CAR SERVICE SYSTEM

TIANG KING JECK

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

CAR SERVICE SYSTEM

TIANG KING JECK

This report is submitted in partial fulfillment of the requirements for the Bachelor of Computer Science (Software Development) with Honours.

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2021

DECLARATION

I hereby declare that this project report entitled

[CAR SERVICE SYSTEM]

is written by me and is my own effort and that no part has been plagiarized

without citations.

STUDENT : [illegible] Date : 08/09/2021

(TIANG KING JECK)

I hereby declare that I have read this project report and found

this project report is sufficient in term of the scope and quality for the award of

Bachelor of Computer Science (Software Development) with Honours.

SUPERVISOR : [illegible] Date : 10 September 2021

([PROFESOR MADYA TS. DR. MOHD SANUSI BIN AZMI])

DECLARATION

DEDICATION

I dedicated this project work to my beloved parents. A special feeling of gratitude to my loving parents whose words of encouragement and push for tenacity ring in my ears. I also dedicated the project work to my supervisor, Dr. Mohd Sanusi bin Azmi who had supported me throughout the process. I will always appreciate all he done for helping me develop my technology skills. Last but not least, I dedicated this work and give special thanks to my friends whose had gave me many useful and helpful advises when I get into a bottleneck.

ACKNOWLEDGEMENTS

In this project, without the aid and guidance from many people and my supervisor, Dr. Mohd Sanusi bin Azmi, this would not have been possible. I would like to express my heartfelt appreciation to all these people.

I would like to thank Dr. Mohd Sanusi bin Azmi for giving assistant to complete this project successfully. I am very obliged to Dr. Mohd Sanusi bin Azmi for his encouragement, continuous supervision and knowledge about the project and support for it in the completion of the project. His advice is very beneficial to the project completion. I would also like to thank my beloved parents who have been giving me support and motivation throughout my project.

I want to express my particular gratitude as well as thanks to Universiti Teknikal Malaysia Melaka (UTeM), the Faculty of Information and Communications technology (FTMK), for providing me with the opportunity to learn how to cooperate and incorporate our expertise in the project.

I also thank our colleagues for their contributions and the individuals who have helped me voluntarily with their expertise.

ABSTRACT

This project is aimed to developing a real-time car servicing system for the car owner and the car service industry. The Car Service System (CaSS) consist of two applications which are the desktop application and the mobile application. The system had the own web service as the back-end to handle the data processing between the front-end and database. This system is able to tracking the whole process of the car servicing in real-time in order to help the staffs or technicians to control the servicing process with their customers. In conclusion, CaSS is able to replace the current system with more features and solved the problem of the traditional servicing shop.

ABSTRAK

Projek ini bertujuan untuk membangunkan sistem servis kereta masa nyata untuk pemilik kereta dan industri perkhidmatan kereta. Sistem Servis Kereta (CaSS) terdiri daripada dua aplikasi iaitu aplikasi desktop dan aplikasi mudah alih. Sistem ini mempunyai perkhidmatan web sendiri sebagai back-end untuk menangani pemprosesan data antara front-end dan pangkalan data. Sistem ini dapat mengesan keseluruhan proses servis kereta dalam masa nyata untuk membantu kakitangan atau juruteknik mengawal proses servis dengan pelanggan mereka. Kesimpulannya, CaSS dapat menggantikan sistem semasa dengan lebih banyak ciri dan menyelesaikan masalah kedai servis tradisional.

TABLE OF CONTENTS

LSIT OF TABLES

PAGE

LIST OF FIGURES

PAGE

LIST OF ABBREVIATIONS

CaSS	**-**	**Car Service System**
OOAD	**-**	**Object-Oriented Analysis Design**
SDLC	**-**	**Software Development Life Cycle**
REST	**-**	**Representational State Transfer**
JSON	**-**	**JavaScript Object Notation**
ORM	**-**	**Object–Relational Mapping**
npm	**-**	**Node Package Manager**
SQA	**-**	**Software Quality Assurance**

LIST OF ATTACHMENTS

CHAPTER 1: INTRODUCTION

1.1 Introduction

In Malaysia, whether students or office workers, most people have a car to travel. This also means that people often need to go to their respective car service centers to regularly service their cars, especially for the newly bought cars. The process of servicing a car is not only time-consuming and complicated, so it requires frequent communication between the customer and the repairman, and sometimes even the counter personnel need to communicate the opinions of both of them through the phone call.

Not only that, booking in advance for car services is also a very troublesome process. Car owners need to call the service center corresponding to their car to make a repair appointment and since the car owner does not know which time has not been reserved, the owner can only waste time inquiring one by one in order to find a time when both of them are available to service the car.

Car owner needs to service the car when the car has travelled a certain distance or time so the car owner needs to check the distance recorder in the car and also the time past from the last service made. Besides, because most of the car owners are not familiar with car-related knowledge, and if they are not lucky enough to encounter more greedy maintenance or service personnel, the car owners will face the situation of extremely high fees from the service center.

1.2 Problem Statement

- Lack of platform for processing the service of car that led to time-consuming and complicated.
- Lack of visualization of reservation for servicing car that would cause disputes due to duplication of reservation, conflict in scheduling and so on.
- Unable to track the service process and the result of service.

1.3 Objective

- To develop a mobile application which provide the automation features for servicing the car such as the record of the service logbook, the confirmation of service parts of the car after checking and so on.
- To provide a visualized schedule in the calendar for each of the car service center that based on the location and distance of the car owner.
- To track all the process of the servicing and update to the developed platform from time to time. The serviced parts and results would be generated in form of e-receipt with the description and purpose.

1.4 Scope

The scope of the project contains the target users and the modules that will be develop in the system. There are three target users and several modules which will be developed in three different platforms.

1.4.1 Target Users

- Car owner: a person who own at least a car and act as customer to service the car in the car service center by using the customer mobile application.
- Car servicer: a staff who service the car of the customer in the car service center with the assistance of branch desktop application.

1.4.2 Modules

The following modules will be developed into three different platforms which will be used by corresponding target users. The customer mobile application is used by car owner and branch desktop application is used by car servicer.

1.4.2.1 Customer Mobile Application

- Authentication module: a module which handles the authentication of the customer.
- Car Module: a module which displays and manages the cars of the customer added.
- Reservation Module: a module which enables the customer to view and find the car service centers and handles the reservation of car service such as make, modify, and cancel the reservation.
- Service Module: a module which enables the customer to track and update the current servicing status.
- Profile Module: a module which enables the customer to manage the profile of the customer.

1.4.2.2 Branch Desktop Application

- Authentication module: a module which handles the authentication of the staff of the branch.
- Dashboard Module: a module which displays the dashboard with the summary and visualized information of the reservation.
- Customer Module: a module which manages the customer by adding, modifying, and disabling customer.
- Reservation Module: a module which enables the staff handles the reservation of car service such as make, modify, and cancel the reservation.
- Service Module: a module which enables the staff to start the servicing and update the current servicing status.

1.5 Project Significance

Car Service System is a project which developed for automobile industry and focus only for car. The proposed system is digitalized from the current system which the proposed system will provides the platforms for each of the users to improve the efficiency of the whole business process and also avoid the time consuming. The reservation feature will be improved and become independent from both car owner and car servicer. This offers the ultimate freedom to the car owner to make, modify and cancel the reservation of the service and the car servicer also does not require to confirm these operations one by one at the same time. The digitalized system also achieved the data centralization which enable manager of the system can collect, process, and store the data information effectively without data redundancy. In general, this project benefits all of the current users in the current system.

1.6 Expected Output

The project is expected to develop a system that form by two subsystems which run in different platforms and use by customer and branch. The subsystem of the customer side will be developed in mobile application (only for Android) as the front-end by using Dart programming language with Flutter framework. The subsystem of the branch side will be developed in desktop application (only for Windows) as the front-end by using Dart programming language with Flutter framework. Besides, MySQL is chosen as the database of the system which host in the localhost with the help of XAMPP. Therefore, a web service that also host in the localhost will be developed as the back-end for communicating these subsystems with the database by using JavaScript with Node.js.

1.7 Conclusion

In Malaysia, car service is a potential trade that can be improved and digitalized. Most of the current car service systems are still using pure human resource to handle all of the process which led to time-consuming, data redundancy, and unable to keep records for a long time. Therefore, this project aims to develop the automation features of car servicing, provide clear and visualized schedule to enhance the service reservation, and also digitalize the data to enable the record to be stored and tracked. This system is developed for car owner, car servicer, and manager who will use the customer mobile app, branch desktop app, and manager web app respectively. The project is expected to be developed a mobile app, a desktop app, a web app, and a web service that will communicate these apps with the MySQL database in localhost.

CHAPTER 2: LITERATURA REVIEW AND PROJECT METHODOLOGY

2.1 Introduction

Planning is an important aspect of a software development cycle, and without proper planning, problems may arise later in development. This chapter will describe the methodology that was used to plan and develop the system, namely, the Object-Oriented Analysis Design, and also the requirements, the relevant researches, schedules, development technique, and previous works.

2.2 Facts and Findings

This part will list out the facts and findings related to the project that was done before development was commenced.

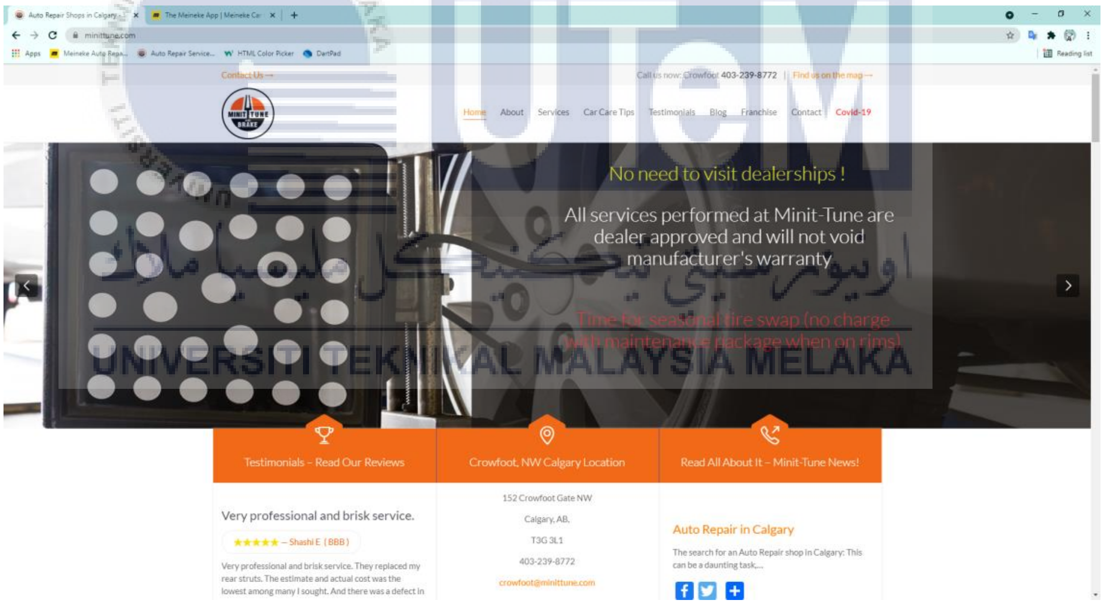
2.2.1 Domain

The Car Service System is about the reservation of car service and the tracking of the servicing process. The system provides a friendly and visualize reservation interface to ease the car owner from scheduling a reservation of service. The system will also enable the car owner to tracking the servicing process after the staff start the service. The tracking process will be updated in real-time through the web service from the database whenever there is a state change from the staff who servicing the car. The car owner also will be able to interact with the servicing process which can selected the actions that need to apply to the car such as replace specific parts. In general, car owner can communicate with staff more efficiently through the system to save the resources and avoid unnecessary misunderstandings.

2.2.2 Existing System

This section describes and states about the approach and related or past research, references, case study and other finding that relate to Car Service System. In Malaysia, the systems about the car servicing are mostly informational only which only provide the information about the service provided. There are several existing systems that can be the approach for this project are the web application from foreign country. These systems are related about the online reservation for the car servicing.

Minit-Tune is one of the existing systems for this project. This system is a web application which only can be access though the website. This website provided many of services with the full descriptive tips about the car caring. This website enables car owner to book the reservation after login to the system. Figure 2.1 shows the home page of the website.



minittune.com
Contact Us →
Call us now: Crowfoot 403-239-8772 | Find us on the map →
Home About Services Car Care Tips Testimonials Blog Franchise Contact Covid-19
No need to visit dealerships !
All services performed at Minit-Tune are dealer approved and will not void manufacturer's warranty
Time for seasonal tire swap (no charge with maintenance package when on rims)
Testimonials – Read Our Reviews
Crowfoot, NW Calgary Location
Read All About It – Minit-Tune News!
Very professional and brisk service.
★★★★★ – Shashi E (BBB)
152 Crowfoot Gate NW
Calgary, AB,
T3G 3L1
403-239-8772
crowfoot@minittune.com
Auto Repair in Calgary
The search for an Auto Repair shop in Calgary: This can be a daunting task,...

Figure 2.1: Home Page of Minit-Tune Website

Meineke is another existing system for this project. This system not only provide the web application and also integrated into a mobile application. This system also enables the car owner to book the reservation. The mobile application of this system provides the bonus and reward to the user. Figure 2.2 shows the home page of the Meineke website.

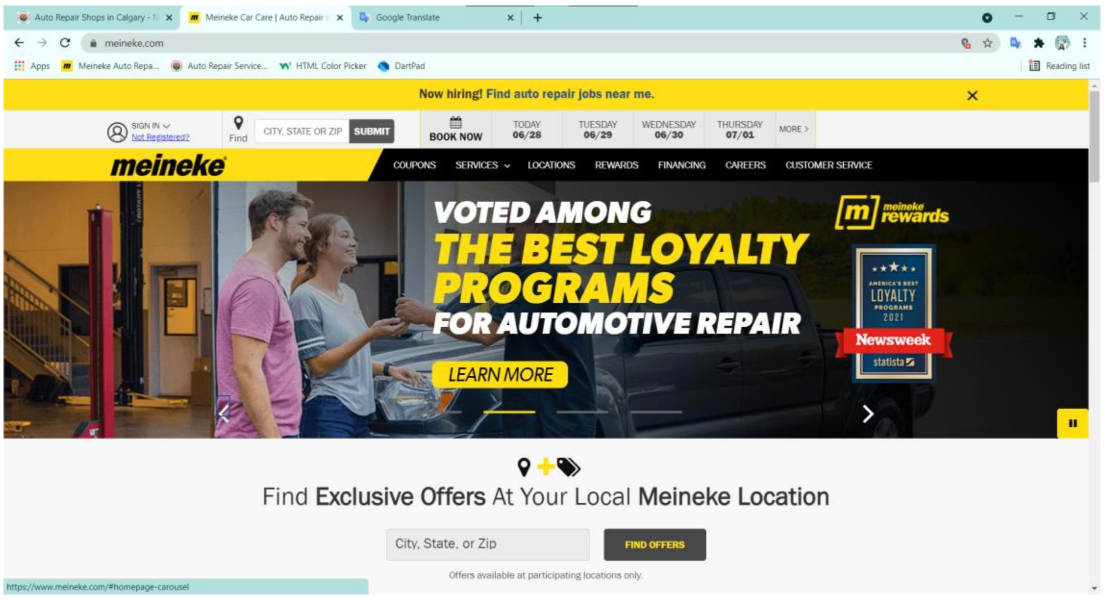


Figure 2.2: Home Page of Meineke Website

Both of these existing systems provided the basic features for booking the reservation only. The proposed system is aimed to develop a platform based on these systems with additional features. Table 2.1 shows the comparison between the existing systems and the proposed system

Table 2.1: Comparison Between Existing System and Proposed System

Feature	Existing System	Proposed System
Authenticate user	Yes	Yes
Book reservation	Yes	Yes
Review reservation history	Yes	Yes
Manage car information	No	Yes
Tracking service process	No	Yes
Customize repairing options	No	Yes

2.2.3 Technique

This section discusses about the other techniques or approaches which are also applicable and related. This system is able to apply the blockchain technology to handle the tracking process of the car servicing. This is because blockchain is a specific type of database which differs from the traditional database in the way to store the data. Blockchain stores data in blocks that are then chained together in chronological order with decentralized structured. Due to these characteristics, blockchains are immutable which mean the car servicing process can be guarantee irreversible. In other words, the authenticity of the servicing process will be ensured. Unfortunately, because of the decentralized nature of blockchain, all transactions or processes can be transparently viewed by every user that using the system and this will violate the privacy of other users. Besides, the blockchain is complicated and applying blockchain technology in this project is just make a big fuss over a minor issue which will cause unnecessary waste of resources.

2.3 Project Methodology

This project is developed based on the Object-Oriented Analysis Design (OOAD). Object-oriented life cycle model considers "objects" as the basis of the software engineering process. The development team starts by observing and analysing the system they intend to develop before defining the requirements. Once the process is over, they focus on identifying the objects of the system. Now, an object could be anything; it can have a physical existence like a customer, car, etc. An object also constitutes intangible elements like a process or a project. The primary objectives of the Object-Oriented Model are object-oriented analysis (OOA), object-oriented design (OOD), and object-oriented implementation (OOI). (Nanda, 2021)

The object-oriented analysis consists of the process where a development team evaluates the system and organizes the requirements as objects. Contrary to traditional structural analysis, the OOA heavily depends on advanced data like Use Cases and Object Models. The OOA starts with analysing the problem domain and produce a conceptual model by thoroughly evaluating the information in the given area. Once the analysis is complete, the development team prepares a conceptual model describing the system's functionalities and requirements. (Nanda, 2021)

It is the next development stage of the object-oriented life cycle model where the analysts design the desired system's overall architecture. The system is divided into a set of interacting subsystems. The analyst considers the specifications from the system analysis. It all about evaluating what the end-users expect from the new system. As per the object-oriented design, the system is considered a collection of objects, with each object handling a specific state data. (Nanda, 2021)

Object-oriented implementation phase, developers translate the class objects and the interrelationships of classes and code them using a programming language. This is the phase to create databases and establish functionalities for the system. The object-oriented methodology focuses on identifying objects in the system. Developers closely observe each object to identify characteristics and behavioural patterns. The developers ensure that the object recognizes and responds perfectly to an event. (Nanda, 2021)

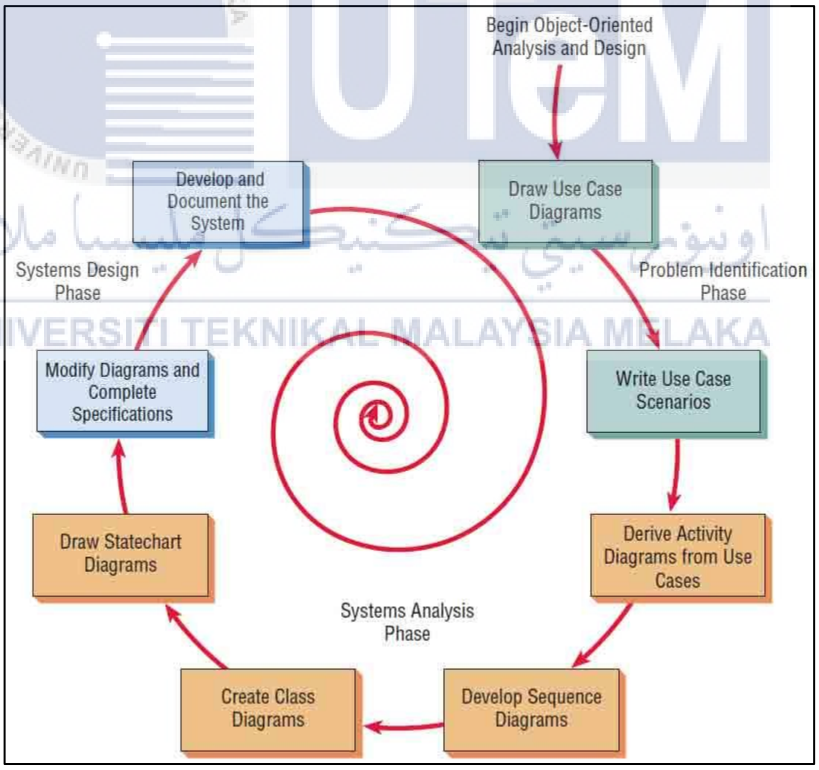

Begin Object-Oriented Analysis and Design
Draw Use Case Diagrams
Problem Identification Phase
Write Use Case Scenarios
Derive Activity Diagrams from Use Cases
Develop Sequence Diagrams
Create Class Diagrams
Draw Statechart Diagrams
Modify Diagrams and Complete Specifications
Systems Design Phase
Develop and Document the System
Systems Analysis Phase

Figure 2.3: Steps in UML Development Process

2.4 Project Requirements

Project requirements describe about the detail of the software and hardware requirements of this project. There is no other requirement that to be used in the project.

2.4.1 Software Requirement

- Visual Studio Code v1.55.2
- Microsoft Visual Studio Professional 2019 v16.9.4
- Android Studio v.4.1.3
- Google Chrome v90.0.4430.93
- Flutter v2.3.0-0.1.pre
- Node.js v14.15.4
- XAMPP Control Panel v3.2.4
- MySQL v15.1
- Apache v2.4.46
- phpMyAdmin v5.1.0
- Windows 10 Home Single Language v20H2

2.4.2 Hardware Requirement

- Laptop
- Android smart phone

2.4.3 Other Requirement

N/A

2.5 Project Schedule and Milestones

This section discusses about the schedule and milestones of the project. Table 2.2 shows the project milestone of this project which undergoes five main activities according to the OOAD methodology.

At the start of the weeks, the requirements of the system will be identified in problem identification phase. The requirements need to be transformed into a Use Case Diagram with the use case scenarios. At week two, the system analysis phase should be started. The use cases in Use Case Diagram need to derive into Activity Diagrams and Sequence Diagrams. The Class Diagram also need to be created. Next will proceed to design phase and the diagrams should be modified and also complete the specification. After that, the system will be documented.

The implementation phase is started at week six. The modules are developed in this phase. At the same time, the unit testing also perform during the development of modules. Then the modules will be combines and integrated after completed. Following by the implementation phase, the testing phase will be started. The test plan will be created first. Then testers will start to execute the test cases and the results and analysis will be recorded after the testing process is done.

During the implementation and testing phase, the schedule will be repeated if the requirements need to be changed. The process is repeat and repeat according to the development life cycle.

Table 2.2: Project Milestone of CaSS

Activity	Period (Week)														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Problem Identification Phase • Draw Use Case Diagram • Write Use Case Scenarios	▨	▨			▨	▨									
System Analysis Phase • Derive Activity Diagrams from Use Cases • Develop Sequence Diagrams • Create Class Diagram		▨	▨			▨				▨	▨				
System Design Phase • Modify Diagrams and Complete Specifications • Develop and Document the System			▨	▨	▨		▨				▨				
Implementation Phase • Develop Modules • Perform Unit Testing in Modules • Combine Modules to Subsystem						▨	▨	▨	▨	▨	▨	▨	▨		

• Integrate and Compile the Subsystems						▨	▨	▨	▨	▨	▨	▨	▨		
Testing Phase • Create Test Plan • Write and Run Test Cases • Document Test Results and Analysis							▨	▨		▨	▨		▨	▨	▨

2.6 Conclusion

This chapter stated the facts and finding from different aspects such as domain, existing system, and also technique. These findings will be used for analysis the problem and the requirement of the proposed system. The project methodology is decided and the project milestones are scheduled according to this methodology. Next chapter will discuss about the analysis of the current system for identifying the requirements.

CHAPTER 3: ANALYSIS

3.1 Introduction

This chapter details the analysis that was done on this system. Analysis was done by analyzing the existing system, finding out the problems, and implementing new ideas and innovation as solutions. This is done to define the goals and objectives.

3.2 Problem Analysis

The problem analysis discusses about the current system scenario and situation. Figure 3.1 shows the activity diagram of the make reservation for the current system and Figure 3.2 shows the activity diagram of the servicing process for the current system.

To make a new reservation, customer needs to make a call to the service center or the branch. Then the customer will choose the date and time of reservation with the help and discussion of the staff to make sure the date time reserved is available and not duplicated. The staff will ask for the customer to provide the personal information such as name and phone number, and also the detail of car that will been serviced for this reservation.

On the reserved date, the customer will drive the to the service center. After confirm the information of reservation, the staff will let the technician to checking the car and lists out the problems of the car with suggested solutions to fix the problems. The staff will call the customer to talk about the result of checking car and ask the customer to choose and agree the suggested solutions. After the customer selected the solutions that want to apply to the car, the staff will let the technician to repairing the car according to the solutions that customer selected. After finish repairing the car, staff will call customer again to announce the customer to pick up the car.

The process of servicing car is complicated and involved a lot of communication between customer and staff. The best way of the communication for the current system is through the phone calling so that the customer needs to give out the decision in a short time during the call. The progress and time of servicing process is unpredictable, so the customer unable to prepare for the arrival of each stage of the process.

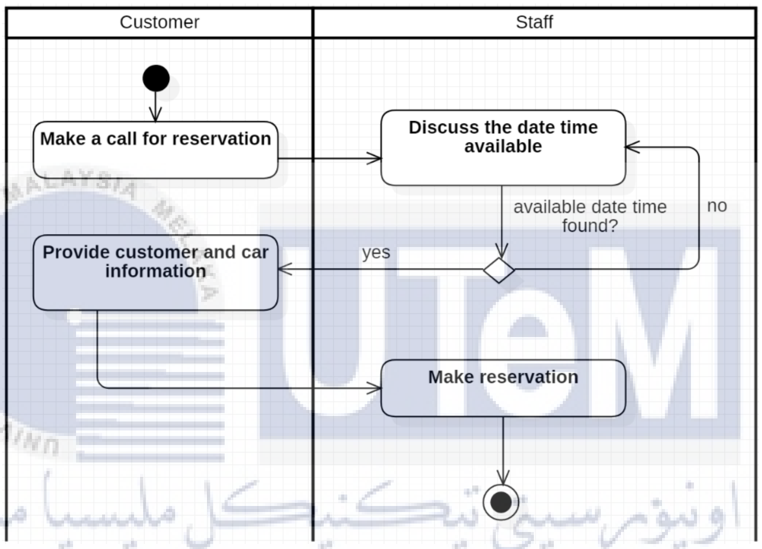

Customer
Staff
Make a call for reservation
Discuss the date time available
available date time found?
no
yes
Provide customer and car information
Make reservation

Figure 3.1: Activity Diagram of Make Reservation for Current System

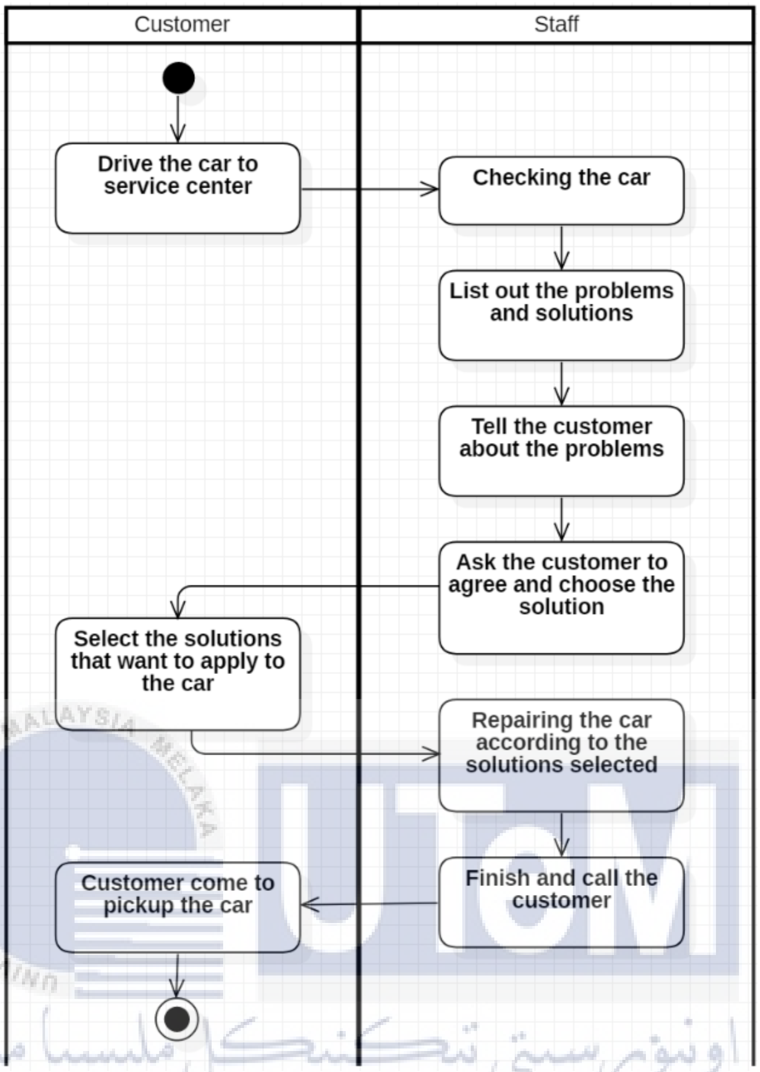

Customer
Staff
Drive the car to service center
Checking the car
List out the problems and solutions
Tell the customer about the problems
Ask the customer to agree and choose the solution
Select the solutions that want to apply to the car
Repairing the car according to the solutions selected
Finish and call the customer
Customer come to pickup the car

Figure 3.2: Activity Diagram of Service Car for Current System

3.3 Requirement Analysis

The requirement analysis discusses about some important requirements in this project which are the data requirement, functional requirement and also non-functional requirement.

3.3.1 Data Requirement

This section discusses about the system input and output, and also the data that store internally in the system. The data requirement is represented by using data dictionary and there are thirteen tables are involved.

Table 3.1: Data Dictionary of Customer

Name	Type	Null	Constraint	Description
customer_id	integer	no	Primary Key	Unique ID of customers
name	varchar	no	-	Name of customers
phone_no	varchar	no	Unique	Phone number of customers
email	varchar	yes	Unique	Email of customers
password	varchar	yes	-	Password of customers who are app user only
type	varchar	no	-	Type of customers which have “app user” and “normal user”

Table 3.2: Data Dictionary of Car

Name	Type	Null	Constraint	Description
car_id	integer	no	Primary Key	Unique ID of cars
plate_no	varchar	no	Unique	Plate number of cars
model_id	integer	no	Foreign Key	Unique ID of car models
customer_id	integer	no	Foreign Key	Unique ID of customers

Table 3.3: Data Dictionary of Model

Name	Type	Null	Constraint	Description
model_id	integer	no	Primary Key	Unique ID of car models
name	varchar	no	Unique	Name of car models
brand_id	integer	no	Foreign Key	Unique ID of car brands

Table 3.4: Data Dictionary of Brand

Name	Type	Null	Constraint	Description
brand_id	integer	no	Primary Key	Unique ID of car brands
name	varchar	no	Unique	Name of car brands

Table 3.5: Data Dictionary of Branch

Name	Type	Null	Constraint	Description
branch_id	integer	no	Primary Key	Unique ID of branches
name	varchar	no	Unique	Name of branches
location	varchar	no	-	Location of branches
email	varchar	no	Unique	Email of branches
password	varchar	no	-	Password of branches
datetime_register	datetime	no	-	Date time of registration of branches

Table 3.6: Data Dictionary of Branch_Service

Name	Type	Null	Constraint	Description
bs_id	integer	no	Primary Key	Unique ID of service that had provided by branch
branch_id	integer	no	Foreign Key	Unique ID of branches
service_id	integer	no	Foreign Key	Unique ID of service
bs_availability	boolean	no	-	True if the service of branch is available, false otherwise

Table 3.7: Data Dictionary of Service

Name	Type	Null	Constraint	Description
service_id	integer	no	Primary Key	Unique ID of services
name	varchar	no	Unique	Name of services
description	varchar	no	-	Description of services
fee	decimal	no	-	Fee of services in RM

Table 3.8: Data Dictionary of Task

Name	Type	Null	Constraint	Description
task_id	integer	no	Primary Key	Unique ID of tasks
description	varchar	no	-	Description of tasks
service_id	integer	no	Foreign Key	Unique ID of services

Table 3.9: Data Dictionary of Action

Name	Type	Null	Constraint	Description
action_id	integer	no	Primary Key	Unique ID of actions
description	varchar	no	-	Description of actions
price	decimal	no	-	Price of actions
task_id	integer	no	Foreign Key	Unique ID of tasks
model_id	integer	no	Foreign Key	Unique ID of car models

Table 3.10: Data Dictionary of Reservation

Name	Type	Null	Constraint	Description
reservation_id	integer	no	Primary Key	Unique ID of reservations
datetime_reserved	datetime	no	-	Date time created of reservations
datetime_to_service	datetime	no	-	Date time to service of reservations
status	varchar	no	-	Status of reservations which have “reserved”, “servicing”, “serviced”, “completed”, and “cancelled”.
remark	varchar	yes	-	Remark of reservations
car_id	integer	no	Foreign Key	Unique ID of cars
branch_id	integer	no	Foreign Key	Unique ID of branches

Table 3.11: Data Dictionary of History

Name	Type	Null	Constraint	Description
history_id	integer	no	Primary Key	Unique ID of histories
reservation_id	integer	no	Foreign Key	Unique ID of reservations
payment_id	integer	no	Foreign Key	Unique ID of car payments

Table 3.12: Data Dictionary of Payment

Name	Type	Null	Constraint	Description
payment_id	integer	no	Primary Key	Unique ID of payments
datetime_paid	datetime	no	-	Date time of payments
total_paid	decimal	no	-	Total paid of payments in RM

Table 3.13: Data Dictionary of Implemented_Action

Name	Type	Null	Constraint	Description
i_action_id	integer	no	Primary Key	Unique ID of implemented actions
history_id	integer	no	Foreign Key	Unique ID of histories
action_id	integer	no	Foreign Key	Unique ID of actions

3.3.2 Functional Requirement

This section discusses about the functions of the system, how it records, compute, trans- forms, and transmits data. The functional requirement is represented by using Use Case Diagram with the specification.

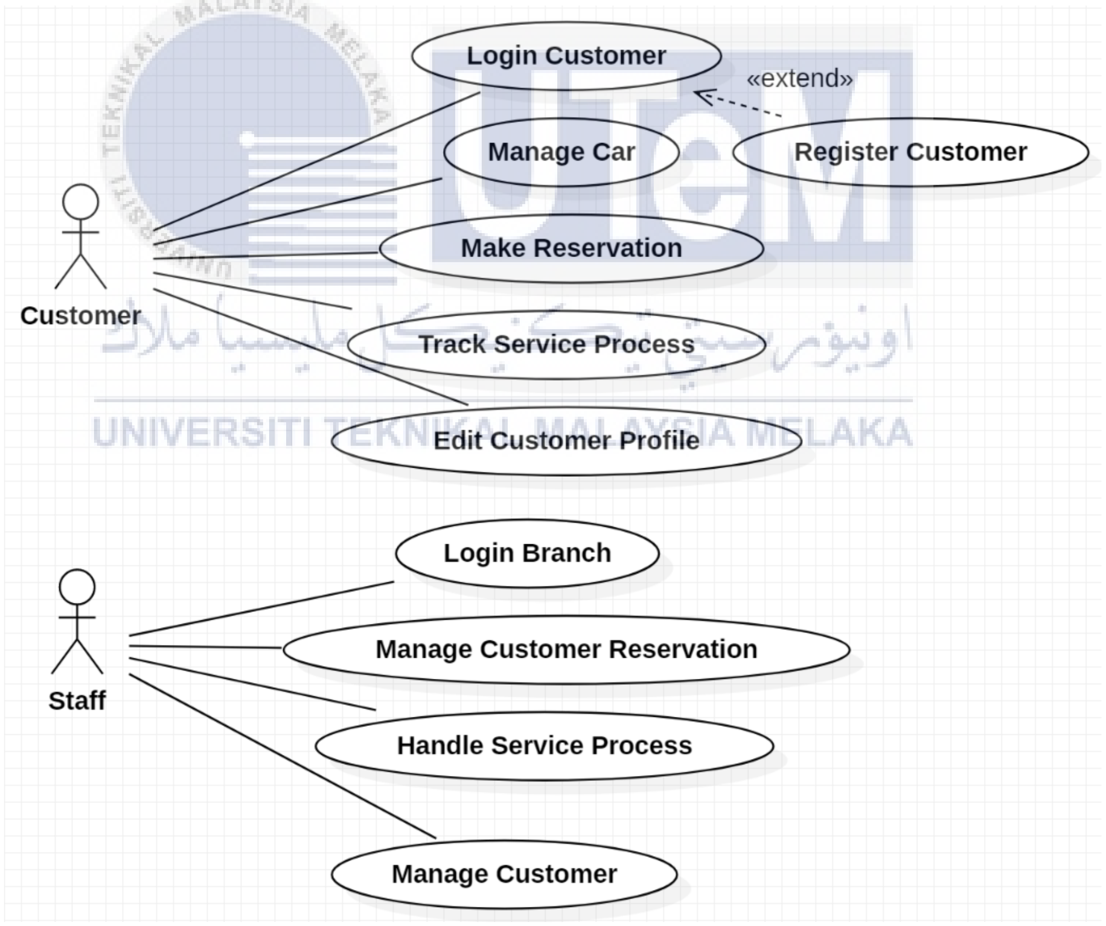


Figure 3.3: Use Case Diagram of CaSS

Figure 3.3 shows the Use Case Diagram of the system. The customer should be able to login and register to the system. The customer should be able to manage the car by adding, updating, and removing the car from the system. The customer should be able to make the reservation for the car that had been added to the system. The customer also should be able to track the service process once the service had started. The customer should be able to edit the profile such as change the email address or phone number that registered in the system. Besides, the staff should be able to authenticate by the branch account with the password. The staff should be able to manage the reservations of the customers with their cars. The staff also should be able to handle the service process which can update the service progress. The staff should be able to manage the customers such as adding a new customer or update the customer information.

3.3.3 Non-Functional Requirement

Non-functional requirement is the requirement that can specify how well the system performs its intended function. Table 3.14 shows the non-functional requirements that involved in this system.

Table 3.14: Non-Functional Requirements of CaSS

Category	Description
Performance	The system should be able to load the page within 10 seconds.
Security	The system should be able to distinguish between authorized and non-authorized users.
Availability	The system should be available for 24 hours.
Integrity	Whenever a change is made, the changes shall be updated in the database.

3.3.4 Others Requirement

Other requirement is the software, hardware and requirements that will be used in the system. The internet connection should be available all the time while using the system. Only the smart phone with the Android OS can be use the customer mobile application while only the computer with Windows OS can be use the branch desktop application.

3.4 Conclusion

This chapter analyzed the problem and requirement for the project. There are many requirements such as data requirement, functional requirement, non-functional requirement and also other requirement are analyzed from the current system. Next chapter will start to design the proposed system according to these requirements.

CHAPTER 4: DESIGN

4.1 Introduction

This chapter will discuss the system design and the activities involved. The activities involved includes converting information, functional, and non-functional requirements that was identified in the analysis phase into the design specification. This chapter will also discuss about the detailed design which included the software design and physical database design.

4.2 High-Level Design

This section discusses about the high-level view of the structure and interior of the system. The high-level design contains the system architecture, user interface design, and also the database design.

4.2.1 System Architecture

Figure 4.1 shows the architecture view of the system. There are three tiers involved in the architecture design which are presentation tier, logic tier and data tier. User access to the presentation tier with different devices according to the user type. The presentation tier will communicate with logic tier, which is a RESTful API web service through the HTTP request. Then, the web service will send the query to the data tier asynchronously. After the query had been execute by database, the data will be sent back to the web service and the web service will encapsulate the data into JSON format. The JSON data is send back to the presentation tier through HTTP response with the response status.

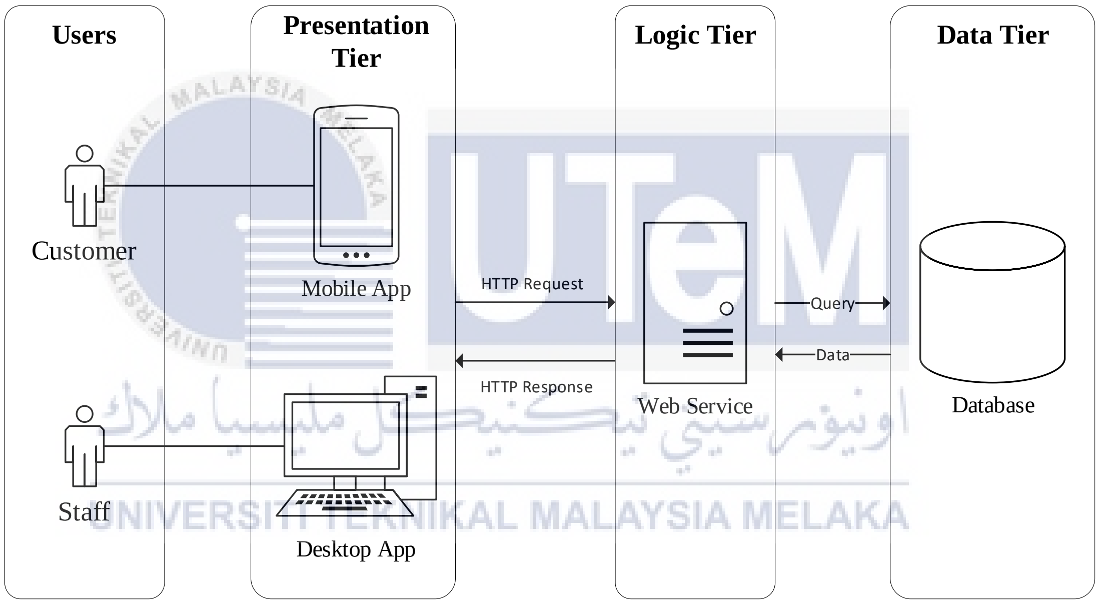


Figure 4.1: Architecture View of CaSS

4.2.2 User Interface Design

This section discusses about the user interface design of the system which included the navigation design, input design and output design. The navigation design defined the navigation flow and types of navigation control. Input design defined the screen used to enter the information, as well as any forms on which users write or type information with the validation rule for each input field. Output design defined the types of outputs including detail reports, summary reports, turnaround documents and graphs and the output is classified in term of periodically or ad-hoc basis.

4.2.2.1 UI Design of CaSS Branch

Figure 4.2 shows the login screen of CaSS Branch. The Staff need to enter the branch email with the correct password to login to the system. The email and password fields are required and the format of email will be validated.

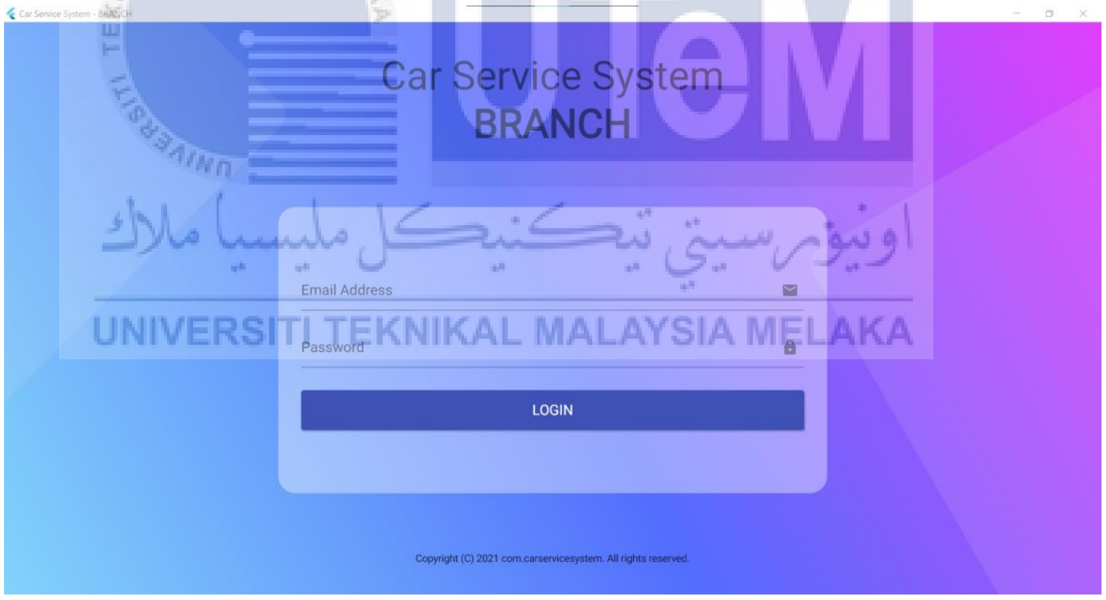

Car Service System - BRANCH
Car Service System
BRANCH
Email Address
Password
LOGIN
Copyright (C) 2021 com.carservicesystem. All rights reserved.

Figure 4.2: Login Screen of CaSS Branch

Figure 4.3 and Figure 4.4 show the dashboard screen of CaSS Branch. There are three panels in this screen which are the reservation overview panel, reservation statistic panel and service ranking panel. Reservation overview panel shows the total and today reservations that divided according to the status of reservation. Reservation statistic panel shows the graph of reservations in a specific year. Service ranking panel shows the ranking of the service which the customer chosen the most in the reservation.

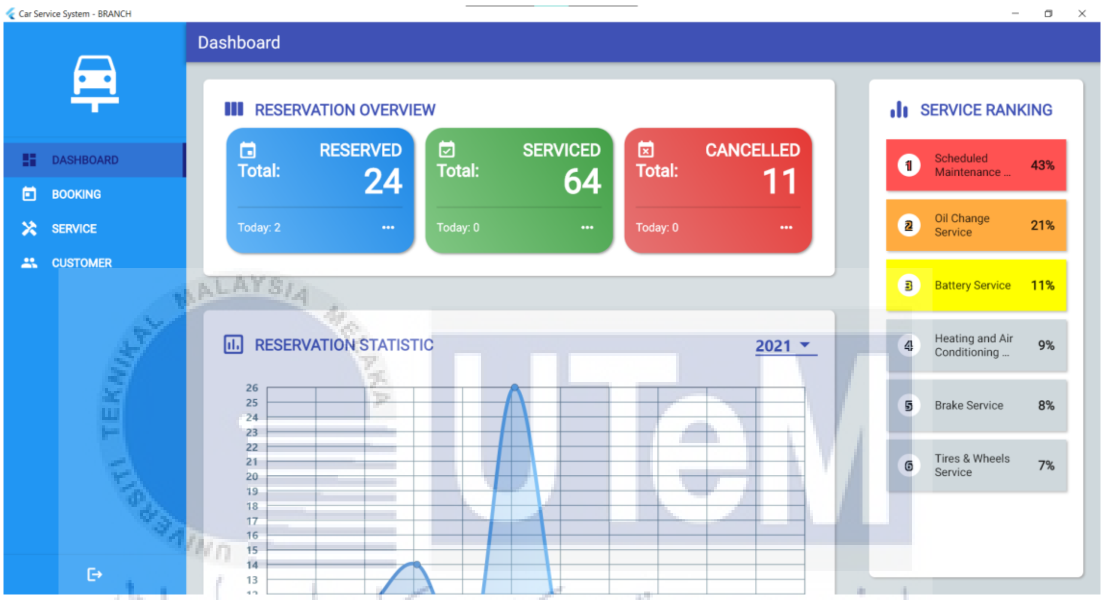


Figure 4.3: Dashboard Screen 1 of CaSS Branch

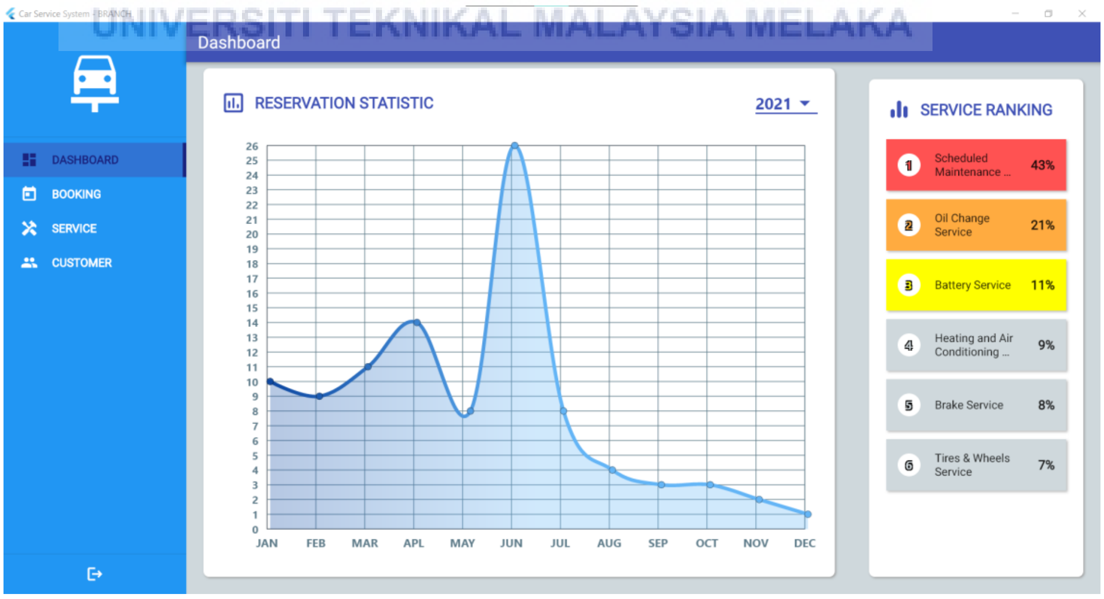


Figure 4.4: Dashboard Screen 2 of CaSS Branch

Figure 4.5, Figure 4.6 and Figure 4.7 show the booking screen of CaSS Branch in different view. The staff is able to view the reservation detail by clicking the events on the calendar. The staff is able to filter the reservation displayed by the status of the reservation. The staff is also able to make a new reservation to the calendar. Figure 4.8 shows the add reservation dialog.

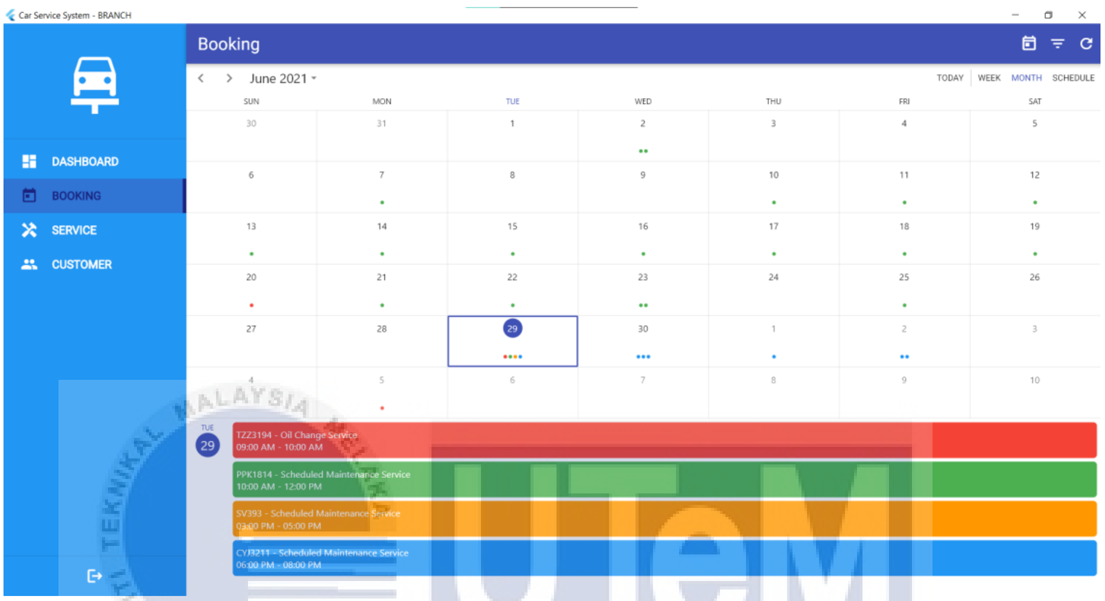

Car Service System - BRANCH
Booking
DASHBOARD
BOOKING
SERVICE
CUSTOMER
June 2021
TODAY
WEEK
MONTH
SCHEDULE
SUN
MON
TUE
WED
THU
FRI
SAT
TZZ3194 - Oil Change Service
09:00 AM - 10:00 AM
PPK1814 - Scheduled Maintenance Service
10:00 AM - 12:00 PM
SV393 - Scheduled Maintenance Service
03:00 PM - 05:00 PM
CYJ3211 - Scheduled Maintenance Service
06:00 PM - 08:00 PM

Figure 4.5:Booking Screen in Month View of CaSS Branch

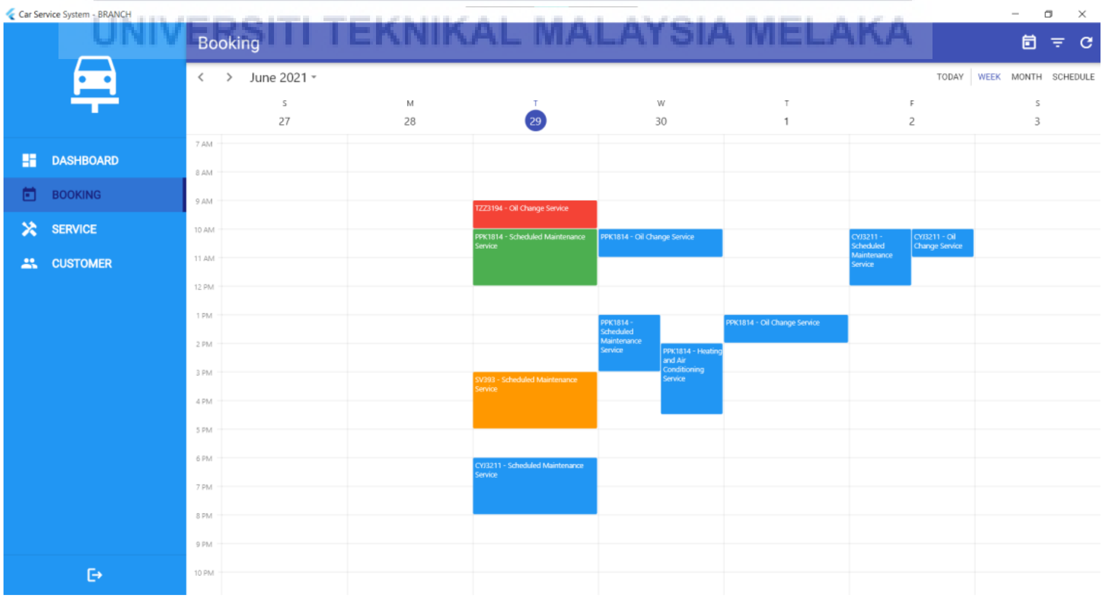
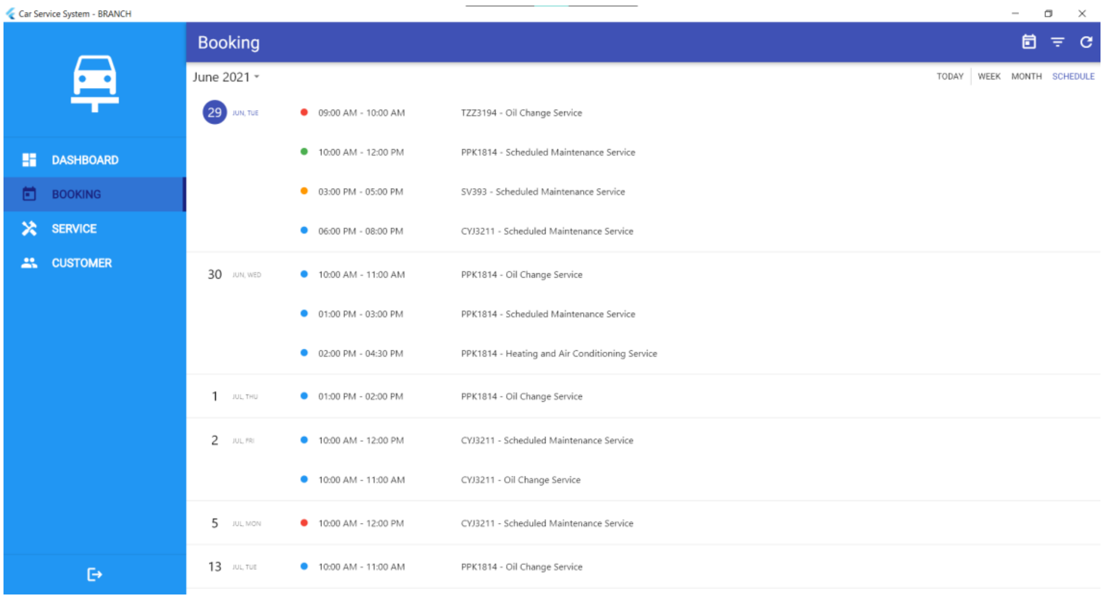

Car Service System - BRANCH
Booking
DASHBOARD
BOOKING
SERVICE
CUSTOMER
June 2021
TODAY
WEEK
MONTH
SCHEDULE
TZZ3194 - Oil Change Service
PPK1814 - Scheduled Maintenance Service
PPK1814 - Oil Change Service
CYJ3211 - Scheduled Maintenance Service
CYJ3211 - Oil Change Service
PPK1814 - Scheduled Maintenance Service
PPK1814 - Heating and Air Conditioning Service
PPK1814 - Oil Change Service
SV393 - Scheduled Maintenance Service
CYJ3211 - Scheduled Maintenance Service

Figure 4.6:Booking Screen in Week View of CaSS Branch



June 2021

29 JUN, TUE ● 09:00 AM - 10:00 AM TZZ3194 - Oil Change Service
● 10:00 AM - 12:00 PM PPK1814 - Scheduled Maintenance Service
● 03:00 PM - 05:00 PM SV393 - Scheduled Maintenance Service
● 06:00 PM - 08:00 PM CYJ3211 - Scheduled Maintenance Service

30 JUN, WED ● 10:00 AM - 11:00 AM PPK1814 - Oil Change Service
● 01:00 PM - 03:00 PM PPK1814 - Scheduled Maintenance Service
● 02:00 PM - 04:30 PM PPK1814 - Heating and Air Conditioning Service

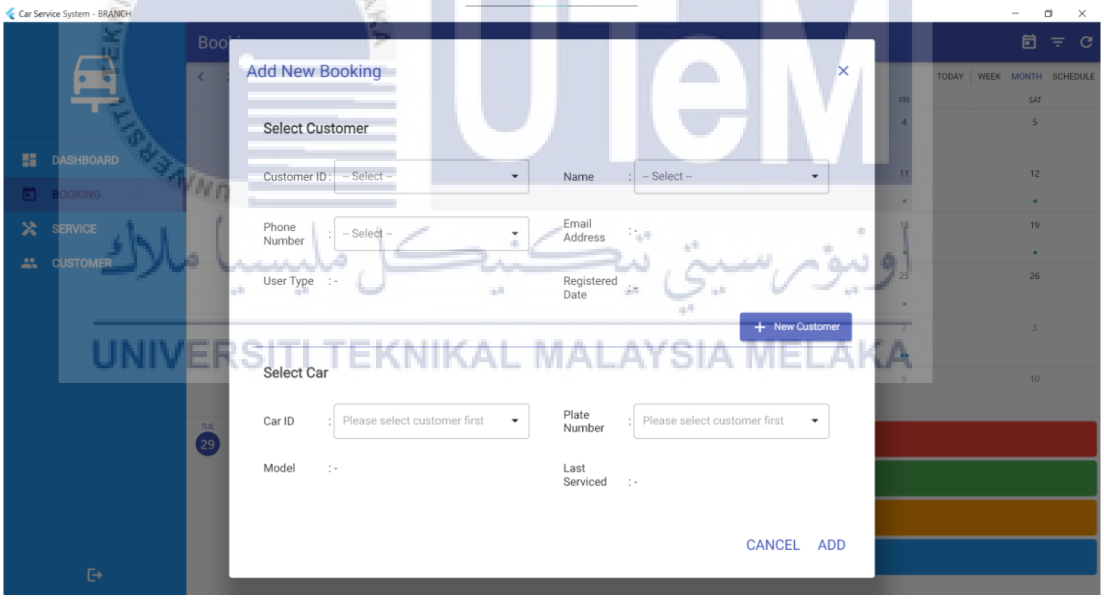
1 JUL, THU ● 01:00 PM - 02:00 PM PPK1814 - Oil Change Service

2 JUL, FRI ● 10:00 AM - 12:00 PM CYJ3211 - Scheduled Maintenance Service
● 10:00 AM - 11:00 AM CYJ3211 - Oil Change Service

5 JUL, MON ● 10:00 AM - 12:00 PM CYJ3211 - Scheduled Maintenance Service

13 JUL, TUE ● 10:00 AM - 11:00 AM PPK1814 - Oil Change Service

Figure 4.7:Booking Screen in Schedule View of CaSS Branch



Add New Booking

Select Customer

Customer ID : -- Select --
Name : -- Select --
Phone Number : -- Select --
Email Address : -
User Type : -
Registered Date : -

Select Car

Car ID : Please select customer first
Plate Number : Please select customer first
Model : -
Last Serviced : -

Figure 4.8: Add New Booking Dialog of CaSS Branch

Figure 4.9, Figure 4.10, Figure 4.11, and Figure 4.12 show the service screen of CaSS Branch with different status of reservation selected. The staff is able to click and view the today reservation at the today reservation aside which arranged the reservations according to the status (cancelled reservation will not be shown). The staff can click the start servicing button for the upcoming reservation to start the servicing process. The service screen is able to process multitask so the staff can start multiple servicing at the same times.

Figure 4.12, Figure 4.13, Figure 4.14, and Figure 4.15 show the service screen of CaSS Branch that the servicing reservation in the different progress step. In progress 1, the staff is able to select the actions according to the checking result of each task. In progress 2, the staff need to wait the customer response by the customer mobile app. If the customer does not response, the staff need to contact the customer manually and help the customer to response by clicking the button response at the bottom right corner. In progress 3, the staff need to update the repairing progress. In the progress 4, the servicing process is completed and needed to waiting the customer to pick up the car after payment is made. The status will be updated to "Completed" after customer had made the payment.

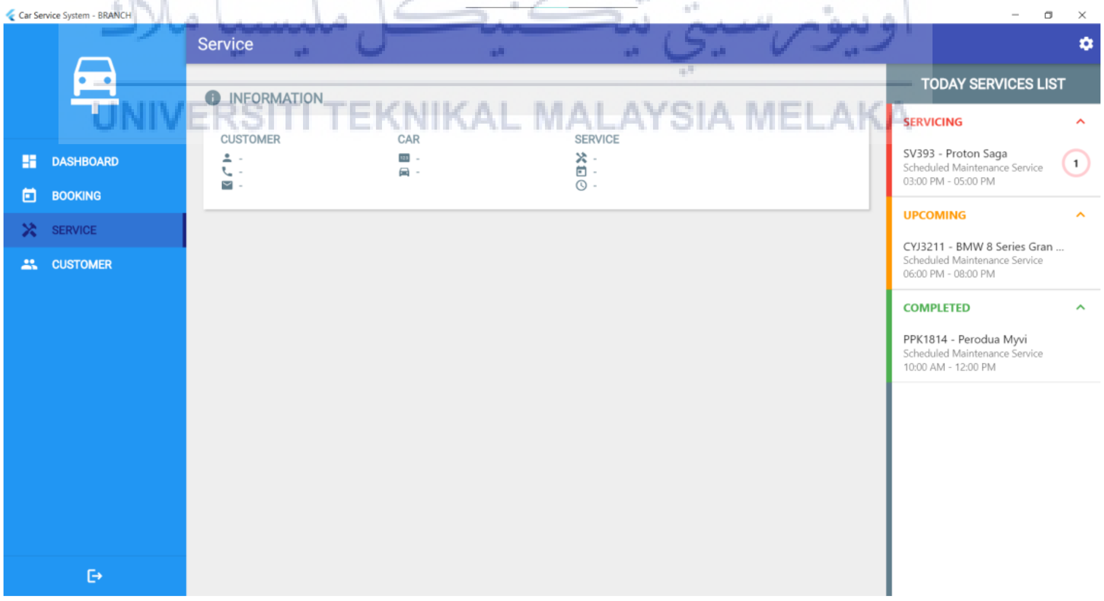
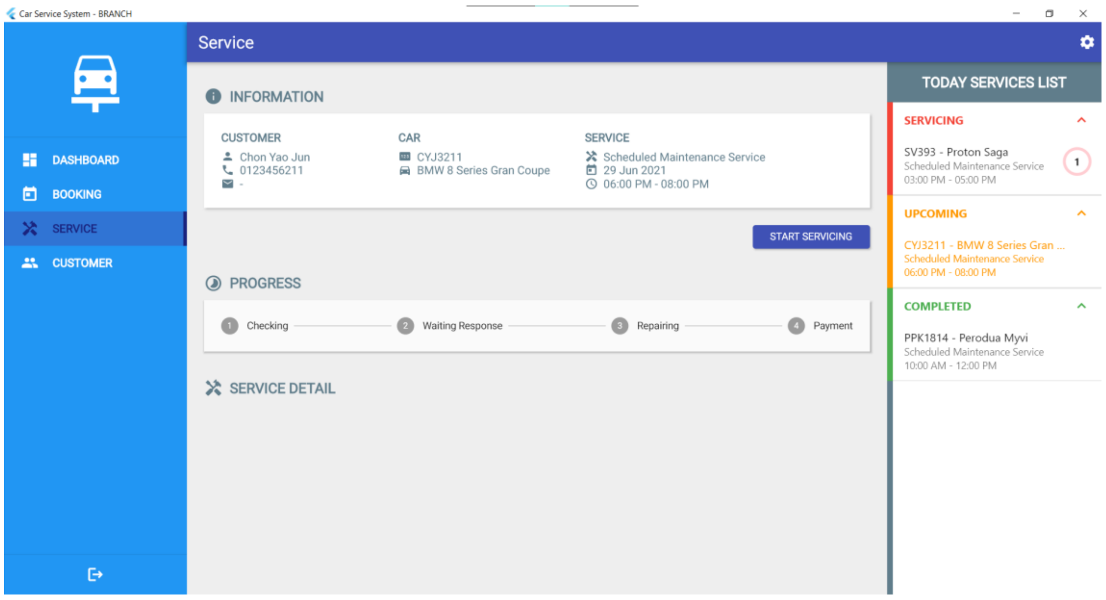


Figure 4.9: Service Screen No Selected of CaSS Branch


Car Service System - BRANCH

Service

DASHBOARD
BOOKING
SERVICE
CUSTOMER

INFORMATION

CUSTOMER
Chon Yao Jun
0123456211
-

CAR
CYJ3211
BMW 8 Series Gran Coupe

SERVICE
Scheduled Maintenance Service
29 Jun 2021
06:00 PM - 08:00 PM

START SERVICING

PROGRESS

1 Checking — 2 Waiting Response — 3 Repairing — 4 Payment

SERVICE DETAIL

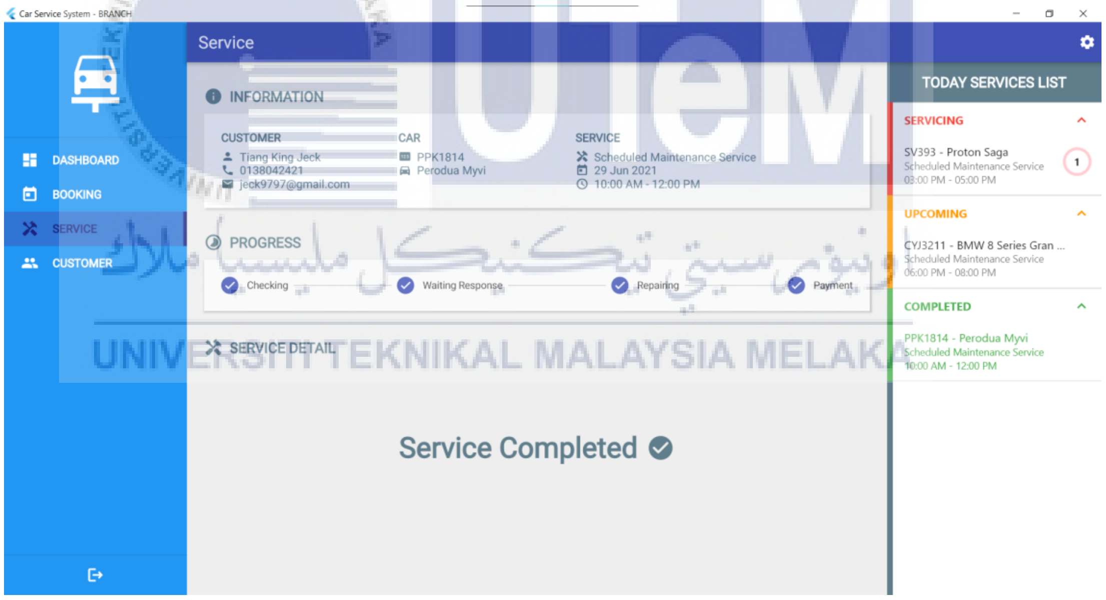
TODAY SERVICES LIST

SERVICING
SV393 - Proton Saga
Scheduled Maintenance Service
03:00 PM - 05:00 PM
1

UPCOMING
CYJ3211 - BMW 8 Series Gran ...
Scheduled Maintenance Service
06:00 PM - 08:00 PM

COMPLETED
PPK1814 - Perodua Myvi
Scheduled Maintenance Service
10:00 AM - 12:00 PM

Figure 4.10: Service Screen Upcoming of CaSS Branch


Car Service System - BRANCH

Service

DASHBOARD
BOOKING
SERVICE
CUSTOMER

INFORMATION

CUSTOMER
Tiang King Jeck
0138042421
jeck9797@gmail.com

CAR
PPK1814
Perodua Myvi

SERVICE
Scheduled Maintenance Service
29 Jun 2021
10:00 AM - 12:00 PM

PROGRESS

Checking — Waiting Response — Repairing — Payment

SERVICE DETAIL

Service Completed

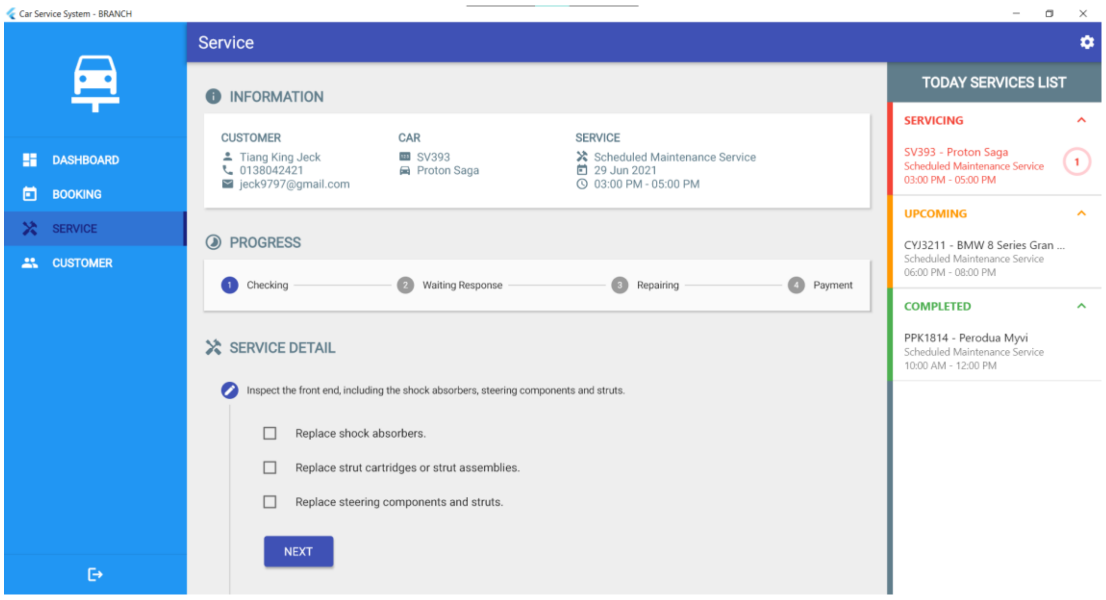
TODAY SERVICES LIST

SERVICING
SV393 - Proton Saga
Scheduled Maintenance Service
03:00 PM - 05:00 PM
1

UPCOMING
CYJ3211 - BMW 8 Series Gran ...
Scheduled Maintenance Service
06:00 PM - 08:00 PM

COMPLETED
PPK1814 - Perodua Myvi
Scheduled Maintenance Service
10:00 AM - 12:00 PM

Figure 4.11: Service Screen Completed of CaSS Branch



INFORMATION

CUSTOMER	CAR	SERVICE
Tiang King Jeck	SV393	Scheduled Maintenance Service
0138042421	Proton Saga	29 Jun 2021
jeck9797@gmail.com		03:00 PM - 05:00 PM

PROGRESS

1 Checking — 2 Waiting Response — 3 Repairing — 4 Payment

SERVICE DETAIL

Inspect the front end, including the shock absorbers, steering components and struts.

- [ ] Replace shock absorbers.
- [ ] Replace strut cartridges or strut assemblies.
- [ ] Replace steering components and struts.

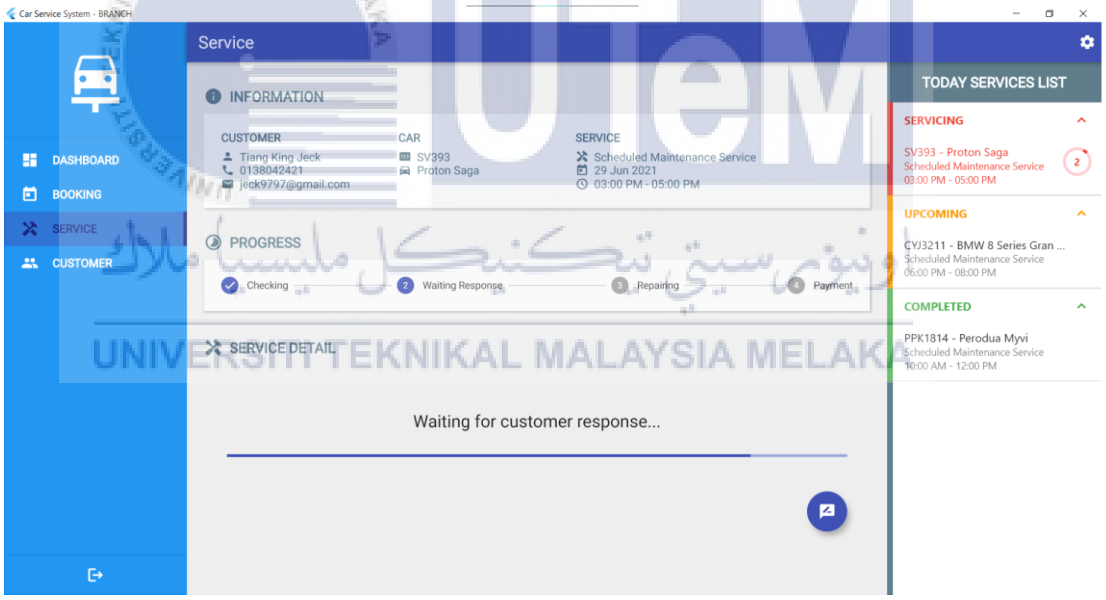
TODAY SERVICES LIST

SERVICING
SV393 - Proton Saga
Scheduled Maintenance Service
03:00 PM - 05:00 PM
1

UPCOMING
CYJ3211 - BMW 8 Series Gran ...
Scheduled Maintenance Service
06:00 PM - 08:00 PM

COMPLETED
PPK1814 - Perodua Myvi
Scheduled Maintenance Service
10:00 AM - 12:00 PM

Figure 4.12: Service Screen Servicing 1 of CaSS Branch



INFORMATION

CUSTOMER	CAR	SERVICE
Tiang King Jeck	SV393	Scheduled Maintenance Service
0138042421	Proton Saga	29 Jun 2021
jeck9797@gmail.com		03:00 PM - 05:00 PM

PROGRESS

Checking — 2 Waiting Response — 3 Repairing — 4 Payment

SERVICE DETAIL

Waiting for customer response...

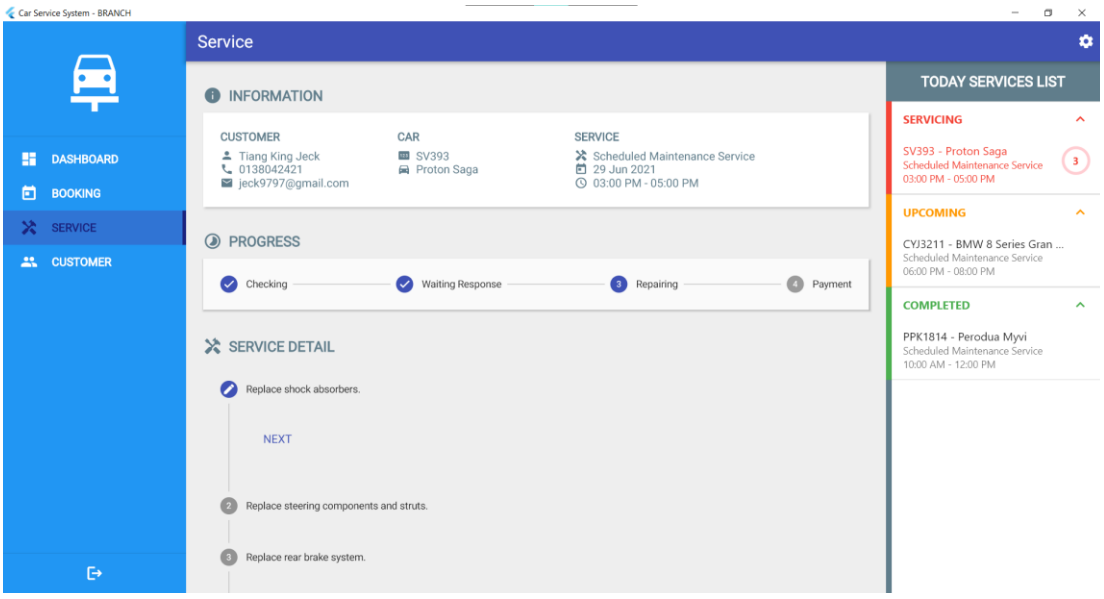
TODAY SERVICES LIST

SERVICING
SV393 - Proton Saga
Scheduled Maintenance Service
03:00 PM - 05:00 PM
2

UPCOMING
CYJ3211 - BMW 8 Series Gran ...
Scheduled Maintenance Service
06:00 PM - 08:00 PM

COMPLETED
PPK1814 - Perodua Myvi
Scheduled Maintenance Service
10:00 AM - 12:00 PM

Figure 4.13: Service Screen Servicing 2 of CaSS Branch



Car Service System - BRANCH

Service

INFORMATION

CUSTOMER	CAR	SERVICE
Tiang King Jeck	SV393	Scheduled Maintenance Service
0138042421	Proton Saga	29 Jun 2021
jeck9797@gmail.com		03:00 PM - 05:00 PM

PROGRESS

Checking — Waiting Response — 3 Repairing — 4 Payment

SERVICE DETAIL

Replace shock absorbers.

NEXT

2 Replace steering components and struts.

3 Replace rear brake system.

TODAY SERVICES LIST

SERVICING

SV393 - Proton Saga
Scheduled Maintenance Service
03:00 PM - 05:00 PM

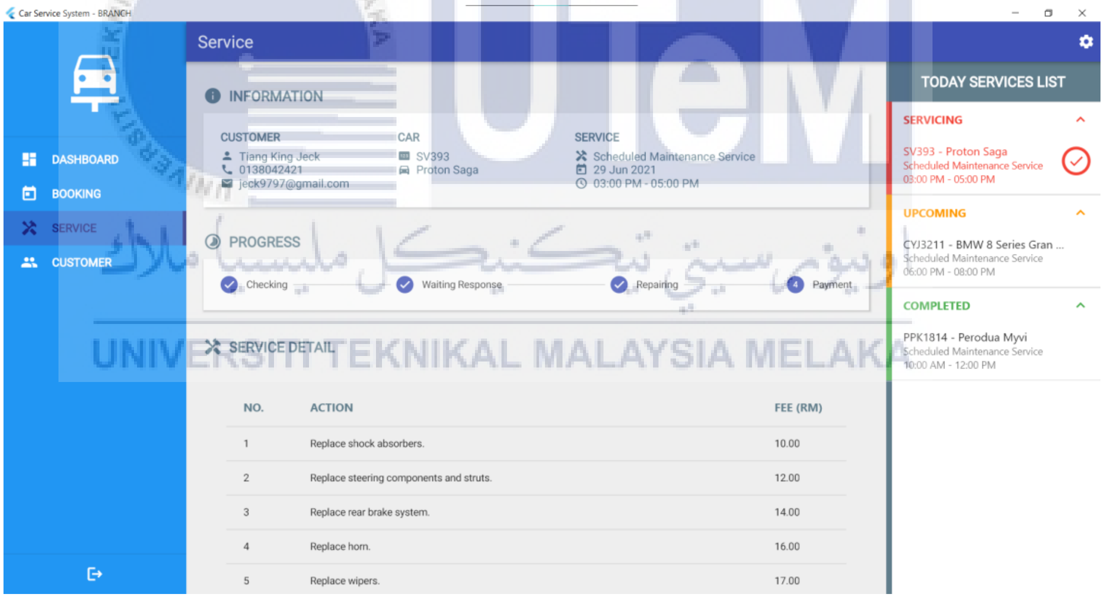
UPCOMING

CYJ3211 - BMW 8 Series Gran ...
Scheduled Maintenance Service
06:00 PM - 08:00 PM

COMPLETED

PPK1814 - Perodua Myvi
Scheduled Maintenance Service
10:00 AM - 12:00 PM

Figure 4.14: Service Screen Servicing 3 of CaSS Branch



Car Service System - BRANCH

Service

INFORMATION

CUSTOMER	CAR	SERVICE
Tiang King Jeck	SV393	Scheduled Maintenance Service
0138042421	Proton Saga	29 Jun 2021
jeck9797@gmail.com		03:00 PM - 05:00 PM

PROGRESS

Checking — Waiting Response — Repairing — 4 Payment

SERVICE DETAIL

NO.	ACTION	FEE (RM)
1	Replace shock absorbers.	10.00
2	Replace steering components and struts.	12.00
3	Replace rear brake system.	14.00
4	Replace horn.	16.00
5	Replace wipers.	17.00

TODAY SERVICES LIST

SERVICING

SV393 - Proton Saga
Scheduled Maintenance Service
03:00 PM - 05:00 PM

UPCOMING

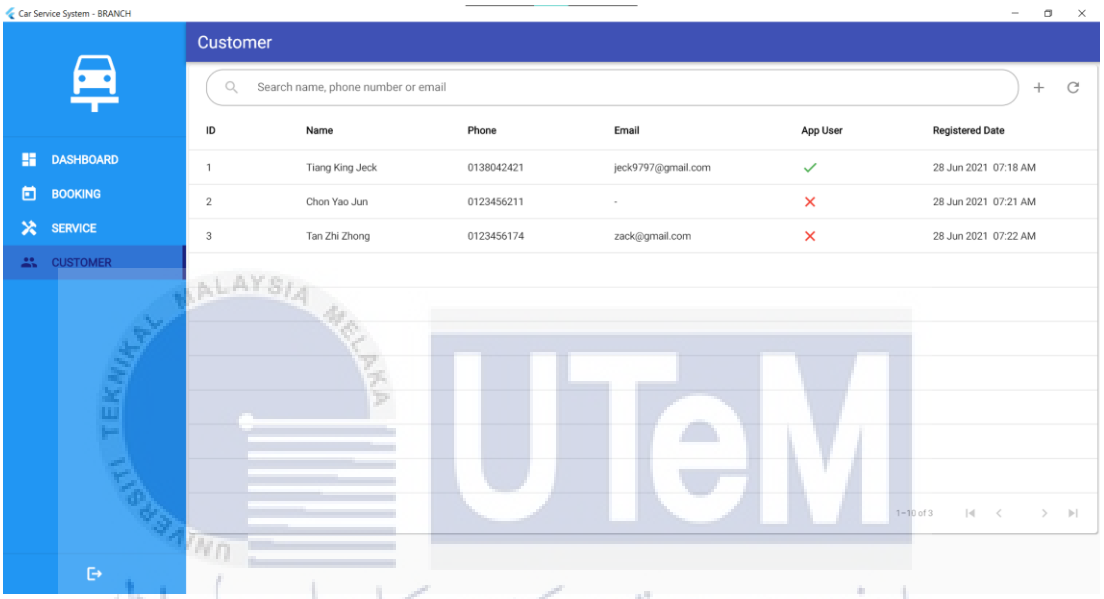
CYJ3211 - BMW 8 Series Gran ...
Scheduled Maintenance Service
06:00 PM - 08:00 PM

COMPLETED

PPK1814 - Perodua Myvi
Scheduled Maintenance Service
10:00 AM - 12:00 PM

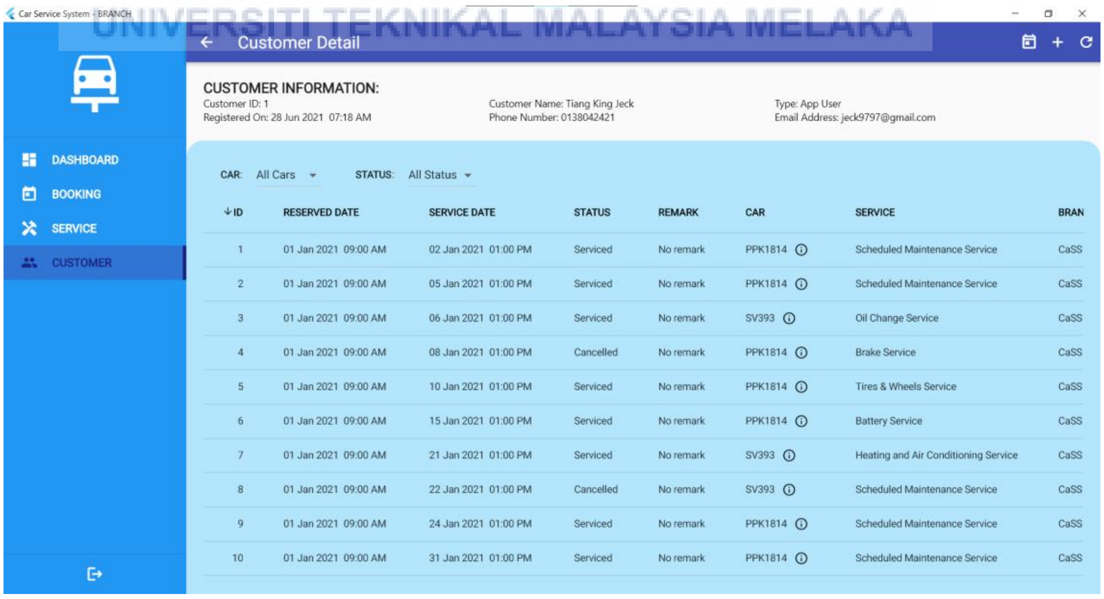
Figure 4.15: Service Screen Servicing 4 of CaSS Branch

Figure 4.16 shows the customer screen of CaSS Branch. This screen will show all the customers in a pagination table The staff are able to search the customer by the customer ID or name. The staff also able to add a new customer for those customers who do not use the customer mobile app. The staff are able to see the detail of specific customer by clicking the customer at the table. Figure 4.17 shows the detail customer screen of CaSS Branch. The staff are able to view the reservations made by this customer and also add a new car information for this customer.


Car Service System - BRANCH
Customer
DASHBOARD
BOOKING
SERVICE
CUSTOMER
Search name, phone number or email
1–10 of 3

ID	Name	Phone	Email	App User	Registered Date
1	Tiang King Jeck	0138042421	jeck9797@gmail.com	✓	28 Jun 2021 07:18 AM
2	Chon Yao Jun	0123456211	-	✗	28 Jun 2021 07:21 AM
3	Tan Zhi Zhong	0123456174	zack@gmail.com	✗	28 Jun 2021 07:22 AM

Figure 4.16: Customer Screen of CaSS Branch



CUSTOMER INFORMATION:
Customer ID: 1
Registered On: 28 Jun 2021 07:18 AM
Customer Name: Tiang King Jeck
Phone Number: 0138042421
Type: App User
Email Address: jeck9797@gmail.com

CAR: All Cars STATUS: All Status

↓ID	RESERVED DATE	SERVICE DATE	STATUS	REMARK	CAR	SERVICE	BRAN
1	01 Jan 2021 09:00 AM	02 Jan 2021 01:00 PM	Serviced	No remark	PPK1814	Scheduled Maintenance Service	CaSS
2	01 Jan 2021 09:00 AM	05 Jan 2021 01:00 PM	Serviced	No remark	PPK1814	Scheduled Maintenance Service	CaSS
3	01 Jan 2021 09:00 AM	06 Jan 2021 01:00 PM	Serviced	No remark	SV393	Oil Change Service	CaSS
4	01 Jan 2021 09:00 AM	08 Jan 2021 01:00 PM	Cancelled	No remark	PPK1814	Brake Service	CaSS
5	01 Jan 2021 09:00 AM	10 Jan 2021 01:00 PM	Serviced	No remark	PPK1814	Tires & Wheels Service	CaSS
6	01 Jan 2021 09:00 AM	15 Jan 2021 01:00 PM	Serviced	No remark	PPK1814	Battery Service	CaSS
7	01 Jan 2021 09:00 AM	21 Jan 2021 01:00 PM	Serviced	No remark	SV393	Heating and Air Conditioning Service	CaSS
8	01 Jan 2021 09:00 AM	22 Jan 2021 01:00 PM	Cancelled	No remark	SV393	Scheduled Maintenance Service	CaSS
9	01 Jan 2021 09:00 AM	24 Jan 2021 01:00 PM	Serviced	No remark	PPK1814	Scheduled Maintenance Service	CaSS
10	01 Jan 2021 09:00 AM	31 Jan 2021 01:00 PM	Serviced	No remark	PPK1814	Scheduled Maintenance Service	CaSS

Figure 4.17: Customer Detail Screen of CaSS Branch

4.2.2.2 UI Design of CaSS Customer

Figure 4.18 shows the authentication screen of CaSS Customer. The customer need to login to the system by using email and password. The customer also able to register a new account by filling the required information.

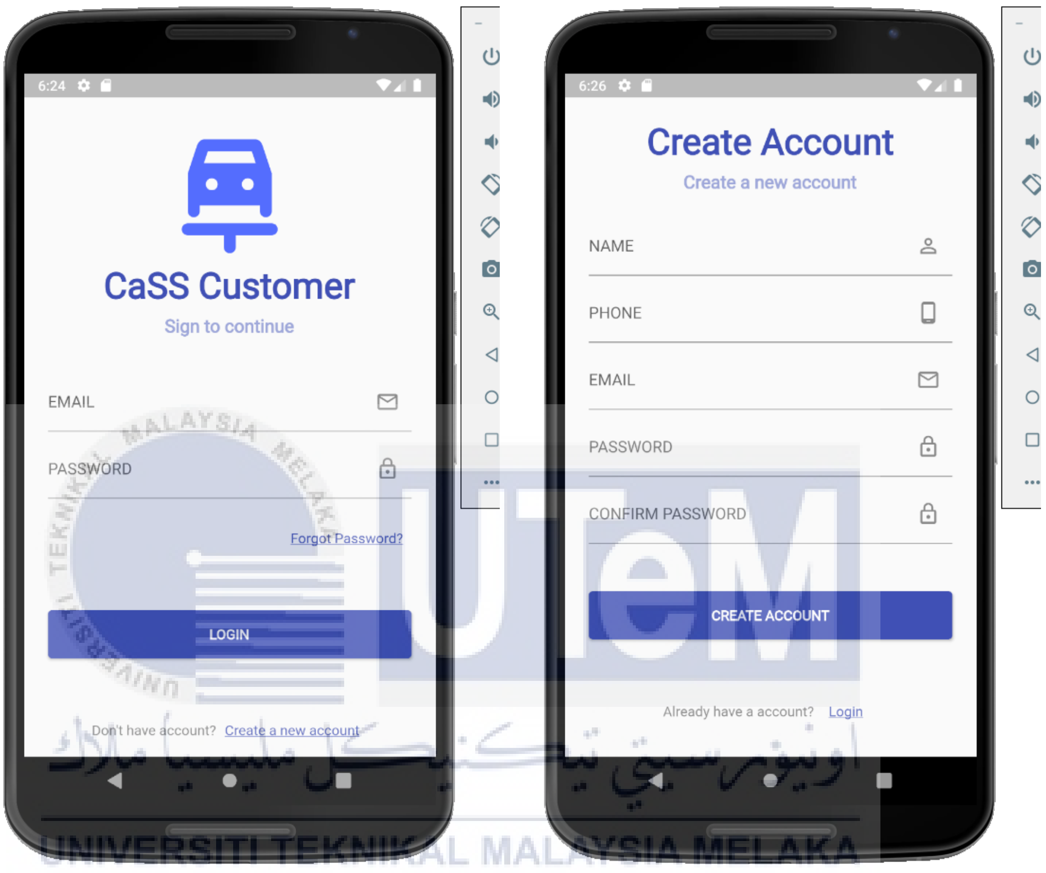


Figure 4.18: Authentication Screen of CaSS Customer

Figure 4.19 shows the car screen of CaSS Customer. The customer is able to view the added cars or add a new car to the system.

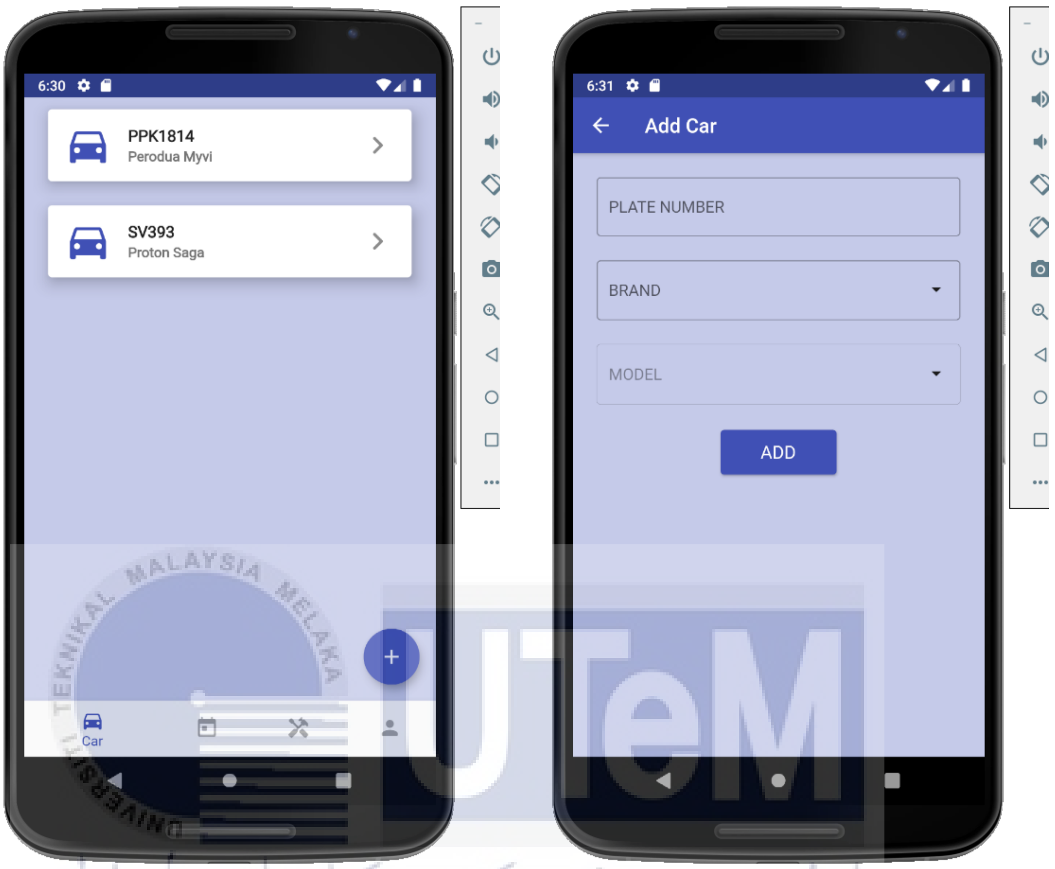
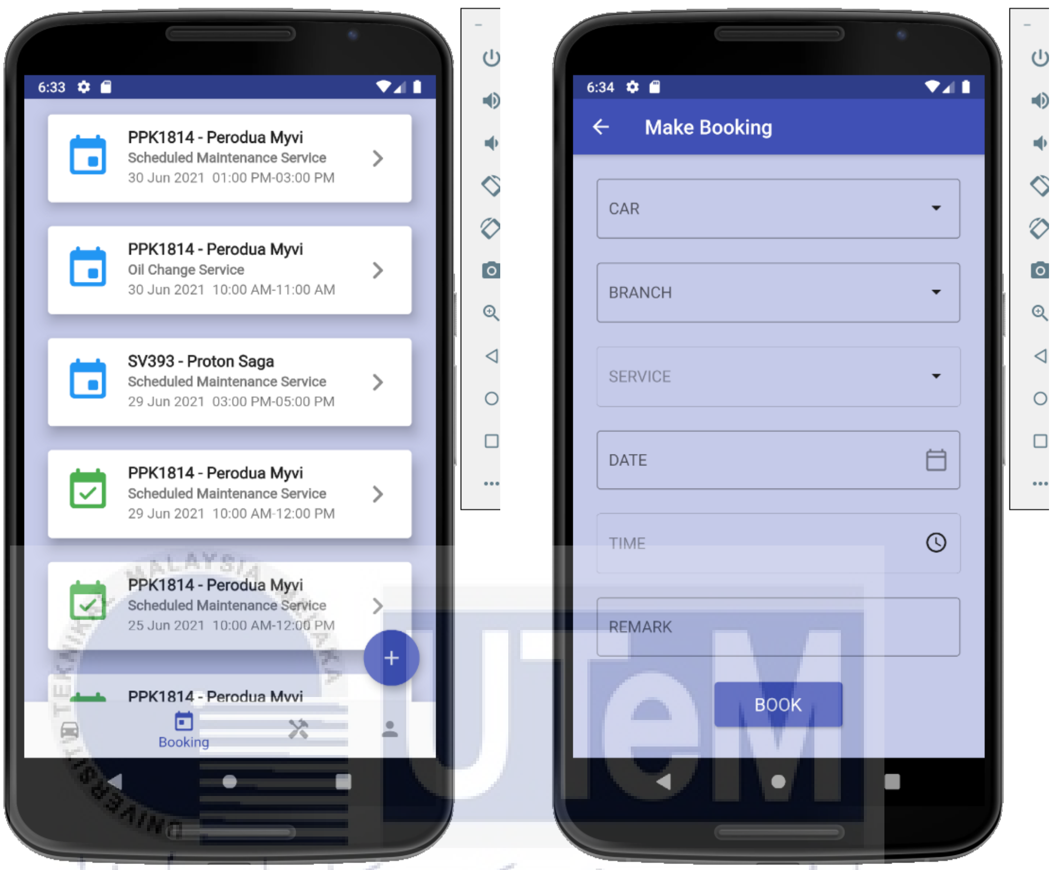


Figure 4.19: Car Screen of CaSS Customer

Figurer 4.20 shows the booking screen of CaSS Customer. The customer is able to view all the made booking or add a new booking for the car.


6:33

PPK1814 - Perodua Myvi
Scheduled Maintenance Service
30 Jun 2021 01:00 PM-03:00 PM

PPK1814 - Perodua Myvi
Oil Change Service
30 Jun 2021 10:00 AM-11:00 AM

SV393 - Proton Saga
Scheduled Maintenance Service
29 Jun 2021 03:00 PM-05:00 PM

PPK1814 - Perodua Myvi
Scheduled Maintenance Service
29 Jun 2021 10:00 AM-12:00 PM

PPK1814 - Perodua Myvi
Scheduled Maintenance Service
25 Jun 2021 10:00 AM-12:00 PM

PPK1814 - Perodua Myvi

6:34

Make Booking

Figure 4.20: Booking Screen of CaSS Customer

Figure 4.21, Figure 4.22, Figure 4.23, Figure 4.24, and Figure 4.25 shows the service screen of CaSS Customer with different progress step. Figure 4.21 show the service screen that does not have servicing car.

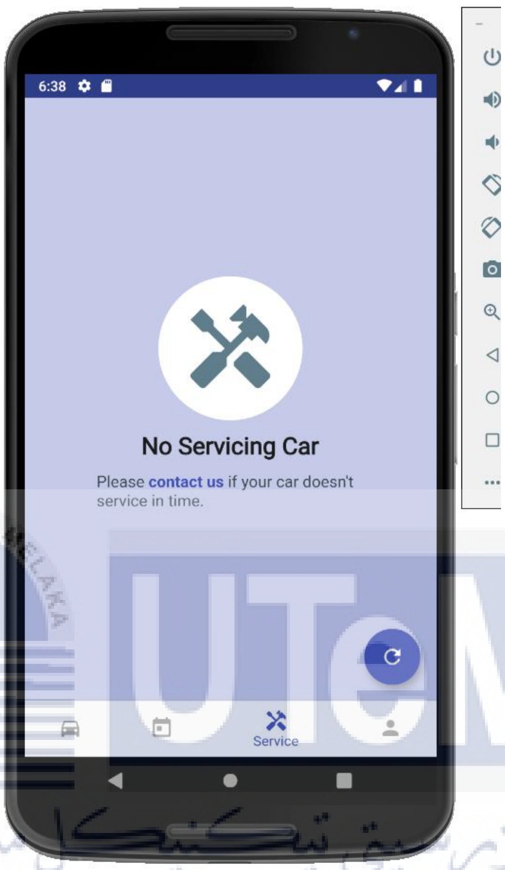


Figure 4.21: Service Screen No Servicing of CaSS Customer

Figure 4.22 shows the service screen in progress 1. The screen is update to progress 1 after the service process is started by the branch. The screen will show the progress that updated in real-time from the branch. The current task and the progress percentage will be tracked and shown.

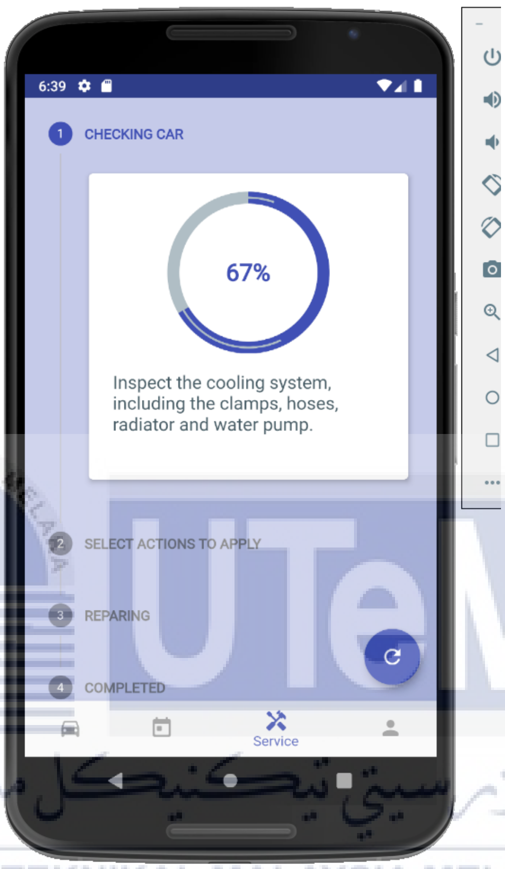
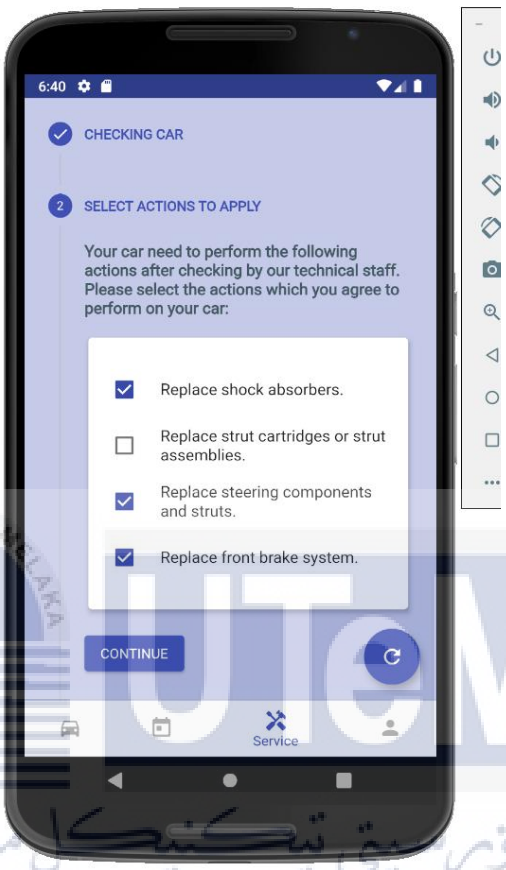

6:39
1 CHECKING CAR
67%
Inspect the cooling system, including the clamps, hoses, radiator and water pump.
2 SELECT ACTIONS TO APPLY
3 REPARING
4 COMPLETED
Service

Figure 4.22: Service Screen Progress 1 of CaSS Customer

Figure 4.23 shows the service screen in progress 2. The screen is updated to progress two after the progress 1 is done. The customer is able to select the action that want to apply to the car. After clicking continue, the progress will update and proceed.


6:40

CHECKING CAR

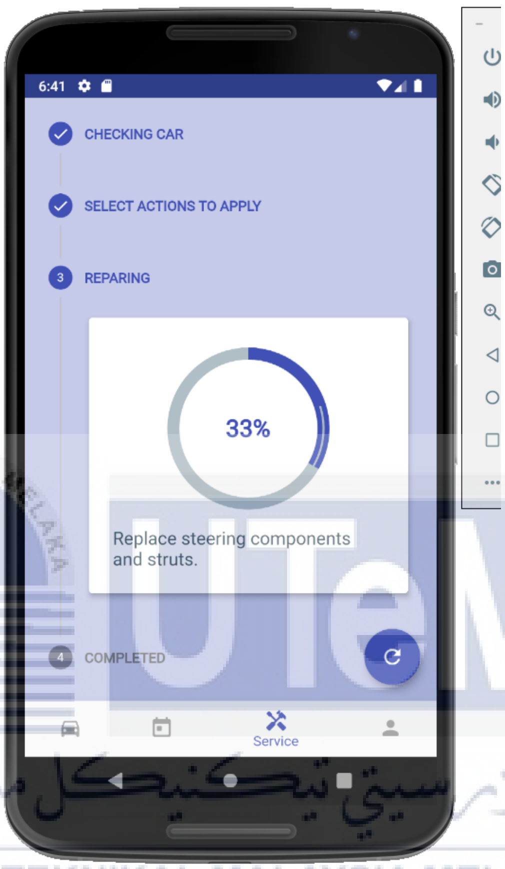
2 SELECT ACTIONS TO APPLY

Your car need to perform the following actions after checking by our technical staff. Please select the actions which you agree to perform on your car:

- [x] Replace shock absorbers.
- [ ] Replace strut cartridges or strut assemblies.
- [x] Replace steering components and struts.
- [x] Replace front brake system.

Figure 4.23: Service Screen Progress 2 of CaSS Customer

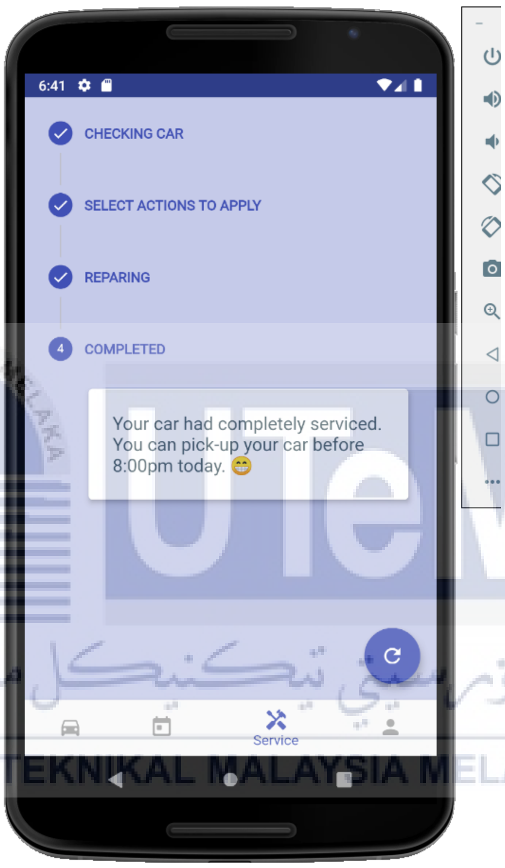
Figure 4.24 shows the service screen in progress 3. The screen will show the percentage of the progress that the actions to be implemented to the car. The customer is able to track the progress in real-time of the repairing process. The current implemented action will be shown with the current progress.



Replace steering components and struts.

Figure 4.24: Service Screen Progress 3 of CaSS Customer

Figure 4.25 shows the service screen in progress 4. The screen is updated to progress 4 after all the repairing process had been done. This screen shows the message that notifies the customer which the car had been done for servicing and prompt the customer to go to the branch for picking up the car. After the payment had been made by customer, this screen will be reset to the no servicing screen as shown in the Figure 4.21.



Your car had completely serviced. You can pick-up your car before 8:00pm today. 😁

Figure 4.25: Service Screen Progress 4 of CaSS Customer

4.2.3 Database Design

This section discusses about the conceptual and logical database design. The database design is represented by using Entity Relationship Diagram (ERD) which had been normalized to avoid the data redundancy.

4.2.3.1 Conceptual and Logical Database Design

Figure 4.26 shows the conceptual database design by using ER diagram which constructed according the business rules of the system. Figure 4.27 shows the logical database design which enriched from conceptual database design by defining explicitly the columns in each entity and introducing operational and transactional entities. The business rules are listed at the below statements:

- Customers should own one or more cars.
- Customers can reserve the service for their cars.
- Branches should provide at least one service.
- The service should have at least one checking task.
- The task should able to implement the one or more actions such as replace specific part to the car after the task had been perform.

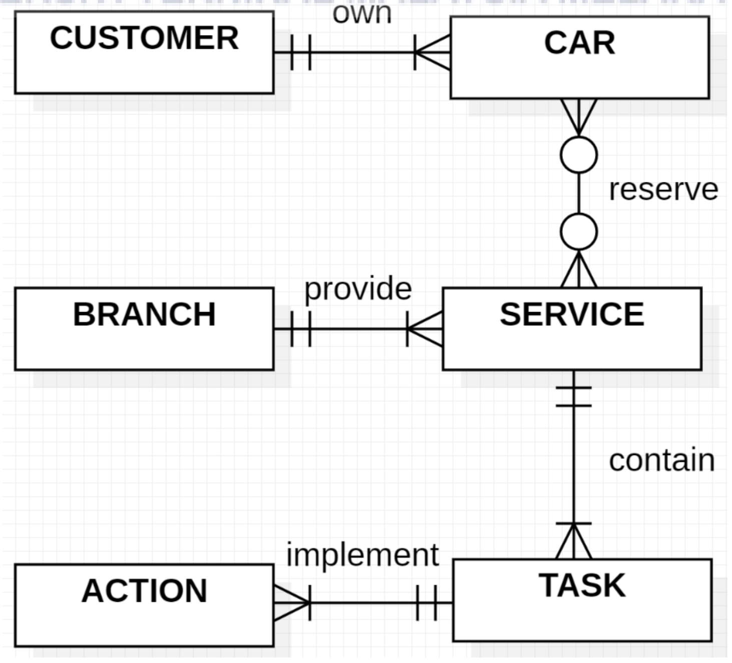


Figure 4.26: Conceptual Database Design of CaSS

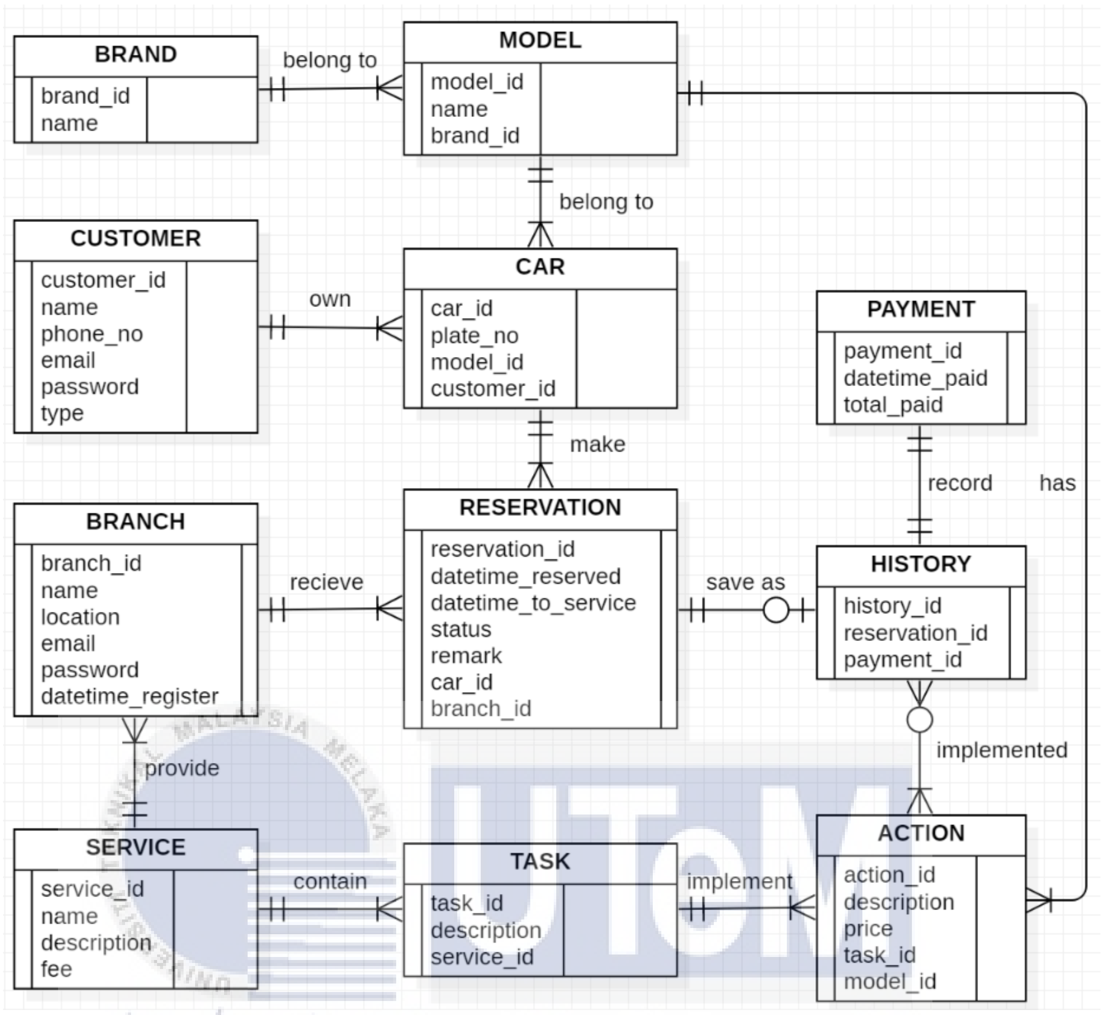


Figure 4.27: Logical Database Design of CaSS

4.3 Detail Design

This section discusses about the logic of the design and the approach to satisfying the requirement. The software design will describe about how will the system function in the classes of each method. The physical database design will describe about the final design of the database based on the DBMS of the system.

4.3.1 Software Design

4.3.1.1 Class LoginPage

Responsibility	Display the login page
Attributes	-

i. onLogin

Responsibility	Handle the login process
Input Parameter	email, password
Output Parameter	-
Pre-Condition	email and password are validated
Post-Condition	initialize customer static instance

ii. onRegister

Responsibility	Navigate to register page
Input Parameter	-
Output Parameter	-
Pre-Condition	-
Post-Condition	-

4.3.1.2 Class RegisterPage

Responsibility	Display the register page
Attributes	-

i. onLogin

Responsibility	Navigate to login page
Input Parameter	-
Output Parameter	-
Pre-Condition	-
Post-Condition	-

ii. onRegister

Responsibility	Handle the register process
Input Parameter	name, phone number, email, password
Output Parameter	-
Pre-Condition	input parameter had been validated
Post-Condition	-

4.3.1.3 Class CarPage

Responsibility	Display the car page
Attributes	list of cars

i. fetchCars

Responsibility	Get cars from API
Input Parameter	customer id
Output Parameter	list of cars
Pre-Condition	call before page loaded
Post-Condition	-

ii. onCarClick

Responsibility	Navigate to car detail page
Input Parameter	selected car
Output Parameter	-
Pre-Condition	-
Post-Condition	-

4.3.1.4 Class BookingPage

Responsibility	Display the booking page
Attributes	list of bookings

i. fetchBookings

Responsibility	Get bookings from API

Input Parameter	customer id
Output Parameter	list of bookings
Pre-Condition	call before page loaded
Post-Condition	-

ii. onBookingClick

Responsibility	Navigate to booking detail page
Input Parameter	selected booking
Output Parameter	-
Pre-Condition	-
Post-Condition	-

4.3.1.5 Class ServicePage

Responsibility	Display the service page
Attributes	servicing booking

i. fetchBooking

Responsibility	Get booking from API
Input Parameter	booking status
Output Parameter	booking
Pre-Condition	status is set to “Servicing”
Post-Condition	-

ii. onRefreshClick

Responsibility	Reload the service page
Input Parameter	-
Output Parameter	-
Pre-Condition	-
Post-Condition	page refreshed

4.3.1.6 Class ProfilePage

Responsibility	Display the profile page
Attributes	

iii. onUpdate

Responsibility	Update the field selected
Input Parameter	field, value
Output Parameter	-
Pre-Condition	-
Post-Condition	-

iv. onLogout

Responsibility	Navigate to login page
Input Parameter	-
Output Parameter	-
Pre-Condition	-
Post-Condition	Clear customer static instance

4.3.2 Physical Database Design

This section discusses about the physical database design which translated from logical to target DBMS (MySQL) – base tables. Figure 4.28 shows the physical database design of the system.

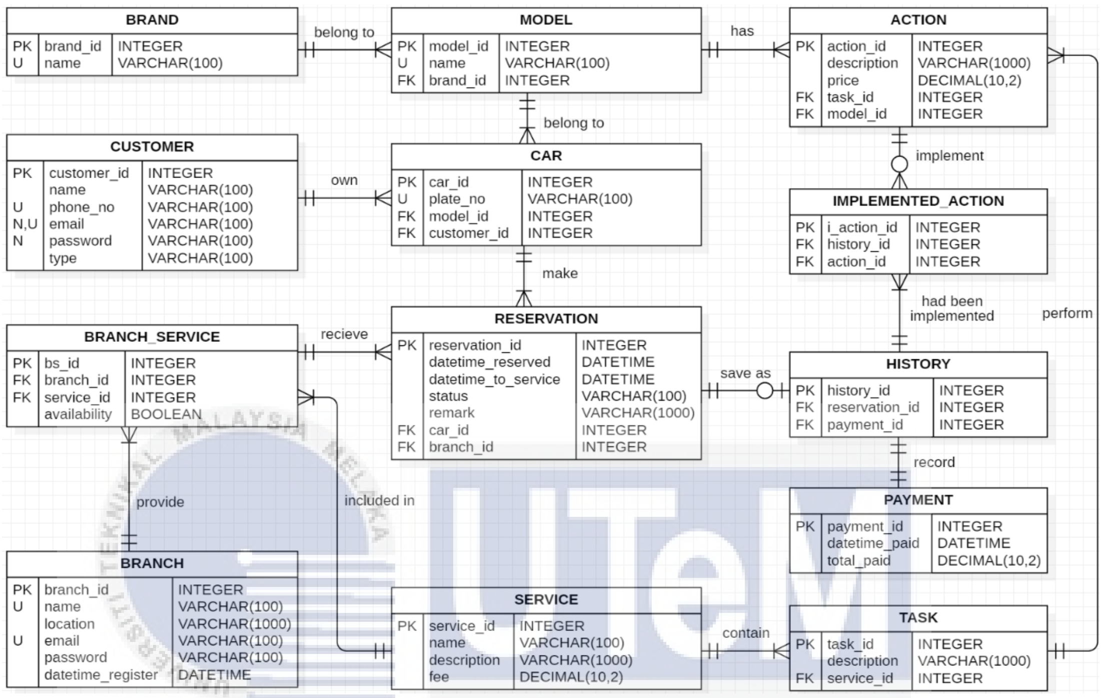


Figure 4.28: Physical Database Design of CaSS

4.4 Conclusion

The application will need to be designed as discussed in this chapter to ensure proper integration, as the system will interact with multiple environments which are a mobile application, a desktop application, a web service, and the localhost database. The user interfaces designed to be user-friendly.

CHAPTER 5: IMPLEMENTATION

5.1 Introduction

This chapter discusses about the implementation of this project. The implementation includes setup of the software development environment and setup of the software configuration environment. The version control of the project is importance and the procedure will be discussed also in this chapter.

5.2 Software Development Environment Setup

Figure 5.1 shows the deployment diagram of the system. The development environment is setup with four device nodes which are the mobile device, personal computer (PC), web server and database server. The CaSS customer application will be installed in the mobile device with the Android operating system. The CaSS branch application will be installed in the branch PC with the Windows operating system. These two device nodes are communicated with web server node by using JSON through the REST architecture. The web server is always listening to the port after executing app.js in the NodeJS environment. The web server act as a RESTful API web service which handle the request and response from the clients. The web server will retrieve the data from the database server by sequelization when received the request from clients and the raw data will be encapsulated in to JSON format with the response status then response to the clients. The implemented sample source codes of CaSS Branch, CaSS Customer and CaSS API are attached in Appendix A, Appendix B and Appendix C respectively.

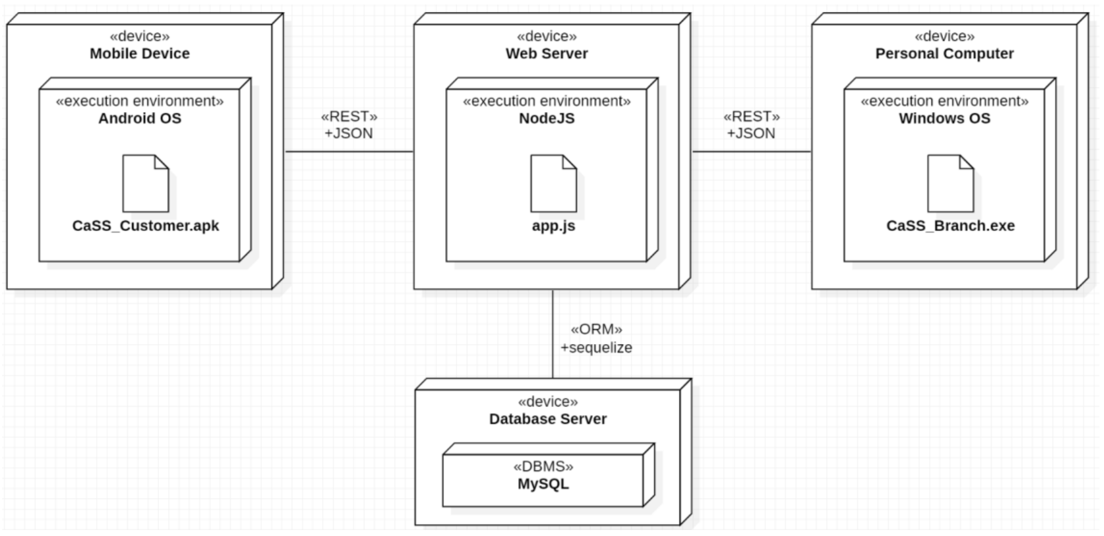


Figure 5.1: Deployment Diagram of CaSS

5.3 Software Configuration Management

This section discusses about the software configuration management which included the configuration environment setup and the version control procedure.

5.3.1 Configuration Environment Setup

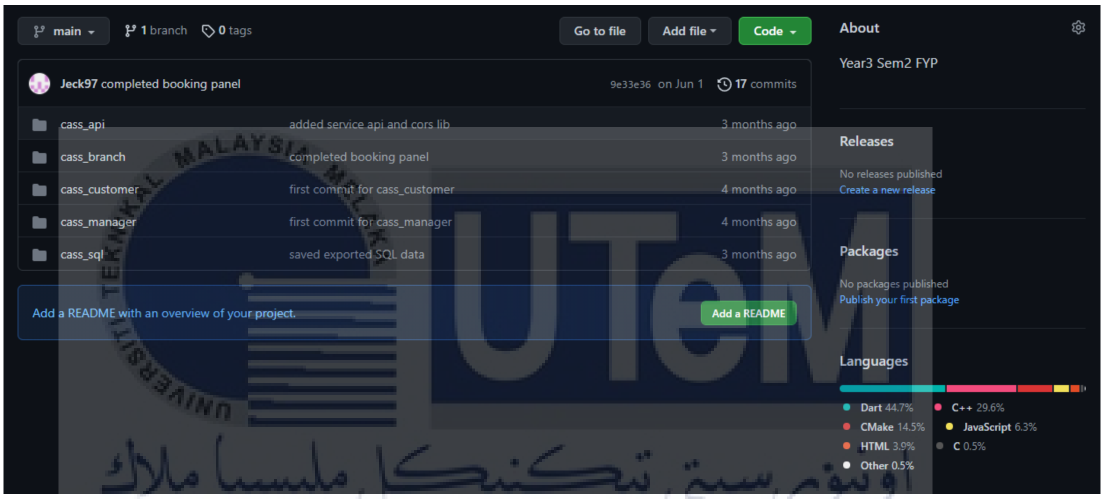
To setup the configuration environment, Visual Studio Code is chosen as the integrated development environment (IDE) to develop the artifacts that stated in deployment diagram in Figure 5.

Flutter is installed as a framework to develop the front-end of the system. As default, Flutter provided the environment to develop the mobile application so the CaSS Customer can directly start to development without adding the additional configuration. To enable the development of desktop application with Flutter, Flutter had to update as a beta version and add the configuration to enable the desktop development with specific platform.

Web server of the system is a RESTful API which developed by using Express. To use the Express, NodeJS is needed to be installed first so the package of Express can be installed through the npm command.

5.3.2 Version Control Procedure

Version control is an important process of tracking and managing different versions of the project. The source code of the project is managing remotely by using GitHub. Visual Studio Code provides the feature of source control by using Git and also GitHub. After setup the configuration environments, the first commit is committed on the main branch by Git locally and then push to the GitHub. Whenever a module is completed, it will be committed to the Git and GitHub with a short comment. Figure 5.2 shows the screenshot of the CaSS on GitHub with the committed comments and date.



Year3 Sem2 FYP

Jeck97 completed booking panel 9e33e36 on Jun 1 17 commits

cass_api	added service api and cors lib	3 months ago
cass_branch	completed booking panel	3 months ago
cass_customer	first commit for cass_customer	4 months ago
cass_manager	first commit for cass_manager	4 months ago
cass_sql	saved exported SQL data	3 months ago

Add a README with an overview of your project.

No releases published
Create a new release

No packages published
Publish your first package

Dart 44.7%	C++ 29.6%
CMake 14.5%	JavaScript 6.3%
HTML 3.9%	C 0.5%
Other 0.5%	

Figure 5.2: Main Page of CaSS on GitHub

5.4 Implementation Status

This section discusses about the progress of the development status for each module. Table 5.1 shows the progress of the development status of the CaSS Customer while Table 5.2 shows the progress of the development status of the CaSS Branch.

Table 5.1: Development Status of CaSS Customer

Module Name	Description	Duration to Complete

Authentication Module	A module which handles the authentication of the customer.	3 days
Car Module	A module which displays and manages the cars of the customer added.	5 days
Reservation Module	A module which enables the customer to view and find the car service centers and handles the reservation of car service such as make, modify, and cancel the reservation.	10 days
Service Module	A module which enables the customer to track and update the current servicing status.	14 days
Profile Module	A module which enables the customer to manage the profile of the customer.	3 days

Table 5.2: Development Status of CaSS Branch

Module Name	Description	Duration to Complete
Authentication Module	A module which handles the authentication of the staff of the branch.	3 days
Dashboard Module	A module which displays the dashboard with the summary and visualized information of the reservation.	5 days
Customer Module	A module which manages the customer by adding, modifying, and disabling customer.	5 days
Reservation Module	A module which enables the staff handles the reservation of car service such as make, modify, and cancel the reservation.	10 days
Service Module	A module which enables the staff to start the servicing and update the current servicing status.	14 days

5.5 Conclusion

The implementation phase is the building phase of the system, something which is crucial to be kept track of. This chapter presents the environment and the configuration required to build the system to ensure smooth development.

CHAPTER 6: TESTING

6.1 Introduction

Software Testing is the processes used to ensure that we can complete our project with minimized error and bugs in our functional and non-functional requirements. Testing method and SDLC that we had chosen are closely related as Software Testing is an integral part of any development methodology. This chapter discusses the testing process which included the test plan, test strategy, test design, and test results and analysis.

6.2 Test Plan

This section discusses about the test plan of the system. Test plan is a detailed document to perform testing for a software product. Test plan is important to help the client to understand the detail of testing. The test plan of this project is mainly focus on describing the test organization, test environment, and test schedule of the system.

6.2.1 Test Organization

Test organization is a procedure of defining roles in the testing process. It defines who is responsible for which activities in testing process. Table 6.1 shows the testing team structure of the project. In this project, I will play all the following roles and responsibilities throughout the whole testing process. However, some people are invited as a tester to execute the test cases of the project. Table 6.2 shows the information of testers in test organization.

Table 6.1: Testing Team Structure of CaSS

Role	Responsibility
Test Manager	Manage the testing process of whole project and define the testing direction of the project.
Tester	Create or execute the test cases and record the log results.
Developer in Test	Create a test program or automation scripts to test the code.
Test Administrator	Setup and ensure the test environment and support the team to use the test environment for test execution.
SQA Members	Check to confirm whether the testing process is meeting specified requirements.

Table 6.2: Test Organization of CaSS

Tester ID	Name	Age	Job
T001	Tiang King Jeck	24	Student
T002	Chon Yao Jun	23	Student
T003	Lee Jong Feng	23	Student
T004	Tan Zhi Zhong	23	Student
T005	Koh Kok Sheng	23	Student

6.2.2 Test Environment

A testing environment is a setup of software and hardware on which the testing team is going to execute test cases. The test environment consists of real business and user environment, as well as physical environments, such as server, front-end running environment.

The location of the testing to be carried out is one of the important testing environments. The testing location is depended on the type of the subsystems. Since CaSS Branch is the system used only at the front desk of the car service centre, the test location will only be set indoors. However, Cass Customer is the mobile application that allows customers to use the system from time to time, so the testing location will be set in anywhere, either indoor or outdoor, as long as able to access the internet.

In addition to the testing location, hardware, firmware configurations, preparations and training prior to testing also involved in testing environment setup. Table 6.3 shows the testing environment setup of the system.

Table 6.3: Testing Environment Setup of CaSS

Testing Environment	Description
Hardware	• Laptop or desktop with Windows operating system • Smart Phone with Android operating system • Router
Firmware Configuration	Not applicable.
Preparation	• Setup the internet connection to all devices • Start the database server and MySQL service • Run the web server with the CaSS API • Launch CaSS Branch and CaSS Customer
Training	Training of testing will not be conducted as the whole testing process will be handle by me.

6.2.3 Test Schedule

A test schedule includes the testing tasks or steps with the duration and responsibilities. The test schedule of this project will focus on the cycles and duration of the test to be conducted. Table 6.4 shows the test schedule of CaSS Branch and CaSS Customer with their modules. The test cycle is about the number of testing cycle needed in the testing process. The number of test cycle for each module is depended on significance and complexity of the module in the whole project.

Table 6.4: Test Schedule of CaSS

Subsystem	Module	Cycle	Duration
CaSS Customer	Authentication	1	5 minutes
	Car	3	15 minutes
	Reservation	3	15 minutes
	Service	5	30 minutes
	Profile	1	5 minutes
CaSS Branch	Authentication	1	5 minutes
	Dashboard	1	5 minutes
	Customer	3	15 minutes
	Reservation	3	15 minutes
	Service	5	30 minutes

6.3 Test Strategy

Test Strategy in software testing is defined as a set of guiding principles that determines the test design and regulates how the software testing process will be done. The objective of the test strategy is to provide a systematic approach to the software testing process in order to ensure the quality, traceability, reliability and better planning. (Hamilton, 2021) This project will use the White Box Testing, Black Box Testing and the Integration Testing with Bottom up approach to perform the testing process.

6.3.1 Classes of Tests

There are many types of testing can be executed which depended on the requirements of the system. In this project, unit testing, integration testing and system testing are emphasized.

Unit testing is a White Box Testing technique that is usually performed by the developer and it is done during the coding phase. Therefore, unit testing will be executed during the implementation phase by using local unit test features provided by Flutter with Visual Studio Code. By adding the dependencies from test package and flutter_test package to the Flutter test file, we can test all the functions and methods in each class without running on devices or emulators. For the unit testing that need the complex interactions with the framework, Mockito package are using to configure mock objects for return some specific value that invoked and needed in testing.

Integration testing is a testing that focuses to expose the defects and the data communication in the interaction between the integrated modules. This project will use the bottom-up approach on the integration testing where the lowest level modules will be tested first. Integration testing will be started after the unit testing had been done and the testing will be executed based on the modules of the systems. Since the project are developing the distributed systems which consist of two subsystems communicated by using the web service, hence the integration testing will be executed with two testing processes for CaSS Customer and CaSS Branch. The test cases are defined according functionality of each module.

System testing is a Black Box Testing technique that validates the completed software application as per the requirements. There are more than fifty types of system testing that could be perform during the testing process. Usability testing, also known as user experience (UX) testing is one of the common types of system testing to measures user-friendliness of a software application. This project will perform the usability testing in the system testing phase during the testing process.

6.4 Test Design

This section discusses the design of testing which included test description and test data. The test design is designed based on the test plan and test strategy that stated in the previous subtopics. Unit testing, integration testing and system testing will be involved in this test design.

6.4.1 Test Description

Test description describes the detail of the testing execution of the system. This project uses the test cases to perform the test execution. Test case identification, test scenario, test step and expected result for each module are designed and documented.

Table 6.5 shows the test cases of CaSS Customer while Table 6.6 shows the test cases of CaSS Branch.

Table 6.5: Test Cases of CaSS Customer

Test Case ID	Test Scenario	Test Steps	Expected Result
CC1001	Check customer register with valid data.	1. Open CaSS Customer App. 2. Click register account link. 3. Enter name, phone number, email, password and confirmed password. 4. Click create account button.	A dialog shows that registration is successful.
CC1002	Check customer register with invalid data.	1. Open CaSS Customer App. 2. Click register account link. 3. Left the fields empty. 4. Click create account button.	The fields show the error text to prompt user enter data.
CC1003	Check customer login with valid data.	1. Open CaSS Customer App. 2. Enter email and password. 3. Click login button.	The application directs from login page to car page.
CC1004	Check customer login with invalid data.	1. Open CaSS Customer App. 2. Left the fields empty. 3. Click login button.	The fields show the error text to prompt user enter data.
CC1005	Check customer forgot password with valid email.	1. Open CaSS Customer App. 2. Click forgot password link. 3. Enter email. 4. Click send button.	A dialog shows that reset password link had been send to the given email.
CC1006	Check customer forgot password with invalid email.	1. Open CaSS Customer App. 2. Click forgot password link. 3. left the field empty. 4. Click send button.	The field shows the error text to prompt user enter data.

CC2001	Check customer adds car with valid data.	1. Login CaSS Customer App. 2. Click “+” icon. 3. Enter car plate number. 4. Select car brand and model. 5. Click add button.	A dialog shows that car is added successfully.
CC2002	Check customer adds car with invalid data.	1. Login CaSS Customer App. 2. Click “+” icon. 3. Left the fields empty. 4. Click add button.	The fields show the error text to prompt user enter data.
CC2003	Check customer removes car.	1. Login CaSS Customer App. 2. Select a car from list. 3. Click remove button. 4. Confirm the remove.	A dialog shows that car is removed successfully.
CC3001	Check customer makes reservation with valid data.	1. Login CaSS Customer App. 2. Navigate to booking page. 3. Click “+” icon. 4. Select car, branch, service, date and time. 5. Click booking button.	A dialog shows that reservation is booked successfully.
CC3002	Check customer makes reservation with invalid data.	1. Login CaSS Customer App. 2. Navigate to booking page. 3. Click “+” icon. 4. Left the field empty. 5. Click booking button.	The fields show the error text to prompt user select data.
CC3003	Check customer cancels reservation.	1. Login CaSS Customer App. 2. Navigate to booking page. 3. Select a reservation from list. 4. Click cancel button. 5. Confirm the cancel.	A dialog shows that reservation is cancelled.
CC4001	Check servicing start on time.	1. Login CaSS Customer App. 2. Navigate to service page. 3. Wait until service start. 4. Click refresh icon.	A progress of servicing progress step is shown.

CC4002	Check customer selects actions after car had been checked.	1. Login CaSS Customer App. 2. Navigate to service page. 3. Click refresh after service is started. 4. Wait the car checking process finished. 5. Select the actions. 6. Click continue button.	The servicing progress step is proceeded to repair step.
CC5001	Check customer edits profile with valid data.	1. Login CaSS Customer App. 2. Navigate to profile page. 3. Click email or password. 4. Enter new email and new password. 5. Click save button.	A dialog shows that profile is edited and saved successfully.
CC5002	Check customer edits profile with invalid data.	1. Login CaSS Customer App. 2. Navigate to profile page. 3. Click email or password. 4. Left the fields empty. 5. Click save button.	The fields show the error text to prompt user enter data.
CC5003	Check customer logout.	1. Login CaSS Customer App. 2. Navigate to profile page. 3. Click logout button. 4. Confirm the logout.	The application directs to login page.

Table 6.6: Test Cases of CaSS Branch

Test Case ID	Test Scenario	Test Steps	Expected Result
CB1001	Check staff login with valid data.	1. Open CaSS Branch App. 2. Enter email and password. 3. Click login button.	The application directs to dashboard page.

CB1002	Check staff login with invalid data.	1. Open CaSS Branch App. 2. Left the fields empty. 3. Click login button.	The fields show the error text to prompt user enter data.
CB2001	Check staff select year to filter statistic.	1. Login CaSS Branch App. 2. Scroll to graph panel. 3. Select year from dropdown.	The graph is changed to the statistic of selected year.
CB3001	Check staff makes reservation with valid data.	1. Login CaSS Branch App. 2. Navigate to booking panel. 3. Click calendar icon. 4. Select customer, car, service, date and time. 5. Click add button.	A dialog shows that reservation is booked successfully.
CB3002	Check staff makes reservation with invalid data.	1. Login CaSS Branch App. 2. Navigate to booking panel. 3. Click calendar icon. 4. Left the field empty. 5. Click add button.	The fields show the error text to prompt user select data.
CB3003	Check staff cancels reservation.	1. Login CaSS Branch App. 2. Navigate to booking panel. 3. Select a reservation from calendar. 4. Click cancel button. 5. Confirm the cancel.	A dialog shows that reservation is cancelled.
CB3004	Check staff filters reservation by status.	1. Login CaSS Branch App. 2. Navigate to booking panel. 3. Click filter icon. 4. Select statuses. 5. Click filter button.	The calendar is refreshed and shown only the reservations with filtered status.
CB4001	Check staff starts service on timed.	1. Login CaSS Branch App. 2. Navigate to service panel.	The service is started and the

		3. Select a reservation from today services list. 4. Wait the time match on time. 5. Click start service button.	servicing progress is updated.
CB4002	Check staff selects actions during the car checking process.	1. Login CaSS Branch App. 2. Navigate to service panel. 3. Select a reservation from today services list. 4. Wait the time match on time. 5. Click start service button. 6. Select the tasks and click next button. 7. Repeat step 6 until progress 1 is done.	The servicing progress step is proceeded and updated.
CB4003	Check staff updates progress after done an action.	1. Login CaSS Branch App. 2. Navigate to service panel. 3. Select a reservation from today services list. 4. Wait the time match on time. 5. Click start service button. 6. Proceed to repairing step. 7. Click next after the repair action done. 8. Repeat step 7 until progress 3 is done.	The servicing progress step is proceeded and updated.
CB5001	Check staff adds a customer with valid data	1. Login CaSS Branch App. 2. Navigate to customer panel. 3. Click “+” icon. 4. Enter name, phone number and email. 5. Click add button.	A dialog shows that customer is added successfully.
CB5002	Check staff adds a customer	1. Login CaSS Branch App. 2. Navigate to customer panel.	The fields show the error text to

	with invalid data.	3. Click "+" icon. 4. Left the fields empty. 5. Click add button.	prompt user enter data.
CB5003	Check staff searches customer.	1. Login CaSS Branch App. 2. Navigate to customer panel. 3. Enter search key in search bar and click search.	The list of customers updated with search results.
CB5004	Check staff adds a car for the customer with valid data.	1. Login CaSS Branch App. 2. Navigate to customer panel. 3. Select a customer from list. 4. Click "+" icon. 5. Enter car plate number. 6. Select car brand and model. 7. Click add button.	A dialog shows that car is added successfully.
CB5005	Check staff adds a car for the customer with invalid data.	1. Login CaSS Branch App. 2. Navigate to customer panel. 3. Select a customer from list. 4. Click "+" icon. 5. Left the fields empty. 6. Click add button.	The fields show the error text to prompt user enter data.
CB5006	Check staff filter the reservations of customer by car.	1. Login CaSS Branch App. 2. Navigate to customer panel. 3. Select a customer from list. 4. Select a car from dropdown.	The list of reservations filtered with the car selected.
CB6001	Check staff logout.	1. Login CaSS Branch App. 2. Click logout button. 3. Confirm the logout.	The application directs to login page.

6.4.2 Test Data

The test data had been prepared for the tester to input during the execution of the test cases according to the test case ID. Table 6.7 shows the test data for the test cases of CaSS Customer while Table 6.8 shows the test data for the test cases of CaSS Branch.

Table 6.7: Test Data of CaSS Customer

Test Case ID	Test Field	Input Method	Test Data
CC1001	Name	Key in	Tiang King Jeck
	Email	Key in	jeck9797@gmail.com
	Phone Number	Key in	0138042421
	Password	Key in	P@ssw0rd
	Confirmed Password	Key in	P@ssw0rd
CC1002	Name	Key in	(Left empty)
	Email	Key in	(Left empty)
	Phone Number	Key in	(Left empty)
	Password	Key in	(Left empty)
	Confirmed Password	Key in	(Left empty)
CC1003	Email	Key in	jeck9797@gmail.com
	Password	Key in	P@ssw0rd
CC1004	Email	Key in	(Left empty)
	Password	Key in	(Left empty)
CC1005	Email	Key in	jeck9797@gmail.com
CC1006	Email	Key in	(Left empty)
CC2001	Plate Number	Key in	PKK1884
	Car Brand	Select	Perodua
	Car Model	Select	Myvi
CC2002	Plate Number	Key in	(Left empty)
	Car Brand	Select	(Not select)
	Car Model	Select	(Not select)
CC2003	(No field needed)	Click	(No data needed)
CC3001	Car	Select	PKK1884

	Branch	Select	CaSS Jelutong
	Service	Select	Oil Change Service
	Date	Select	22 August 2021
	Time	Select	14:00
CC3002	Car	Select	(Not select)
	Branch	Select	(Not select)
	Service	Select	(Not select)
	Date	Select	(Not select)
	Time	Select	(Not select)
CC3003	(No field needed)	Click	(No data needed)
CC4001	(No field needed)	Click	(No data needed)
CC4002	Actions	Select	Refill and replace oil
CC5001	Email	Key in	tiangkingjeck@gmail.com
	Password	Key in	n3w_P@ssword
CC5002	Email	Key in	(Left empty)
	Password	Key in	(Left empty)
CC5003	(No field needed)	Click	(No data needed)

Table 6.8: Test Cases of CaSS Branch

Test Case ID	Test Field	Input Method	Test Data
CB1001	Email	Key in	cassb001@mail.com
	Password	Key in	P@ssw0rd
CB1002	Email	Key in	(Left empty)
	Password	Key in	(Left empty)
CB2001	Year	Select	2020
CB3001	Customer	Select	Tiang King Jeck
	Car	Select	PKK1884
	Service	Select	Scheduled Maintenance Service
	Date	Select	23 August 2021
	Time	Select	16:00
CB3002	Customer	Select	(Not select)

	Car	Select	(Not select)
	Service	Select	(Not select)
	Date	Select	(Not select)
	Time	Select	(Not select)
CB3003	(No field needed)	Click	(No data needed)
CB3004	Status	Select	Cancelled
CB4001	(No field needed)	Click	(No data needed)
CB4002	Task 1	Select	Action 2, Action 3
	Task 2	Select	(Not Select)
	Task 3	Select	Action 1, Action 4
	Task 4	Select	Action 5
	Task 5	Select	(Not select)
	Task 6	Select	Action 1, Action 2, Action 3
CB4003	(No field needed)	Click	(No data needed)
CB5001	Name	Key in	Koh Kok Sheng
	Phone Number	Key in	012457812
	Email	Key in	koksheng@gmail.com
CB5002	Name	Key in	(Left empty)
	Phone Number	Key in	(Left empty)
	Email	Key in	(Left empty)
CB5003	Search key	Key in	sheng
CB5004	Plate Number	Key in	QAA2142
	Car Brand	Select	Toyota
	Car Model	Select	Vios
CB5005	Plate Number	Key in	(Left empty)
	Car Brand	Select	(Not select)
	Car Model	Select	(Not select)
CB5006	Car	Select	QAA2142
CB6001	(No field needed)	Click	(No data needed)

6.5 Test Results and Analysis

Test results and analysis describes the results of testing after executing the test cases defined in test description. The test is analyzed with test case identification, tester identification, test case results (Success/Fail) and detailed documentation on the failed test case. Table 6.9 shows the test results of CaSS Customer while Table 6.10 shows the test results of CaSS Branch.

Each of the test cases are tested according to their test cycle and the test results is simplified if the test case is succeeded. From the test results, the percentage of success for Test Cases of CaSS Customer is 89.47% and the percentage of success for Test Cases of CaSS Branch is 94.44%. The failures are due to unexpected error that not from the function of the system because all the functions in the system is passed after tested through Unit Testing.

Table 6.9: Test Cases of CaSS Customer

Test Case ID	Tester ID	Test Case Result	Reason of Failed
CC1001	T001	Fail	Web service is not started.
		Success	-
CC1002	T001	Success	-
CC1003	T001	Success	-
CC1004	T001	Success	-
CC1005	T001	Success	-
CC1006	T001	Success	-
CC2001	T002	Success	-
CC2002	T002	Success	-
CC2003	T002	Success	-
CC3001	T003	Success	-
CC3002	T003	Success	-
CC3003	T003	Success	-
CC4001	T004	Fail	Time zone of database not match with time zone of system.

		Success	-
CC4002	T004	Success	-
CC5001	T005	Success	-
CC5002	T005	Success	-
CC5003	T005	Success	-

Table 6.10: Test Cases of CaSS Branch

Test Case ID	Tester ID	Test Case Result	Reason of Failed
CB1001	T001	Success	-
CB1002	T001	Success	-
CB2001	T002	Success	-
CB3001	T003	Success	-
CB3002	T003	Success	-
CB3003	T003	Success	-
CB3004	T003	Success	-
CB4001	T004	Success	-
CB4002	T004	Success	-
CB4003	T004	Success	-
CB5001	T005	Fail	Customer phone number duplicated.
		Success	-
CB5002	T005	Success	-
CB5003	T005	Success	-
CB5004	T005	Success	-
CB5005	T005	Success	-
CB5006	T005	Success	-
CB6001	T001	Success	-

6.6 Conclusion

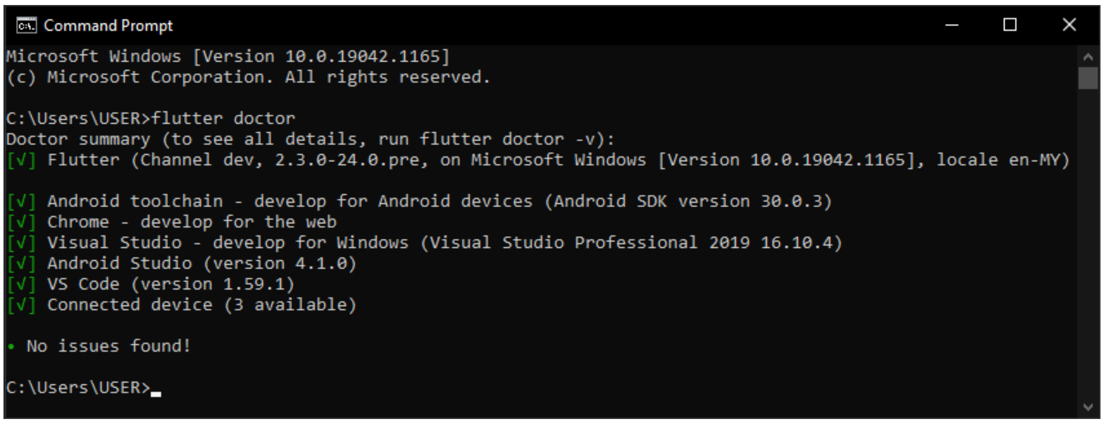
This chapter is designed to ensure that the system is performed according to the system specifications and that no error occurs. Device monitoring is then carried out to know if the system is operating exactly as expected. It is also essential to recognize system constraint to prepare for the challenges faced by this project.

CHAPTER 7: PROJECT CONCLUSION

7.1 Observation on Weaknesses and Strengths

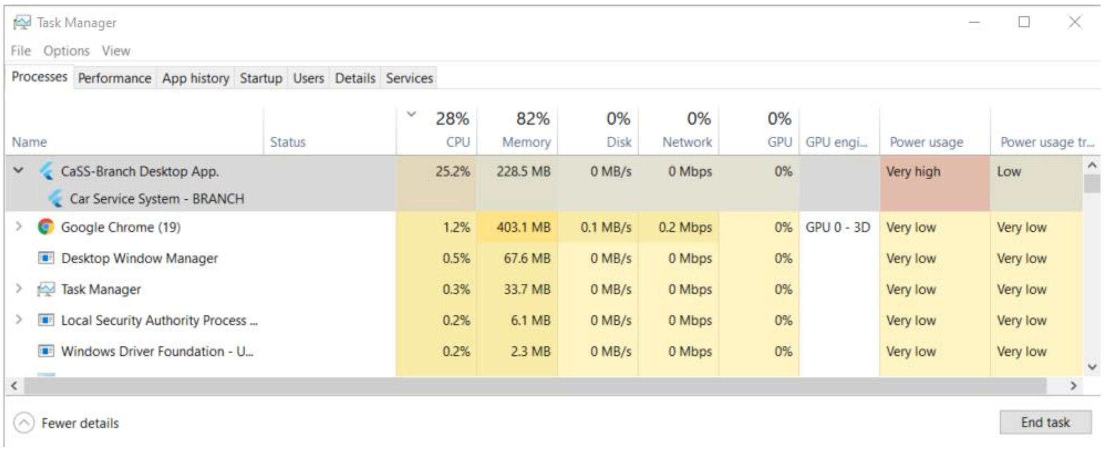
From the development of this project and testing, it can be seen that the project has multiple weaknesses it could improve on. However, it also has many strengths that sets it apart from other systems in its genre.

One of the weaknesses of the system is the restriction of platform. The CaSS Customer can only support on mobile smart phone with Android operating system while the CaSS Branch can only support on personal computer with Windows operating system. Figure 7.1 shows the platforms that can be installed by using Flutter which include Windows and Android platforms. Besides, the performance of CaSS Branch occupies too much resources from computer compared with normal desktop application, especially for CPU and power usage of the computer. Figure 7.2 shows the task manager of the computer that run the CaSS Branch application which occupied ¼ of the CPU usage.



```
Microsoft Windows [Version 10.0.19042.1165]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel dev, 2.3.0-24.0.pre, on Microsoft Windows [Version 10.0.19042.1165], locale en-MY)

[√] Android toolchain - develop for Android devices (Android SDK version 30.0.3)
[√] Chrome - develop for the web
[√] Visual Studio - develop for Windows (Visual Studio Professional 2019 16.10.4)
[√] Android Studio (version 4.1.0)
[√] VS Code (version 1.59.1)
[√] Connected device (3 available)

• No issues found!

C:\Users\USER>
```

Figure 7.1: Result of Flutter Doctor



Task Manager

File Options View

Processes | Performance | App history | Startup | Users | Details | Services

Name	Status	28% CPU	82% Memory	0% Disk	0% Network	0% GPU	GPU engi...	Power usage	Power usage tr...
CaSS-Branch Desktop App.		25.2%	228.5 MB	0 MB/s	0 Mbps	0%		Very high	Low
Car Service System - BRANCH									
Google Chrome (19)		1.2%	403.1 MB	0.1 MB/s	0.2 Mbps	0%	GPU 0 - 3D	Very low	Very low
Desktop Window Manager		0.5%	67.6 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Task Manager		0.3%	33.7 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Local Security Authority Process ...		0.2%	6.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Driver Foundation - U...		0.2%	2.3 MB	0 MB/s	0 Mbps	0%		Very low	Very low

Fewer details

End task

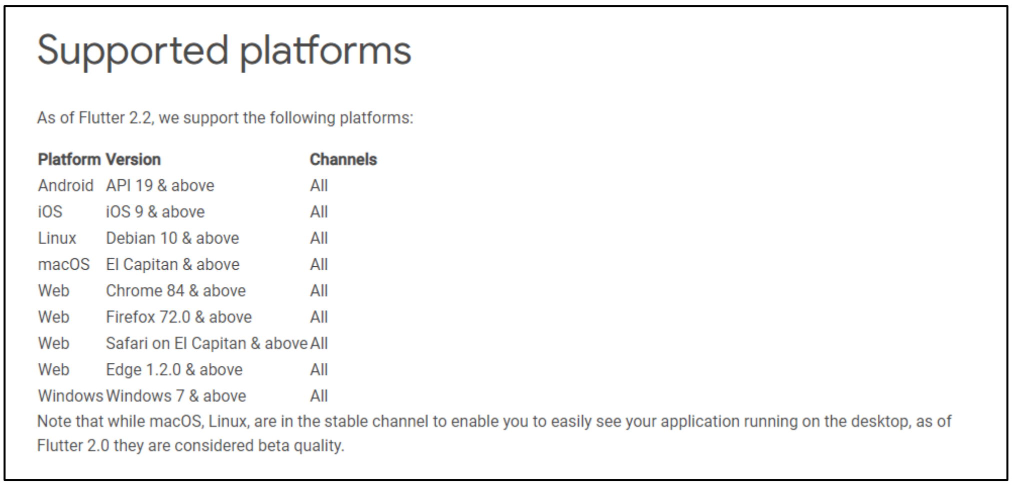
Figure 7.2: Performance of CaSS Branch

The strength of the system is it provides a unique real-time tracking feature for the servicing process. The real-time tracking process is handling by the web service that designed for this system which ensure the tracking process is secure and efficient.

7.2 Propositions for Improvement

From the weaknesses of the system, we can see that there are many propositions and improvement for this system.

The system can be developed to support more platforms. For example, the CaSS Customer can be developed in IOS platform while the CaSS Branch can be developed in MacOS and Linux platforms. Fortunately, Flutter supports many platforms with one codebase development. Figure 7.3 shows the platforms that supported by Flutter.



Supported platforms

As of Flutter 2.2, we support the following platforms:

Platform	Version	Channels
Android	API 19 & above	All
iOS	iOS 9 & above	All
Linux	Debian 10 & above	All
macOS	El Capitan & above	All
Web	Chrome 84 & above	All
Web	Firefox 72.0 & above	All
Web	Safari on El Capitan & above	All
Web	Edge 1.2.0 & above	All
Windows	Windows 7 & above	All

Note that while macOS, Linux, are in the stable channel to enable you to easily see your application running on the desktop, as of Flutter 2.0 they are considered beta quality.

Figure 7.3: Supported Platforms by Flutter

Besides, the web service can be hosted to online server. The current web service is running in localhost with the server which only allow the client applications request to the API while accessed to the same Local Area Network (LAN) with the server. Therefore, the web service or CaSS API should be hosted to online server to enable the client application to access the web service in anywhere and anytime as long as connected to the internet. Same to the database, the current local database should also be stored in online. For example, databases can be stored in cloud services which allow provide enough space for the users to store large amounts of data.

7.3 Project Contribution

The project mainly contributed to the automotive industry and concerned more about the communication of data in real-time.

This project promotes the digitalization of the automotive industry. All of the data of the car owners are stored into the database instead of recording on the papers or sheets. The data are encapsulated to standard information and encrypted before stored into the database. Besides, most of the activities also can be done without conversing to each other such as make a reservation of service and check the status of car which is servicing.

This project also promotes the technology of ubiquitous computing in Malaysia. CaSS focused on the feature of tracking the process of servicing a car by exchange the data in real-time. The percentage of the process completed is computed and updated through the web service from time to time and simultaneously to all users. Therefore, the users can now their servicing status in anytime and everywhere.

7.4 Conclusion

Car Service System really can be helpful and useful application to customer and staff of car service center. Even though this system still has flaws and weakness, I definitely keep hoping that the weakness can be resolved soon. I also hoping that all the users will be happy and satisfied in using this system.

REFERENCES

Conway, L. (2021, June 1). *Blockchain Explained*. Retrieved from Investopedia: https://www.investopedia.com/terms/b/blockchain.asp

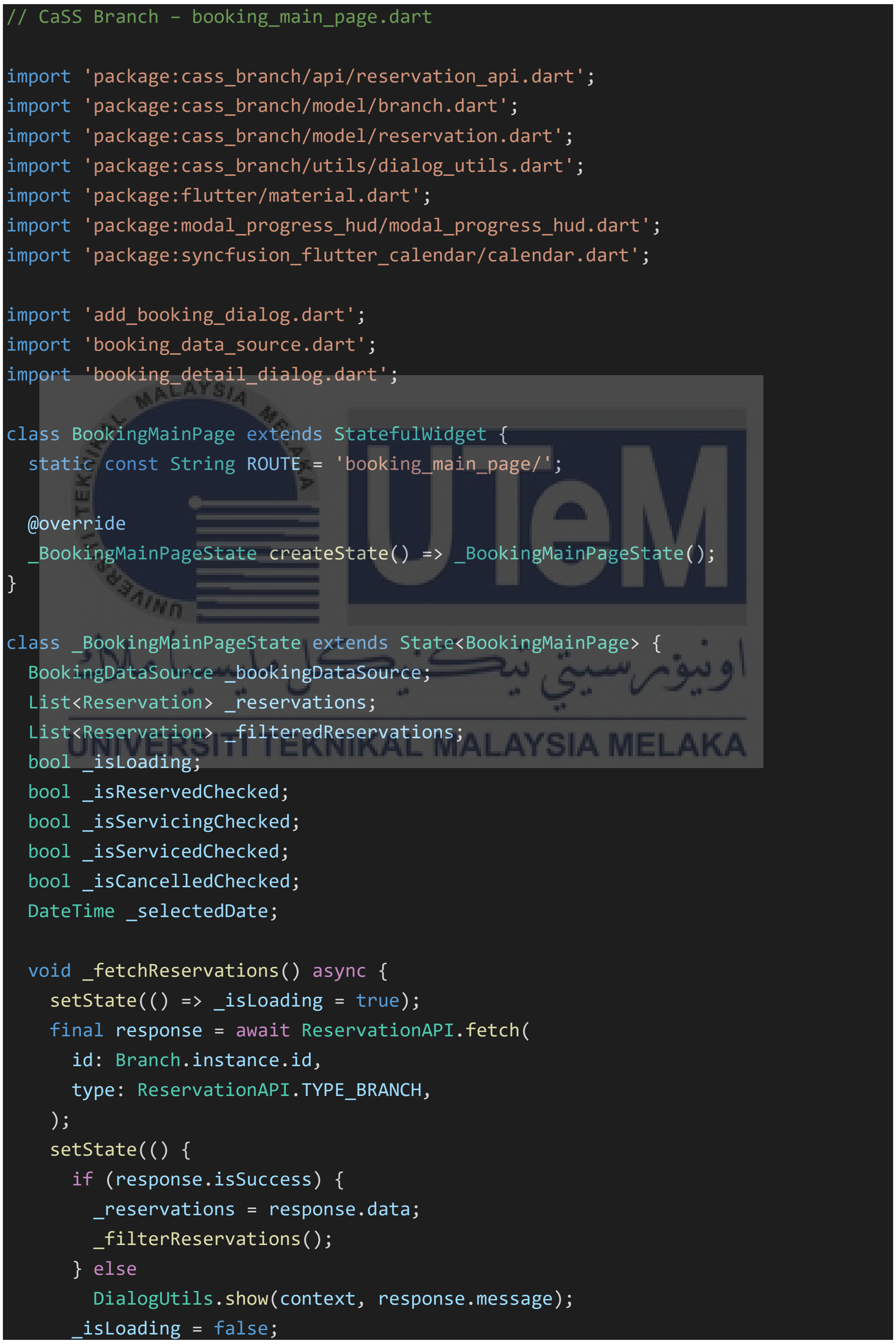
Data Modeling: Conceptual vs Logical vs Physical Data Model. (n.d.). Retrieved from Visual Paradigm: https://online.visual-paradigm.com/knowledge/visual-modeling/conceptual-vs-logical-vs-physical-data-model/

Hamilton, T. (2021, August 30). *TEST PLAN: What is, How to Create (with Example)*. Retrieved from Guru99: https://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html

Nanda, V. (2021, March 8). *Object-oriented Life Cycle Model in Software Engineering*. Retrieved from tutorialspoint: https://www.tutorialspoint.com/object-oriented-life-cycle-model-in-software-engineering

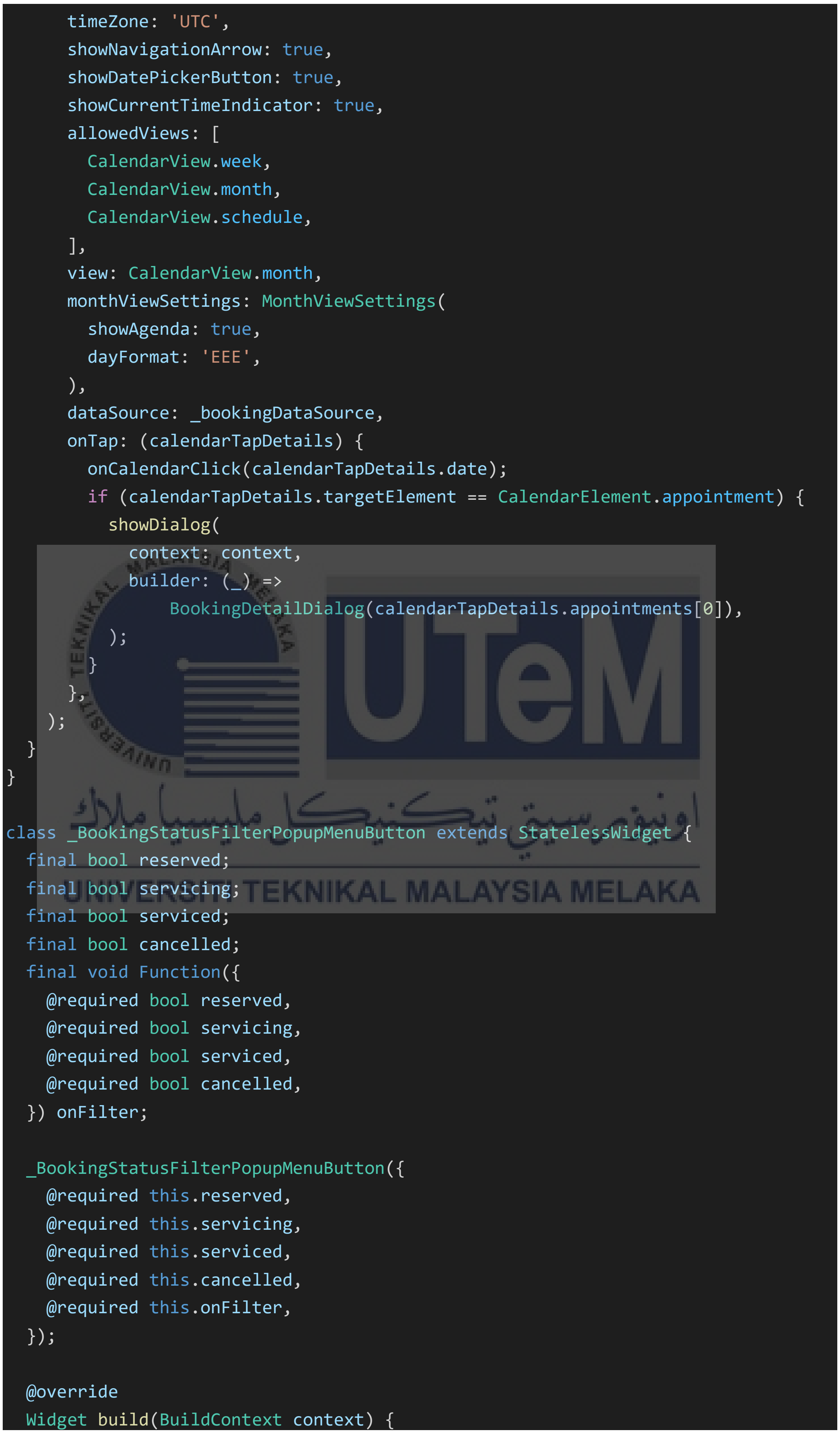
APPENDICES

Appendix A – Sample Source Code 1

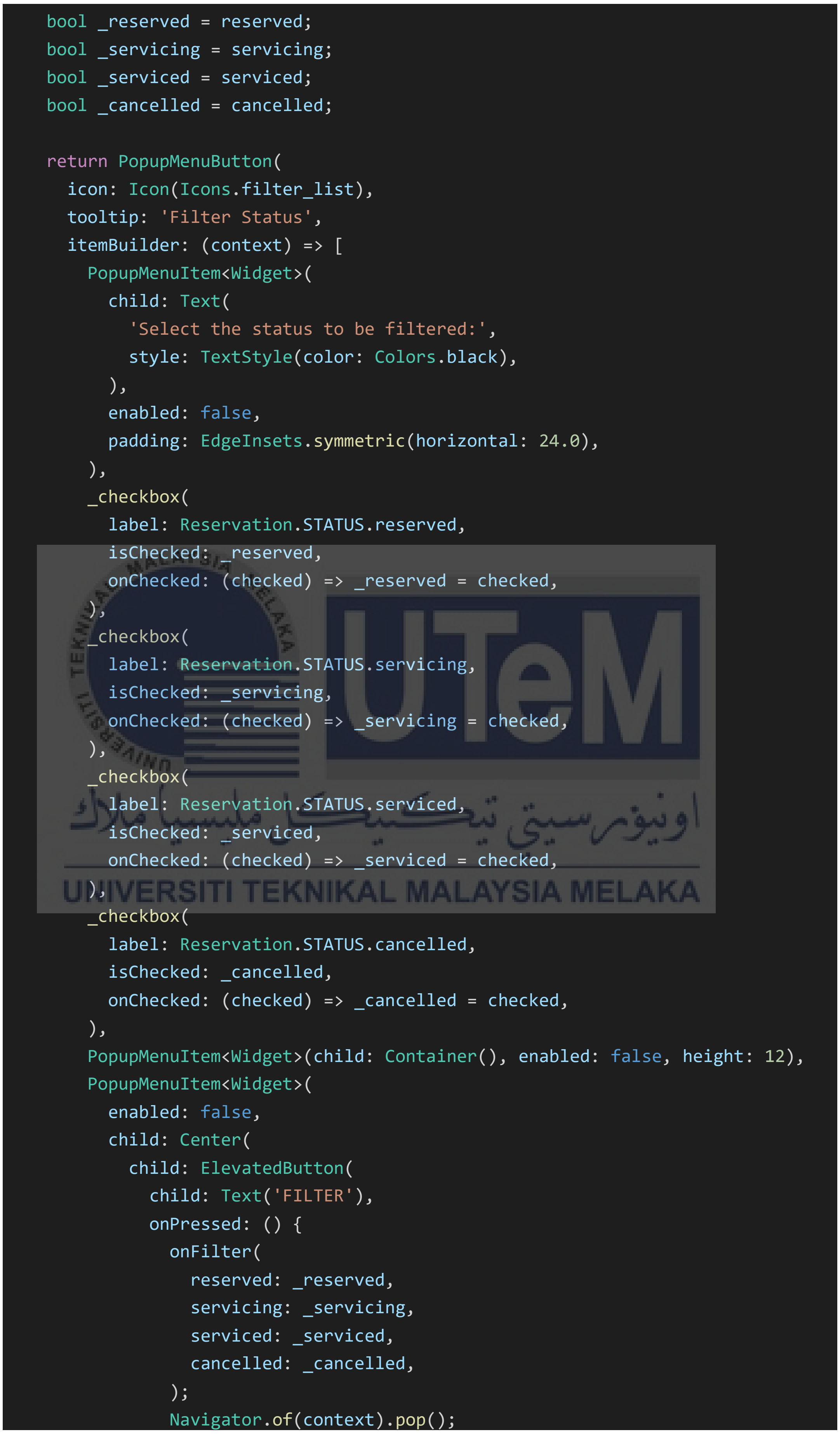


```dart
// CaSS Branch - booking_main_page.dart

import 'package:cass_branch/api/reservation_api.dart';
import 'package:cass_branch/model/branch.dart';
import 'package:cass_branch/model/reservation.dart';
import 'package:cass_branch/utils/dialog_utils.dart';
import 'package:flutter/material.dart';
import 'package:modal_progress_hud/modal_progress_hud.dart';
import 'package:syncfusion_flutter_calendar/calendar.dart';

import 'add_booking_dialog.dart';
import 'booking_data_source.dart';
import 'booking_detail_dialog.dart';

class BookingMainPage extends StatefulWidget {
  static const String ROUTE = 'booking_main_page/';

  @override
  _BookingMainPageState createState() => _BookingMainPageState();
}

class _BookingMainPageState extends State<BookingMainPage> {
  BookingDataSource _bookingDataSource;
  List<Reservation> _reservations;
  List<Reservation> _filteredReservations;
  bool _isLoading;
  bool _isReservedChecked;
  bool _isServicingChecked;
  bool _isServicedChecked;
  bool _isCancelledChecked;
  DateTime _selectedDate;

  void _fetchReservations() async {
    setState(() => _isLoading = true);
    final response = await ReservationAPI.fetch(
      id: Branch.instance.id,
      type: ReservationAPI.TYPE_BRANCH,
    );
    setState(() {
      if (response.isSuccess) {
        _reservations = response.data;
        _filterReservations();
      } else
        DialogUtils.show(context, response.message);
      _isLoading = false;
    });
  }

  void _filterReservations() {
    _filteredReservations = (_isReservedChecked &&
            _isServicingChecked &&
            _isServicedChecked &&
            _isCancelledChecked)
        ? _reservations
        : _reservations
            .where((r) =>
                (r.status == Reservation.STATUS.reserved &&
                    _isReservedChecked) ||
                (r.status ==
                        Reservation.STATUS.servicing &&
                    _isServicingChecked) ||
                (r.status == Reservation.STATUS.serviced &&
                    _isServicedChecked) ||
                (r.status == Reservation.STATUS.cancelled &&
                    _isCancelledChecked))
            .toList();
    _bookingDataSource = BookingDataSource(_filteredReservations);
  }

  void _onFilter({
    @required bool reserved,
    @required bool servicing,
    @required bool serviced,
    @required bool cancelled,
  }) {
    setState(() {
      _isReservedChecked = reserved;
      _isServicingChecked = servicing;
      _isServicedChecked = serviced;
      _isCancelledChecked = cancelled;
      _filterReservations();
    });
  }

  @override
  void initState() {
    _reservations = [];
    _isLoading = false;
    _isReservedChecked = true;
    _isServicingChecked = true;
    _isServicedChecked = true;
    _isCancelledChecked = true;
    _fetchReservations();
    super.initState();
  }
```

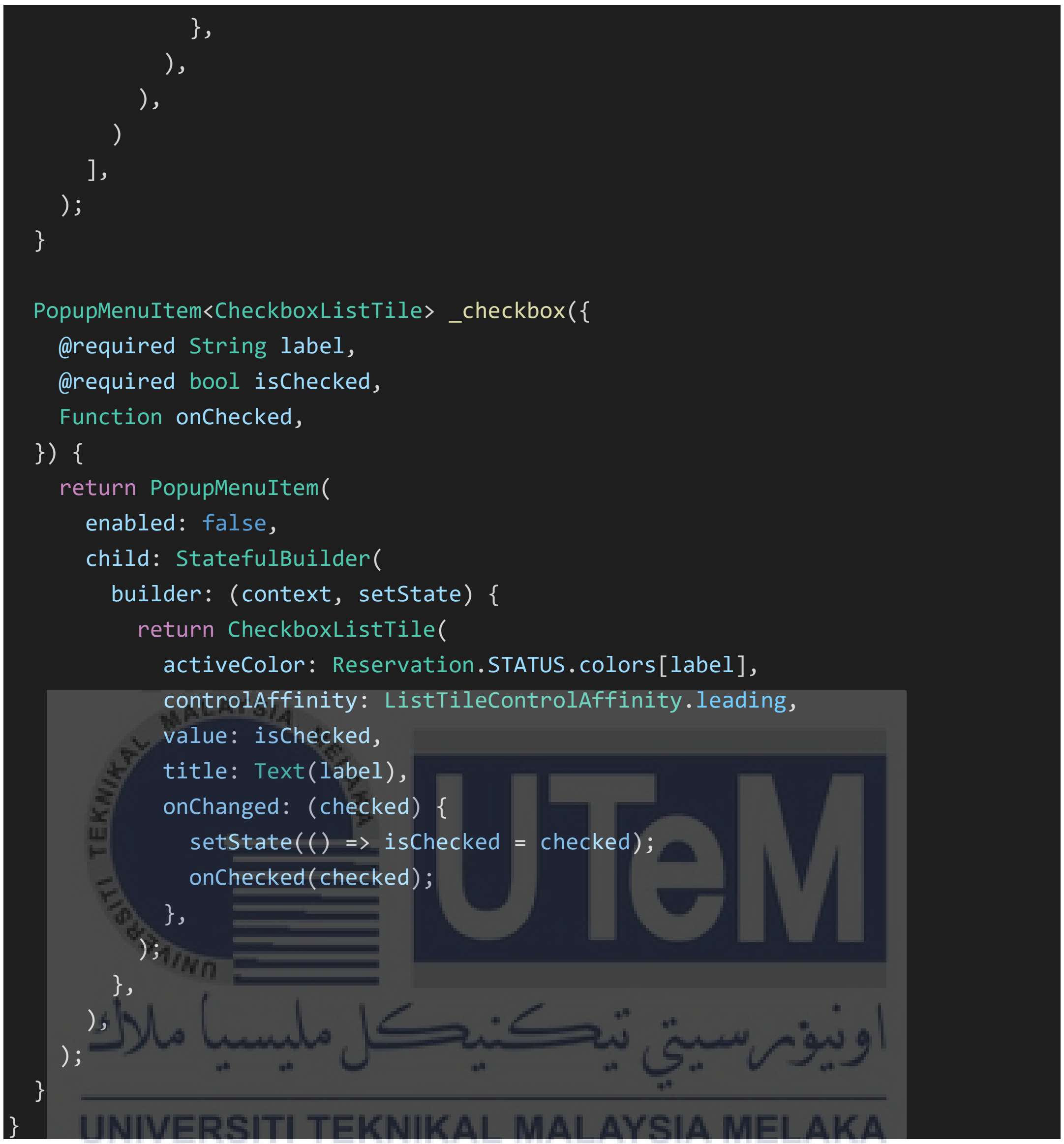
```
  @override
  Widget build(BuildContext context) {
    return ModalProgressHUD(
      inAsyncCall: _isLoading,
      child: Scaffold(
        appBar: AppBar(
          title: Text(
            'Booking',
            style: Theme.of(context)
                .textTheme
                .headline5
                .copyWith(color: Colors.white),
          ),
          actions: [
            IconButton(
              icon: Icon(Icons.today_outlined),
              tooltip: 'Make booking',
              onPressed: () => showDialog(
                context: context,
                barrierDismissible: false,
                builder: (_) => AddBookingDialog(date: _selectedDate),
              ),
            ),
            _BookingStatusFilterPopupMenuButton(
              reserved: _isReservedChecked,
              servicing: _isServicingChecked,
              serviced: _isServicedChecked,
              cancelled: _isCancelledChecked,
              onFilter: _onFilter,
            ),
            IconButton(
              icon: Icon(Icons.refresh),
              tooltip: 'Refresh',
              onPressed: _fetchReservations,
            ),
          ],
        ),
        body: _BookingCalendar(_bookingDataSource, (d) => _selectedDate = d),
      ),
    );
  }
}

class _BookingCalendar extends StatelessWidget {
  final BookingDataSource _bookingDataSource;
  final void Function(DateTime) onCalendarClick;
  _BookingCalendar(this._bookingDataSource, this.onCalendarClick);

  @override
  Widget build(BuildContext context) {
    return SfCalendar(
```



```
      timeZone: 'UTC',
      showNavigationArrow: true,
      showDatePickerButton: true,
      showCurrentTimeIndicator: true,
      allowedViews: [
        CalendarView.week,
        CalendarView.month,
        CalendarView.schedule,
      ],
      view: CalendarView.month,
      monthViewSettings: MonthViewSettings(
        showAgenda: true,
        dayFormat: 'EEE',
      ),
      dataSource: _bookingDataSource,
      onTap: (calendarTapDetails) {
        onCalendarClick(calendarTapDetails.date);
        if (calendarTapDetails.targetElement == CalendarElement.appointment) {
          showDialog(
            context: context,
            builder: (_) =>
                BookingDetailDialog(calendarTapDetails.appointments[0]),
          );
        }
      },
    );
  }
}

class _BookingStatusFilterPopupMenuButton extends StatelessWidget {
  final bool reserved;
  final bool servicing;
  final bool serviced;
  final bool cancelled;
  final void Function({
    @required bool reserved,
    @required bool servicing,
    @required bool serviced,
    @required bool cancelled,
  }) onFilter;

  _BookingStatusFilterPopupMenuButton({
    @required this.reserved,
    @required this.servicing,
    @required this.serviced,
    @required this.cancelled,
    @required this.onFilter,
  });

  @override
  Widget build(BuildContext context) {
```



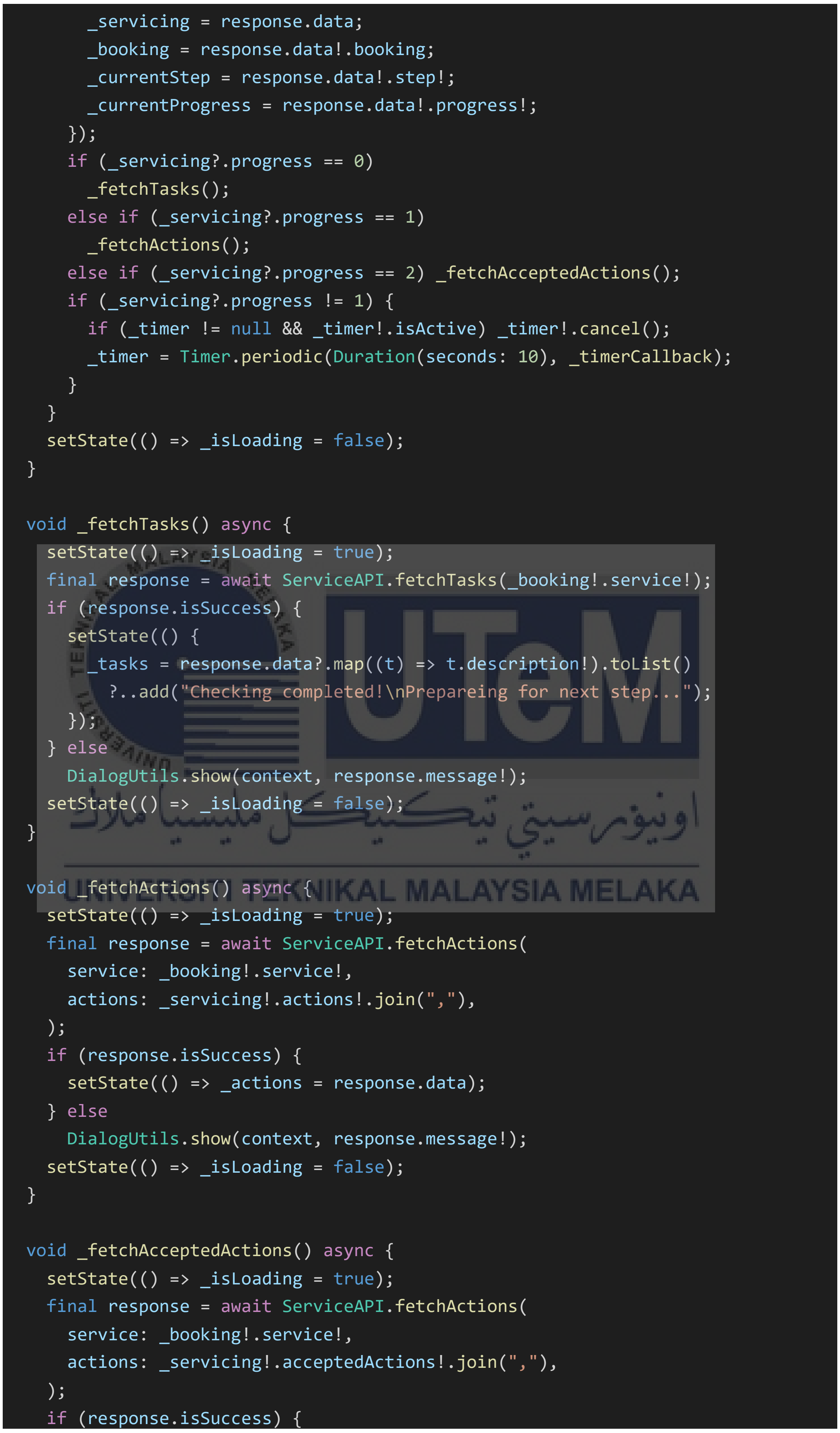
```dart
    bool _reserved = reserved;
    bool _servicing = servicing;
    bool _serviced = serviced;
    bool _cancelled = cancelled;

    return PopupMenuButton(
      icon: Icon(Icons.filter_list),
      tooltip: 'Filter Status',
      itemBuilder: (context) => [
        PopupMenuItem<Widget>(
          child: Text(
            'Select the status to be filtered:',
            style: TextStyle(color: Colors.black),
          ),
          enabled: false,
          padding: EdgeInsets.symmetric(horizontal: 24.0),
        ),
        _checkbox(
          label: Reservation.STATUS.reserved,
          isChecked: _reserved,
          onChecked: (checked) => _reserved = checked,
        ),
        _checkbox(
          label: Reservation.STATUS.servicing,
          isChecked: _servicing,
          onChecked: (checked) => _servicing = checked,
        ),
        _checkbox(
          label: Reservation.STATUS.serviced,
          isChecked: _serviced,
          onChecked: (checked) => _serviced = checked,
        ),
        _checkbox(
          label: Reservation.STATUS.cancelled,
          isChecked: _cancelled,
          onChecked: (checked) => _cancelled = checked,
        ),
        PopupMenuItem<Widget>(child: Container(), enabled: false, height: 12),
        PopupMenuItem<Widget>(
          enabled: false,
          child: Center(
            child: ElevatedButton(
              child: Text('FILTER'),
              onPressed: () {
                onFilter(
                  reserved: _reserved,
                  servicing: _servicing,
                  serviced: _serviced,
                  cancelled: _cancelled,
                );
                Navigator.of(context).pop();
```



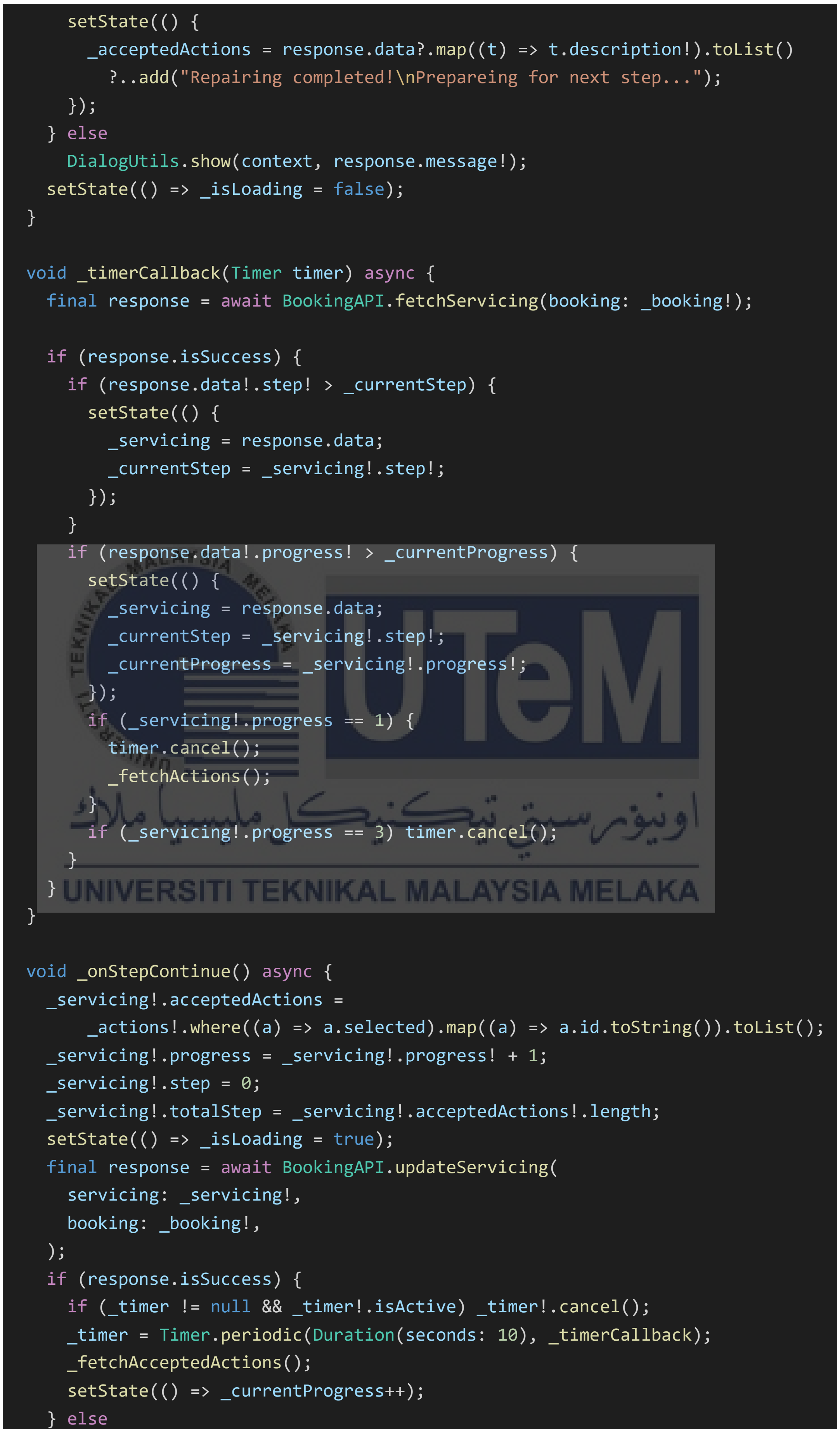
```
              },
            ),
          ),
        )
      ],
    );
  }

  PopupMenuItem<CheckboxListTile> _checkbox({
    @required String label,
    @required bool isChecked,
    Function onChecked,
  }) {
    return PopupMenuItem(
      enabled: false,
      child: StatefulBuilder(
        builder: (context, setState) {
          return CheckboxListTile(
            activeColor: Reservation.STATUS.colors[label],
            controlAffinity: ListTileControlAffinity.leading,
            value: isChecked,
            title: Text(label),
            onChanged: (checked) {
              setState(() => isChecked = checked);
              onChecked(checked);
            },
          );
        },
      ),
    );
  }
}
```

Appendix B – Sample Source Code 2

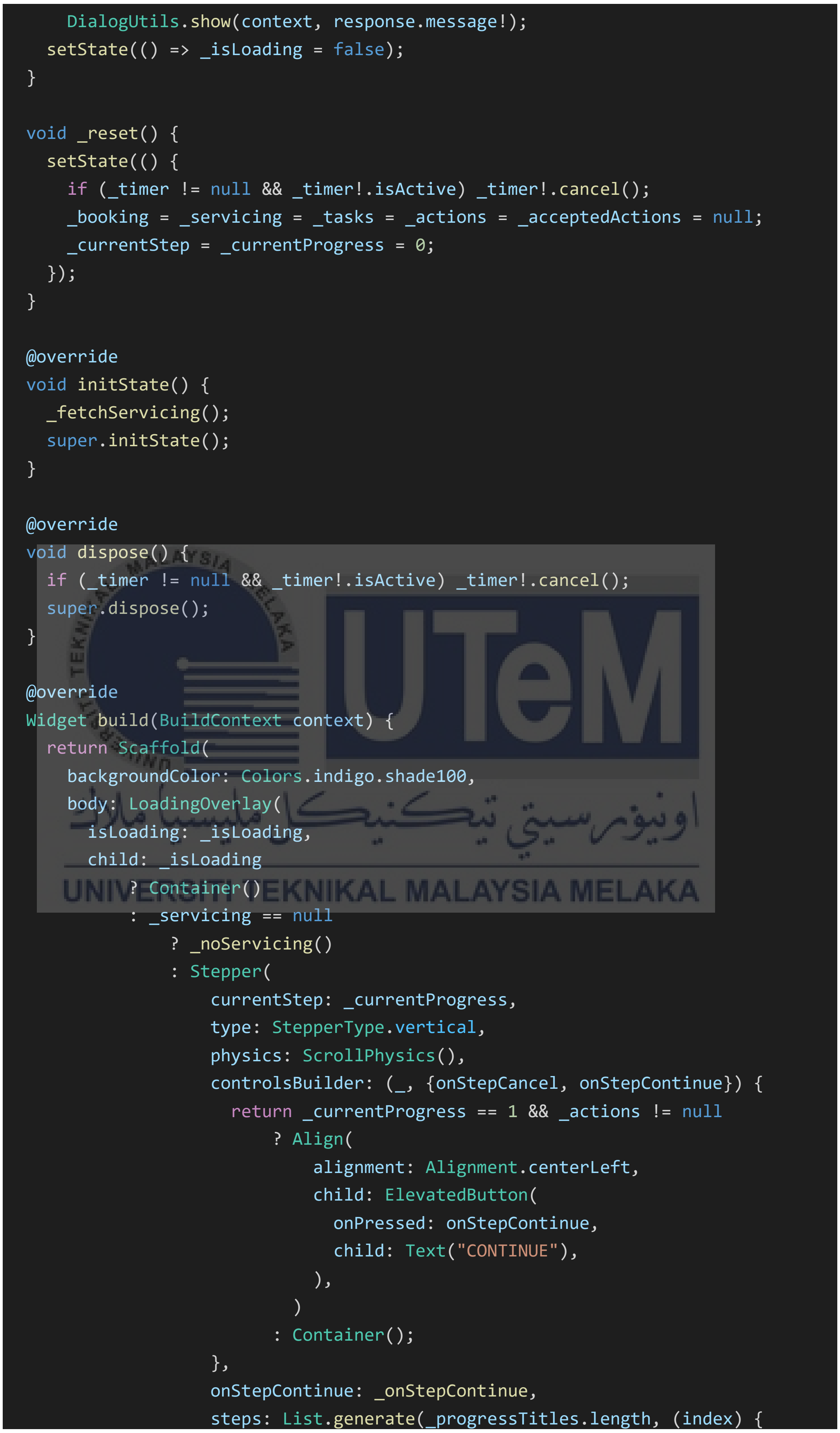
```dart
// CaSS Customer - service_page.dart

import 'dart:async';
import 'dart:ui';
import 'package:cass_customer/api/booking_api.dart';
import 'package:cass_customer/api/service_api.dart';
import 'package:cass_customer/model/action.dart' as a;
import 'package:cass_customer/model/booking.dart';
import 'package:cass_customer/model/customer.dart';
import 'package:cass_customer/model/servicing.dart';
import 'package:cass_customer/utils/dialog_utils.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/gestures.dart';
import 'package:flutter/material.dart';
import 'package:flutter/painting.dart';
import 'package:flutter/rendering.dart';
import 'package:flutter/widgets.dart';
import 'package:loading_overlay/loading_overlay.dart';

class ServicePage extends StatefulWidget {
  @override
  _ServicePageState createState() => _ServicePageState();
}

class _ServicePageState extends State<ServicePage> {
  Booking? _booking;
  Servicing? _servicing;
  List<String>? _tasks;
  List<a.Action>? _actions;
  List<String>? _acceptedActions;
  Timer? _timer;
  bool _isLoading = false;
  int _currentStep = 0;
  int _currentProgress = 0;

  final _progressTitles = [
    "CHECKING CAR",
    "SELECT ACTIONS TO APPLY",
    "REPARING",
    "COMPLETED",
  ];

  void _fetchServicing() async {
    setState(() => _isLoading = true);
    _reset();
    final response =
        await BookingAPI.fetchServicing(customer: Customer.instance!);
    if (response.isSuccess) {
      setState(() {
```



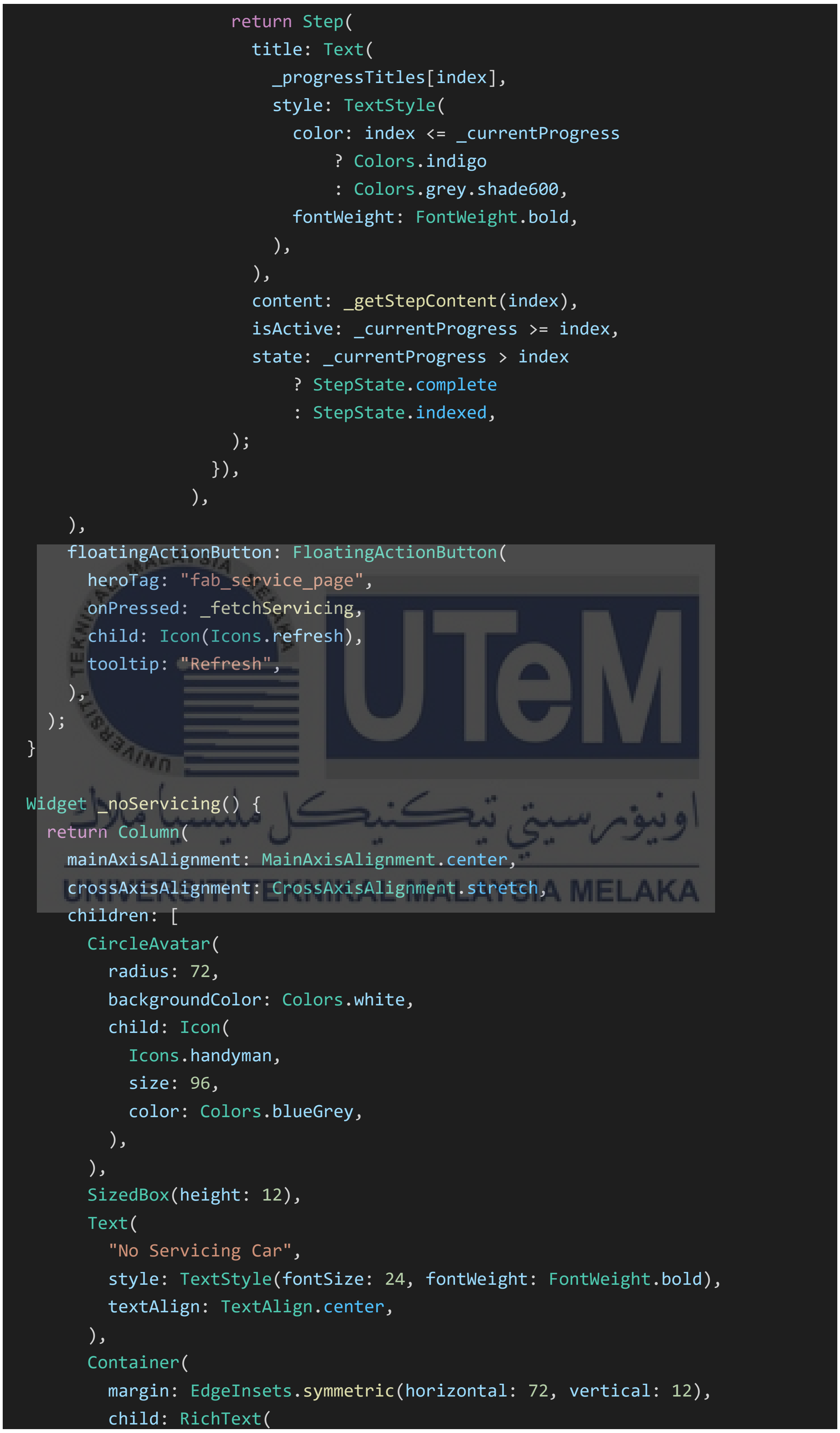
```
      _servicing = response.data;
      _booking = response.data!.booking;
      _currentStep = response.data!.step!;
      _currentProgress = response.data!.progress!;
    });
    if (_servicing?.progress == 0)
      _fetchTasks();
    else if (_servicing?.progress == 1)
      _fetchActions();
    else if (_servicing?.progress == 2) _fetchAcceptedActions();
    if (_servicing?.progress != 1) {
      if (_timer != null && _timer!.isActive) _timer!.cancel();
      _timer = Timer.periodic(Duration(seconds: 10), _timerCallback);
    }
  }
  setState(() => _isLoading = false);
}

void _fetchTasks() async {
  setState(() => _isLoading = true);
  final response = await ServiceAPI.fetchTasks(_booking!.service!);
  if (response.isSuccess) {
    setState(() {
      _tasks = response.data?.map((t) => t.description!).toList()
        ?..add("Checking completed!\nPrepareing for next step...");
    });
  } else
    DialogUtils.show(context, response.message!);
  setState(() => _isLoading = false);
}

void _fetchActions() async {
  setState(() => _isLoading = true);
  final response = await ServiceAPI.fetchActions(
    service: _booking!.service!,
    actions: _servicing!.actions!.join(","),
  );
  if (response.isSuccess) {
    setState(() => _actions = response.data);
  } else
    DialogUtils.show(context, response.message!);
  setState(() => _isLoading = false);
}

void _fetchAcceptedActions() async {
  setState(() => _isLoading = true);
  final response = await ServiceAPI.fetchActions(
    service: _booking!.service!,
    actions: _servicing!.acceptedActions!.join(","),
  );
  if (response.isSuccess) {
```



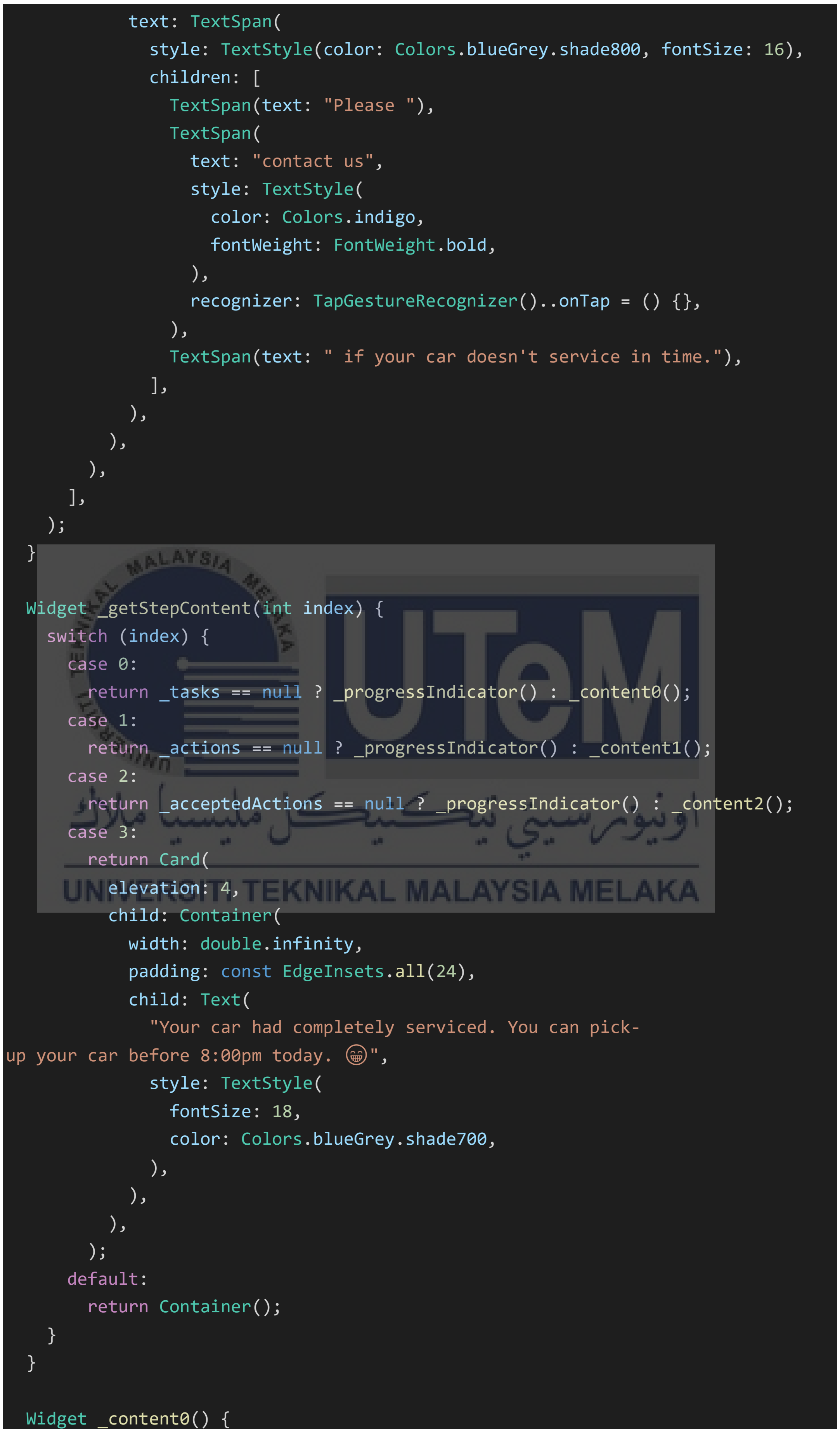
```
    setState(() {
      _acceptedActions = response.data?.map((t) => t.description!).toList()
        ?..add("Repairing completed!\nPrepareing for next step...");
    });
  } else
    DialogUtils.show(context, response.message!);
  setState(() => _isLoading = false);
}

void _timerCallback(Timer timer) async {
  final response = await BookingAPI.fetchServicing(booking: _booking!);

  if (response.isSuccess) {
    if (response.data!.step! > _currentStep) {
      setState(() {
        _servicing = response.data;
        _currentStep = _servicing!.step!;
      });
    }
    if (response.data!.progress! > _currentProgress) {
      setState(() {
        _servicing = response.data;
        _currentStep = _servicing!.step!;
        _currentProgress = _servicing!.progress!;
      });
      if (_servicing!.progress == 1) {
        timer.cancel();
        _fetchActions();
      }
      if (_servicing!.progress == 3) timer.cancel();
    }
  }
}

void _onStepContinue() async {
  _servicing!.acceptedActions =
      _actions!.where((a) => a.selected).map((a) => a.id.toString()).toList();
  _servicing!.progress = _servicing!.progress! + 1;
  _servicing!.step = 0;
  _servicing!.totalStep = _servicing!.acceptedActions!.length;
  setState(() => _isLoading = true);
  final response = await BookingAPI.updateServicing(
    servicing: _servicing!,
    booking: _booking!,
  );
  if (response.isSuccess) {
    if (_timer != null && _timer!.isActive) _timer!.cancel();
    _timer = Timer.periodic(Duration(seconds: 10), _timerCallback);
    _fetchAcceptedActions();
    setState(() => _currentProgress++);
  } else
```



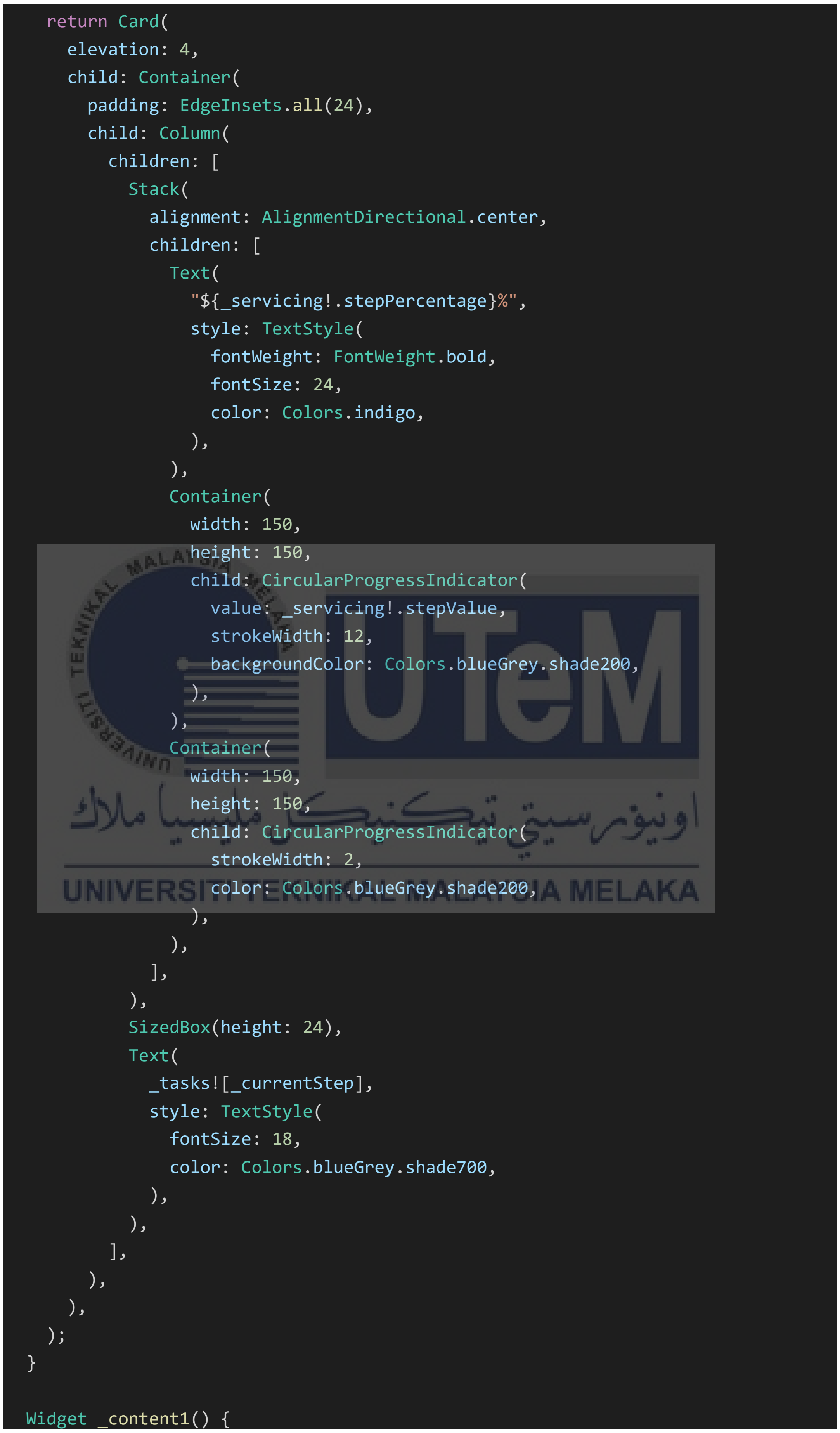
```
      DialogUtils.show(context, response.message!);
    setState(() => _isLoading = false);
  }

  void _reset() {
    setState(() {
      if (_timer != null && _timer!.isActive) _timer!.cancel();
      _booking = _servicing = _tasks = _actions = _acceptedActions = null;
      _currentStep = _currentProgress = 0;
    });
  }

  @override
  void initState() {
    _fetchServicing();
    super.initState();
  }

  @override
  void dispose() {
    if (_timer != null && _timer!.isActive) _timer!.cancel();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.indigo.shade100,
      body: LoadingOverlay(
        isLoading: _isLoading,
        child: _isLoading
            ? Container()
            : _servicing == null
                ? _noServicing()
                : Stepper(
                    currentStep: _currentProgress,
                    type: StepperType.vertical,
                    physics: ScrollPhysics(),
                    controlsBuilder: (_, {onStepCancel, onStepContinue}) {
                      return _currentProgress == 1 && _actions != null
                          ? Align(
                              alignment: Alignment.centerLeft,
                              child: ElevatedButton(
                                onPressed: onStepContinue,
                                child: Text("CONTINUE"),
                              ),
                            )
                          : Container();
                    },
                    onStepContinue: _onStepContinue,
                    steps: List.generate(_progressTitles.length, (index) {
```



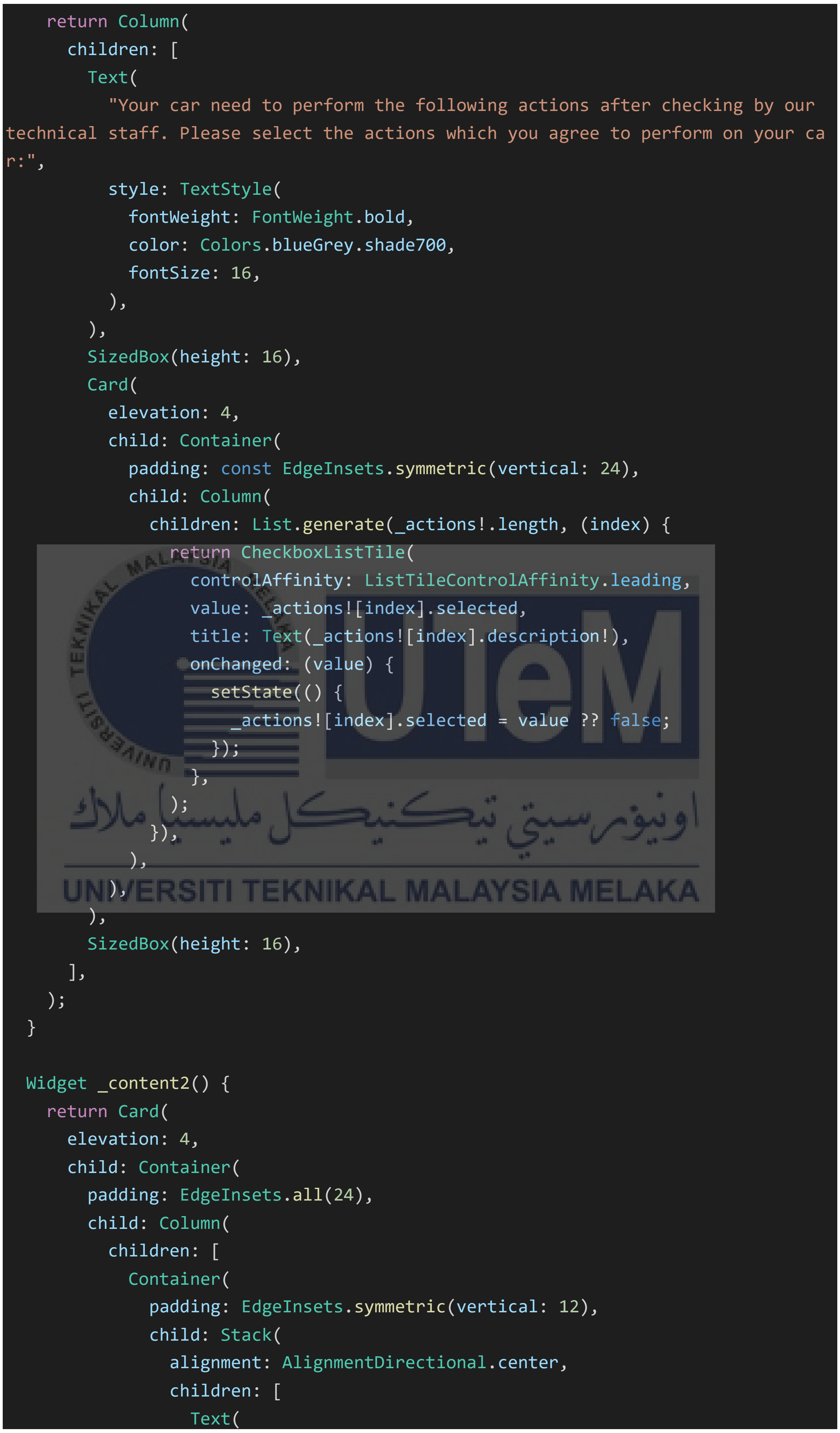
```
          return Step(
           title: Text(
            _progressTitles[index],
            style: TextStyle(
             color: index <= _currentProgress
               ? Colors.indigo
               : Colors.grey.shade600,
             fontWeight: FontWeight.bold,
            ),
           ),
           content: _getStepContent(index),
           isActive: _currentProgress >= index,
           state: _currentProgress > index
             ? StepState.complete
             : StepState.indexed,
          );
         }),
        ),
  ),
  floatingActionButton: FloatingActionButton(
   heroTag: "fab_service_page",
   onPressed: _fetchServicing,
   child: Icon(Icons.refresh),
   tooltip: "Refresh",
  ),
 );
}

Widget _noServicing() {
 return Column(
  mainAxisAlignment: MainAxisAlignment.center,
  crossAxisAlignment: CrossAxisAlignment.stretch,
  children: [
   CircleAvatar(
    radius: 72,
    backgroundColor: Colors.white,
    child: Icon(
     Icons.handyman,
     size: 96,
     color: Colors.blueGrey,
    ),
   ),
   SizedBox(height: 12),
   Text(
    "No Servicing Car",
    style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
    textAlign: TextAlign.center,
   ),
   Container(
    margin: EdgeInsets.symmetric(horizontal: 72, vertical: 12),
    child: RichText(
```



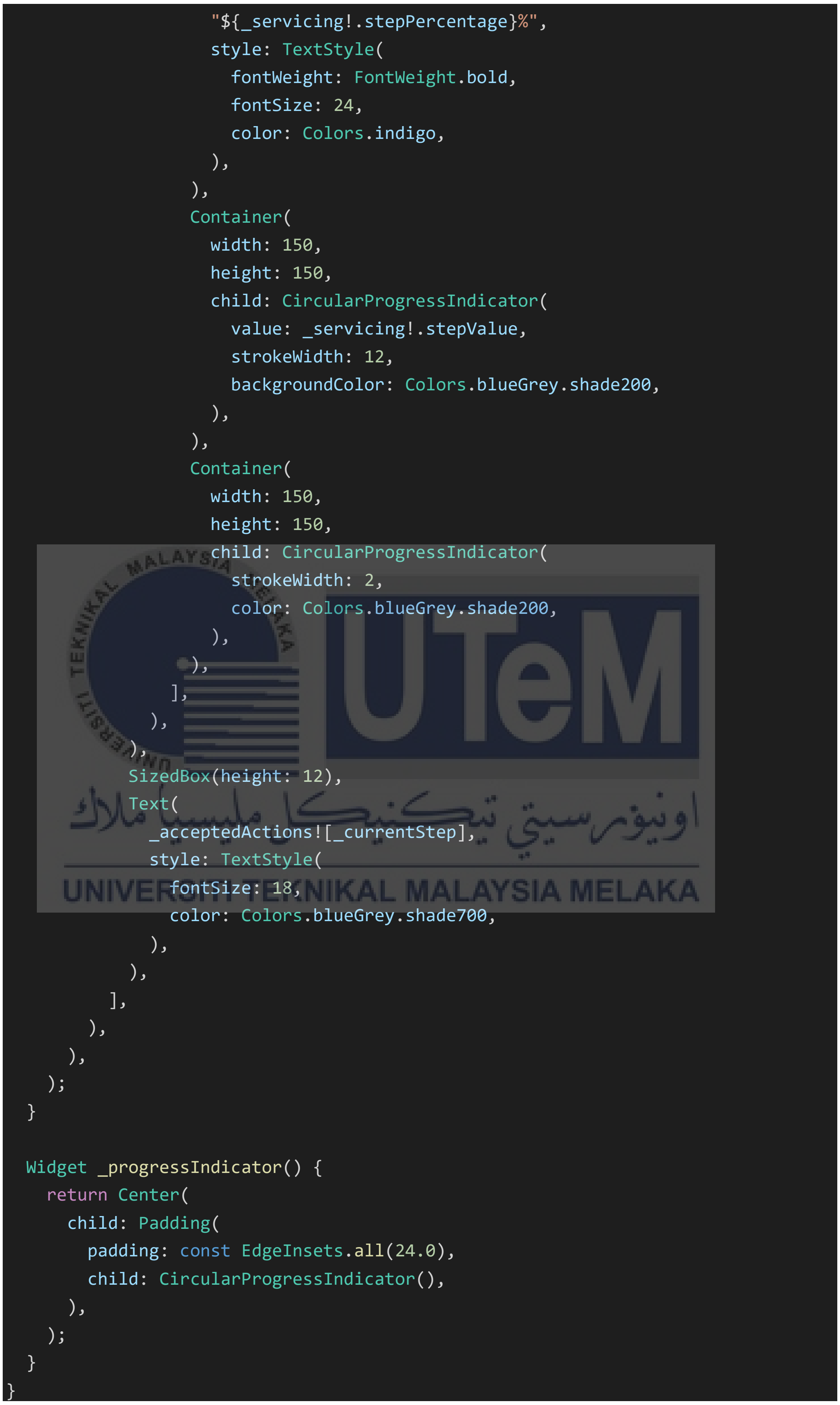
```
      text: TextSpan(
       style: TextStyle(color: Colors.blueGrey.shade800, fontSize: 16),
       children: [
        TextSpan(text: "Please "),
        TextSpan(
         text: "contact us",
         style: TextStyle(
          color: Colors.indigo,
          fontWeight: FontWeight.bold,
         ),
         recognizer: TapGestureRecognizer()..onTap = () {},
        ),
        TextSpan(text: " if your car doesn't service in time."),
       ],
      ),
     ),
    ),
   ],
  );
 }

 Widget _getStepContent(int index) {
  switch (index) {
   case 0:
    return _tasks == null ? _progressIndicator() : _content0();
   case 1:
    return _actions == null ? _progressIndicator() : _content1();
   case 2:
    return _acceptedActions == null ? _progressIndicator() : _content2();
   case 3:
    return Card(
     elevation: 4,
     child: Container(
      width: double.infinity,
      padding: const EdgeInsets.all(24),
      child: Text(
       "Your car had completely serviced. You can pick-
up your car before 8:00pm today. 😁",
       style: TextStyle(
        fontSize: 18,
        color: Colors.blueGrey.shade700,
       ),
      ),
     ),
    );
   default:
    return Container();
  }
 }

 Widget _content0() {
```



```
  return Card(
    elevation: 4,
    child: Container(
      padding: EdgeInsets.all(24),
      child: Column(
        children: [
          Stack(
            alignment: AlignmentDirectional.center,
            children: [
              Text(
                "${_servicing!.stepPercentage}%",
                style: TextStyle(
                  fontWeight: FontWeight.bold,
                  fontSize: 24,
                  color: Colors.indigo,
                ),
              ),
              Container(
                width: 150,
                height: 150,
                child: CircularProgressIndicator(
                  value: _servicing!.stepValue,
                  strokeWidth: 12,
                  backgroundColor: Colors.blueGrey.shade200,
                ),
              ),
              Container(
                width: 150,
                height: 150,
                child: CircularProgressIndicator(
                  strokeWidth: 2,
                  color: Colors.blueGrey.shade200,
                ),
              ),
            ],
          ),
          SizedBox(height: 24),
          Text(
            _tasks![_currentStep],
            style: TextStyle(
              fontSize: 18,
              color: Colors.blueGrey.shade700,
            ),
          ),
        ],
      ),
    ),
  );
}

Widget _content1() {
```

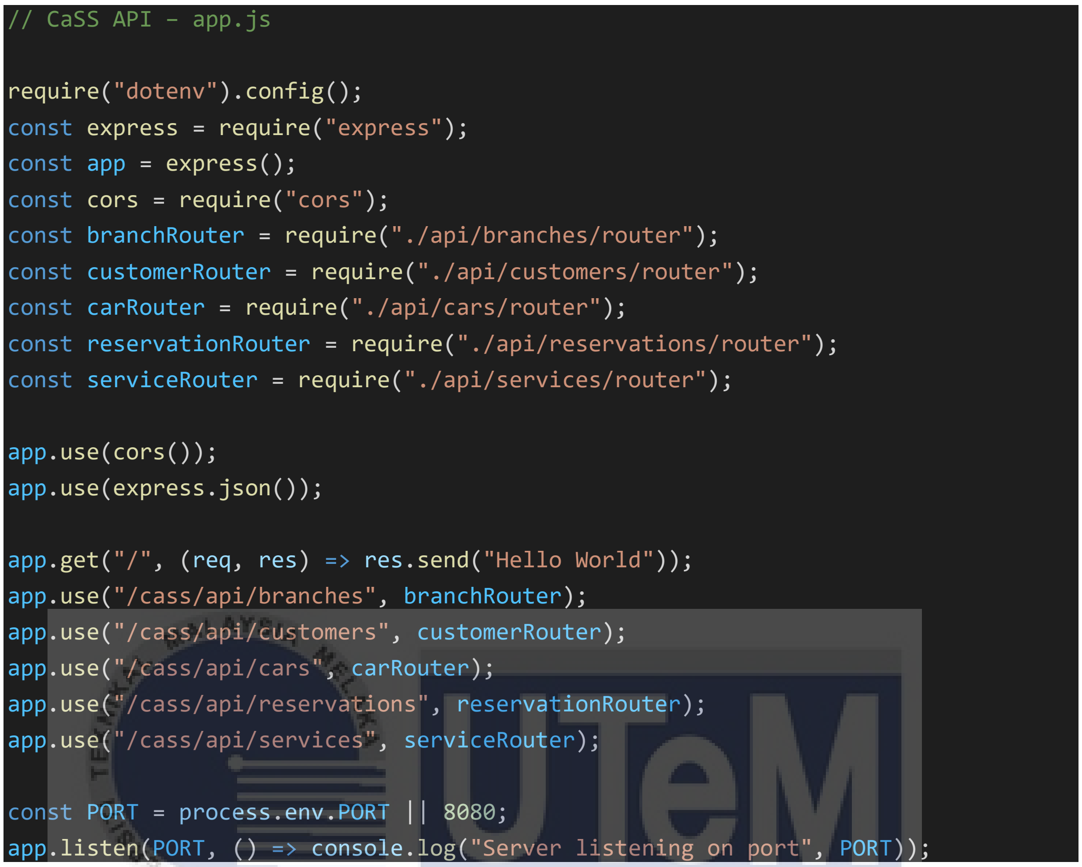


```
    return Column(
      children: [
        Text(
          "Your car need to perform the following actions after checking by our
technical staff. Please select the actions which you agree to perform on your ca
r:",
          style: TextStyle(
            fontWeight: FontWeight.bold,
            color: Colors.blueGrey.shade700,
            fontSize: 16,
          ),
        ),
        SizedBox(height: 16),
        Card(
          elevation: 4,
          child: Container(
            padding: const EdgeInsets.symmetric(vertical: 24),
            child: Column(
              children: List.generate(_actions!.length, (index) {
                return CheckboxListTile(
                  controlAffinity: ListTileControlAffinity.leading,
                  value: _actions![index].selected,
                  title: Text(_actions![index].description!),
                  onChanged: (value) {
                    setState(() {
                      _actions![index].selected = value ?? false;
                    });
                  },
                );
              }),
            ),
          ),
        ),
        SizedBox(height: 16),
      ],
    );
  }

  Widget _content2() {
    return Card(
      elevation: 4,
      child: Container(
        padding: EdgeInsets.all(24),
        child: Column(
          children: [
            Container(
              padding: EdgeInsets.symmetric(vertical: 12),
              child: Stack(
                alignment: AlignmentDirectional.center,
                children: [
                  Text(
```



```
                    "${_servicing!.stepPercentage}%",
                    style: TextStyle(
                      fontWeight: FontWeight.bold,
                      fontSize: 24,
                      color: Colors.indigo,
                    ),
                  ),
                  Container(
                    width: 150,
                    height: 150,
                    child: CircularProgressIndicator(
                      value: _servicing!.stepValue,
                      strokeWidth: 12,
                      backgroundColor: Colors.blueGrey.shade200,
                    ),
                  ),
                  Container(
                    width: 150,
                    height: 150,
                    child: CircularProgressIndicator(
                      strokeWidth: 2,
                      color: Colors.blueGrey.shade200,
                    ),
                  ),
                ],
              ),
            ),
            SizedBox(height: 12),
            Text(
              _acceptedActions![_currentStep],
              style: TextStyle(
                fontSize: 18,
                color: Colors.blueGrey.shade700,
              ),
            ),
          ],
        ),
      ),
    );
  }

  Widget _progressIndicator() {
    return Center(
      child: Padding(
        padding: const EdgeInsets.all(24.0),
        child: CircularProgressIndicator(),
      ),
    );
  }
}
```

Appendix C – Sample Source Code 3



```
// CaSS API - app.js

require("dotenv").config();
const express = require("express");
const app = express();
const cors = require("cors");
const branchRouter = require("./api/branches/router");
const customerRouter = require("./api/customers/router");
const carRouter = require("./api/cars/router");
const reservationRouter = require("./api/reservations/router");
const serviceRouter = require("./api/services/router");

app.use(cors());
app.use(express.json());

app.get("/", (req, res) => res.send("Hello World"));
app.use("/cass/api/branches", branchRouter);
app.use("/cass/api/customers", customerRouter);
app.use("/cass/api/cars", carRouter);
app.use("/cass/api/reservations", reservationRouter);
app.use("/cass/api/services", serviceRouter);

const PORT = process.env.PORT || 8080;
app.listen(PORT, () => console.log("Server listening on port", PORT));
```