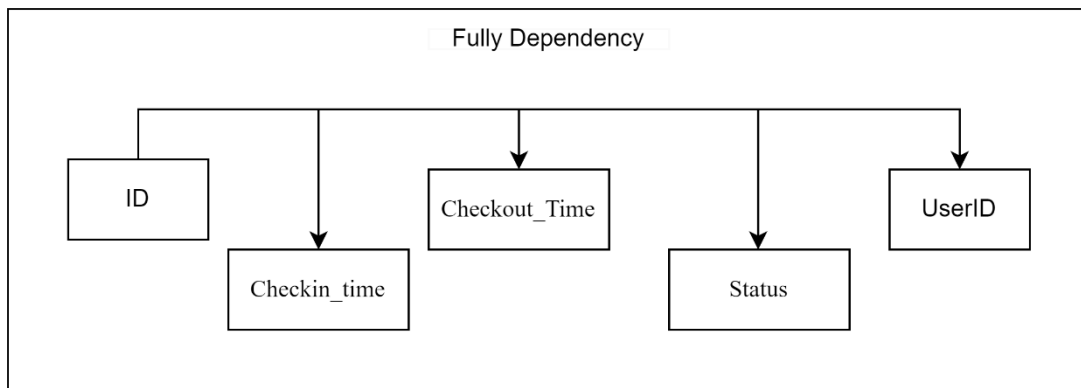


Figure 4.6: 3 NF of Checkin Table

Figure 4.7 shows that Bus table is in Third Normal Form (3NF).

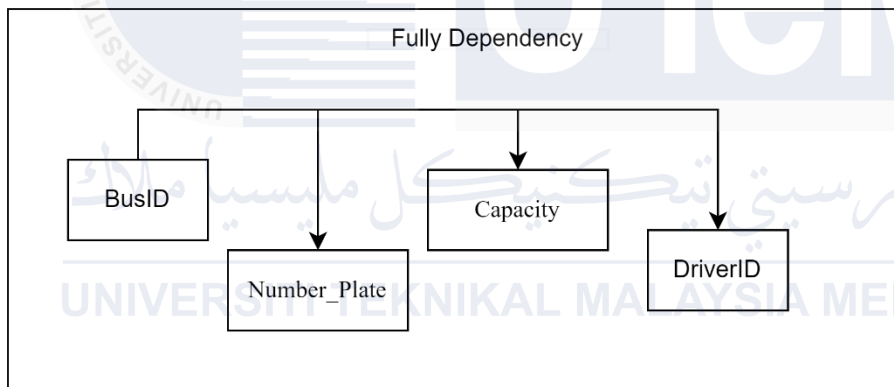


Figure 4.7 : 3 NF of Bus Table

Figure 4.8 shows that Semester table is in Third Normal Form (3NF).

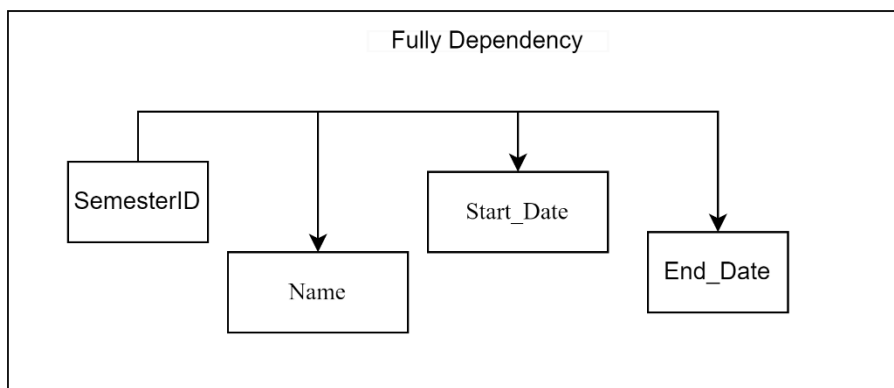


Figure 4.8: 3 NF of Semester Table

Figure 4.9 shows that Schedule table is in Third Normal Form (3NF).

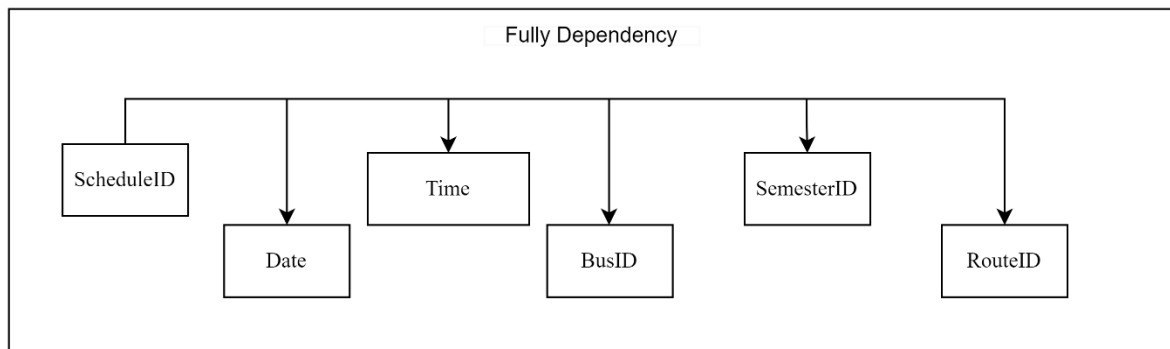


Figure 4.9 : 3 NF of Schedule Table

Figure 4.10 shows that Route table is in Third Normal Form (3NF).

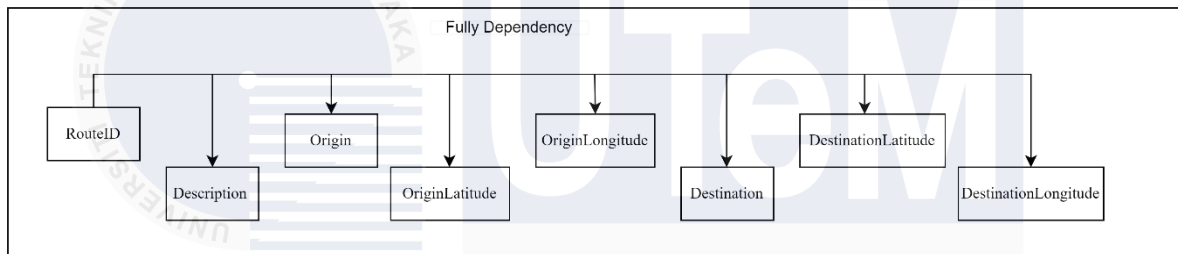


Figure 4.10: 3 NF of Route table

Figure 4.11 shows that Announcement table is in Third Normal Form (3NF).

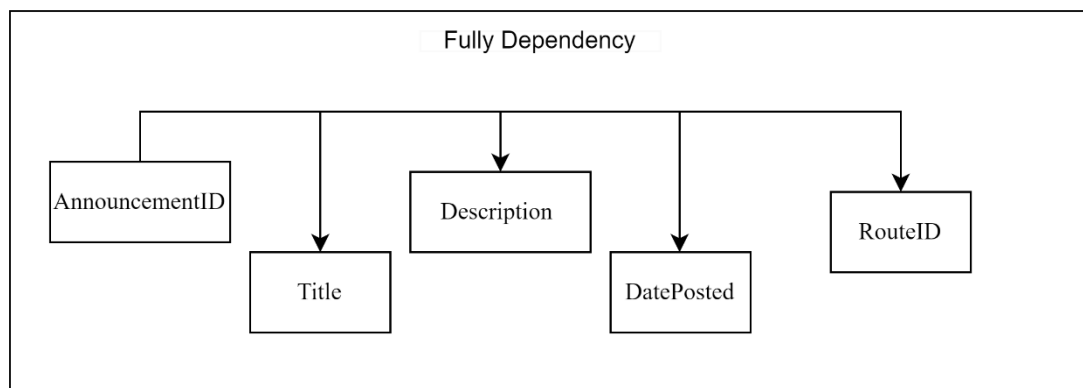


Figure 4.11: 3 NF of Announcement Table

Figure 4.12 shows that Stop table is in Third Normal Form (3NF).

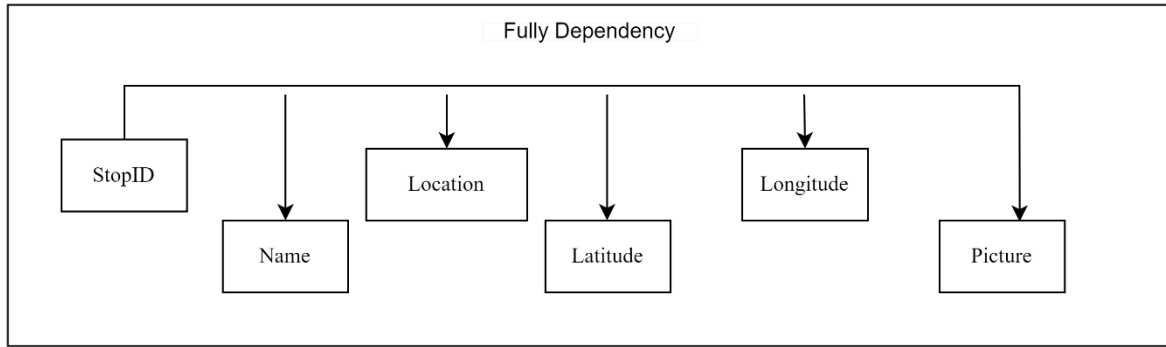


Figure 4.12 : 3 NF of Stop Table

Figure 4.13 shows that Stop_Schedule table is in Third Normal Form (3NF).

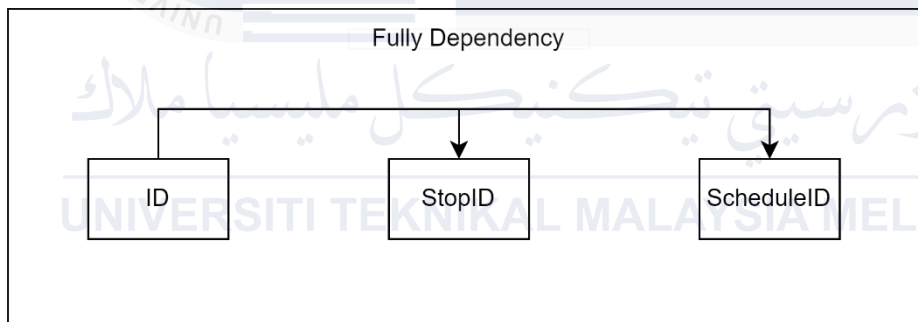


Figure 4.13: 3 NF of Stop_Schedule Table

4.3.2.2 Query Design

Various query structures able to produce different types of outputs, each tailored to specific requirements, reasons, and purposes. Table 4.15 will provide examples of query designs.

Table 4.15 : Query Design of UTeM Bus Scheduling Management System

Type of Query	Query	Explanation
Simple Query	SELECT COUNT(*) FROM users WHERE usertype = 'admin';	To retrieve the count of users where the user type is administrator.
Join Table Query	SELECT routes.*, schedules.* FROM routes LEFT JOIN schedules ON routes.RouteID = schedules.RouteID;	To retrieve the routes and schedules detail with join using RouteID.
	SELECT b.*, d.* FROM buses b LEFT JOIN drivers d ON b.DriverID = d.DriverID WHERE b.DriverID IS NOT NULL;	To retrieve the buses and drivers' detail with join using DriverID.
Aggregate and Grouping Query	SELECT faculty, COUNT(*) as student_count FROM users WHERE usertype = 'user' GROUP BY faculty;	To retrieve the faculty and count the students where user type is user group by faculty.

4.3.2.3 Usage of Stored Procedures and Triggers

Table 4.16: Triggers Relates Database Object Detail

Trigger	Database table	Query	Explanation
Before Delete	Drivers	<pre>CREATE DEFINER=`root`@`localhost` TRIGGER `delete_driver_trigger` BEFORE DELETE ON `drivers` FOR EACH ROW BEGIN DELETE FROM users WHERE users.usertype = 'driver' AND users.phone = OLD.phone; END</pre>	To delete users before delete the driver where user type is driver and user phone number is driver phone number
Before Insert	Checkins	<pre>CREATE DEFINER=`root`@`localhost` TRIGGER `prevent_new_checkin_before_checkout` BEFORE INSERT ON `checkins` FOR EACH ROW BEGIN IF (SELECT COUNT(*) FROM checkins WHERE user_id = NEW.user_id) > 0 THEN IF (SELECT COUNT(*) FROM checkins WHERE user_id = NEW.user_id AND checkout_time IS NULL) > 0 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot create a new checkin before checking out.'; END IF; END IF; END</pre>	This trigger prevents a new checkin from being inserted if there is already an open checkin

Table 4.17 : Stored Procedure Relates Database Object Detail

Procedure	Database table	Query	Explanation
Insert	Schedules	<pre> CREATE DEFINER=`root`@`localhost` PROCEDURE `add_schedule`(IN p_date DATE, IN p_Time TIME, IN p_BusID BIGINT, IN p_SemesterID BIGINT, IN p_RouteID BIGINT) BEGIN INSERT INTO schedules (Date, Time, BusID, SemesterID, RouteID) VALUES (p_date, p_Time, p_BusID, p_SemesterID, p_RouteID); END </pre>	To insert new schedule
Update	Drivers	<pre> CREATE DEFINER=`root`@`localhost` PROCEDURE `update_driver`(IN p_DriverID BIGINT, IN p_name VARCHAR(255), IN p_license_number VARCHAR(255), IN p_phone VARCHAR(20)) BEGIN </pre>	To update drivers information

		<pre> UPDATE drivers SET name = p_name, license_number = p_license_number, phone = p_phone, updated_at = NOW() WHERE DriverID = p_DriverID; END </pre>	
Delete	Announcements	<pre> CREATE DEFINER=`root`@`localhost` PROCEDURE `delete_announcement`(IN p_AnnouncementID BIGINT) BEGIN DELETE FROM announcements WHERE AnnouncementID = p_AnnouncementID; END </pre>	To delete announcement

4.3.2.4 Security Mechanism

The security mechanism in the system validates the user's email and password to ensure they are correct and determines the user's role (admin, normal user, or driver). Based on their role, the user is directed to the appropriate page. For example, if a user enters a valid email and password for a driver account, they will log in as a driver and be navigated to the driver-specific page..

```

2 references | 0 implementations
class HomeController extends Controller
{
    1 reference | 0 overrides
    public function redirect()
    {
        if (Auth::check()) {
            $user = Auth::user();
            if ($user->usertype == 'admin') {
                return redirect()->route('dashboard');
            } elseif ($user->usertype == 'user') {
                return redirect()->route('user.main');
            }
            elseif ($user->usertype == 'driver') {
                return redirect()->route('checkin.index');
            }
        }
        return redirect()->route('login');
    }
}

```

4.4 Graphical User Interface (GUI) Design

. The way users will engage with a system and the kinds of inputs it may process are delineated in a graphical user interface (GUI) design. It includes forms and screens for data entry in addition to screen displays that make navigating the system easier. The three primary parts of GUI design are input, output, and navigation design.

4.4.1 Navigation Design

The user's easy access and guidance are guaranteed by the interface design's navigation component. The UTeM Bus Scheduling Management System's navigation architecture and path are shown in Figure 4.14.

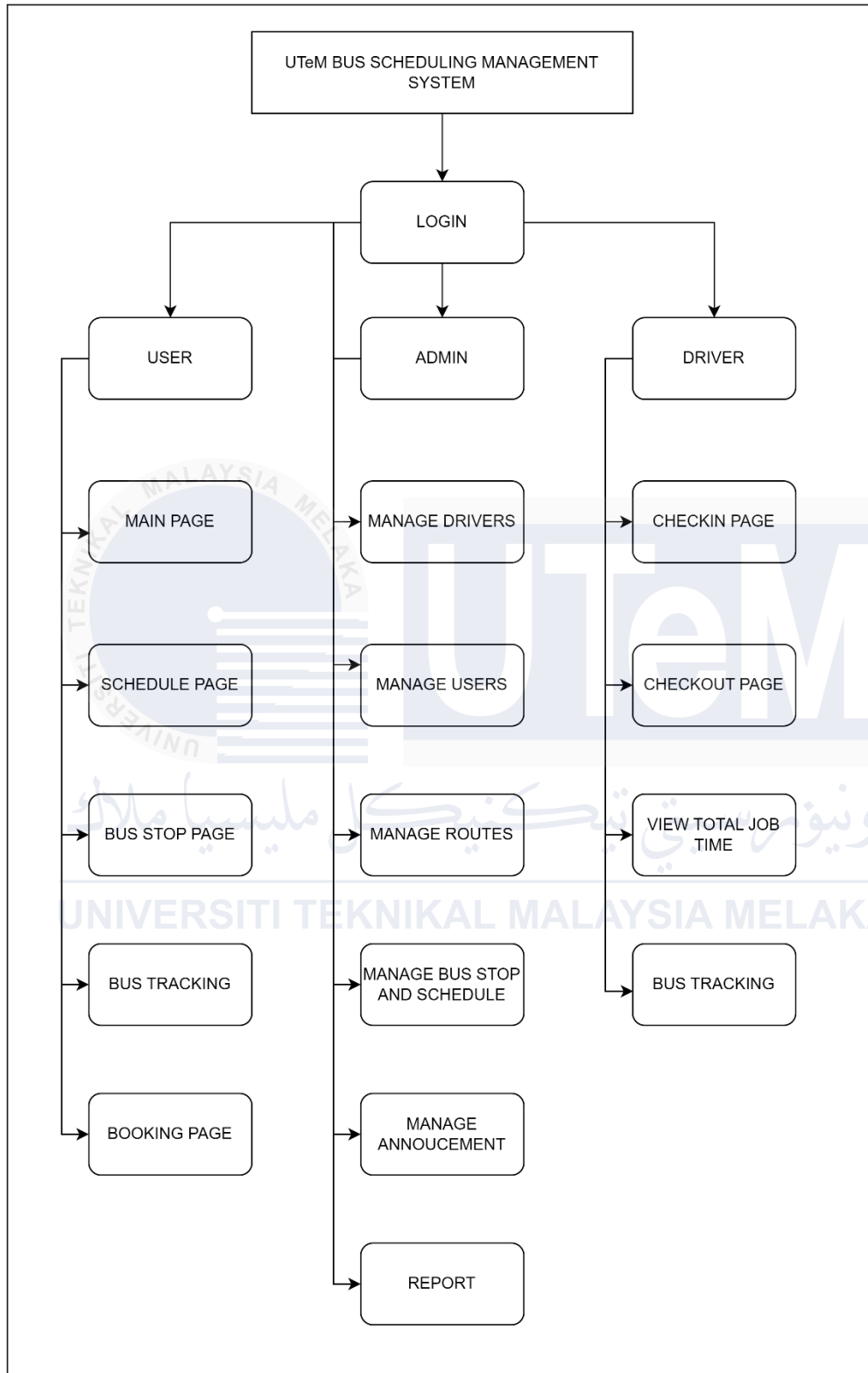


Figure 4.14: Navigation Path of UTeM BUS SCHEDULING MANAGEMENT SYSTEM

4.4.2 Input Design

The input design is concerned with how admin enter both organised and unstructured data into the system. Information for a completed action is stored in the forms and screen to be used by the system. Appendix A contains the illustrations of the input design, which range from Figure 8.1 to Figure 8.8.

4.4.3 Output Design

The output design focuses on presenting the system's retrieved information on the screen or in forms. Figures 8.9 through 8.22, which can be referenced in Appendix A, illustrate the output design.

4.5 Conclusion

This chapter concludes by detailing a methodical approach to building the UTeM Bus Scheduling Management System, addressing both the functional and non-functional requirements identified in Chapter 3. The requirements document from the analysis phase's problems are the focus of the design phase's work. The shift from the issue domain to the solution domain is signified by this phase. This phase's design document will be used as a guide for the stages of implementation, testing, and maintenance.

CHAPTER V



5.1 Introduction

This chapter delineates the procedure for executing the database design. It outlines the installation and configuration processes for the database. MySQL was installed on a Windows 11 platform during the database implementation phase. This phase entailed the execution of Data Definition Language (DDL) and Data Manipulation Language (DML) SQL statements in the database. The status of implementation for each module is also detailed.

5.2 System Development Environment Setup

In the UTeM bus scheduling management system, the software development environment must be set up before developing the system. In addition, the development environment of the system has 4 main components which are built-in PHP development servers from Laravel, MySQL server, PHP Web Programming Language using Laravel and MySQL

Workbench Database Management Tool. In addition, MySQL workbench may be downloaded for free from the internet for the Windows platform.

5.2.1 Steps of Installation Setup

Installation of MySQL workbench

Step 1: Download the installer from <https://dev.mysql.com/downloads/workbench/> using a web browser.

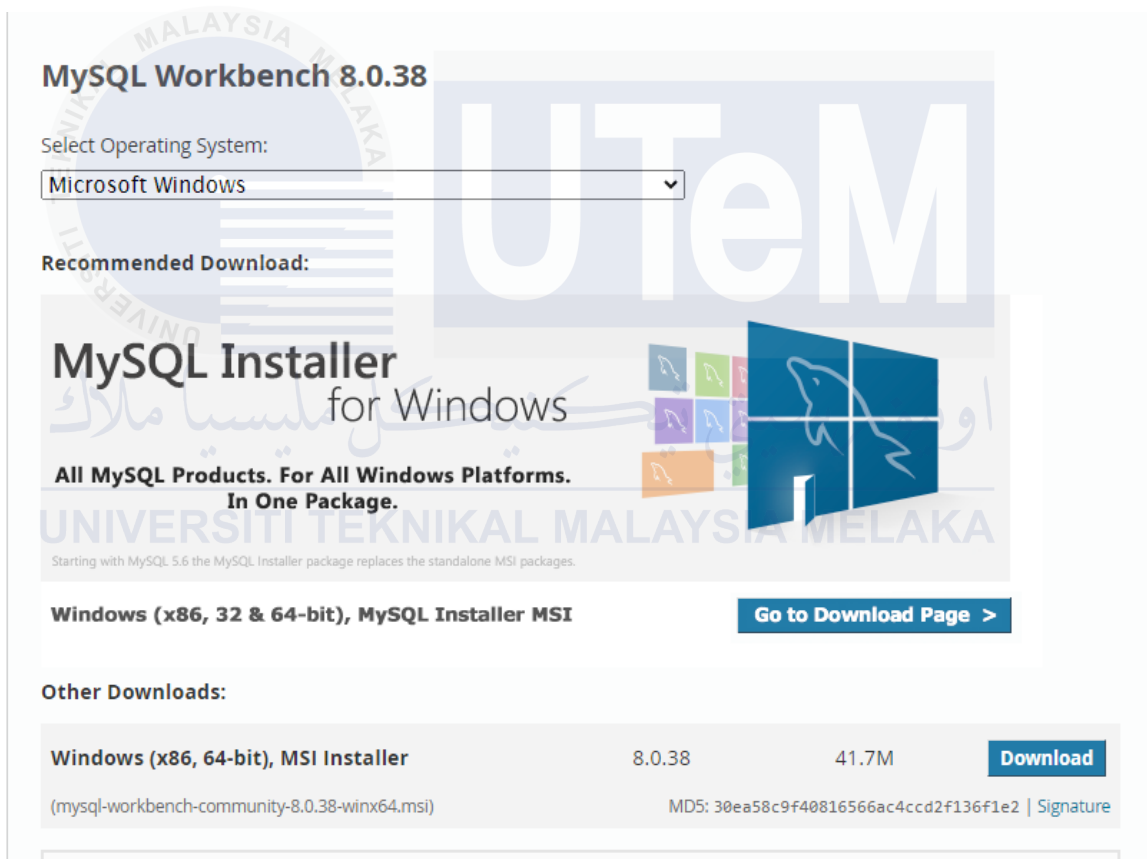


Figure 5.1 : Download MySQL installer for Windows

Step 2: After the download, open the installer as shown in Figure 5.2.

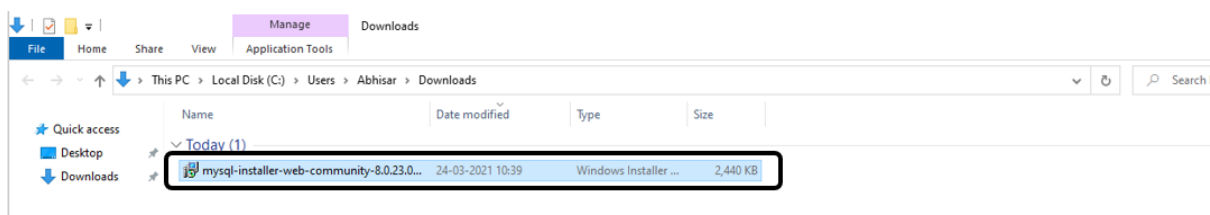


Figure 5.2 : MySQL installer

Step 3: Click Custom for installation of MySQL in Figure 5.3. Click on Next.

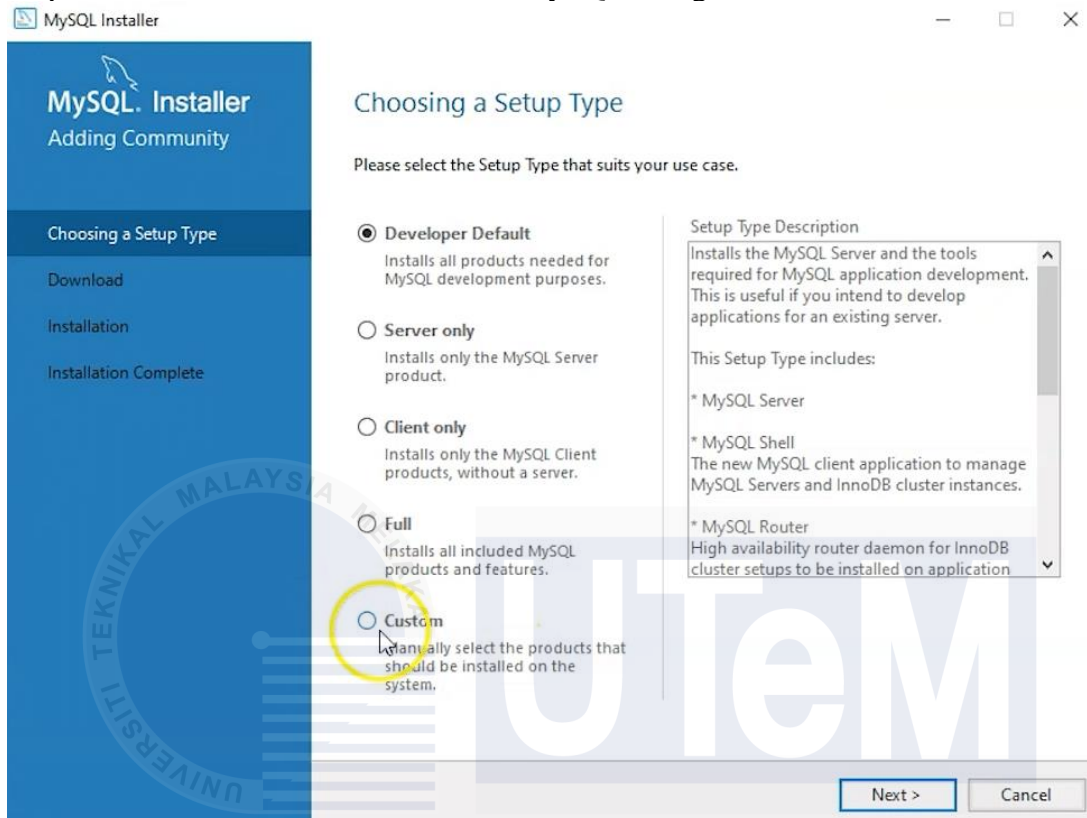


Figure 5.3 : Select Custom Setup Type

Step 4: Select MySQL Server, MySQL Workbench, and MySQL Shell as shown at Figure 5.4.

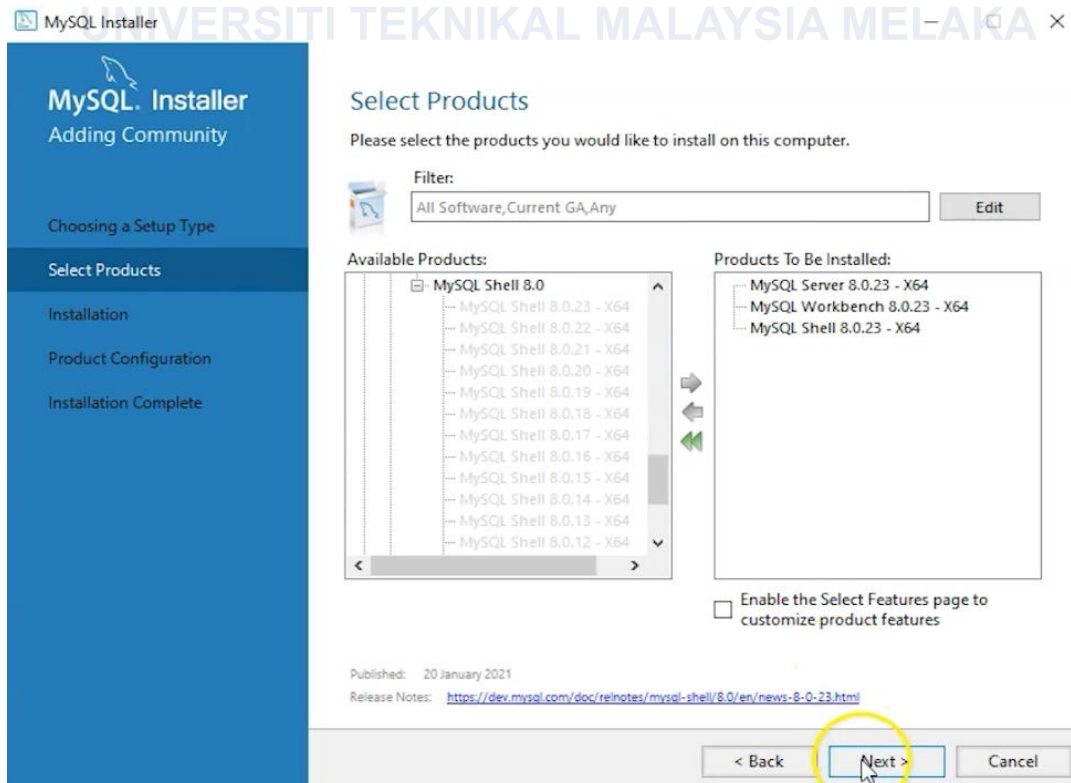


Figure 5.4 : Select MySQL Products

Step 5: Choose the server configuration type for the MySQL Server installation and click Next in Figure 5.5.

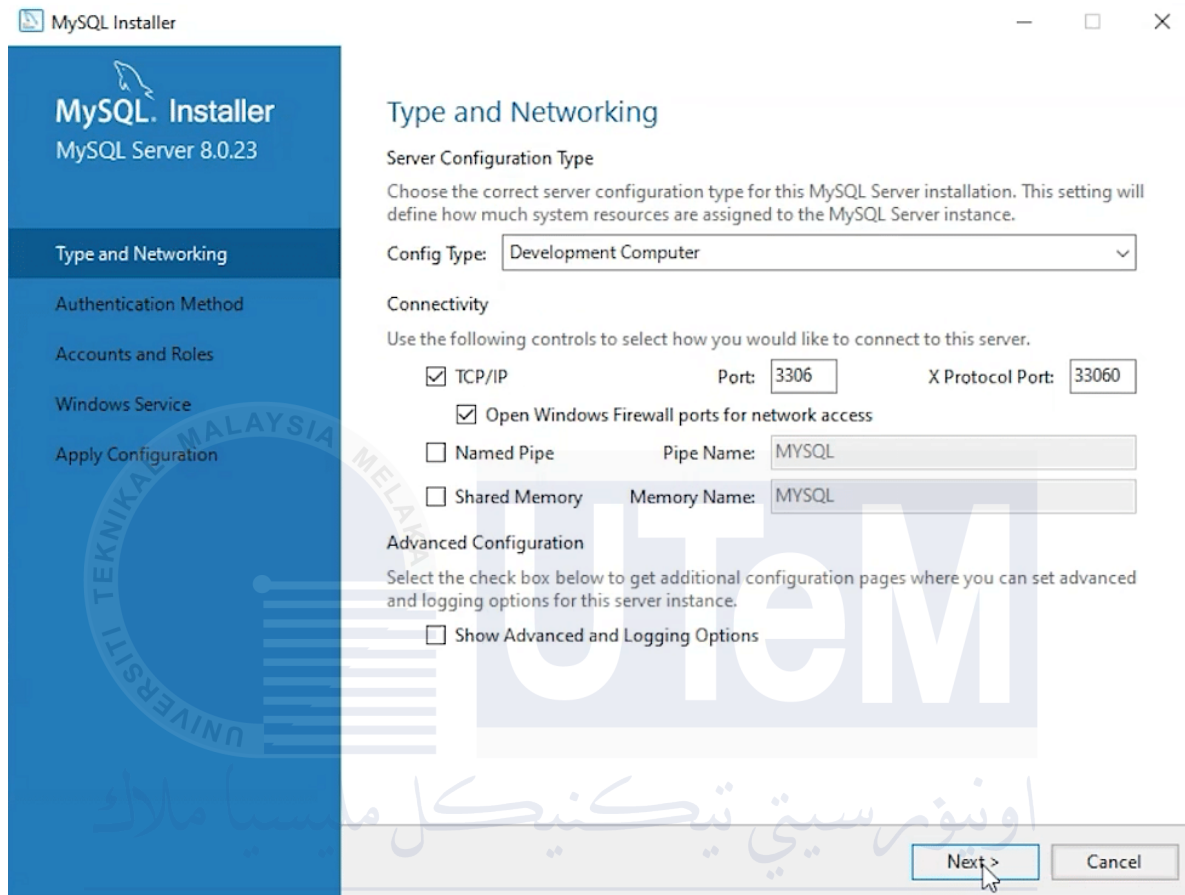


Figure 5.5 : Server Configuration Type

Step 6: Set the MySQL root password and click on Next in Figure 5.6.

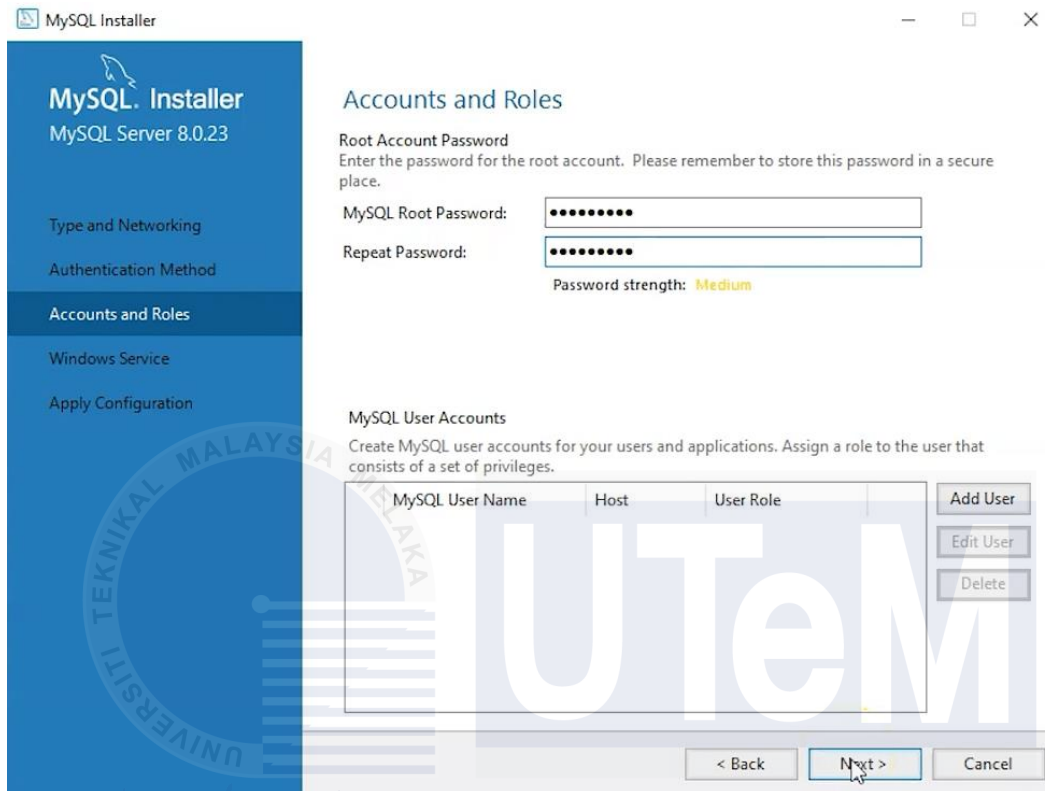


Figure 5.6 : MySQL Settings

Step 7: After the configuration in Figure 5.7 is done, click on finish.

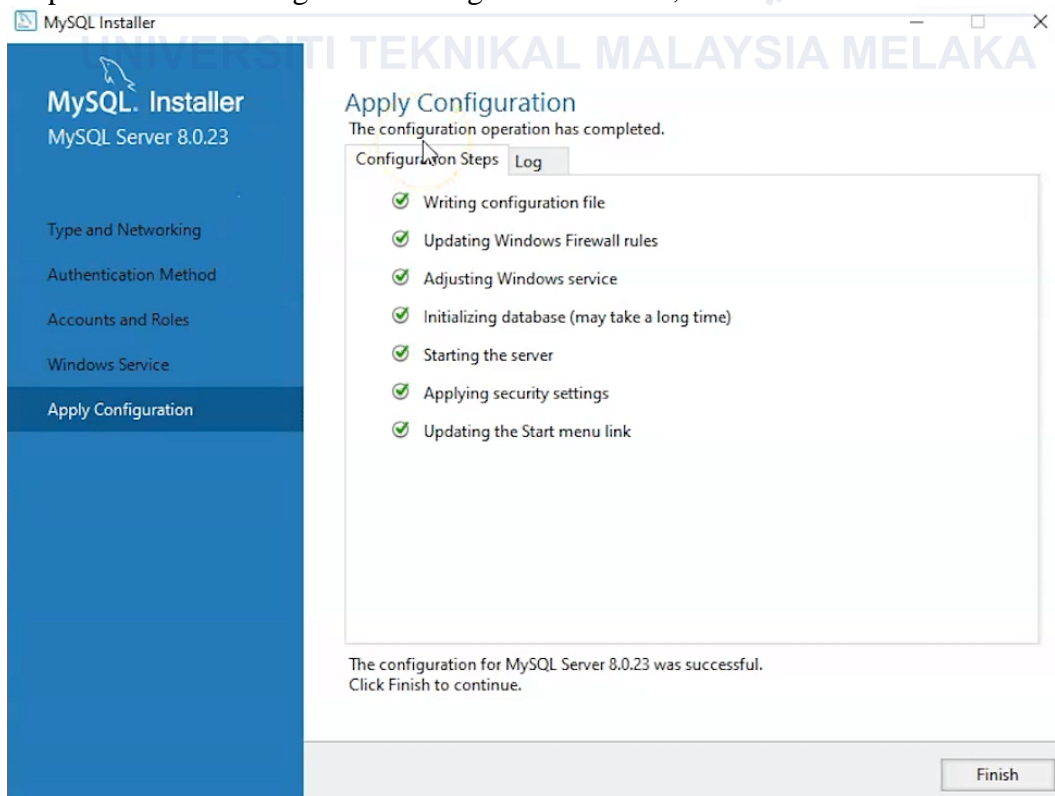


Figure 5.7 : Apply Configuration

Step 8: Launch the MySQL workbench and click the local instance in Figure 5.8.

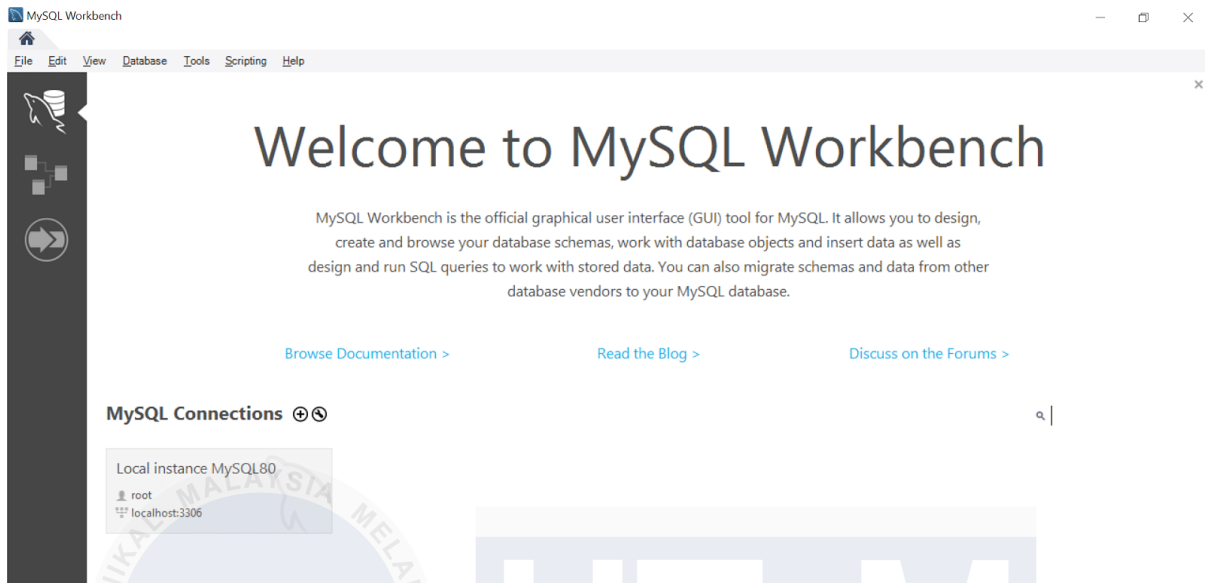


Figure 5.8 : MySQL workbench interface

Step 9: Login into the MySQL workbench with the password in figure 5.6.

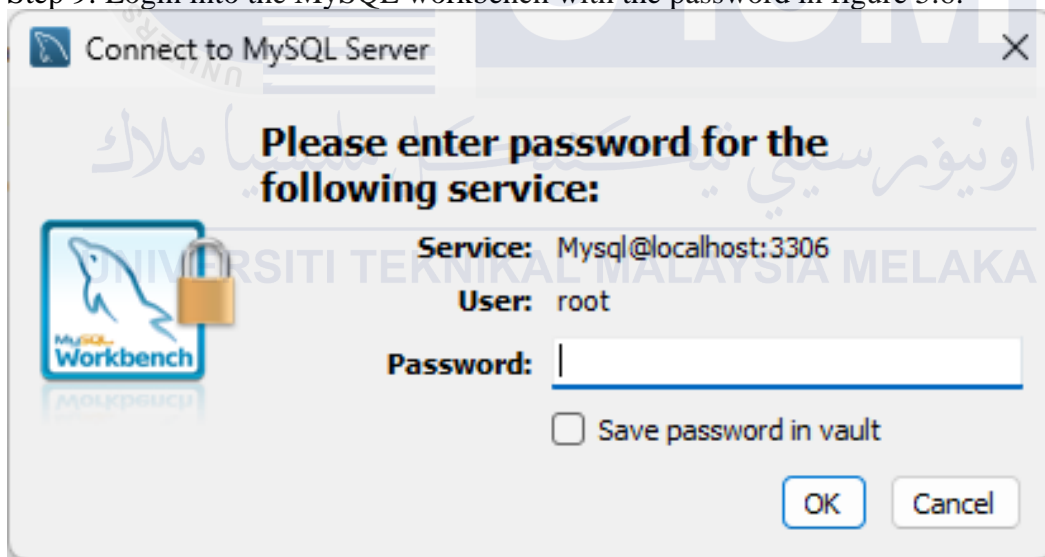


Figure 5.9 : Enter the root password

Step 10: Figure 5.10 shows the MySQL workbench page. Click the schema icon to create a schema.

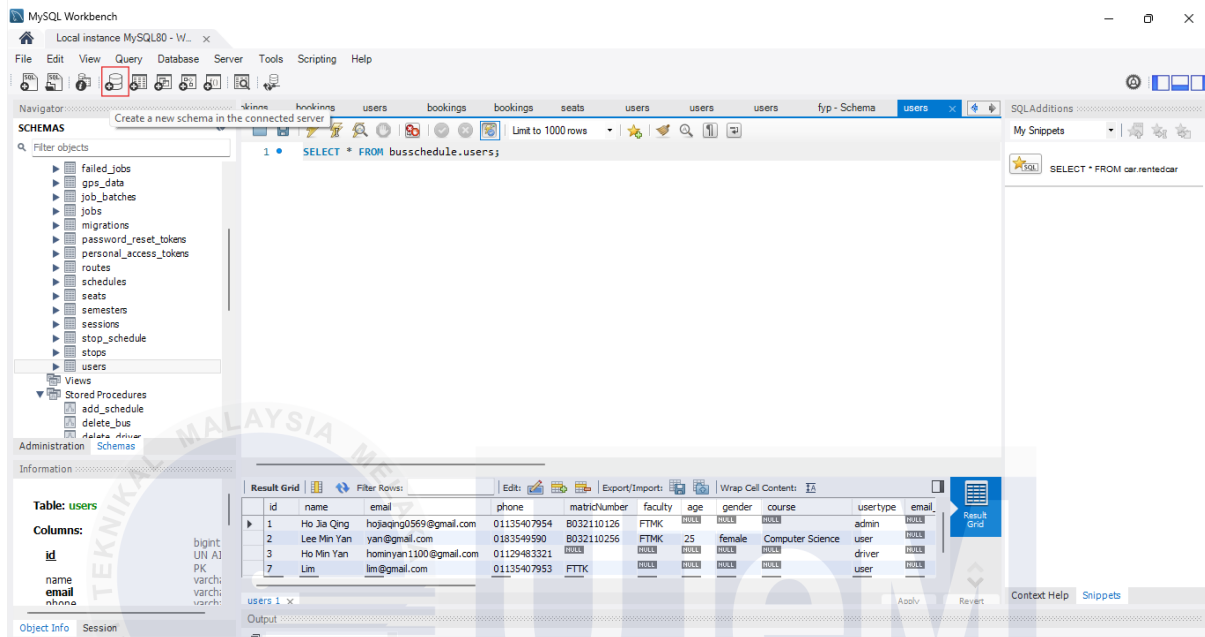


Figure 5.10 : Enter MySQL workbench main page

Step 11: The database schema was successfully constructed after entering the database name in the name column in Figure 5.11.

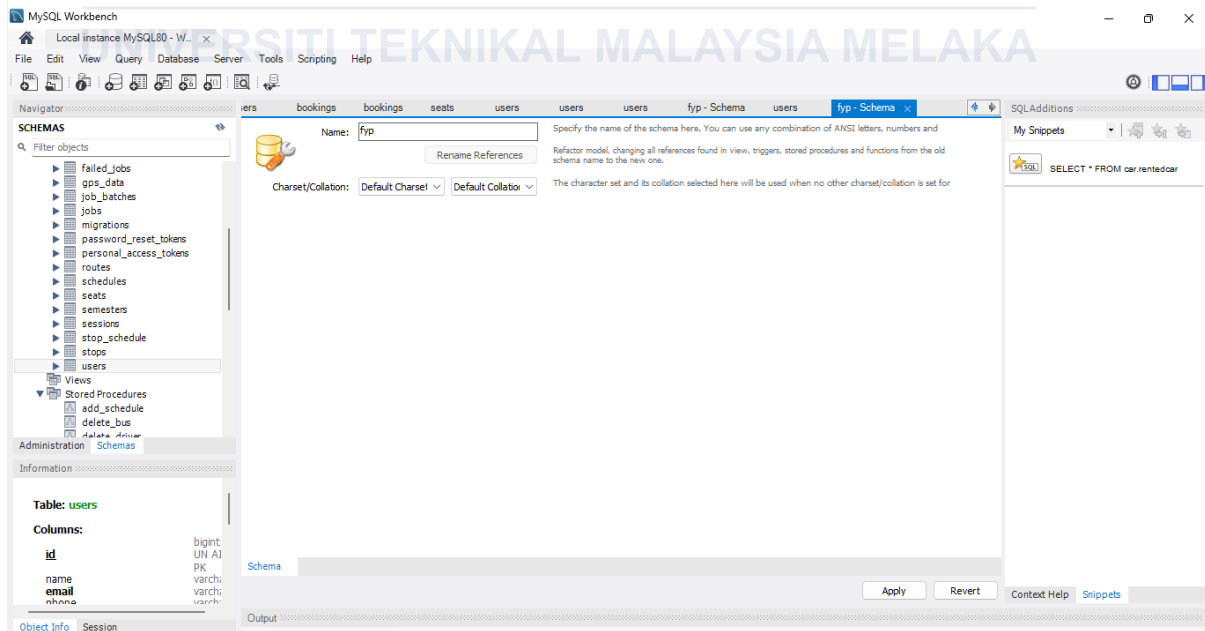


Figure 5.11 : Create database schema

Step 12: In Figure 5.12 click the Create Table button under the schema that created.

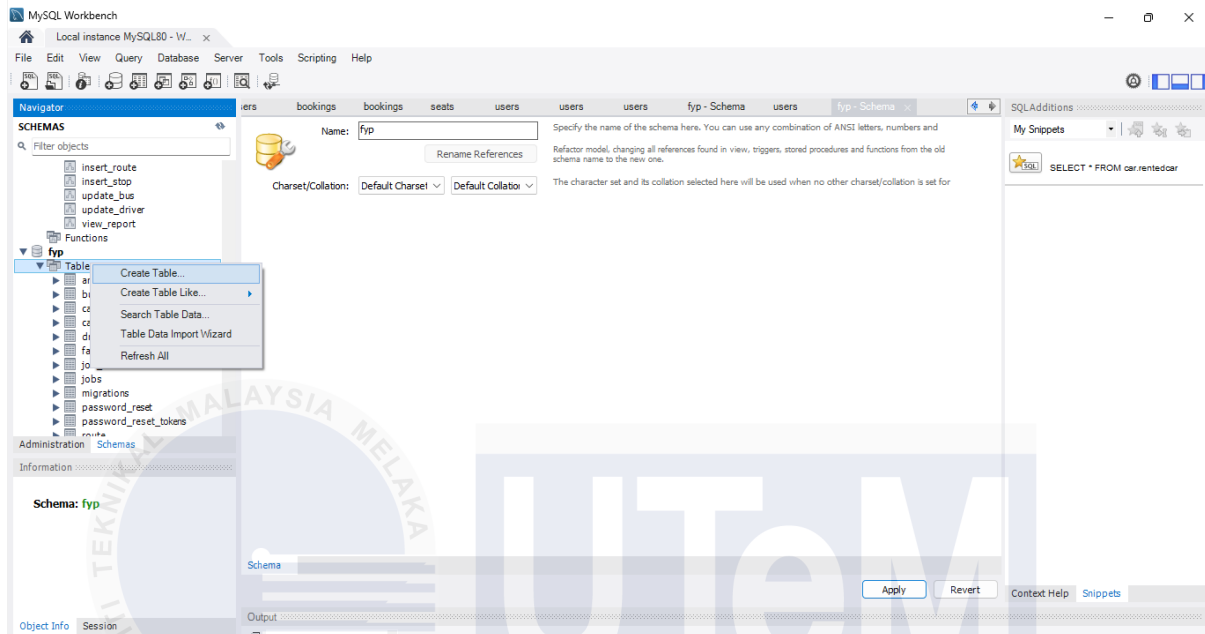


Figure 5.12 : Add New Table into Database

Step 13: Type the name of the database table into the assigned text box, fill in the text field with the name, type, length, and index of each attribute, and click Save. As shown in Figure 5.13, the insertion of attributes into the database table was completed successfully.

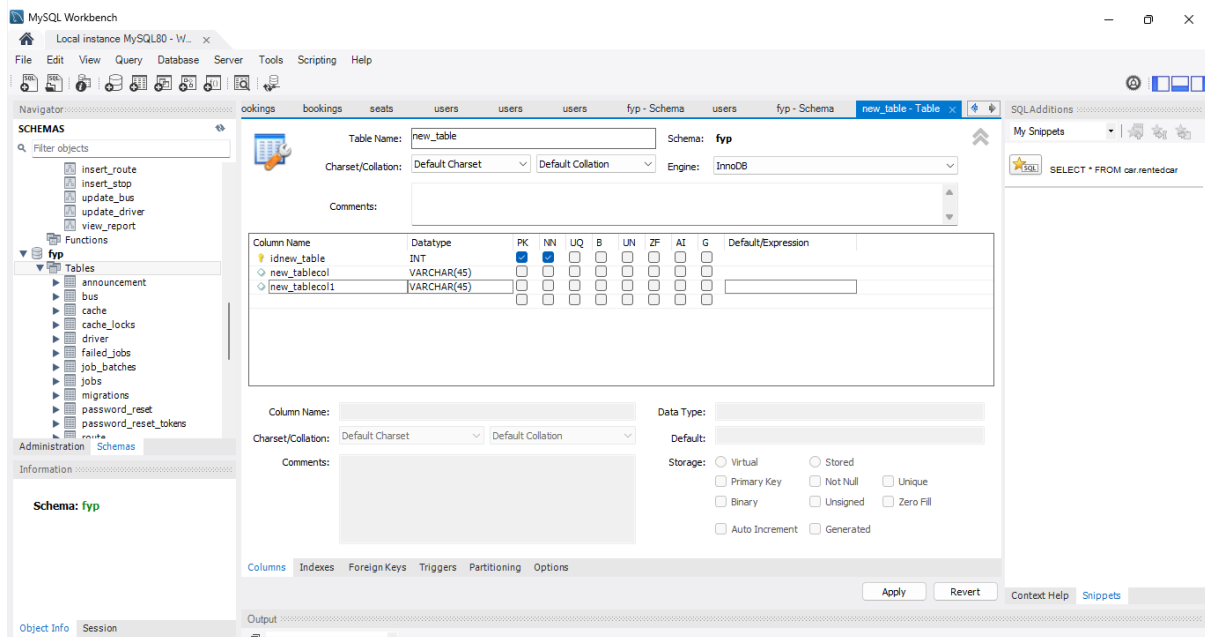


Figure 5.13 : Add Attribute into Table

5.3 Database Implementation

In the database implementation for the UTeM bus schedule management system, the database is used to evaluate database queries, including basic and complicated queries, aggregate functions, stored procedures, and triggers.

5.3.1 Data Definition Language (DDL)

Data Definition Language (DDL) pertains to the SQL commands utilised for the creation and modification of tables within a relational database. DDL statements are employed to build, modify, and remove objects within the database, such as tables, indexes, and triggers, for the UTeM bus scheduling management system.

5.3.1.1 Create Table Commands

Figure 5.14 to Figure 5.28 will show the database queries of each create table commands.

```
CREATE TABLE `announcements` (
  `AnnouncementID` bigint unsigned NOT NULL AUTO_INCREMENT,
  `Title` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `Description` varchar(100) COLLATE utf8mb4_unicode_ci NOT NULL,
  `DatePosted` date NOT NULL,
  `RouteID` bigint unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`AnnouncementID`),
  KEY `announcements_routeid_foreign` (`RouteID`),
  CONSTRAINT `announcements_routeid_foreign` FOREIGN KEY (`RouteID`)
REFERENCES `routes` (`RouteID`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

Figure 5.14 : Create Table Announcement


```

CREATE TABLE `bookings` (
  `BookingID` bigint unsigned NOT NULL AUTO_INCREMENT,
  `UserID` bigint unsigned NOT NULL,
  `ExternalBusID` bigint unsigned NOT NULL,
  `SeatNumber` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `BookingDate` datetime NOT NULL,
  `departure_date` date NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`BookingID`),
  KEY `bookings_userid_foreign` (`UserID`),
  KEY `bookings_externalbusid_foreign` (`ExternalBusID`),
  CONSTRAINT `bookings_externalbusid_foreign` FOREIGN KEY (`ExternalBusID`)
REFERENCES `external_buses` (`ExternalBusID`) ON DELETE CASCADE,
  CONSTRAINT `bookings_userid_foreign` FOREIGN KEY (`UserID`) REFERENCES
`users` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

Figure 5.15 : Create Table Booking

```

CREATE TABLE `buses` (
  `BusID` bigint unsigned NOT NULL AUTO_INCREMENT,
  `NumberPlate` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `Capacity` int NOT NULL,
  `DriverID` bigint unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`BusID`),
  KEY `buses_driverid_foreign` (`DriverID`),
  CONSTRAINT `buses_driverid_foreign` FOREIGN KEY (`DriverID`) REFERENCES
`drivers` (`DriverID`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=17 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

Figure 5.16 : Create Table Bus


```

CREATE TABLE `checkins` (
  `id` bigint unsigned NOT NULL AUTO_INCREMENT,
  `UserID` bigint unsigned NOT NULL,
  `checkin_time` timestamp NULL DEFAULT NULL,
  `checkout_time` timestamp NULL DEFAULT NULL,
  `status` enum('work','rest') COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT 'rest',
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `checkins_userid_foreign` (`UserID`),
  CONSTRAINT `checkins_userid_foreign` FOREIGN KEY (`UserID`) REFERENCES
`users` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

Figure 5.17 : Create Table Checkin

```

CREATE TABLE `departure_dates` (
  `DepartureDateID` bigint unsigned NOT NULL AUTO_INCREMENT,
  `departure_date` date NOT NULL,
  `ExternalBusID` bigint unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`DepartureDateID`),
  KEY `departure_dates_externalbusid_foreign` (`ExternalBusID`),
  CONSTRAINT `departure_dates_externalbusid_foreign` FOREIGN KEY (`ExternalBusID`)
REFERENCES `external_buses` (`ExternalBusID`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

Figure 5.18 : Create Table Departure_Date

```

CREATE TABLE `drivers` (
  `DriverID` bigint unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `license_number` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `phone` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`DriverID`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

Figure 5.19 : Create Table Driver

```

CREATE TABLE `external_buses` (
  `ExternalBusID` bigint unsigned NOT NULL AUTO_INCREMENT,
  `NumberPlate` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `Capacity` int NOT NULL,
  `Zone` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`ExternalBusID`),
  UNIQUE KEY `external_buses_numberplate_unique` (`NumberPlate`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

Figure 5.20 : Create Table External_Bus

```

CREATE TABLE `gps_data` (
  `id` bigint unsigned NOT NULL AUTO_INCREMENT,
  `UserID` bigint unsigned NOT NULL,
  `latitude` double NOT NULL,
  `longitude` double NOT NULL,
  `timestamp` timestamp NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `gps_data_userid_foreign` (`UserID`),
  CONSTRAINT `gps_data_userid_foreign` FOREIGN KEY (`UserID`) REFERENCES
`users` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

Figure 5.21 : Create Table GPS_Data

```

CREATE TABLE `routes` (
  `RouteID` bigint unsigned NOT NULL AUTO_INCREMENT,
  `Description` varchar(100) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  `Origin` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `OriginLatitude` double NOT NULL,
  `OriginLongitude` double NOT NULL,
  `Destination` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `DestinationLatitude` double NOT NULL,
  `DestinationLongitude` double NOT NULL,
  PRIMARY KEY (`RouteID`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

Figure 5.22 : Create Table Route

```

CREATE TABLE `schedules` (
  `ScheduleID` bigint unsigned NOT NULL AUTO_INCREMENT,
  `date` date NOT NULL,
  `Time` time NOT NULL,
  `BusID` bigint unsigned NOT NULL,
  `SemesterID` bigint unsigned NOT NULL,
  `RouteID` bigint unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`ScheduleID`),
  KEY `schedules_busid_foreign` (`BusID`),
  KEY `schedules_routeid_foreign` (`RouteID`),
  KEY `schedules_semesterid_foreign` (`SemesterID`),
  CONSTRAINT `schedules_busid_foreign` FOREIGN KEY (`BusID`) REFERENCES
`buses` (`BusID`),
  CONSTRAINT `schedules_routeid_foreign` FOREIGN KEY (`RouteID`) REFERENCES
`routes` (`RouteID`),
  CONSTRAINT `schedules_semesterid_foreign` FOREIGN KEY (`SemesterID`)
REFERENCES `semesters` (`SemesterID`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

Figure 5.23 : Create Table Schedule

```

CREATE TABLE `seats` (
  `SeatID` bigint unsigned NOT NULL AUTO_INCREMENT,
  `ExternalBusID` bigint unsigned NOT NULL,
  `SeatNumber` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `IsBooked` tinyint(1) NOT NULL DEFAULT '0',
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`SeatID`),
  KEY `seats_externalbusid_foreign` (`ExternalBusID`),
  CONSTRAINT `seats_externalbusid_foreign` FOREIGN KEY (`ExternalBusID`)
REFERENCES `external_buses` (`ExternalBusID`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=295 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

Figure 5.24 : Create Table Seat

```

CREATE TABLE `semesters` (
  `SemesterID` bigint unsigned NOT NULL AUTO_INCREMENT,
  `Name` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `Start_Date` date NOT NULL,
  `End_Date` date NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`SemesterID`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

Figure 5.25 : Create Table Semester

```

CREATE TABLE `stop_schedule` (
  `StopID` bigint unsigned NOT NULL,
  `ScheduleID` bigint unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`StopID`,`ScheduleID`),
  KEY `stop_schedule_scheduleid_foreign` (`ScheduleID`),
  CONSTRAINT `stop_schedule_scheduleid_foreign` FOREIGN KEY (`ScheduleID`)
REFERENCES `schedules` (`ScheduleID`),
  CONSTRAINT `stop_schedule_stopid_foreign` FOREIGN KEY (`StopID`)
REFERENCES `stops` (`StopID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

```

Figure 5.26 : Create Table Stop_Schedule

```

CREATE TABLE `stops` (
  `StopID` bigint unsigned NOT NULL AUTO_INCREMENT,
  `Name` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `Location` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `Latitude` decimal(10,7) DEFAULT NULL,
  `Longitude` decimal(10,7) DEFAULT NULL,
  `picture` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`StopID`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

Figure 5.27 : Create Table Stop

```

CREATE TABLE `users` (
  `id` bigint unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `email` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `phone` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `matricNumber` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `faculty` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `age` int DEFAULT NULL,
  `gender` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `course` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `usertype` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT 'user',
  `email_verified_at` timestamp NULL DEFAULT NULL,
  `password` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `two_factor_secret` text COLLATE utf8mb4_unicode_ci,
  `two_factor_recovery_codes` text COLLATE utf8mb4_unicode_ci,
  `two_factor_confirmed_at` timestamp NULL DEFAULT NULL,
  `remember_token` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `current_team_id` bigint unsigned DEFAULT NULL,
  `profile_photo_path` varchar(2048) COLLATE utf8mb4_unicode_ci DEFAULT
  NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `users_email_unique` (`email`)
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

```

Figure 5.28 : Create Table User

5.3.2 Data Manipulation Language

```
INSERT INTO semesters (Name, Start_Date, End_Date) VALUES
('Week 1 to Week 7', '2024-03-10', '2024-12-15'),
('Week 8 to Week 15', '2024-05-10', '2024-07-03');
```

Figure 5.29 : Insert Statement – insert data into table semester

```
UPDATE buses
SET NumberPlate = 'ABC1234', Capacity = 40
WHERE BusID = 1;
```

Figure 5.30 : Update Statement – update table bus

```
DELETE FROM drivers WHERE DriverID = 1;
```

Figure 5.31 : Delete Statement – delete table driver

```
SELECT drivers.name, drivers.license_number, drivers.phone, users.email,
checkins.checkin_time, checkins.checkout_time FROM drivers JOIN users ON
drivers.phone = users.phone JOIN checkins ON users.id = checkins.UserID;
```

Figure 5.32 : Select Statement – retrieve table driver

5.3.3 Stored Procedures

Table 5.1: Stored Procedures Query

Description	Stored Procedure Query
To insert a new schedule	<pre>CREATE DEFINER=`root`@`localhost` PROCEDURE `add_schedule`(IN p_date DATE, IN p_time VARCHAR(255), IN p_BusID INT, IN p_SemesterID INT, IN p_RouteID INT) BEGIN INSERT INTO schedules (date, Time, BusID, SemesterID, RouteID) VALUES (p_date, p_time,</pre>

	<pre>p_BusID, p_SemesterID, p_RouteID); END;</pre>
To insert a new bus	<pre>CREATE DEFINER=`root`@`localhost` PROCEDURE `insert_bus`(IN number_plate VARCHAR(255), IN capacity INT, IN driver_id INT) BEGIN INSERT INTO buses (NumberPlate, Capacity, DriverID) VALUES (number_plate, capacity, driver_id); END;</pre>
To insert a new route	<pre>CREATE DEFINER=`root`@`localhost` PROCEDURE `insert_route`(IN origin VARCHAR(50), IN origin_latitude DECIMAL(10, 6), IN origin_longitude DECIMAL(10, 6), IN destination VARCHAR(50), IN destination_latitude DECIMAL(10, 6), IN destination_longitude DECIMAL(10, 6), IN description VARCHAR(100)) BEGIN INSERT INTO routes (Origin, OriginLatitude, OriginLongitude, Destination, DestinationLatitude, DestinationLongitude, Description) VALUES (origin, origin_latitude, origin_longitude, destination, destination_latitude, destination_longitude, description); END;</pre>
To update an existing bus record in the buses table with new values for the number plate and capacity, based on the provided busID.	<pre>CREATE DEFINER=`root`@`localhost` PROCEDURE `update_bus`(IN busID BIGINT, IN numberPlate VARCHAR(255), IN capacity INT) BEGIN UPDATE buses SET NumberPlate = numberPlate, Capacity = capacity WHERE BusID = busID; END;</pre>

<p>To update an existing driver record in the drivers table with new values for the driver's name, license number, and phone number, based on the provided driver_id.</p>	<pre>CREATE DEFINER=`root`@`localhost` PROCEDURE `update_driver`(IN driver_id INT, IN driver_name VARCHAR(255), IN license_number VARCHAR(255), IN phone VARCHAR(15)) BEGIN UPDATE drivers SET name = driver_name, license_number = license_number, phone = phone WHERE DriverID = driver_id; END;</pre>
<p>To retrieve detailed information about a specific driver, including all driver details and the associated user's email, based on the provided driver_id</p>	<pre>CREATE DEFINER=`root`@`localhost` PROCEDURE `driver_detail`(IN driver_id BIGINT) BEGIN SELECT drivers.*, users.email FROM drivers JOIN users ON drivers.phone = users.phone WHERE drivers.DriverID = driver_id; END;</pre>
<p>To delete a bus record from the buses table based on the provided bus_id, retrieves and stores the associated DriverID, and returns a success message</p>	<pre>CREATE DEFINER=`root`@`localhost` PROCEDURE `delete_bus`(IN bus_id INT) BEGIN DECLARE driver_id INT; SELECT DriverID INTO driver_id FROM buses WHERE BusID = bus_id; DELETE FROM buses WHERE BusID = bus_id; SELECT CONCAT('Bus with BusID ', bus_id, ' deleted successfully.') AS Message; END;</pre>

5.3.4 Triggers

In UTeM Bus Scheduling Management System, triggers are used to maintain the integrity of the information on the database.

Table 5.2: Triggers Query

Explanation	Trigger Query
To prevent users from making multiple bookings by checking if a user already has an existing booking before allowing a new one to be inserted.	<pre>CREATE DEFINER=`root`@`localhost` TRIGGER `prevent_multiple_bookings` BEFORE INSERT ON `bookings` FOR EACH ROW BEGIN IF EXISTS (SELECT 1 FROM bookings WHERE UserID = NEW.UserID) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'You can only book one bus.'; END IF; END;</pre>
To prevent multiple check-ins by checking if the user already has an active check-in before allowing a new check-in to be inserted	<pre>CREATE DEFINER=`root`@`localhost` TRIGGER `prevent_multiple_checkins` BEFORE INSERT ON `checkins` FOR EACH ROW BEGIN DECLARE active_checkin_count INT; SELECT COUNT(*) INTO active_checkin_count FROM checkins WHERE UserID = NEW.UserID AND checkout_time IS NULL; IF active_checkin_count > 0 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'You are already checked in.'; END IF; END;</pre>
To ensure that a check-out operation is only allowed if there is an active check-in (i.e., a check-in where checkout_time is NULL).	<pre>CREATE DEFINER=`root`@`localhost` TRIGGER `prevent_invalid_checkout` BEFORE UPDATE ON `checkins` FOR EACH ROW BEGIN DECLARE active_checkin_count INT; IF OLD.checkout_time IS NULL AND NEW.checkout_time IS NOT NULL THEN SELECT COUNT(*) INTO active_checkin_count FROM checkins WHERE UserID = OLD.UserID AND checkout_time IS NULL; IF active_checkin_count = 0 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No check-in found to check out.'; END IF; END IF; END;</pre>
is designed to automatically delete the GPS data associated with a user after they check out, provided that the check-in status was previously 'work'.	<pre>CREATE DEFINER=`root`@`localhost` TRIGGER `delete_gps_data_after_checkout` AFTER UPDATE ON `checkins` FOR EACH ROW BEGIN IF NEW.checkout_time IS NOT NULL AND OLD.status = 'work' THEN</pre>

	DELETE FROM gps_data WHERE UserID = NEW.UserID; END IF; END;
To ensure that before an UPDATE is made to the external_buses table, the NumberPlate is checked against existing entries in the buses table. If a duplicate NumberPlate is found, the update is blocked, and an error message "Duplicate NumberPlate found in buses table" is raised, preventing potential conflicts or inconsistencies in the NumberPlate data across the two tables	CREATE DEFINER=`root`@`localhost` TRIGGER `check_number_plate_update` BEFORE UPDATE ON `external_buses` FOR EACH ROW BEGIN IF EXISTS (SELECT 1 FROM buses WHERE NumberPlate = NEW.NumberPlate) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Duplicate NumberPlate found in buses table'; END IF; END;

5.3.5 Data Loading Process

The UTeM Bus Scheduling Management System is a system that stores and processes data to give drivers and users relevant information. All the data such as schedule details, booking details, ticket details and driver location was stored in the database. In UTeM Bus Scheduling Management System, the Extract, Transform, Load (ETL) process is utilized to manage and process large volumes of both unstructured and structured data. **Extraction** involves collecting and reading data from various database sources. **Transformation** encompasses converting and preparing the data for storage in a different database. **Loading** refers to the process of inserting the transformed data into the target database.

5.4 Implementation Status

Every module's description, the time needed to finish it, and the date of completion are all included in the implementation status. This summary monitors the implementation process's development.

Module	Description	Duration to complete/days	Date completes
User Authentication Module	To allow users to create secure accounts with passwords, ensuring that their personal information is protected	10	25-04-2024

Searching and Reporting Module	To allow users to search the schedule based on the location and time and generate pdf from the page.	10	4-05-2024
Notification Module	To allow users to receive notification in real-time when administrator put notifications	15	13-05-2024
Admin Dashboard Module	To allow administrator to manage the data related bus scheduling system based on the web -based system	15	28-05-2024
Tracking Module	To allow users can track the location of the drivers	15	12-06-2024
Booking Module	To allow user can make booking at the web-based system	15	28-06-2024

5.5 Conclusion

In summary, this chapter outlines the setup of the website development environment, detailing the procedures for installing MySQL server, MySQL workbench, Laravel, and MySQL on a Windows platform. It also covers the implementation of the database to manage the system's processes. Additionally, the system will be developed based on business logic, incorporating Data Definition Language (DDL) and Data Manipulation Language (DML) commands, as well as stored procedures, triggers, and the creation of database tables and constraints.

CHAPTER VI



6.1 Introduction

Testing is a process designed to assess the functionality of a software application to determine whether it meets the specified requirements and to identify any defects, ensuring the system is as error-free as possible. This chapter focuses on the verification and validation of the UTeM Bus Scheduling Management System. The primary objectives of testing the system are:

- i. To demonstrate that the system meets the user requirements.
- ii. To identify any bugs or faults in UTeM Bus Scheduling Management System using various testing strategies.

Furthermore, a critical stage in the Database Life Cycle (DBLC) is system testing. A test plan that describes the test's structure, timing, and setting is part of the system's testing process.

6.2 Test Plan

A test plan is a technical report that describes the objectives, scope, personnel, software, hardware, test schedule, and deliverables of the test. It provides a thorough

description of the system's operations and procedures, outlining how each component will be evaluated to make sure the system works as intended, find any defects, and determine its limitations.

6.2.1 Test Organization

In the UTeM Bus Scheduling Management System, the test organization includes three types of users: students, drivers, and administrators. Every kind of user will have both functional and non-functional needs checked. Table 6.1 shows the testing procedures that these three users go through according to their different roles.

Table 6.1 : User Responsibilities List

Tester ID	Users	Responsibilities
T1	Student	<ul style="list-style-type: none"> • Testing the system by using the provided test script • Examining the student/user module • Finding bugs and defects
T2	Driver	<ul style="list-style-type: none"> • Testing the system by using the provided test script • Examining the driver module • Finding bugs and defects
T3	Administrator	<ul style="list-style-type: none"> • Testing the system by using the provided test script • Examining the admin module

		<ul style="list-style-type: none"> Finding bugs and defects
--	--	--

6.2.2 Test Environment

The test environment consists of the hardware, software, operating system, tools, and network configurations required for the testing teams to execute test cases. The hardware and software requirements for the UTeM Bus Scheduling Management System test environment are listed in Tables 6.2 and 6.3.

Table 6.2 : Test Environment

Environment Details	Description
Laptop	Acer Nitro 5
Processor	Core i5 Intel
Mouse and Keyboard	VGN
Random Access Memory (RAM)	16 GB

Table 6.3 : Test Environment Software List

Environment	Description
Database	MySQL To administer data within the database table operating on a server
Web Server	Built-in php environment server Php artisan server (cmd line)
Operating System/ Platform	Windows 11 To administer the resources for computer hardware and software, and then provide the service or instrument needed to run computer programs.
Web Browser	Google Chrome To run the Laravel source code and assess the operation of the system interface
Visual Studio Code	To manage Laravel framework
Microsoft Word 2024 / PowerPoint 2024	To draft the completed report and make a PowerPoint show.

6.2.3 Test Schedule

A test schedule serves as a summary which lists the most important test deadlines and milestones. To guarantee that system testing is completed on time, test activities are scheduled according to predetermined dates. Estimating dates and making necessary adjustments are part of creating the test schedule. The test schedule for the UTeM Bus Scheduling Management System development is shown in Table 6.4.

Table 6.4 : Test Schedule

Testing Task	Testing Activity	Start Date	End Date
Login	Unit Testing, Integration Testing and User acceptance	15-7-2024	18-7-2024
Driver Registration	Unit Testing, Integration Testing and User acceptance	19-7-2024	21-7-2024
Add Bus	Unit Testing, Integration Testing and User acceptance	23-7-2024	25-7-2024
Manage User	Unit Testing, Integration Testing and User acceptance	26-7-2024	27-7-2024
Manage Route	Unit Testing, Integration Testing and User acceptance	28-7-2024	29-7-2024
Manage Bus Schedule	Unit Testing, Integration Testing and User acceptance	30-7-2024	1-8-2024
Manage Announcement	Unit Testing, Integration Testing and User acceptance	3-8-2024	6-8-2024
External Bus Registration	Unit Testing, Integration Testing and User acceptance	7-8-2024	12-8-2024
Searching Schedule	Unit Testing, Integration Testing and User acceptance	15-8-2024	16-8-2024

Booking Seats	Unit Testing, Integration Testing and User acceptance	17-8-2024	20-8-2024
Driver Check-in	Unit Testing, Integration Testing and User acceptance	21-8-2024	22-8-2024
Driver Check-out	Unit Testing, Integration Testing and User acceptance	23-8-2024	24-8-2024

6.3 Test Strategy

A test strategy is a collection of instructions that describes test design and establishes which modules should be tested, the method to be used, and if testing is required.

A software testing technique called "black box" testing assesses a system's functioning without revealing its core code structure. Typically, testers who are not familiar with implementation or programming carry out this kind of testing. The goal of black box testing is to evaluate the system's functional and non-functional needs. Usually, it is used in high-level testing phases like user acceptance and system testing.

On the other side, White Box Testing entails testing the code's internal structure while having a solid understanding of its architecture. Software developers, who require implementation and programming skills, implement this strategy. The focus of white box testing is on examining logic, branching, and code organisation. High-level testing stages like unit and integration testing frequently use it.

Unit Testing is a software testing level where individual components or units of the software are tested to ensure that each performs as intended. In UTeM Bus Scheduling Management System, unit modules such as user login and user registration were tested to verify the functionality. For integration testing, it involves combining and testing individual units as a group to identify faults in their interactions. In UTeM Bus Scheduling Management, this means that after user fill out the registration form, the details of the username and password are recorded, and they are directed to the main page of the website. The system must make sure user able to check the schedules, bus tracking and booking status .

System testing assesses the integrated software as a whole to make sure it satisfies predetermined standards.. In UTeM Bus Scheduling Management, the entire system flow was tested to confirm that it operates according to the requirements outlined in Figure 3.2.

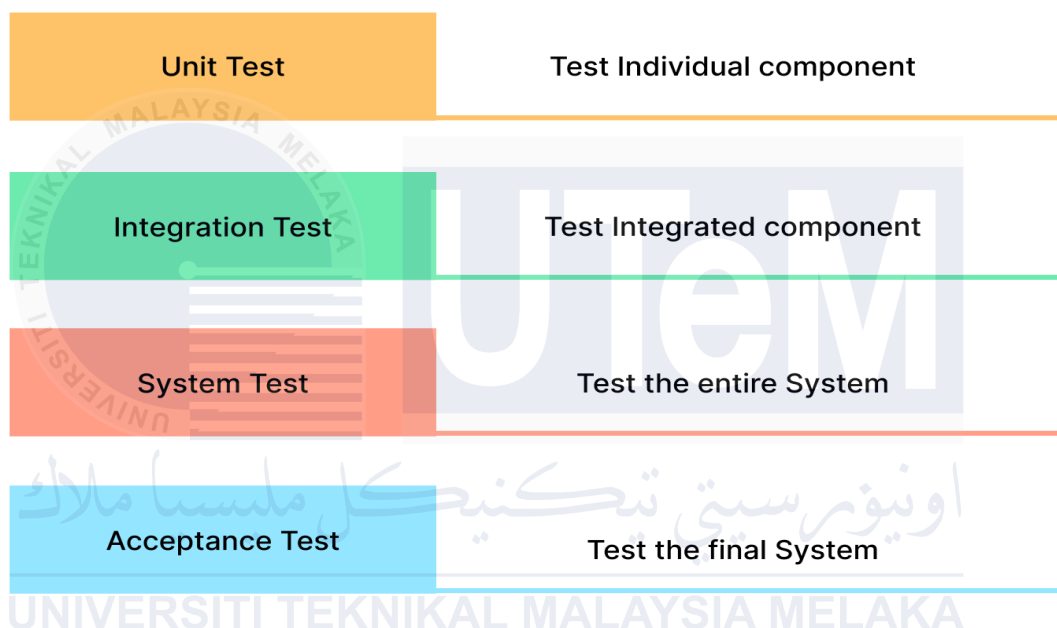
User Acceptance Testing is performed by the user to verify that the system meets the agreed-upon requirements.

Table 6.5 : Type of test and test design techniques for white box and black box testing

	White Box	Black Box
Type of Test	<p><u>Unit Testing</u> Evaluating discrete parts or components of the software to verify their functionality aligns with expectations.</p> <p><u>Integration Testing</u> Evaluating the interaction of integrated units/modules to detect faults in their interfaces.</p>	<p><u>Functional Testing</u> Evaluating the software's functions in accordance with the defined requirements.</p> <p><u>Non-Functional Testing</u> Evaluating factors include performance, usability, and security.</p> <p><u>Acceptance Testing</u> Evaluating the software's compliance with acceptance criteria and its readiness for deployment.</p>
Test Design Techniques	<ul style="list-style-type: none"> • Statement Coverage • Branch Coverage • Path Coverage • Loop Testing 	<ul style="list-style-type: none"> • Equivalence Partitioning • Boundary Value Analysis • Decision Table Testing

		<ul style="list-style-type: none"> • State Transition Testing • Use Case Testing
--	--	--

Figure 6.1 : Level of Testing



The many forms of testing and the associated test design methods for both White Box and Black Box testing are shown in Table 6.5. Only White Box Testing is used in the UTeM Bus Scheduling Management System.

For White Box Testing, techniques such as unit testing and integration testing are employed. These techniques focus on verifying the correctness of data input types and ensuring proper data storage and transfer between modules.

Equivalence partitioning analysis is utilised in Black Box Testing. Through the use of this approach, different input situations may be effectively tested by dividing the software unit's input data into divisions of equal data.

6.3.1 Classes of Tests

The test class descriptions that are being implemented on the UTeM Bus Scheduling Management System are listed below in various formats.

i. Error Handling Test

Ensuring that users may only submit suitable and correct data into the input forms is the aim of this testing. For example, the "Age" field for profile updates is restricted to numeric values only. The purpose of this testing is to ensure that users enter accurate data before it is entered into the database and to indicate errors with missing or wrong data with an error message.

ii. Security Test

During the login procedure, security testing will be used to confirm and validate the user's email address and password. In the system, the password that created is hashed and stored by database. When user enter incorrect password, error message will be returned.

iii. Integration Test

The purpose of the integration test is to verify that, depending on the key in the data, the system appropriately enters the data into the database. It is accomplished by going through every menu item in the user interface. This testing was done in the UTeM Bus Scheduling Management System to ensure that the data are correct and to focus on the data flow between the modules. For example, only the driver check-in successfully, the data location only can appear in map at user interface.

6.4 Test Design

To define and construct test suites for software testing, test designs are described. The goal of the test design is to ensure that the specifications are fulfilled in a way that aligns with the needs and desires of the customer. The test description and test data are the two components that make up the test design.

6.4.1 Test Description

Every module test case has a test description that details the desired output result, the method by which each test case is identified, the kind of testing, the preconditions, the test requirements, and the step-by-step process. Tables 6.6 through 6.13 provide a detailed explanation of the test, broken down by system module.

Table 6.6 : Test Description for User Login (student)

Test ID	T001- Login			
Testing Type	Unit testing and integration testing			
Test Strategy	White box Testing			
Text Class	Security and error handling testing			
Test Case ID	Test Conditions	Pre-condition	Step	Expected Output
TC1_1	Verify that the login process is functioning properly by making sure that access is only allowed if the password and email given are true.	The user's password and email address are correct.	<ol style="list-style-type: none"> 1. Go to the login screen. 2. Send a legitimate email. 3. Enter a working password. 4. Press the Login button. 	Authentication successful
TC1_2	Verify the functionality of the login process. Unavailable if the email or password fields are null.		<ol style="list-style-type: none"> 1. Go to the login screen. 2. Press the Login button 	Failed to log in. Show the error message "Please fill up this field"
TC1_3	Verify the functionality of the login process. If the password and email given are not valid, access is prohibited.		<ol style="list-style-type: none"> 1. Access the login page. 2. Submit a legitimate email. 3. Submit a working password. 4. Press the Login button. 	Failed to log in. Show the failure message "These credentials do not match our records"

Table 6.7 : Test Description of Registration for Driver Account

Test ID	T002- Register Driver account		
Testing Type	Unit testing and integration testing		
Test Strategy	White box Testing		
Pre-condition	User must login as administrator		
Test Case ID	Test Conditions	Step	Expected Output
TC2_1	Verify that when all the given data is valid, you can use the insert driver account function.	<ol style="list-style-type: none"> 1. Open the page for adding drivers. 2. Fill in the appropriate data type in the fields labelled "Name," "License Number," "phone," "email," "password," and "confirm password." 3. Click the Add Driver button 	Insert new driver account successful
TC2_2	Verify that when the input field is blank, the insert driver account function is not accessible.	<ol style="list-style-type: none"> 1. Open the page for adding drivers. 2. Click the Add Driver button 	Insert driver account failed. Show the failure message "Please fill out this field" at the input text field
TC2_3	Verify that when certain input data types are invalid, the driver account insertion function is deactivated.	<ol style="list-style-type: none"> 1. Access the driver page. 2. Input the correct data types into the "Name," "License Number," "Phone," "Email," "Password," and "Confirm Password" fields. 3. Enter invalid "email" 4. Click on Add Driver button 	Insert driver account failed. Show the failure message "Please include an '@' in the email address.
TC2_4	Validate the availability of the update driver account feature.	<ol style="list-style-type: none"> 1. Navigate the edit driver page 2. Update the "License Number" 3. Click on Update Driver button 	Update the driver account successful

Table 6.8 : Test Description of Add Bus for Driver

Test ID	T003- Add Buses for Driver		
Testing Type	Unit testing and integration testing		
Test Strategy	White box Testing		
Pre-condition	User must login as administrator		
Test Case ID	Test Conditions	Step	Expected Output
TC3_1	Verify that when all input data are legitimate, the bus insertion function is available.	<ol style="list-style-type: none"> 1. Navigate the "Add Buses" page. 2. Enter the correct data types into the "Number Plate" and "Capacity" input fields. 3. Select the "Add Buses" button. 	Insert new bus successful
TC3_2	If there is nothing entered in the input fields, confirm that the bus insertion function is not accessible.	<ol style="list-style-type: none"> 1. Navigate the "Add Buses" page. 2. Select the "Add Buses" button without entering any data. 	Insert buses failed. Show the failure message "Please fill out this field" at the "Capacity" input text field
TC3_3	Verify that the bus update function is available and operating as intended.	<ol style="list-style-type: none"> 1. Navigate the "Edit Buses" page. 2. Update the "Capacity" field. 3. Select the "Update Bus" button. 	Update the bus successful

Table 6.9 : Test Description of User Content

Test ID	T004- Manage User		
Testing Type	Unit testing and integration testing		
Test Strategy	White box Testing		
Pre-condition	User must login as administrator		
Test Case ID	Test Conditions	Step	Expected Output
TC4_1	Verify that when all the supplied data is legitimate, the user update function is usable.	<ol style="list-style-type: none"> 1. Navigate the "User Content" page. 2. Enter valid data into the "Name," "Matric Number," "Age," "Phone," "Email," "Faculty," "Course," and "Gender" input fields. 3. Select the "Update" button. 	Update user detail successful
TC4_2	Validate that the user update function is unavailable if entered input data types are invalid.	<ol style="list-style-type: none"> 1. Navigate the "User Content" page. 2. Enter valid data into the "Name," "Matric Number," "Age," "Phone," "Faculty," "Course," and "Gender" fields. 3. Enter an invalid email in the "Email" field. 4. Select the "Update" button. 	Update user detail failed. Show the failure message "Please include an '@' in the email address."
TC4_3	Validate that the user update function is available even if some input data remain unchanged.	<ol style="list-style-type: none"> 1. Navigate to the "User Content" page. 2. Enter the same name for the user as is currently in use. 3. Select the "Update" button. 	Update user detail failed. Show the failure message "The provided name is already in use."

Table 6.10 : Test Description of Bus Route

Test ID	T005- Add Route		
Testing Type	Unit testing and integration testing		
Test Strategy	White box Testing		
Pre-condition	User must login as administrator		
Test Case ID	Test Conditions	Step	Expected Output
TC5_1	Verify that when all the supplied data is valid, the route insertion function is available.	<ol style="list-style-type: none"> 1. Navigate the "Add Route" page. 2. Enter valid data into the "Origin," "Destination," and "Description" input fields, and select locations on the map for both origin and destination. 3. Select the "Add Route" button. 	Insert new route successful
TC5_2	Validate that the route insertion function is unavailable if no data is entered into the input fields.	<ol style="list-style-type: none"> 1. Navigate the "Add Route" page. 2. Select the "Add Route" button without entering any data. 	Insert route failed. Show the failure message "Please fill out this field" at the "origin" input text field
TC5_3	Validate that the route insertion function is unavailable if no location is selected on the map.	<ol style="list-style-type: none"> 1. Navigate to the "Add Route" page. 2. Enter valid data into the "Origin," "Destination," and "Description" fields. 3. Do not select a location on the map. 4. Select the "Add Route" button. 	Insert route failed. Show the failure message "The origin latitude, origin longitude, destination latitude, destination longitude are required."

Table 6.11 : Test Description of Bus Schedule

Test ID	T006- Add Bus Schedule		
Testing Type	Unit testing and integration testing		
Test Strategy	White box Testing		
Pre-condition	User must login as administrator		
Test Case ID	Test Conditions	Step	Expected Output
TC6_1	Verify that when all the supplied data is correct, the schedule insertion function is available.	<ol style="list-style-type: none"> 1. Navigate the "Add Bus" page. 2. Enter valid data into the "Time" and "Date" fields. 3. Select the available bus, semester, and route. 4. Select the "Add Schedule" button. 	Insert new bus schedule successful
TC6_2	Validate that the schedule insertion function is unavailable if no data is entered into the input fields.	<ol style="list-style-type: none"> 1. Navigate to the "Add Schedule" page. 2. Select the "Add Schedule" button without entering any data. 	Insert schedule failed. Show the failure message "Please fill out this field" at the "date" input text field
TC6_3	If the date entered is past due, confirm that the schedule insertion feature is not accessible.	<ol style="list-style-type: none"> 1. Navigate to the "Add Schedule" page. 2. Enter a past date into the "Date" field. 3. Select the "Add Schedule" button. 	Insert bus schedule failed. The previous date cannot be selected.
TC6_4	Validate that the schedule update function is accessible and operational.	<ol style="list-style-type: none"> 1. Navigate to the "Update Schedule" page. 2. Update the "Date" field. 3. Select the "Update Schedule" button. 	Update bus schedule successful

Table 6.12 : Test Description of Manage Stop

Test ID	T007- Manage Stop		
Testing Type	Unit testing and integration testing		
Test Strategy	White box Testing		
Pre-condition	User must login as administrator		
Test Case ID	Test Conditions	Step	Expected Output
TC7_1	Verify that when all the supplied data is legitimate, you may access the stop insertion function.	<ol style="list-style-type: none"> 1. Navigate the "Add Stop" page. 2. Enter valid data into the "Stop Name" and "Location" fields. 3. Choose a picture for the stop. 4. Select the "Add Stop" button. 	Insert new stop information successful
TC7_2	Validate that the stop insertion function is unavailable if no data is entered into the input fields.	<ol style="list-style-type: none"> 1. Navigate the "Add Stop" page. 2. Select the "Add Stop" button without entering any data. 	Insert stop failed. Display the failure message "Please fill this field" appears in the text box for "name."

Table 6.13 : Test Description of Announcement

Test ID	T008- Add Announcement		
Testing Type	Unit testing and integration testing		
Test Strategy	White box Testing		
Pre-condition	User must login as administrator		
Test Case ID	Test Conditions	Step	Expected Output
TC8_1	Verify that when all of the supplied data is legitimate, the announcement insertion function is available.	<ol style="list-style-type: none"> 1. Navigate to the "Add Announcement" page. 2. Enter valid data into the "Title," "Date," and "Description" fields. 3. Select an available route. 4. Select the "Create Announcement" button. 	Insert new announcement successful
TC8_2	Validate that the announcement insertion function is unavailable if no data is entered into the input fields.	<ol style="list-style-type: none"> 1. Navigate to the "Add Announcement" page. 2. Select the "Create Announcement" button without entering any data. 	Insert announcement failed. Show the failure message "Please fill out this field" at the "title" input text field
TC8_3	Validate that the announcement insertion function is unavailable if the date provided is in the past.	<ol style="list-style-type: none"> 1. Navigate to the "Add Announcement" page. 2. Enter a past date in the "Date Posted" field. 3. Select the "Create Announcement" button. 	Insert announcement failed. The previous data cannot be selected.
TC8_4	Verify that the feature for updating announcements is available and functioning.	<ol style="list-style-type: none"> 1. Navigate to the "Update Announcement" page. 2. Update the "Description" field. 3. Select the "Update" button. 	Update announcement successful

Table 6.14 : Test Description of External Buses Registration

Test ID	T009- Add External Buses		
Testing Type	Unit testing and integration testing		
Test Strategy	White box Testing		
Pre-condition	User must login as administrator		
Test Case ID	Test Conditions	Step	Expected Output
TC9_1	Validate that the external buses update function is accessible and operational.	<ol style="list-style-type: none"> 1. Navigate to the "Update Announcement" page. 2. Update the "Capacity" field. 3. Select the "Update" button. 	Update external bus information successful

Table 6.15 : Test Description of Searching Schedule

Test ID	T010- Searching schedule		
Testing Type	Unit testing and integration testing		
Test Strategy	White box Testing		
Pre-condition	User must login as normal user		
Test Case ID	Test Conditions	Step	Expected Output
TC10_1	Verify the availability and functionality of the schedule search feature.	<ol style="list-style-type: none"> 1. Navigate to the main page for the user. 2. Enter a valid destination name. 3. Select the "Search" button. 	Show the details of the schedules you are searching for.
TC10_2	Verify that if no data is given, the search feature is inaccessible.	<ol style="list-style-type: none"> 1. Navigate to the main page for the user. 2. Select the "Search" button without entering any data. 	Searching function failed. Show the failure message "Please fill out this field" at the search bar

Table 6.16 : Test Description of Booking Seats

Test ID	T011- Bus Booking		
Testing Type	Unit testing and integration testing		
Test Strategy	White box Testing		
Pre-condition	User must login as normal user		
Test Case ID	Test Conditions	Step	Expected Output
TC11_1	Verify the availability and functionality of the booking feature.	<ol style="list-style-type: none"> 1. Navigate to the booking page. 2. Select a seat. 3. Select the "Book Now" button. 	Booking successful.
TC11_2	Validate the booking function is Not available if user book twice time	<ol style="list-style-type: none"> 1. Navigate to booking page 2. Select a seat after already booked previously 3. Select the book now button 	Booking function failed. Show the failure message "You can only book one bus."

Table 6.17 : Test Description for Driver Check-in

Test ID	T012- Driver Check-in		
Testing Type	Unit testing and integration testing		
Test Strategy	White box Testing		
Pre-condition	User must login as driver		
Test Case ID	Test Conditions	Step	Expected Output
TC12_1	Verify that the check-in feature is operational.	<ol style="list-style-type: none"> 1. Navigate to check-in page 2. Select check-in date 3. Select the OK from confirmation box. 	Check-in successful
TC12_2	Validate the check-in function is Not available if user check-in twice time	<ol style="list-style-type: none"> 1. Navigate to check-in page 2. Select check-in date 3. Select the OK from confirmation box. 	Check-in function failed. Show the failure message “An error occurred while checking in.”

Table 6.18 : Test Description of Driver Check-out

Test ID	T013- Driver Check-out		
Testing Type	Unit testing and integration testing		
Test Strategy	White box Testing		
Pre-condition	User must login as driver		
Test Case ID	Test Conditions	Step	Expected Output
TC13_1	Verify that the checkout feature is operational.	<ol style="list-style-type: none"> 1. Navigate to check-out page 2. Select check-out date 3. Select the OK from confirmation box. 	Check-out successful
TC13_2	Validate the check-in function is Not available if	<ol style="list-style-type: none"> 1. Navigate to check-out page 2. Select check-out date 	Check-in function failed. Show the failure message “No

	user check-out twice time	3. Select the OK from confirmation box.	check-in found to check out.”
--	---------------------------	---	-------------------------------

1.4.3 Test Data and Test Result

Table 6.19: Test Data of User Login

Test Data ID	Email	Password
TD1_1	yan@gmail.com	abcd123
TD1_2		
TD1_3	yan@gmail.com	1234

Table 6.20 : Test Data of Registration for Driver Account

Test Data ID	Name	License number	Phone	Email	Password
TD2_1	Ho Min Yan	NB001135	01139483321	hominyan1100@gmail.com	abcd1234
TD2_2					
TD2_3	Ho Min Yan	NB001135	01139483321	hominyan1100gmail.com	abcd1234
TD2_4		NB001231			

Table 6.21 : Test Data of Add Bus for Driver

Test Data ID	Number Plate	Capacity
TD3_1	NPA 3214	40
TD3_2		
TD3_3		44

Table 6.22 : Test Data of User Content

Test Data ID	Name	Matric Number	Age	Phone	Email	Faculty	Course	Gender
TD4_1		B0321101256						
TD4_2					lang@gmail.com			
TD4_3	Ho Jia Qing							

Table 6.23 : Test Data of Bus Route

Test Data ID	Origin	Destination	Description	Location
TD5_1	UTeM	Bukit Beruang	From UTeM back bukit beruang	“Choose at map”
TD5_2				
TD5_3	UTeM	Bukit Beruang	From UTeM back bukit beruang	

Table 6.24 : Test Data of Bus Schedule

Test Data ID	Date	Time	Bus	Semester	Route
TD6_1	2024-08-23	10:57:00	15	Week 1 to Week 7	From UTeM back bukit beruang
TD6_2					
TD6_3	2024-07-28				
TD6_4	2024-08-24	11:00:00			

Table 6.25 : Test Data of Manage Stop

Test Data ID	Stop Name	Location	Picture	Location
TD7_1	Bus Stop near ixora	Bukit Beruang	Bus_stop.jpg	“choose in map”
TD7_2				

Table 6.26 : Test Data of Announcement

Test Data ID	Title	Description	Dated Posted	Route
TD8_1	Bus Late	Bus Driver Sick	24/08/2024	From UTeM back Bukit Beruang
TD8_2				
TD8_3			22/	
TD8_4		Have traffic jam		

Table 6.27 : Test Data of External Buses Registration

Test Data ID	Number Plate	Capacity	Zone	Departure Date
TD9_1		40		

Table 6.28 : Test Data of Searching Schedule

Test Data ID	Input
TD10_1	Bukit Beruang
TD10_2	1122

Table 6.29 : Test Data of Booking Seats

Test Data ID	Select
TD11_1	<input type="button" value="1A"/> <input type="button" value="2A"/>
TD11_2	<input type="button" value="1A"/> <input type="button" value="2A"/>

Table 6.30 : Test Data of Driver Check-in

Test Data ID	Select
TD12_1	<input type="button" value="23"/> <input type="button" value="24"/>
TD12_2	<input type="button" value="23"/> <input type="button" value="24"/>

Table 6.31 : Test Data of Driver Check-out

Test Data ID	Select
TD13_1	<input type="button" value="23"/> <input type="button" value="24"/>
TD13_2	<input type="button" value="23"/> <input type="button" value="24"/>

1.5 Test Result and Analysis

Table 6.32 : Test Result of User Login

Test Case ID	Test Data ID	Expected Result	Actual Result	Pass/Fail
TC1_1	TD1_1	Authentication successful	Authentication successful	Pass
TC1_2	TD1_2	Failed to log in. Show the failure message "Please fill up this field"	Show the failure message "Please fill up this field"	Pass
TC1_3	TD1_3	Failed to log in. Show the failure message "These credentials do not match our records"	Show the failure message "These credentials do not match our records"	Pass

Table 6.33 : Test Result of Registration for Driver Account

Test Case ID	Test Data ID	Expected Result	Actual Result	Pass/Fail
TC2_1	TD2_1	Insert new driver account successful	Display success message "Driver added successfully."	Pass
TC2_2	TD2_2	Insert driver account failed. Show the failure message "Please fill out this field" at the input text field	Show the failure message "Please fill out this field" at the input text field	Pass
TC2_3	TD2_3	Insert driver account failed. Show the failure message "Please include an '@' in the email address."	Show the failure message "Please include an '@' in the email address."	Pass
TC2_4	TD2_4	Update the driver account successful	Display success message "Driver updated successfully"	Pass

Table 6.34 : Test Result of Add Bus for Driver

Test Case ID	Test Data ID	Expected Result	Actual Result	Pass/Fail
TC3_1	TD3_1	Insert new bus successful	Display success message “Bus created successfully.”	Pass
TC3_2	TD3_2	Insert buses failed. Show the failure message “Please fill out this field” at the “Capacity” input text field	Show the failure message “Please fill out this field” at the “Capacity” input text field	Pass
TC3_3	TD3_3	Update the bus successful	Display success message “Bus updated successfully.”	Pass

Table 6.35 : Test Result of User Content

Test Case ID	Test Data ID	Expected Result	Actual Result	Pass/Fail
TC4_1	TD4_1	Update user detail successful	Display success message “User updated successfully”	Pass
TC4_2	TD4_2	Update user detail failed. Show the failure message “Please include an ‘@’ in the email address.”	Show the failure message “Please include an ‘@’ in the email address.”	Pass
TC4_3	TD4_3	Update user detail failed. Show the failure message “The provided name is already in use.”	Show the failure message “The provided name is already in use.”	Pass

Table 6.36 : Test Result of Bus Route

Test Case ID	Test Data ID	Expected Result	Actual Result	Pass/Fail
TC5_1	TD5_1	Insert new route successful	Display success message “Route created successfully.”	Pass
TC5_2	TD5_2	Insert route failed. Show the failure message “Please fill out this field” at the “origin” input text field	Show the failure message “Please fill out this field” at the “origin” input text field	Pass
TC5_3	TD5_3	Insert route failed. Show the failure message “The origin latitude, origin longitude, destination latitude, destination longitude are required.”	Show the failure message “The origin latitude, origin longitude, destination latitude, destination longitude are required.”	Pass

Table 6.37 : Test Result of Bus Schedule

Test Case ID	Test Data ID	Expected Result	Actual Result	Pass/Fail
TC6_1	TD6_1	Insert new bus schedule successful	Display success message “Schedule created successfully.”	Pass
TC6_2	TD6_2	Insert schedule failed. Show the failure message “Please fill out this field” at the “date” input text field	Show the failure message “Please fill out this field” at the “date” input text field.	Pass
TC6_3	TD6_3	Insert bus schedule failed. The previous date cannot be selected.	There are not allow choose previous date	Pass

TC6_4	TD6_4	Update bus schedule successful	Display success message "Schedule updated successfully!"	Pass
-------	-------	--------------------------------	--	------

Table 6.38 : Test Result of Manage Stop

Test Case ID	Test Data ID	Expected Result	Actual Result	Pass/Fail
TC7_1	TD7_1	Insert new stop information successful	Display success message "Stop created successfully."	Pass
TC7_2	TD7_2	Insert stop failed. Show the failure message "Please fill out this field" at the "name" input text field	Show the failure message "Please fill out this field" at the "name" input text field	Pass

Table 6.39 : Test Result of Announcement

Test Case ID	Test Data ID	Expected Result	Actual Result	Pass/Fail
TC8_1	TD8_1	Insert new announcement successful	Display success message "Announcement created successfully."	Pass
TC8_2	TD8_2	Insert announcement failed. Show the failure message "Please fill out this field" at the "title" input text field	Show the failure message "Please fill out this field" at the "title" input text field.	Pass
TC8_3	TD8_3	Insert announcement failed. The previous data cannot be selected.	Only allowed choose today at the "date" input text.	Pass

TC8_4	TD8_4	Update announcement successful	Display success message "Announcement updated successfully."	Pass
-------	-------	--------------------------------	--	------

Table 6.40 : Test Result of External Buses Registration

Test Case ID	Test Data ID	Expected Result	Actual Result	Pass/Fail
TC9_1	TD9_1	Update external bus information successful	Display success message "External bus updated successfully."	Pass

Table 6.41 : Test Result of Searching Schedule

Test Case ID	Test Data ID	Expected Result	Actual Result	Pass/Fail
TC10_1	TD10_1	Show the details of the schedules you are searching for.	Show the details of the schedules you are searching for.	Pass
TC10_2	TD10_2	Searching function failed. Show the failure message "Please fill out this field" at the search bar	Show the failure message "Please fill out this field" at the search bar	Pass

Table 6.42 : Test Result of Booking Seats

Test Case ID	Test Data ID	Expected Result	Actual Result	Pass/Fail
TC11_1	TD11_1	Booking successful.	Display success message "Booking confirmed successfully."	Pass
TC11_2	TD11_2	Booking function failed. Show the failure message "You can only book one bus."	Show the failure message "You can only book one bus."	Pass

Table 6.43 : Test Result of Driver Check-in

Test Case ID	Test Data ID	Expected Result	Actual Result	Pass/Fail
TC12_1	TD12_1	Check-in successful	Display success message "Check-in successful."	Pass
TC12_2	TD12_2	Check-in function failed. Show the failure message "An error occurred while checking in."	Show the failure message "An error occurred while checking in"	Pass

Table 6.44 : Test Result of Driver Check-out

Test Case ID	Test Data ID	Expected Result	Actual Result	Pass/Fail
TC13_1	TD13_1	Check-out successful	Display success message “Checkout successful”	Pass
TC13_2	TD13_2	Check-in function failed. Show the failure message “No check-in found to check out.”	Show the failure message “No check-in found to check out.”	Pass

6.6 Conclusion

In summary, a crucial stage of the Database Life Cycle (DBLC) is system testing. Errors and faults must be found early on and addressed by the DBLC testing phase. The testing approaches—white-box—have been used in the UTeM bus schedule management system. The test schedule, test descriptions, and test results are all included in this chapter.

CHAPTER VII



7.1 Introduction

The general conclusion for the UTeM Bus Scheduling Management System will be covered in this chapter, along with a critique of its advantages and disadvantages. This chapter will also describe and expound on recommendations for enhancements that stem from the examination of the project's strengths and weaknesses and contributions.

7.2 Observation Weakness and Strengths

Every method has its benefits and drawbacks. The UTeM Bus Scheduling Management System's qualify are described in section 7.2.1, while its shortcomings were discussed in part 7.2.2.

7.2.1 Strengths

The strengths of the system are:

i. Convenience

The web-based system offers significant benefits to all three types of users. For example, it provides convenience by allowing users to access the system from any location and at any time. UTeM Bus Scheduling Management allow administrator to manage the schedule information and view the buses information. The system normal user can search the schedules information and make booking for seats when certain date. When there are any notifications, the user can check from the system. For driver, they allowed to check-in and check-out to record their attendance, When drivers had done check-in , normal user able to check their location at map .

ii. Human Error Reduction

UTeM Bus Scheduling Management is using trigger and stored procedure to store schedule data into system. The trigger makes sure the number plate of bus cannot insert repeated, make it unique. Trigger also used at check-in and check-out functions to prevent driver can check-in twice times. Besides that , administrator can view the graph at the main page of the admin system.

7.2.2 Weakness

i. Recovery and backup features

The system lacks automatic recovery and backup features. If an administrator accidentally deletes data, it may not be recoverable.

ii. Training Requirement for Administrator

Administrator may need training to effectively manage the system, which could require additional time and resources. That is because the admin side system had a lot of function like insert the bus information and assign the schedules. The administrator can use the system effectively after he was trained.

iii. **Dependence on Internet Connectivity**

The system requires an internet connection to function, which can be a limitation for users in areas with poor or no internet access. The system requires a strong internet connection to check the driver location at the map

7.3 Proposition of Improvement

The following are proposed actions for enhancing the UTeM Bus Scheduling Management System, based on the identified strengths and weaknesses:

i. **Implement Automatic Recovery and Backup Features**

To address the lack of recovery and backup features, it is recommended to implement automatic backup mechanisms. This will ensure that all data is regularly backed up and can be easily restored in case of accidental deletion or system failure. Cloud-based backup solutions could be explored for this purpose, providing a secure and reliable method for data recovery

ii. **Enhanced Training Programs for Administrators**

To ensure that administrators can effectively manage the system, it is essential to develop comprehensive training programs. These programs should cover all aspects of the system, including bus information management, schedule assignments, and troubleshooting common issues. Additionally, providing administrators with user-friendly manuals and ongoing support can help them adapt quickly to the system.

iii. **Optimize Internet Connectivity Requirements**

Considering the system's dependence on a stable internet connection, efforts should be made to optimize the system to function efficiently even in low-bandwidth scenarios. Introducing offline functionality for certain features, such as schedule viewing and ticket booking, could reduce the system's reliance on continuous internet connectivity. Additionally, implementing data compression techniques and optimizing the map tracking feature for lower bandwidth usage could improve the system's performance in areas with poor connectivity

7.4 Contribution

The UTeM Bus Scheduling Management System significantly enhances transportation management at the university by streamlining bus scheduling and resource allocation, improving the user experience for students and staff through real-time tracking and online booking, and supporting data-driven decision-making. It facilitates better communication with integrated announcements and notifications, ensuring users are always informed about updates. Additionally, the system contributes to UTeM's sustainability goals by optimizing bus usage, reducing fuel consumption, and lowering carbon emissions, all of which contribute to a more efficient and environmentally friendly transportation system.

7.5 Conclusion

In conclusion, the objectives and scope outlined in Chapter 1 of the project have been successfully achieved. The UTeM Bus Scheduling Management System aims to replace the paper-based system with a web-based platform, minimize human errors in updating bus schedules, provide easy access to bus schedule information, and generate various types of graphs. The Database Life Cycle (DBLC) methodology was employed, covering phases such as database initial research, design, implementation and loading, training and evaluation, operations, and maintenance. Throughout the development process, testing was conducted to identify and resolve issues within the system. Ultimately, the system has met both the functional and non-functional requirements. However, some areas still require improvement for future use. The system fulfils the requirements for a Bachelor of Computer Science (Database Management) degree and has been successfully completed. Additionally, the system's contributions may be integrated into the university's future scheduling processes.

References

- Altexsoft. (2022, July 26). *Non-functional Requirements: Examples, Types, How to Approach*. AltexSoft. <https://www.altexsoft.com/blog/non-functional-requirements/>
- Black Box Vs White Box Testing*. (n.d.). Wwww.practitest.com. <https://www.practitest.com/resource-center/article/black-box-vs-white-box-testing/>
- Weerasuriya, A. (2023, February 9). The Database System Development Life Cycle (DBLC) - Ayeshan weerasuriya - Medium. *Medium*. <https://medium.com/@ayeshanweerasuriya/the-database-system-development-life-cycle-dblc-d4c6f3518195>
- What is a Data Flow Diagram*. (n.d.-b). Lucidchart. <https://www.lucidchart.com/pages/data-flow-diagram>
- Laravel - Installation*. (n.d.). https://www.tutorialspoint.com/laravel/laravel_installation.htm
- Bus Management System Project Idea For Final Year*. (2023, February 18). LovelyCoding.org. https://www.lovelycoding.org/bus-management-system/?srsltid=AfmBOoqYyKn_QnDpvOyvoMSg9YMrnQqoT6x_gDPIKzPBYMTPsHyP-m-a
- Vpadmin. (2023, October 10). *Navigating System Complexity: A Comprehensive Guide to Data Flow Diagram Levels - Visual Paradigm Guides*. Visual Paradigm Guides. <https://guides.visual-paradigm.com/navigating-system-complexity-a-comprehensive-guide-to-data-flow-diagram-levels/>

Appendix

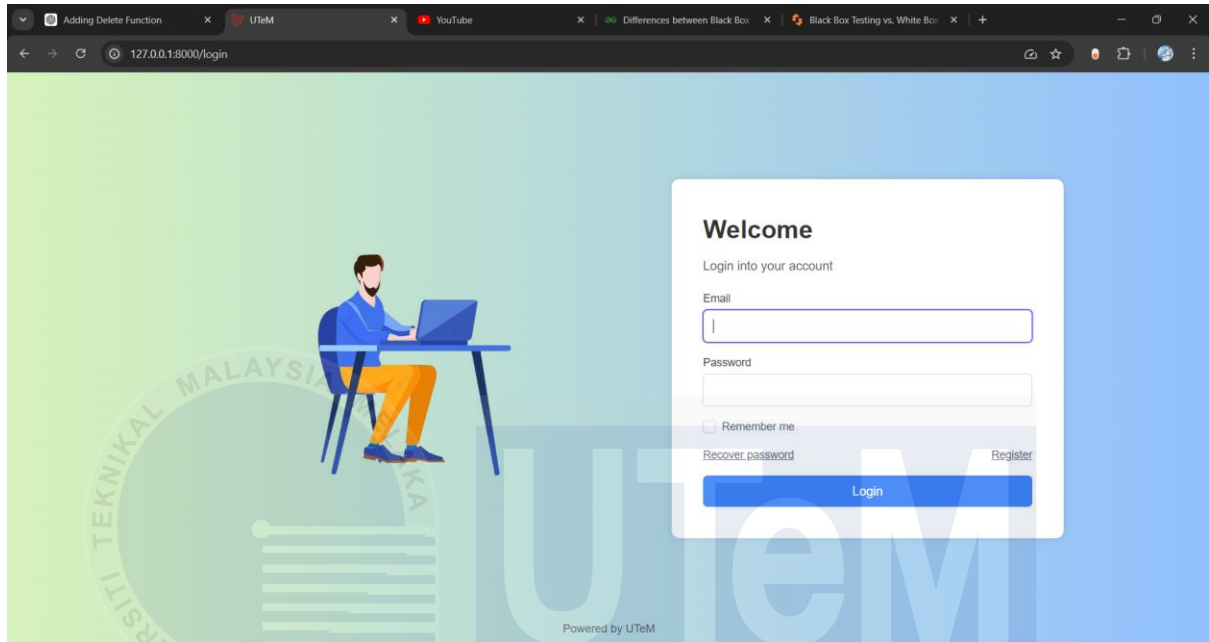


Figure 8.1 : Login Page

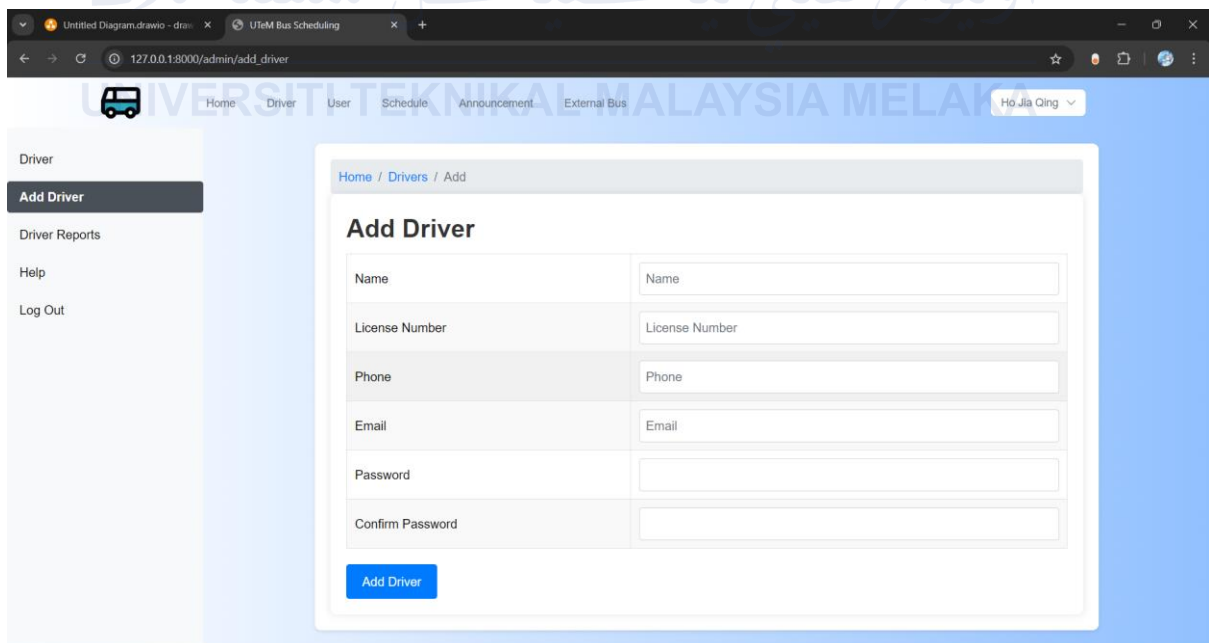


Figure 8.2 : Add Driver

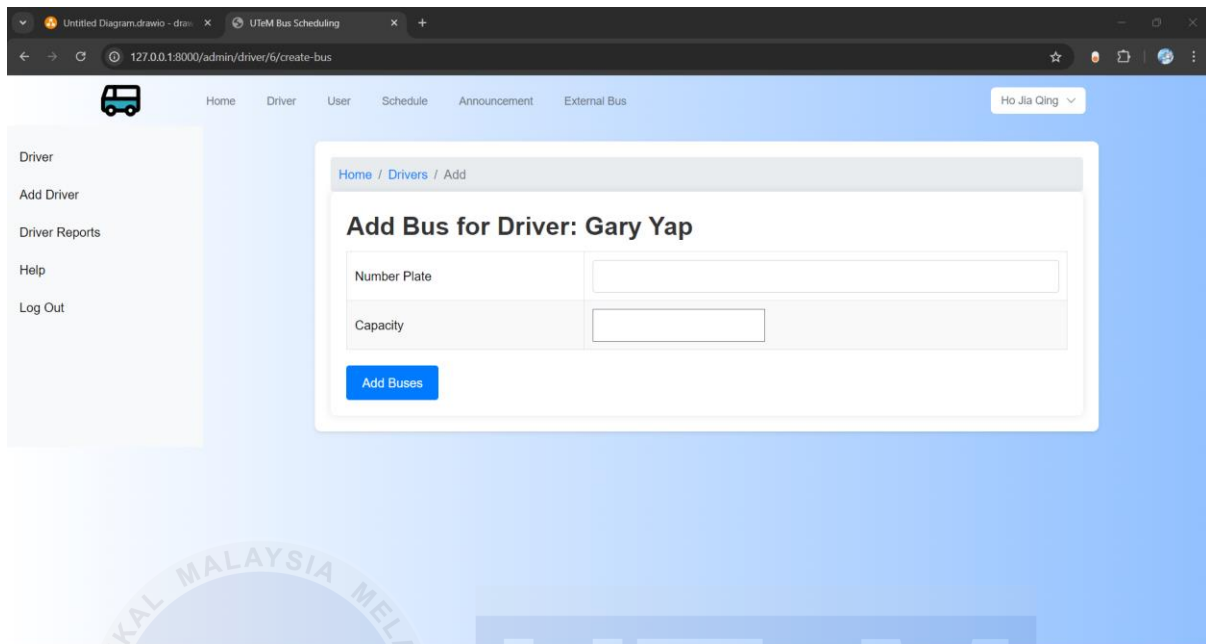


Figure 8.3 : Add Bus

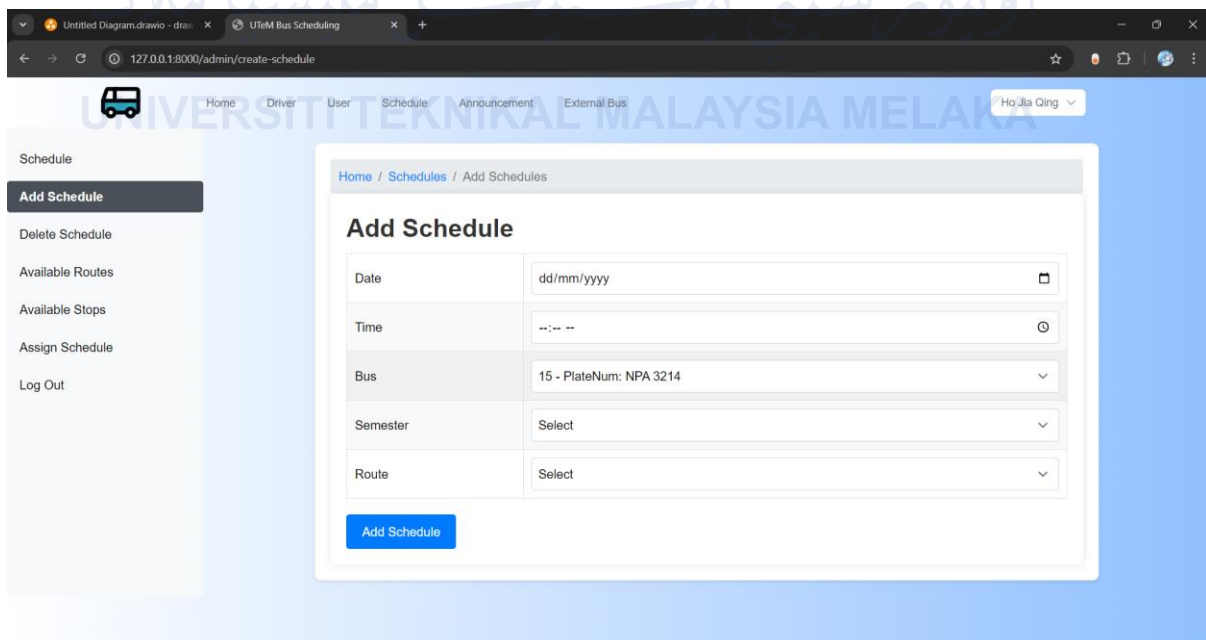


Figure 8.4 : Add Schedule

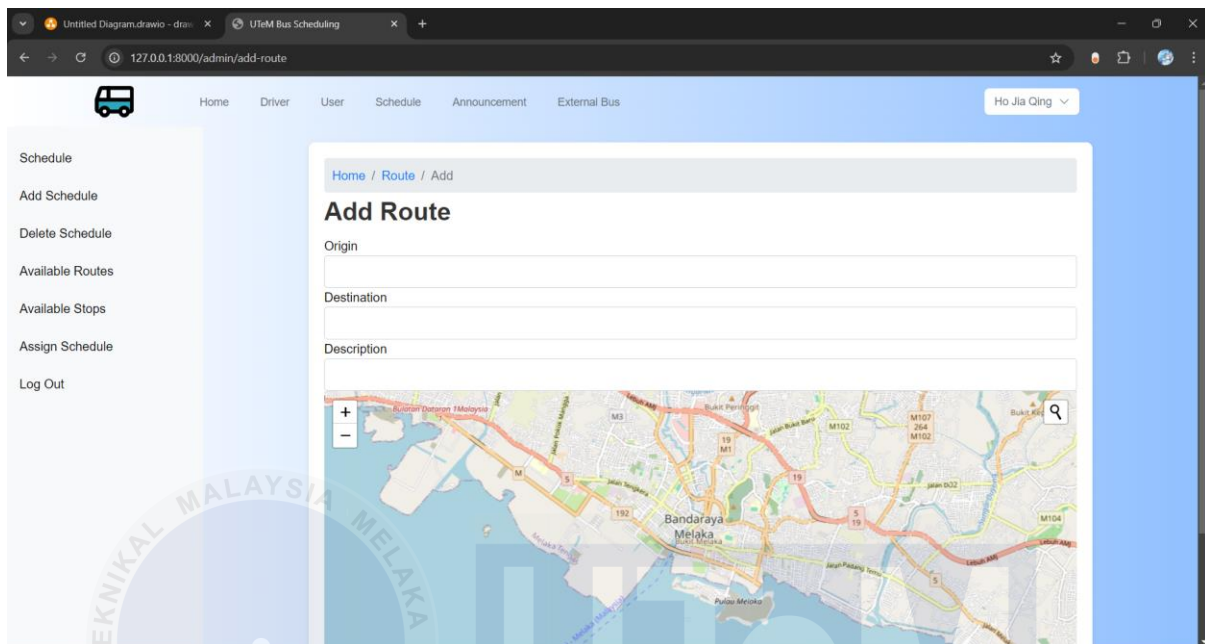


Figure 8.5 : Add Route

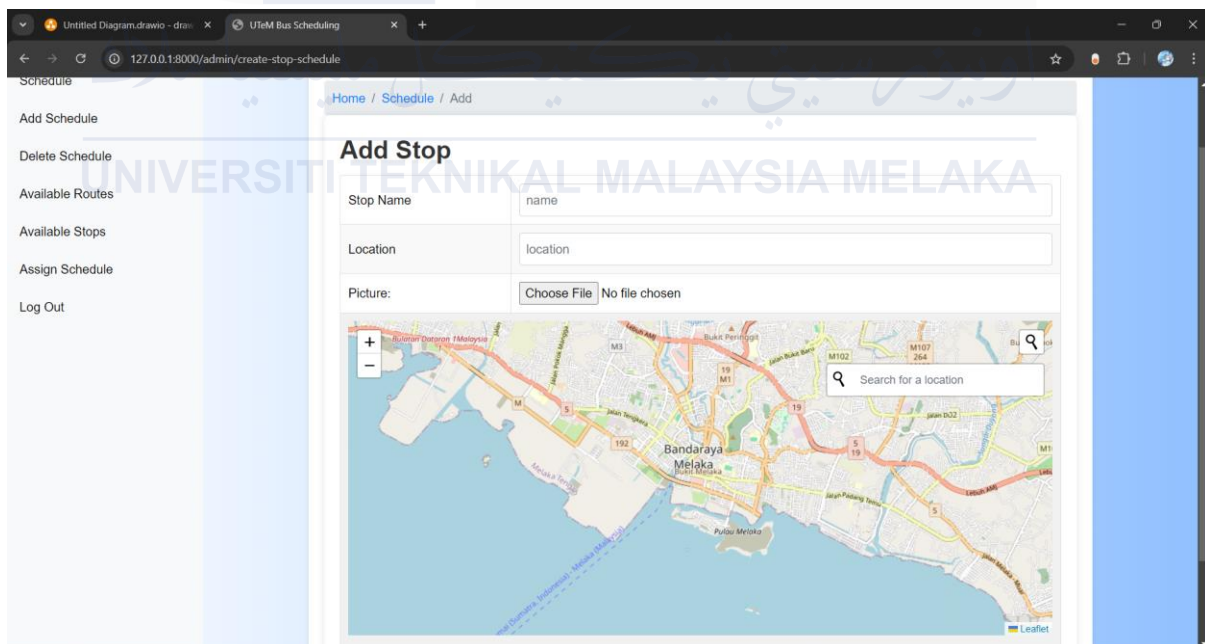
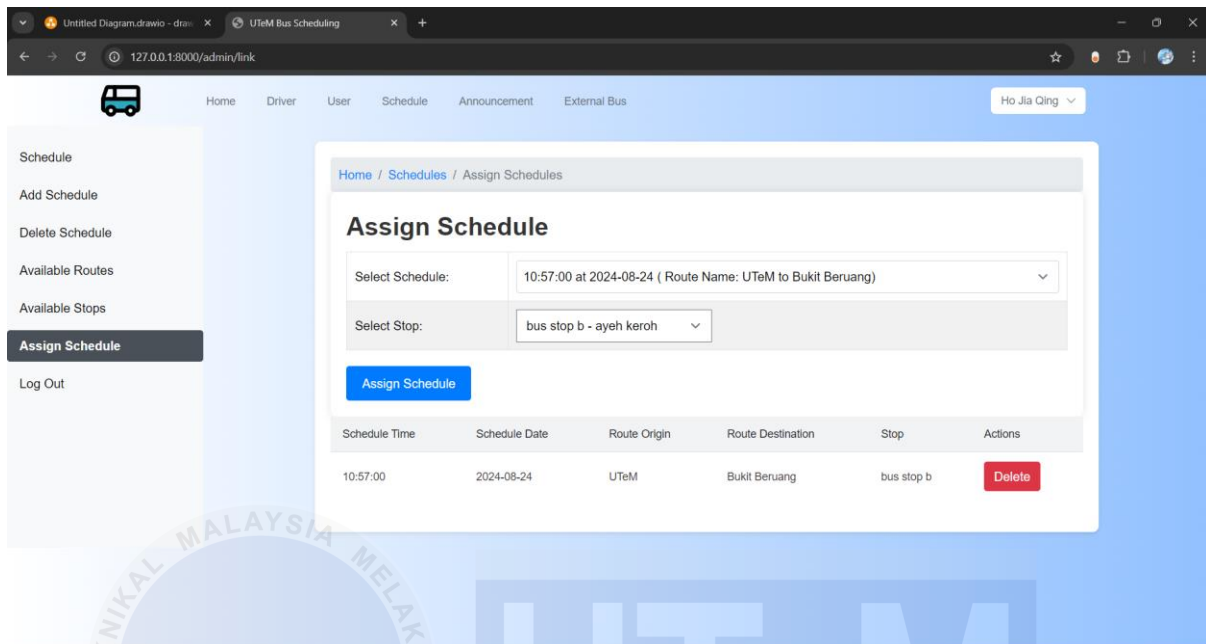


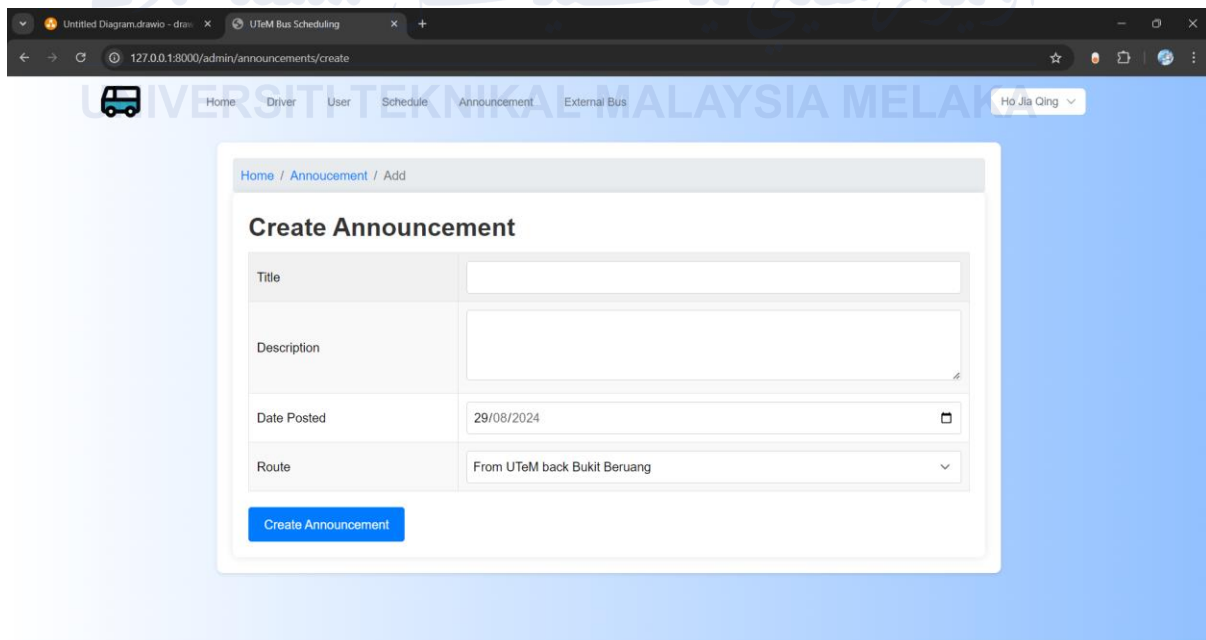
Figure 8.6 : Add Stop



The screenshot shows the 'Assign Schedule' page in the UTeM Bus Scheduling system. The page has a sidebar on the left with the following menu items: Schedule, Add Schedule, Delete Schedule, Available Routes, Available Stops, Assign Schedule (highlighted), and Log Out. The main content area is titled 'Assign Schedule' and contains a form with two dropdown menus: 'Select Schedule:' (showing '10:57:00 at 2024-08-24 (Route Name: UTeM to Bukit Beruang)') and 'Select Stop:' (showing 'bus stop b - ayeh keroh'). Below the form is a blue 'Assign Schedule' button. Underneath the form is a table with the following data:

Schedule Time	Schedule Date	Route Origin	Route Destination	Stop	Actions
10:57:00	2024-08-24	UTeM	Bukit Beruang	bus stop b	Delete

Figure 8.7 : Assign Schedule



The screenshot shows the 'Create Announcement' page in the UTeM Bus Scheduling system. The page has a sidebar on the left with the following menu items: Schedule, Add Announcement, Delete Announcement, Available Routes, Available Stops, Create Announcement (highlighted), and Log Out. The main content area is titled 'Create Announcement' and contains a form with the following fields:

- Title:
- Description:
- Date Posted: 29/08/2024
- Route: From UTeM back Bukit Beruang

Below the form is a blue 'Create Announcement' button.

Figure 8.8 : Add Announcement

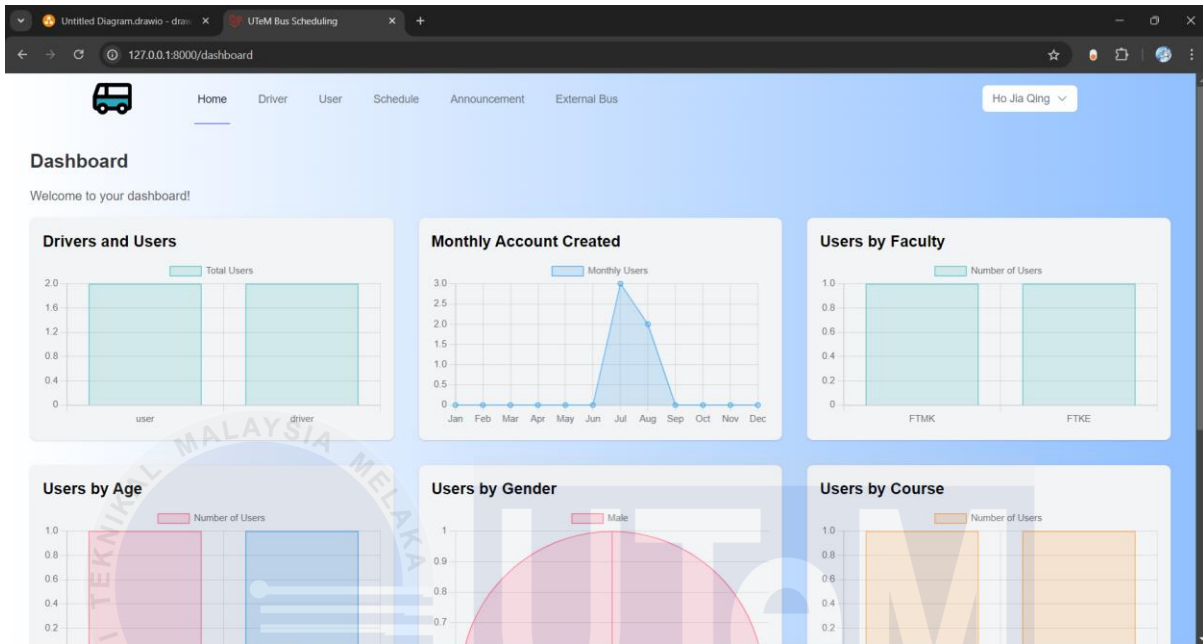


Figure 8.9 : Admin Dashboard

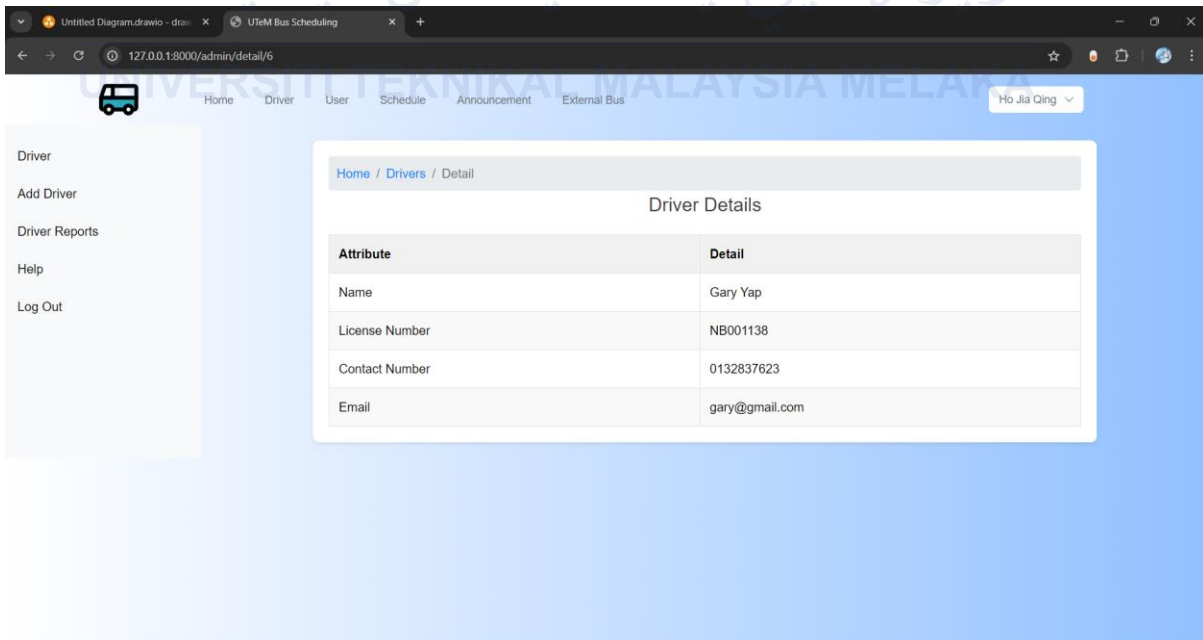


Figure 8.10 : Driver Detail

The screenshot shows the 'Driver Report' page in the UTeM Bus Scheduling system. The breadcrumb trail is 'Home / Drivers / Report'. The page title is 'Driver Report'. A table lists the following data:

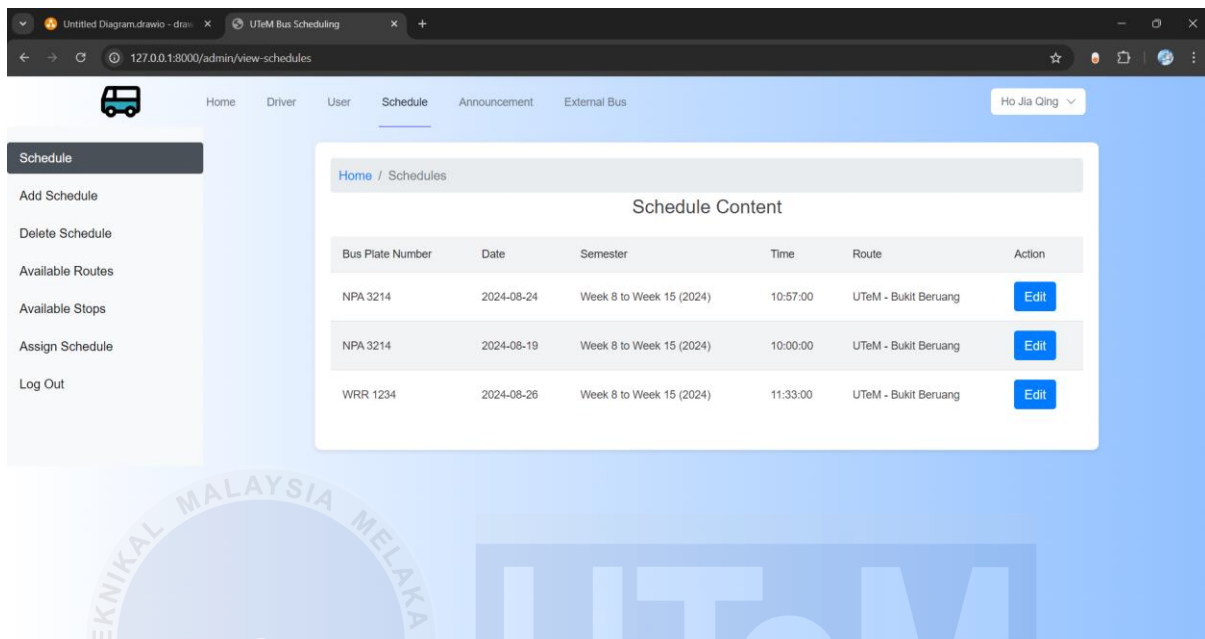
Driver Name	Driver Email	Driver Phone	Driver Check-In	Driver Check-Out	Duration
Gary Yap	gary@gmail.com	0132837623	2024-08-23 08:35:08	2024-08-23 18:37:49	10 hours 2 minutes

Figure 8.11 : Driver Report

The screenshot shows the 'View User' page in the UTeM Bus Scheduling system. The breadcrumb trail is 'Home / Users'. The page title is 'User Content'. A table lists the following data:

Name	Email	Matric Number	Age	Phone	Faculty	Gender	Course	User Type	Action
Ho Jia Qing	hojiaqing0569@gmail.com			01135403313				admin	Edit
Lee Min Yan	min@gmail.com	B032110256	24	0183549590	FTMK	Male	Engineering	user	Edit
Lim Yan	yan@gmail.com	B032110124	23	01135407953	FTKE	Male	Computer Science	user	Edit
Gary Yap	gary@gmail.com			0132837623				driver	Edit
Ho Min Yan	hominyan1100@gmail.com			0183549591				driver	Edit

Figure 8.12 : View User

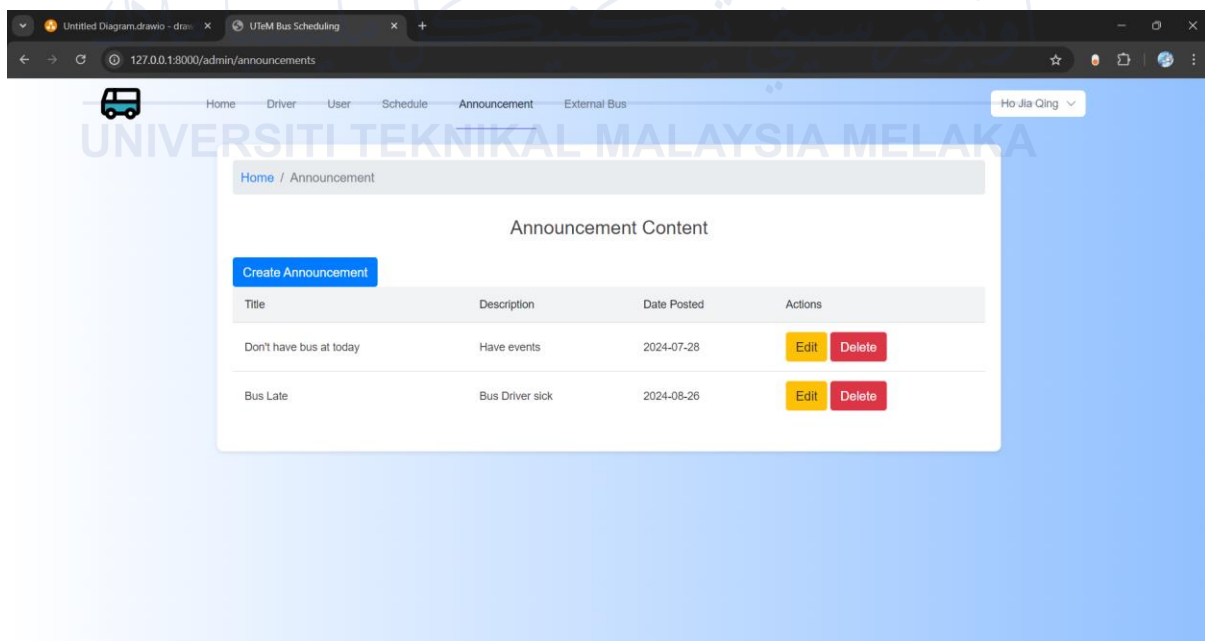


Home / Schedules

Schedule Content

Bus Plate Number	Date	Semester	Time	Route	Action
NPA 3214	2024-08-24	Week 8 to Week 15 (2024)	10:57:00	UTeM - Bukit Beruang	Edit
NPA 3214	2024-08-19	Week 8 to Week 15 (2024)	10:00:00	UTeM - Bukit Beruang	Edit
WRR 1234	2024-08-26	Week 8 to Week 15 (2024)	11:33:00	UTeM - Bukit Beruang	Edit

Figure 8.13 : View Schedule



Home / Announcement

Announcement Content

[Create Announcement](#)

Title	Description	Date Posted	Actions
Don't have bus at today	Have events	2024-07-28	Edit Delete
Bus Late	Bus Driver sick	2024-08-26	Edit Delete

Figure 8.14 : View Announcement

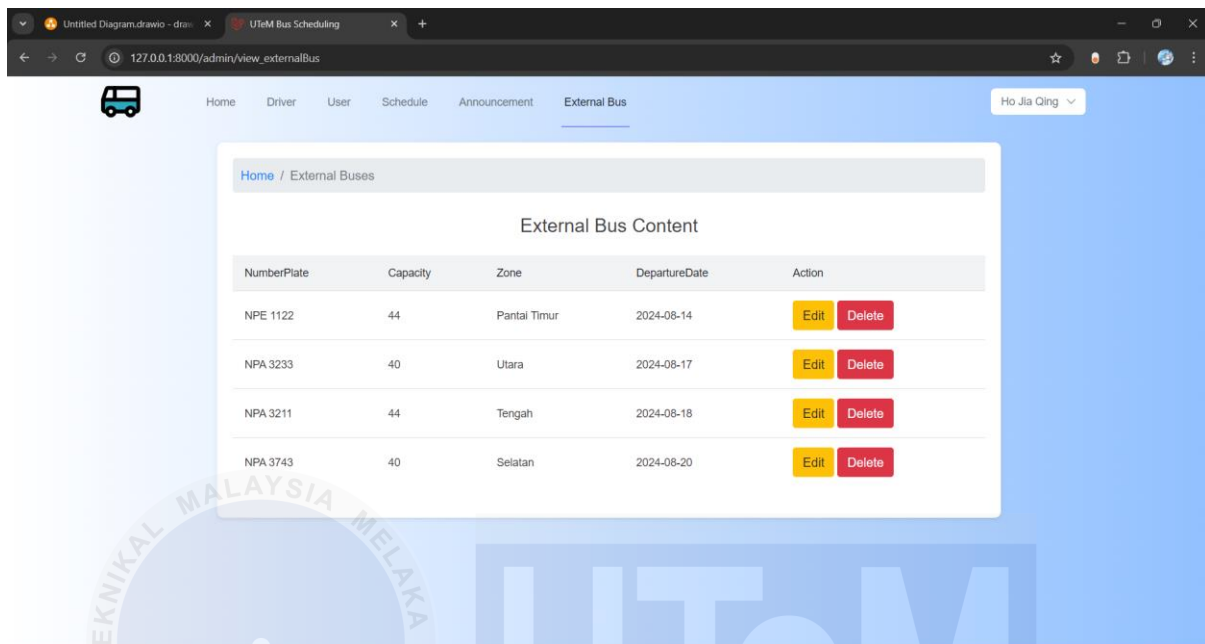


Figure 8.15 : View External Bus

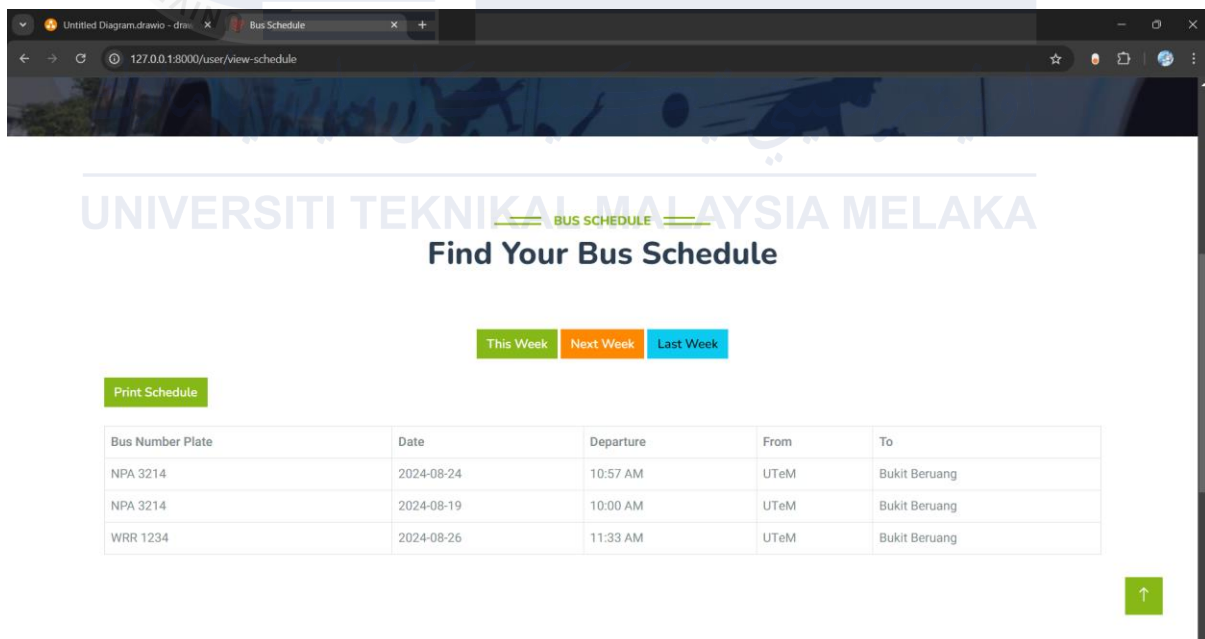


Figure 8.16 : User View Schedule

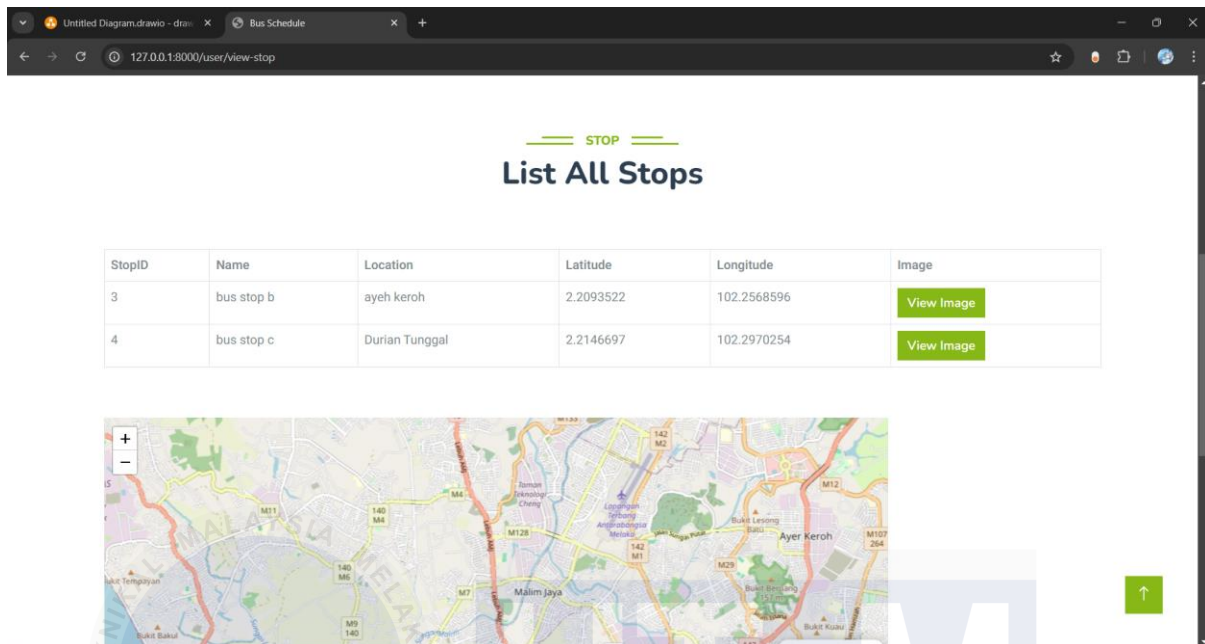


Figure 8.17 : List All Stops

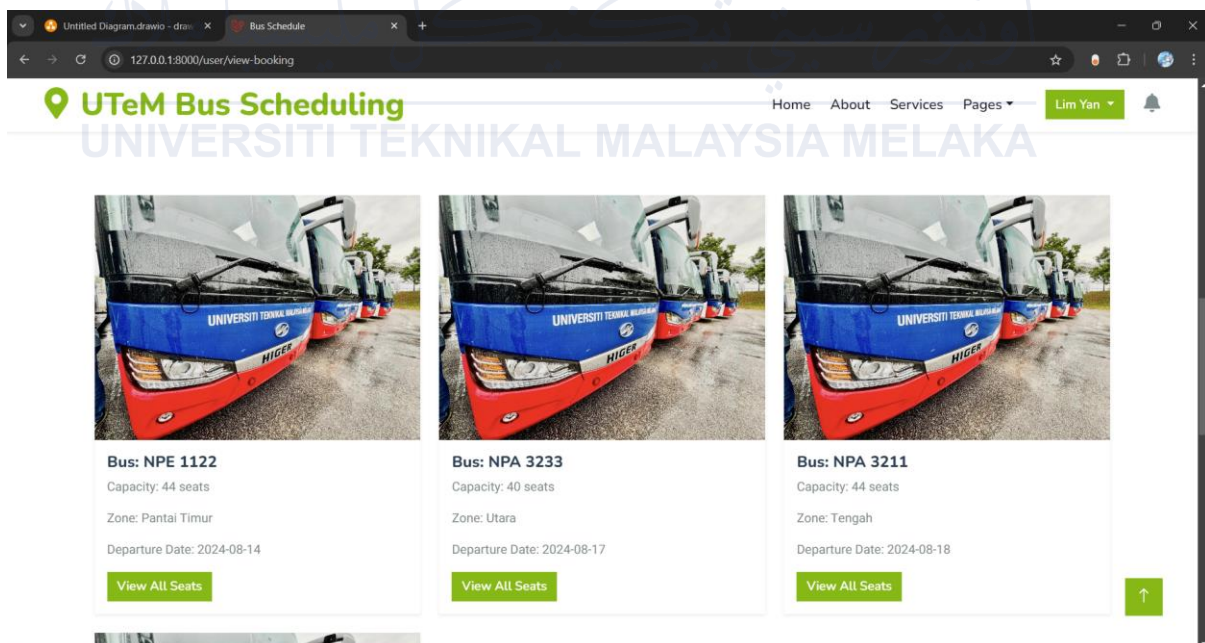


Figure 8.18 : Bus Seat Booking Page

Bus Ticket
Home / Pages / Ticket

UTeM Bus Ticket
Bus Scheduling Management System

Passenger Name: Lim Yan
Bus Number: NPA 3211
Seat Number: 1A
Departure Date: 2024-08-18
Booking Date: 2024-08-17 21:29:21

[Print Ticket](#)

Please bring this ticket with you on the day of departure.

Figure 8.19 : User View Ticket

DRIVER CHECKIN

Calendar

August 2024 today < >

Sun	Mon	Tue	Wed	Thu	Fri	Sat
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17

Figure 8.20 : Driver Check-in / Check-out Page

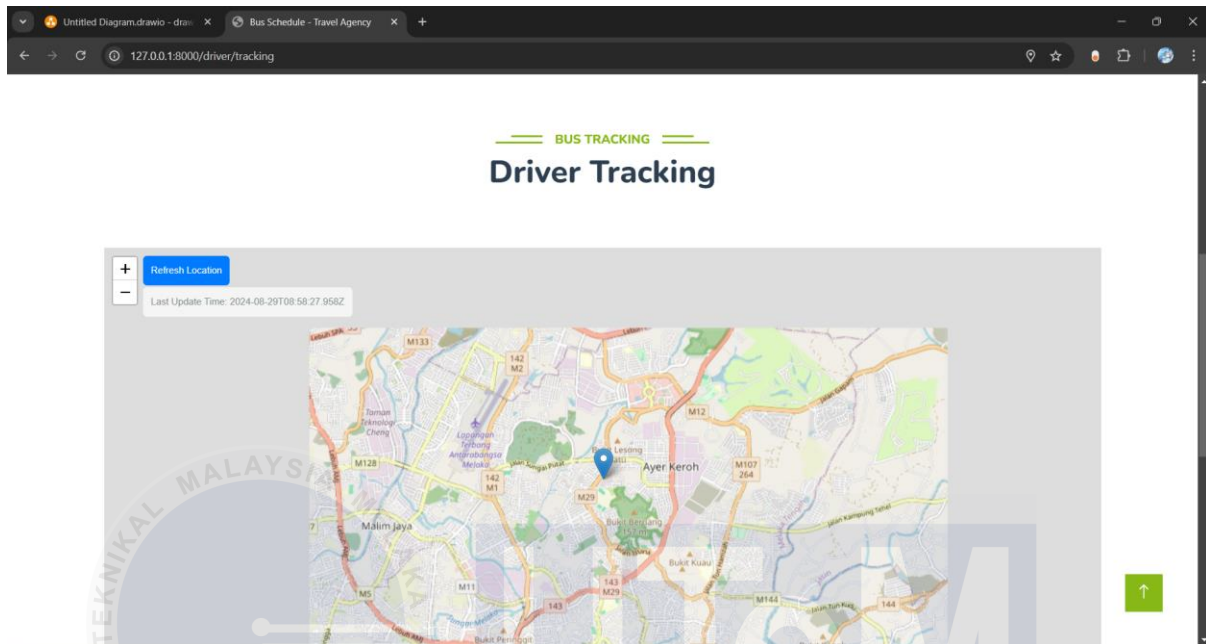


Figure 8.21 : Driver Tracking

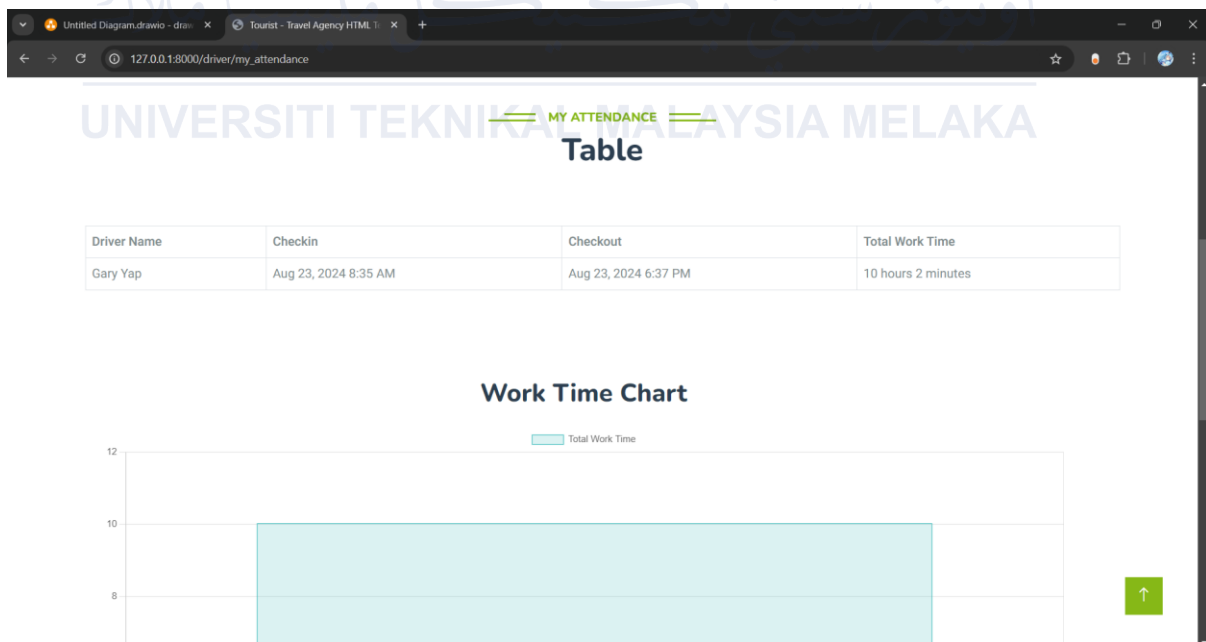


Figure 8.22 : Driver Attendance