APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 6 No. 1 (2025) 920-939
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

The Development of Agos Motor Workshop Booking and Inventory Management System

Nur Izzah Hannani Mohammad[1], Nur Ariffin Mohd Zin[1*]

[1] *Faculty of Science Computer and Information Technology, Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA*

*Corresponding Author: ariffin@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.01.051

Article Info

Received: 18 June 2024
Accepted: 19 June 2025
Available online: 30 June 2025

Keywords

Booking System, Inventory Management System, Motorcycle Workshop

Abstract

Agos Motor workshop is a local motorcycle workshop located in Batu Pahat, Johor. The workshop has been established since the year 1990 focusing on motorcycle services such as modification, repairment, and maintenance, and selling individual motorcycle spare parts. However, as the workshop gained trust from the locals and new customers after being well established for more than 30 years, several problems surfaced which are data redundancy, limited manpower, no-show attendance for appointments, and inefficient manual processes. To address the aforementioned issues, this study proposes the Agos Motor Workshop Booking and Management System, designed to revolutionize these outdated methods by integrating web technology. The system is developed by using PHP, CSS, Javascript, and HTML as the programming language and MySQL as the database. Prototyping model is used to ensure rapid development and iterative testing. This approach allows for continuous feedback and refinement, resulting in a robust and user-friendly solution. Key features include a user-friendly interface, real-time data processing, and secure payment mechanisms. The system also supports integration with third-party services, making it a versatile tool for modern business operations. Results show that the system has successfully addressed the issues mentioned, with most respondents providing positive feedback and an over 80 percent satisfaction rate.

1. Introduction

Statista Research Department [1] stated that the automotive market in Malaysia expanded by more than 40% this year after declining for the previous two years in a row due to the COVID-19 outbreak which leaves everyone around the globe with no choice but to stay at home. With this rapid increment of motor vehicle sales, the automotive services center is needed to cater to the needs of the vehicle to keep them in good condition and subsequently deter unwanted vehicle problems in the long run. Technology is inevitable and irreplaceable in human's daily lives, along with the rapid increment of the motor vehicle sales and automotive market in Malaysia, a workshop booking, and inventory management system can be one of the technologies that may facilitate a small business' daily operation. Agos Motor workshop is a local motorcycle workshop located in Batu Pahat, Johor. The workshop has been established since the year 1990 focusing on motorcycle services such as modification, repairment, and maintenance, and selling individual motorcycle spare parts. Even though the workshop has been established for more than 30 years, the owner has not yet implemented any inventory software in their daily business operation as the owner is not very fond of new technologies. The owner was satisfied with the traditional

recording inventory method he used before. However, as time goes on, the owner is currently selling more spare part products which made traditional method of inventory recording more difficult to keep track of the stock and thus cause him to suffer some profit loss. The workshop's inventory-related difficulties primarily stem from the increasing sales volume of motorcycle spare parts. As the workshop sells more spare parts products, the conventional inventory recording method has proven to be inadequate in accurately tracking stock levels. This limitation has led to difficulties in managing inventory effectively, potentially resulting in profit loss due to inaccuracies and inefficient management practices. This paper consists of five sections. Section one describes the project introduction and research background. In section two, the related work is executed describing the literature study and a comparison study of similar existing system. Section three, namely methodology, describes the system development methodology while section four, namely results and discussion, shows the system testing result. The last section, which is section five namely conclusion, explains about the project's advantages, limitations, and future works of the developed system.

2. Related Work

The domain case study of Agos Motor Workshop will be discussed in this section as well as the system's literature review in the following subsections.

2.1 Booking Appointment Management

An appointment booking system can be integrated with a queue management system to manage customers' experience and to keep track of the customers' visit before, during, and after being served such as stated by SEDCO [2]. By using the booking appointment system, the customer will find it easy to schedule appointments for services and thus enjoy a better user experience. The customer does not have to wait for regular office hours to make an appointment, they can do so whenever and from anywhere. The booking appointment system can help market services more effectively to draw in new customers as well as retain existing ones. This integration empowers the business to receive insightful data to support future planning such as the number of booked versus walk-in customers, no-show customers, and many other data such as peak visiting hours.

2.2 Inventory Management

The concept of inventory management is a crucial analytical aspect of overall management [3]. It revolves around optimizing the resources allocated for maintaining stocks of various materials. Insufficient inventory levels can result in stock-outs, leading to production stoppages. On the other hand, a very high inventory can result in increased cost of production due to high cost of carrying inventory. Thus, the optimization of inventory is essential to strike a balance to ensure that stock levels are neither too low nor excessively high.

2.3 Comparison of existing system

Three existing systems on the market were studied. This study is carried out so that the system developer can analyze and identify the benefits and shortcomings of the existing systems to use it as a reference when developing the proposed system. The three existing systems that have some similarities and are related to the proposed system have been chosen which are UFirst Perodua [4], SDAC Ford Holding [5], and Proton Holding [6].

Table 1 *Comparison between existing system and the proposed system*

Features/System	UFirst Perodua	SDAC Ford Holding	Proton Holding	Agos Motor Workshop
Register	Yes, by using login ID, e-mail address, and phone number	Yes, by using vehicle identification number	Yes, by using e-mail address and phone number	Yes, by using phone number and e-mail address
Login	Yes, by using login ID	Yes, by using e-mail address	Yes, by using e-mail address	Yes, by using phone number
Display dashboard	Yes	Yes	Yes	Yes
Booking service appointment	Yes	Yes	Yes	Yes
Manage service appointment	Yes	Yes	Yes	Yes
Validate time slot	No	No	No	Yes

Table 1 (*cont.*)

Update service status	Yes	Yes	Yes	Yes
Manage spare parts	No	No	No	Yes
Make payment and invoice	No	No	Yes	Yes
Generate report	No	No	No	Yes

Based on the comparison of existing and proposed systems done in Table 1, it can be concluded that there are some similarities and differences in these systems. The results of this comparison will serve as a guide to developing the Agos Motor Workshop Booking and Management System

3. Methodology

Agos Motor Workshop Booking and Management System was developed by using the Prototyping Model methodology. The methodology can be applied by developing, testing, and redevelop the system's prototype repeatedly until the real outcome of the system is satisfied [2]. As the system's prototype is flexible in design and can be redeveloped along in the development process, it can ensure a greater level of customer satisfaction.

3.1 System Development Workflow

The prototype model has a total of five phases. As shown in Table 2, each phase has its assignment and output that needs to be produced during the entire project development.

Table 2 *Software development activities and the tasks*

Phase	Task	Output
Planning	• Proposed the project. • Determine the project schedule, activities, and output. • Determine the project background, problem statement, objectives, scope, expected result, and project significance. • Interview the stakeholders. • Develop a project proposal. • Prepare a schedule of timelines, tasks, and output.	• Project proposal. • Develop Gantt chart.
Analysis	• Analyze system requirements. • Investigate the existing system. • Analyze hardware and software requirements. • Determine the programming language, database, framework, and library that will be used. • Choosing Prototyping Model as the methodology.	• Literature review. • Swimlane diagram (As-Is Model). • Use case diagram. • Activity diagram. • Sequence diagram. • Class diagram. • Requirement definition. • Hardware and Software specification.
Design	• Design the interface design. • Design the database. • Develop the UML diagrams.	• Interface design. • Database schema. • Data dictionary.
System Prototype Implementation	• Develop prototypes. • Collect feedback from stakeholders for the first prototype. • Refined the prototype. • Develop the second prototype. • Collect feedback from stakeholders for the second prototype. • Finalize the design and prototype.	• Prototype 1 (The interfaces of the system). • Prototype 2 (The changes in the interface system). • The final design of the system.
Final Implementation	• Develop the program code for the system interface. • Develop the program code. • Develop the program code to connect the backend and database. • Refactor the code and debug the code. • System Testing (User Acceptance Testing). • Roll out.	• Completed system. • Test cases report. • Requirement Traceability Matrix (Test cases vs Requirement)

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 920-939

3.2 Analysis

This section presents the analysis result, depicted through various visual representations. These include a swimlane diagram, a use case diagram, a class diagram, and a comprehensive requirement definition. These visual aids collectively convey the findings and insights derived from the analysis of the system.

3.2.1 Swimlane Diagram

The swimlane diagram as shown in Fig. 1 outlines the actions the actors take throughout the booking process, starting with the admin assigning appointment sessions with timeslots and date to the mechanic. The customer can view available appointment sessions and book a service appointment session and make payment. Invoices will be issued by the system and the customer can show the invoice at the workshop counter. Lastly, the mechanic can view the sessions assigned to them and prepare for the service appointments. The mechanic can update service status upon completion and the customer can view their motorcycle service status.

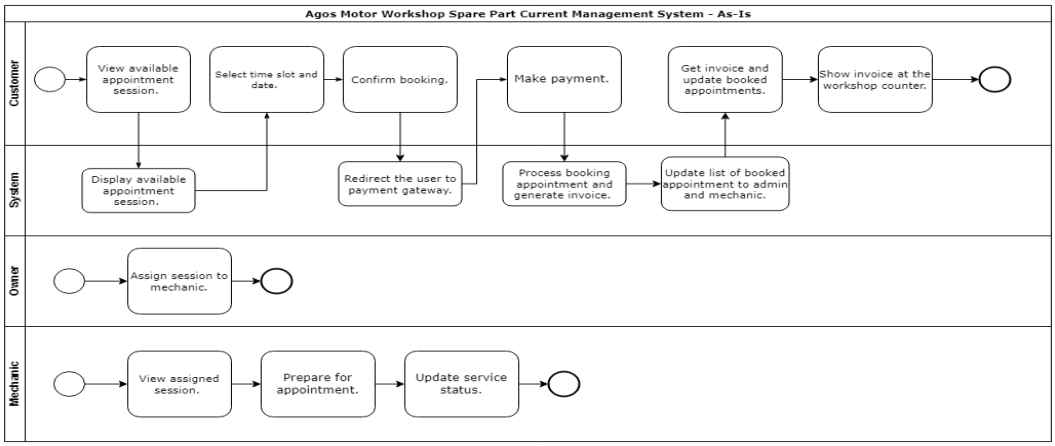

Fig. 1 Swimlane Diagram (To-Be)

3.2.2 Use Case Diagram

A use case diagram as shown in Fig. 2 is developed as a part of the study to analyze, annotate, and illustrate the overall functionality and component of the booking and management system. The role of actors in relation to the system will also be described in the use case diagram. The actors in this system include Workshop Owner, Mechanic, and Customer. A total of nine use cases such as register, login, display dashboard, book service appointment, manage service appointment, update service status, manage spare part, make payment and invoice, and generate reports, are included in this use case.

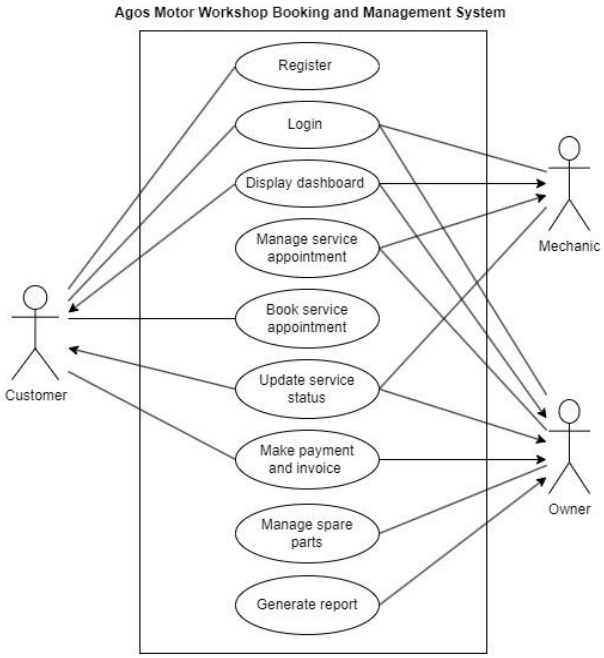

Fig. 2 Use Case Diagram

The use case diagram such as shown in Fig. 2 above illustrates the booking and management process in Agos Motor Workshop business operation. There are three actors that are designated for the system which are customers, mechanic, and workshop owner. The customers can register themselves in the system and book a service appointment.

The mechanic on the other hand is able to view the assigned service task to him and get to update the customer and workshop owner on the service's task status. With the development of this module, the mechanic can manage his time better to focus on a service task at one time, instead of having two or more service tasks coming at him at the same time without a prior notice since a customer will visit the workshop at any time without booking an appointment in advance. In this system, the owner can assign appointment service with the timeslot and date to the mechanic and manage the spare parts products.

3.2.3 Class Diagram

The class diagram for the booking and inventory management system is displayed in Fig. 3. There are 8 classes in the class diagram: Customer, Session, Appointment, Mechanic, Products, Invoice, Owner, and Report. The fields for each class are displayed, and multiplicity is used to illustrate the link between the classes.

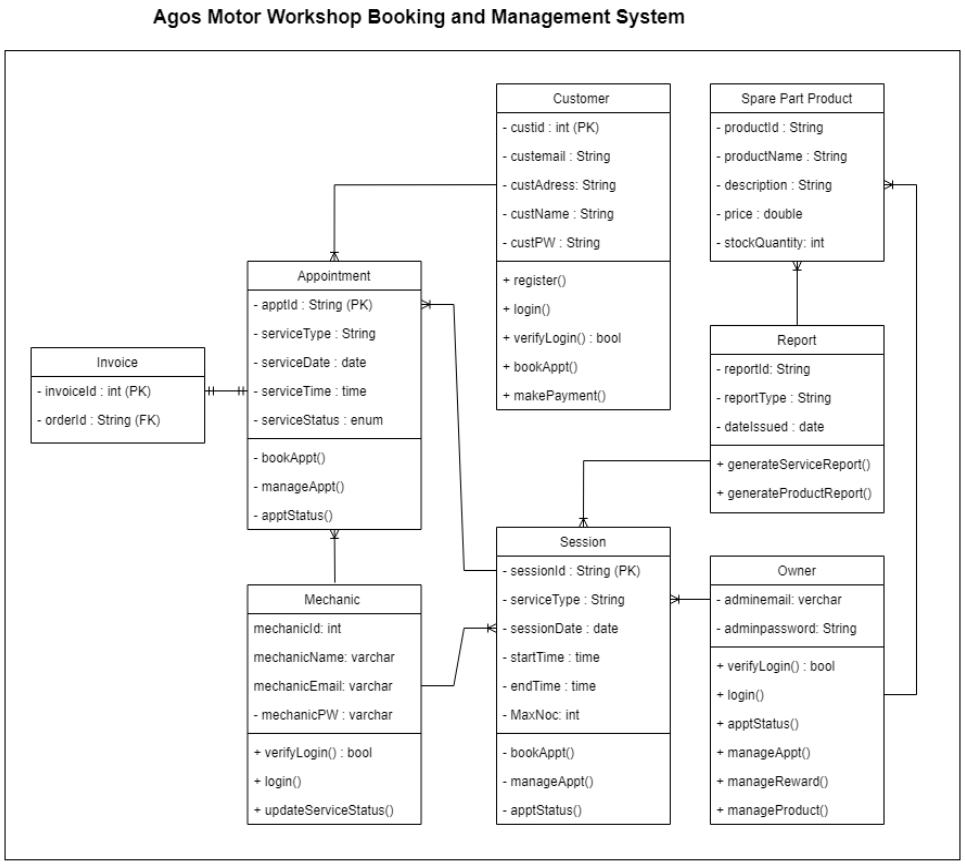

Fig. 3 Class Diagram

3.2.4 System Requirement Analysis

The process of system requirement analysis plays a pivotal role in establishing the anticipated outcomes for the users of a proposed system. This involves gathering all necessary requirements, assessing challenges in system design, and ensuring that the system aligns with its goals, allowing users to efficiently handle inventory information. A thorough requirement analysis is essential for the seamless operation of a mobile application, as failure to meet these requirements may lead to potential issues during installation or impact performance. Functional requirements specify what the system does, non-functional requirements ensure performance and quality, and user requirements align with the user needs and expectations.

(a) User Requirement Analysis

Table 3 shows the user requirement for the proposed system. The user requirements detail the expected functionality of each module.

Table 3 *User Requirements Analysis*

Use Case Name	User Requirement
Register	• The customer must be able to register as a new account to the system by inputting their first name, last name, e-mail address, phone number, and password. • The customer must be redirected to the login page after registering a new account.
Login	• The customer must be able to enter a valid phone number and password to access the system. • The owner and mechanic must be able to enter a valid ID number and password to access the system. • The customer must be able to reset their password for login to the system.
Display Dashboard	• The users must be redirected to the dashboard of the system after logging in to their account. • The user must be able to see the dashboard of the system after logging in to their account.
Book Service Appointment	• One-time customer should be able to create new booking appointments in the system by inputting their e-mail address, phone number, name, motorcycle type and number plate, service type, date, and time.
Manage Service Appointment	• The owner should be able to accept or reject the requested appointment in the system. • The owner should be able to assign the service appointment to the mechanic in the system. • The mechanic must be able to see the list of service appointments assigned to him in the system.
Update Service Status	• The mechanic must be able to update the customer's motorcycle service status in the system. • The customer must be able to receive a notification if their motorcycle's service status is completed. • The owner must be able to see the list of the customer's motorcycle service status in the system.
Manage Spare Parts	• The owner must be able to create, update, and delete the spare parts items and edit the details such as the items' price, description, and quantity of the spare parts in the inventory list.
Make Payment and Invoice	• The customer must be able to pay for their service booking fee. • The customer must be able to print the invoice of their payment.
Generate Report	• The owner should be able to view and print a product report consisting of the list of booked appointments and service report consisting of the list of services.

3.3 Design

The design is articulated through two key components: the system interfaces design and database design.

3.3.1 System interface design

User interface (UI) design is the process through which designers construct interfaces for software or computerized devices, emphasizing the aesthetics and style. The goal is to develop interface that users perceive as user-friendly and enjoyable. UI design encompasses graphical user interfaces and various formats, including interfaces controlled by voice commands.

The Booking and Management System is developed by using Visual Studio Code that serves as an IDE software and MySQL database that were used to store data. The programming languages that have been used are Hypertext Preprocessor (PHP) and Hypertext Markup Language (HTML). The modules that have been involved in this system are Register, Login, Display Dashboard, Manage Service Appointment, Book Service Appointment, Update Service Status, Make Payment and Invoice, Manage Spare Parts, and Generate Report. Each of the modules will be explained via the interfaces and the functions.

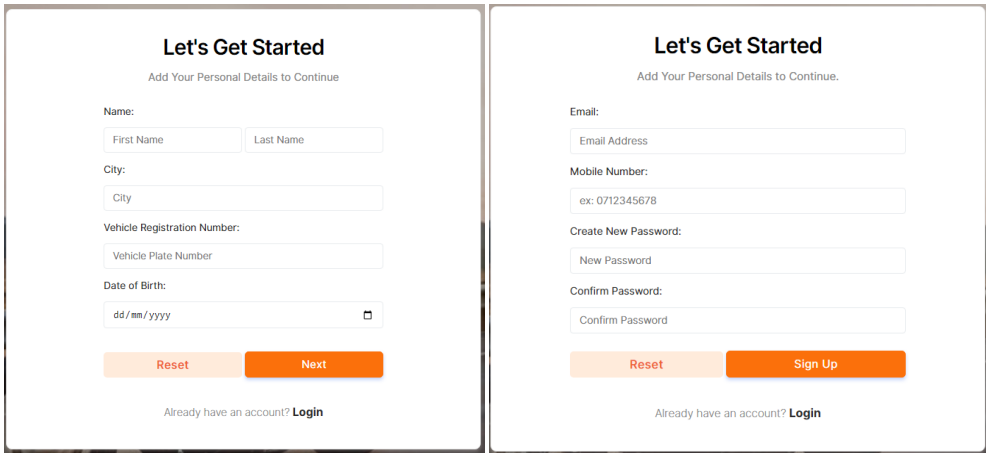

Fig. 5 *Register Interface*

Fig. 5 shows the interface for customers to register a new account in the system. A new customer needs to fill in the registration details and click on the Sign-Up button in the Register interface to register as a new user. Upon clicking the Sign-Up button, the user will be redirected to the customer's dashboard.

```
58  if ($newpassword==$cpassword){
59      $result= $database->query("select * from webuser where email='$email';");
60      if($result->num_rows==1){
61          $error='<label for="promter" class="form-label" style="color:rgb(255, 62, 62);text-align:center;">
62          Already have an account for this Email address.</label>';
63      }else{
64          $hashed_password = password_hash($newpassword, PASSWORD_BCRYPT);
65          $database->query("insert into customer(custemail,custname,custpassword, custcity, platenum,custdob,custnumphone)
66          values('$email','$name','$newpassword','$custcity','$platenum','$dob','$tele');");
67          $database->query("insert into webuser values('$email','c')");
68          $_SESSION["user"]=$email;
69          $_SESSION["usertype"]="c";
70          $_SESSION["username"]=$fname;
71
72          header('Location: customer/index.php');
73          $error='<label for="promter" class="form-label" style="color:rgb(255, 62, 62);text-align:center;"></label>';
74      }
75
76  }else{
77      $error='<label for="promter" class="form-label" style="color:rgb(255, 62, 62);text-align:center;">
78      Password Confirmation Error! Reconfirm Password</label>';
79  }
```

Fig. 6 *Code Segment of Register*

Fig. 6 shows the code segment that is used to check the existence of email address in the database, password, and re-enter password. The system checks the format of the email address entered and the password confirmation. If the email address has been registered, the system will alert the customer that the email has been registered. If the entered password did not possess the same value as the confirm password, another alert will be triggered by the system.

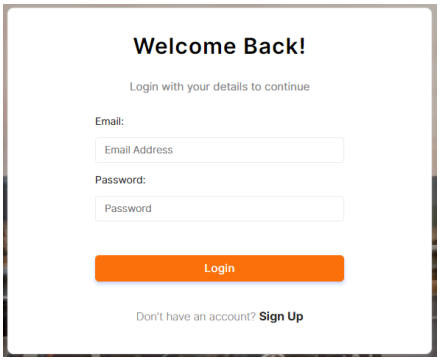

Fig. 7 *Login Interface*

Fig. 7 shows the interface for registered customers, mechanics, and admin to log into the system. The users are prompted to enter their email address and password. The system checks the format of the email address and verifies the email and password in the database. Upon clicking the Login button, the system will prompt an alert message, only if any invalid credential occurs. Otherwise, the user will be redirected to the dashboard according to their role as customer, mechanic, or admin.

```php
    $email=$_POST['useremail'];
    $password=$_POST['userpassword'];
    $error='<label for="promter" class="form-label"></label>';
    $result= $database->query("select * from webuser where email='$email'");

    if($result->num_rows==1){
        $usertype=$result->fetch_assoc()['usertype'];
        if ($usertype=='c'){
            $checker = $database->query("select * from customer where custemail='$email' and custpassword='$password'");
            if ($checker->num_rows==1){
                $row = $checker->fetch_assoc();
                //   Customer dashboard
                $_SESSION['user']=$email;
                $_SESSION['usertype']='c';

                header('location: customer/index.php');
            }else{
                $error='<label for="promter" class="form-label" style="color:rgb(255, 62, 62);text-align:center;">Wrong credentials: Invalid email or password</label>';
            }
    }elseif($usertype=='a'){
        $checker = $database->query("select * from admin where aemail='$email' and apassword='$password'");
        if ($checker->num_rows==1){

            //   Admin dashbord
            $_SESSION['user']=$email;
            $_SESSION['usertype']='a';
            header('location: admin/index.php');
        }else{
            $error='<label for="promter" class="form-label" style="color:rgb(255, 62, 62);text-align:center;">
            Wrong credentials: Invalid email or password</label>';
        }

    }elseif($usertype=='m'){
        $checker = $database->query("select * from mechanic where mechanicemail='$email' and mechanicpassword='$password'");
        if ($checker->num_rows==1){
            //   mechanic dashbord
            $_SESSION['user']=$email;
            $_SESSION['usertype']='m';
            header('location: mechanic/index.php');
        }else{
            $error='<label for="promter" class="form-label" style="color:rgb(255, 62, 62);text-align:center;">
            Wrong credentials: Invalid email or password</label>';
        }
    }
}else{
    $error='<label for="promter" class="form-label" style="color:rgb(255, 62, 62);text-align:center;">
    We cant found any acount for this email.</label>';
}
```

Fig. 8 *Code Segment for Login*

Fig. 8 shows the code segment of Login interface for the customer, mechanic, and admin. The system checks the format of the email and validates the email and password in the database. The system will then check if the email address and password match the user type customer, mechanic, or admin. If any invalid credential occurs, an alert message will be displayed. Otherwise, the user will be redirected to the dashboard according to their role as customer, mechanic, or admin.

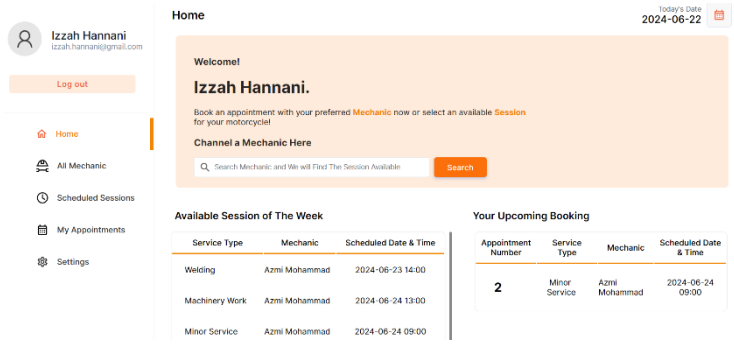

Fig. 9 *Customer's Home Page Interface*

Fig. 9 shows the customer's home page interface. The lists of available sessions within the next seven days and the customer's upcoming booked appointment are also displayed. The menus such as All Mechanics, Scheduled Session, My Appointments, and Settings can be accessed by clicking on the menus on the left-side of the home page.

```php
<?php
$nextweek=date("Y-m-d",strtotime("+1 week"));
    $sqlmain= "select * from session inner join appointment on session.sessionid=appointment.sessionid inner join customer on customer.custid=appointment.custid
    inner join mechanic on session.mechanicid=mechanic.mechanicid where  customer.custid=$userid  and session.sessiondate>='$today' order by session.sessiondate asc";
    $result= $database->query($sqlmain);
    if($result->num_rows==0){
        echo '<tr>
        <td colspan="4">
        <br><br><br><br>
        <center>
        <img src="../img/notfound.svg" width="25%">
        <br>
        <p class="heading-main12" style="margin-left: 45px;font-size:20px;color:rgb(49, 49, 49)">Nothing to show here!</p>
        <a class="non-style-link" href="schedule.php"><button  class="login-btn btn-primary-soft btn"
        style="display: flex;justify-content: center;align-items: center;margin-left:20px;">  Channel a Mechanic  </font></button>
        </a>
        </center>
        <br><br><br><br>
        </td>
        </tr>';

    }
    else{
    for ( $x=0; $x<$result->num_rows;$x++){
        $row=$result->fetch_assoc();
        $sessionid=$row["sessionid"];
        $spe=$row["servicetype"];
        $spcil_res= $database->query("select sname from servicetype where id='$spe'");
        $spcil_array= $spcil_res->fetch_assoc();
        $spcil_name=$spcil_array["sname"];
        $apponum=$row["apponum"];
        $mechanicname=$row["mechanicname"];
        $sessiondate=$row["sessiondate"];
        $starttime=$row["starttime"];
```

Fig. 10 *The Code Segment for Customer's Home Page Interface*

The provided code segment for the customer's home page interface such as shown in Fig. 10 is used to fetch data from the table 'session'. The query fetched from the table will then be displayed as a list of scheduled sessions to provide information to the customer. The query fetched from the table will then be displayed as a list of scheduled sessions to provide information to the customer.

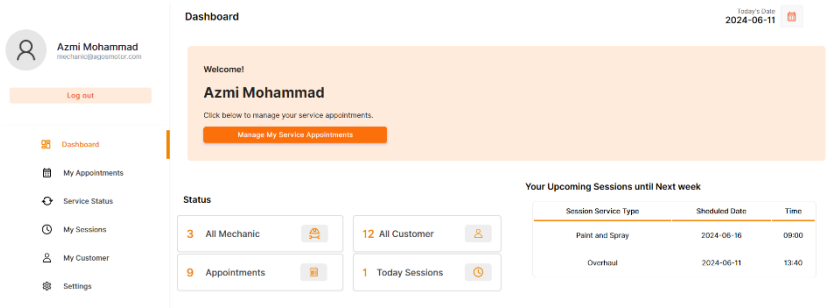

Fig. 11 *Mechanic's Dashboard Interface*

Fig. 11 shows the mechanic's dashboard interface. Statuses of the system such as number of registered mechanics, customers, appointments, and assigned session of the day are displayed. The list of upcoming sessions assigned to the mechanic are also displayed. The mechanic can access and manage modules such as My Appointment, Service Status, My Sessions, My Customer, and Settings by clicking the menus on the left-side of the dashboard page.

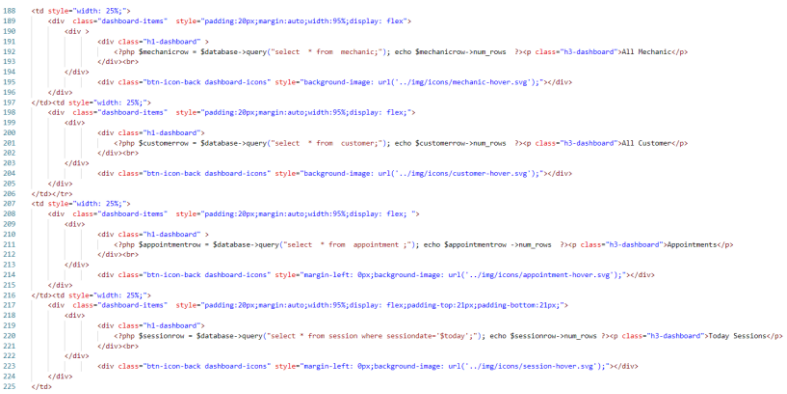

Fig. 12 *The Code Segment for Mechanic's Dashboard Interface*

Fig. 12 shows the code segment for the mechanic's dashboard interface that is used to display the statuses of the system.

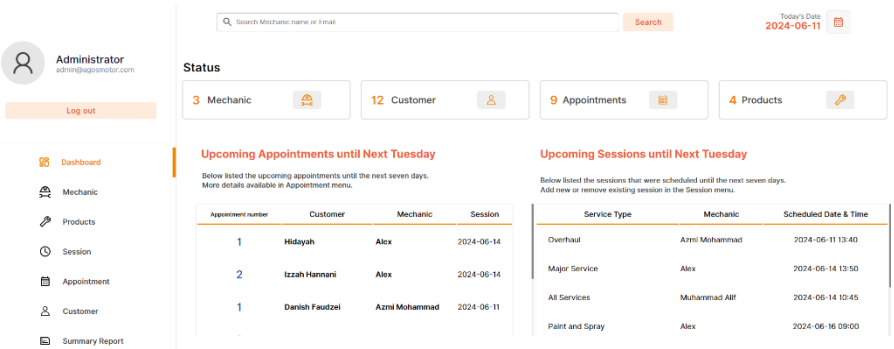

Fig. 13 *Display Dashboard for Admin Interface*

Fig. 13 shows the admin's dashboard interface. Statuses of the system such as number of mechanics, customers, appointments, and spare part products are displayed. The lists of upcoming appointments and sessions in the next seven days are also displayed. The admin can access and manage modules such as Mechanic, Products, Session, Appointment, Customer, and Summary Report by clicking on the menus on the left-side of the dashboard page.

```php
<?php
$nextweek=date("Y-m-d",strtotime("+1 week"));
$sqlmain= "select appointment.appoid,session.sessionid,mechanic.mechanicname,customer.custname,session.sessiondate,appointment.apponum,appointment.appodate
from session inner join appointment on session.sessionid=appointment.sessionid inner join customer on customer.custid=appointment.custid
inner join mechanic on session.mechanicid=mechanic.mechanicid  where session.sessiondate>='$today'  and session.sessiondate<='$nextweek' order by session.sessiondate desc";
    $result= $database->query($sqlmain);
    if($result->num_rows==0){
        echo '<tr>
        <td colspan="3">
        <br><br><br><br>
        <center>
        <img src="../img/motorcycle.svg" width="29%">
        <br>
        <p class="heading-main12" style="margin-left: 45px;font-size:20px;color:rgb(49, 49, 49)">   No appointment booked until the next seven days.</p>
        <a class="non-style-link" href="appointment.php"><button class=" btn-primary-soft btn"  style="display: flex;justify-content: center;align-items: center;margin-left:20px;">
          Show all Appointments  </font></button>
        </a>
        </center>
        <br><br><br><br>
        </td>
        </tr>';
    }
    else{
    for ( $x=0; $x<$result->num_rows;$x++){
        $row=$result->fetch_assoc();
        $appoid=$row["appoid"];
        $mechanicname=$row["mechanicname"];
        $sessiondate=$row["sessiondate"];
        $pname=$row["custname"];
        $apponum=$row["apponum"];
        $appodate=$row["appodate"];
```

Fig. 14 *The Segment Code for Admin's Dashboard Interface*

The code segment of the display dashboard interface for admin is shown in Fig. 14 is used to display the modules such as Mechanic, Products, Session, Appointment, Customer, and Summary Report on the left-side of the admin's dashboard interface. The system will also redirect the admin to the page of the modules if clicked.

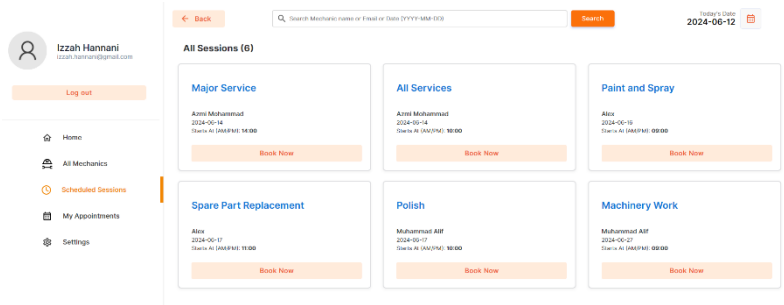

Fig. 15 *Book Service Appointment for Customer's Interface*

Fig. 15 shows the book service appointment interface for customers. The details of each session such as service type, mechanic name, session date and time are displayed. A customer will be able to book a session by clicking on the Book Now button and the system will redirect the customer to the booking page.

```php
for ( $x=0; $x<($result->num_rows);$x++){
    echo "<tr>";
    for($q=0;$q<3;$q++){
        $row=$result->fetch_assoc();
        if (!isset($row)){
            break;
        };
        $sessionid=$row["sessionid"];
        $spe=$row["servicetype"];
        $spcil_res= $database->query("select sname from servicetype where id='$spe'");
        $spcil_array= $spcil_res->fetch_assoc();
        $spcil_name=$spcil_array["sname"];
        $mechanicname=$row["mechanicname"];
        $sessiondate=$row["sessiondate"];
        $starttime=$row["starttime"];
        if($sessionid==""){
            break;}
        echo '
        <td style="width: 25%;">
                <div  class="dashboard-items search-items"  >

                    <div style="width:100%">
                            <div class="h1-search">
                                '.substr($spcil_name,0,30).'
                            </div><br>
                            <div class="h3-search">
                                '.substr($mechanicname,0,30).'
                            </div>
                            <div class="h4-search">
                                '.$sessiondate.'<br>Starts At (AM/PM): <b>'.substr($starttime,0,5).'</b>
                            </div>
                            <br>
                            <a href="booking.php?id='.$sessionid.'" ><button  class="login-btn btn-primary-soft btn "
                             style="padding-top:11px;padding-bottom:11px;width:100%"><font class="tn-in-text">Book Now</font></button></a>
                    </div>

                </div>
            </td>';
```

Fig. 16 *Code Segment for Book Service Appointment for Customer's Interface*

Fig. 16 shows the code segment for book service appointment interface that is used to display the details of the service appointment.

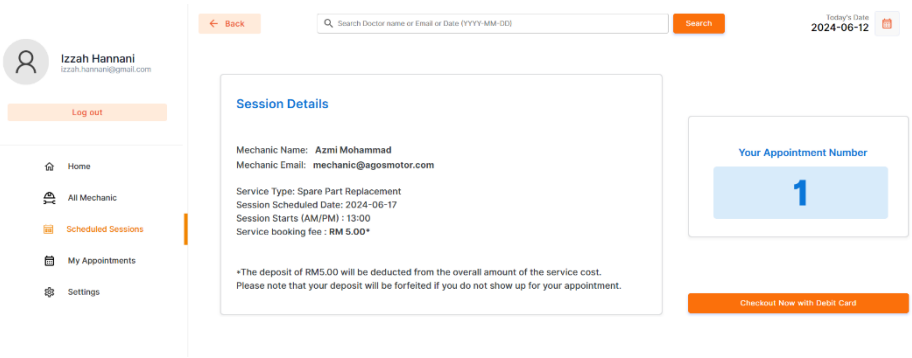

Fig. 17 *Booking Page for Customer's Interface*

Fig. 17 shows the booking page for customers. Extensive details of the service appointment are displayed on this page. The system will redirect the customer to the payment gateway to book a service appointment if the Checkout Now with Debit Card button is clicked.

```
$userrow = $database->query("select * from customer where custemail='$useremail'");
$userfetch=$userrow->fetch_assoc();
$userid= $userfetch["custid"];
$username=$userfetch["custname"];

if($_POST){
    if(isset($_POST["booknow"])){
        $apponum=$_POST["apponum"];
        $date=$_POST["date"];
        $sessionid=$_POST["sessionid"];
        $sql2="insert into appointment(custid,apponum,sessionid,appodate) values ($userid,$apponum,$sessionid,'$date')";
        $result= $database->query($sql2);
        if ($result) {
        $appoid = $database->insert_id;
        header("location: payment/checkout.php");
        exit();
        }else{}
    }
}
```

Fig. 18 *Code Segment for Customer's Booking Page*

The code segment as shown in Fig. 18 is used to insert a new appointment into the 'appointment' table with the provided appointment number, session ID, and date, then redirects the user to the checkout page if the insertion is successful.

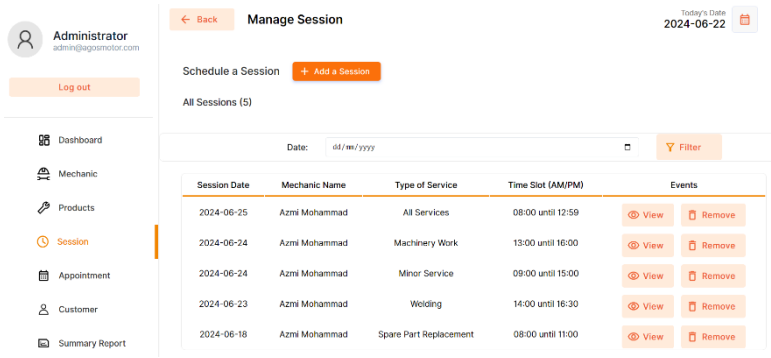

Fig. 19 *Manage Session for Admin's Interface*

Fig. 19 shows the manage session interface. In this page, a list of scheduled sessions will be displayed, and the admin can schedule a new session, view, filter by date, and delete existing session.

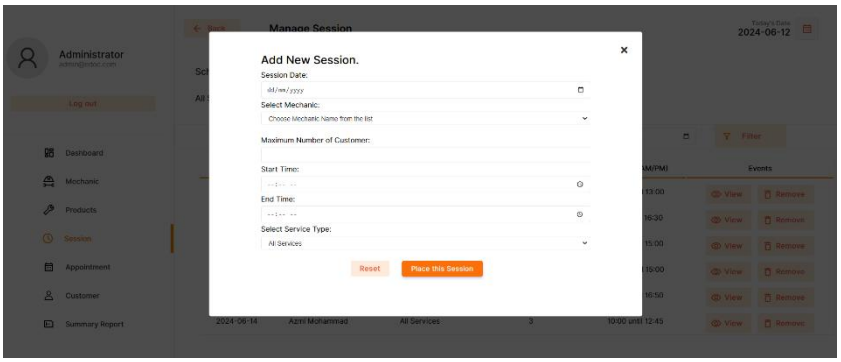

Fig. 20 *Add New Session Form for Admin's Interface*

A modal form will be displayed if the admin clicked on the Add a Session button, such as shown in Fig. 19. The admin is prompted to fill in details such as session date, mechanic to be assigned, maximum number of customers per session, session start time and end time, and service type, to schedule a new session.

```php
if($_POST){
    //import database
    include("../connection.php");

    $sessiondate=$_POST["sessiondate"];
    $maxnoc=$_POST["maxnoc"];
    $starttime=$_POST["starttime"];
    $endtime=$_POST["endtime"];
    $mechanicid=$_POST["mechanicid"];
    $spec=$_POST['spec'];
    $sql="insert into session (mechanicid,sessiondate,maxnoc,starttime,endtime, servicetype) values
    ('$mechanicid','$sessiondate','$maxnoc','$starttime','$endtime',  '$spec');";
    $result= $database->query($sql);
    header("location: session.php?action=session-added&sessionid=$sessionid");

}
```

Fig. 20 *Code Segment to Insert New Session into Database*

Upon clicking the Place this Session button on the add new session form, the system will insert the new session into the 'session' table such as shown in Fig. 20.

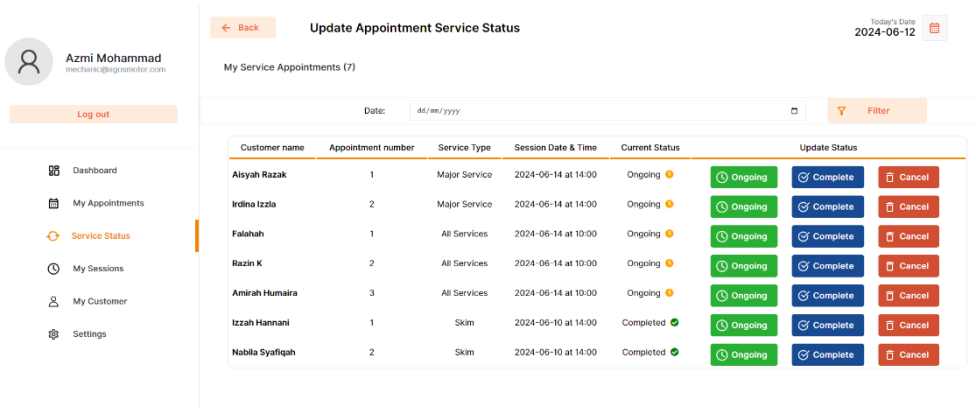

Fig. 21 *Update Service Status for Mechanic's Interface*

Fig. 21 shows the updated appointment service status interface for mechanic. A list of booked appointments will be displayed. The mechanic can update the appointment service status by clicking on the Ongoing, Complete, or Cancel button.

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $appoid = $_POST['appoid'];
    $status = 'Ongoing';
    $custname = $_POST['name'];
    $spcil_name = $_POST['session'];
    $apponum = $_POST['apponum'];

    // Update the session status
    $sql = "UPDATE appointment SET status = ? WHERE appoid = ?";
    $stmt = $database->prepare($sql);
    $stmt->bind_param("si", $status, $appoid);

    if ($stmt->execute()) {
        header("location: service-status.php?action=pending&appoid=$appoid&name=$custname&session=$spcil_name&apponum=$apponum");
    } else {
    }
```

Fig. 22 *Code Segment for 'Ongoing' status*

Fig. 22 shows the code segment for handling a POST request to update the status of an appointment to 'Ongoing' in the database using a prepared statement with query parameters indicating the updated appointment details.

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $appoid = $_POST['appoid'];
    $status = 'Completed';
    $custname = $_POST['name'];
    $spcil_name = $_POST['session'];
    $apponum = $_POST['apponum'];

    // Update the session status
    $sql = "UPDATE appointment SET status = ? WHERE appoid = ?";
    $stmt = $database->prepare($sql);
    $stmt->bind_param("si", $status, $appoid);

    if ($stmt->execute()) {
        header("location: service-status.php?action=complete&appoid=$appoid&name=$custname&session=$spcil_name&apponum=$apponum");
    } else {
        echo "Error updating service status: " . $stmt->error;
    }
```

Fig. 23 Code Segment for 'Completed' status

Fig. 23 shows the code segment for handling a POST request to update the status of an appointment to 'Completed' in the database using a prepared statement with query parameters indicating the updated appointment details.

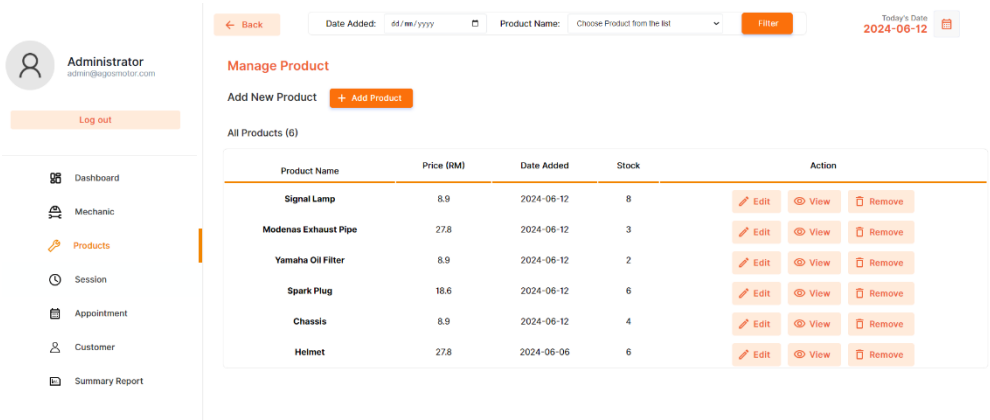

Fig. 24 *Manage Spare Part for Admin's Interface*

Fig. 24 shows the manage spare parts product interface for admin. A list of added spare part products with its' details such as product name, price, date added, and number of stocks are displayed. The admin can add a new spare part product by clicking on the Add Product button. Other actions such as edit, view, and delete products are also can be used.

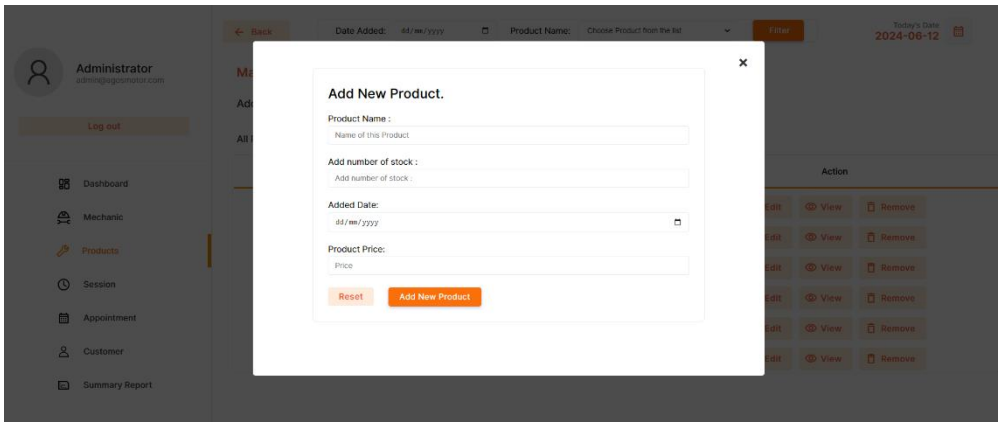

Fig. 25 *Add New Product for Admin's Interface*

Upon clicking the Add Product button, a form in a modal dialog will be displayed to retrieve the product details such as the product name, number of stocks, product added date, and product price, filled by the admin such as shown in Fig. 25.

```php
if($_POST){
    //import database
    include("../connection.php");
    $productid=$_POST["productid"];
    $title=$_POST["title"];
    $productdate=$_POST["productdate"];
    $productprice=$_POST["productprice"];
    $stock=$_POST["stock"];
    $sql="insert into product (title,productdate, productprice,stock)
    values ('$title','$productdate','$productprice','$stock')";
    $result= $database->query($sql);
    header("location: product.php?action=product-added&title=$title");

}
```

Fig. 26 *Code Segment to Add New Product for Admin*

The segment code such as shown in Fig. 26 shows the add new product function by handling a POST request to add a new product to the database into the 'product' table with a query parameter indicating the product was added.

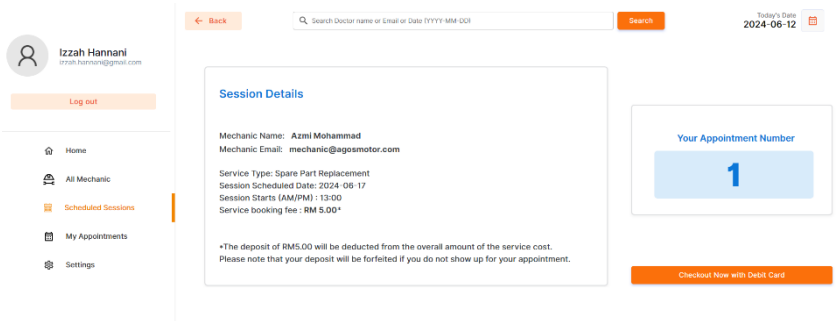

Fig. 27 *Book Service Appointment Page for Customer*

Fig. 27 shows the make payment and invoice interface. The make payment and invoice interface is the continuation from the book service appointment interface.

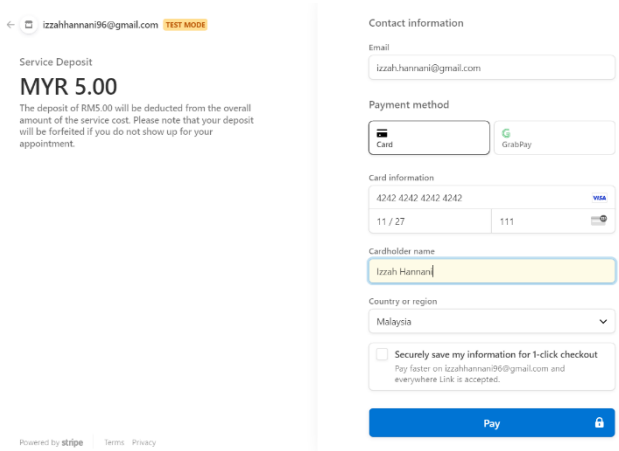

Fig. 28 *Make Payment for Customer Interface*

Upon clicking on the Checkout Now with Debit Card, the customer will be redirected to the payment gateway to pay for the service booking fee such as shown in Fig. 28.

```php
<?php

require_once 'vendor/autoload.php';
require_once 'secrets.php';

\Stripe\Stripe::setApiKey($stripeSecretKey);
header('Content-Type: application/json');

$YOUR_DOMAIN = 'http://localhost/prototype/';

$checkout_session = \Stripe\Checkout\Session::create([
  'line_items' => [[
    'price' => 'price_1PNTEACylju7J6owIMpoqPsG',
    'quantity' => 1,
  ]],
  'mode' => 'payment',
  'success_url' => $YOUR_DOMAIN . './customer/appointment.php',
  'cancel_url' => $YOUR_DOMAIN . './customer/booking.php',
]);
```

Fig. 29 *Code Segment for Integration of Stripe Payment Gateway in PHP*

The script, as shown in Fig. 29 integrates Stripe for processing payments. It starts by including the Stripe library and a secrets file containing the Stripe secret key, which it sets for authentication. The script then creates a Stripe Checkout session with specific line items (e.g., a product with a given price ID and quantity), sets the mode to 'payment'.

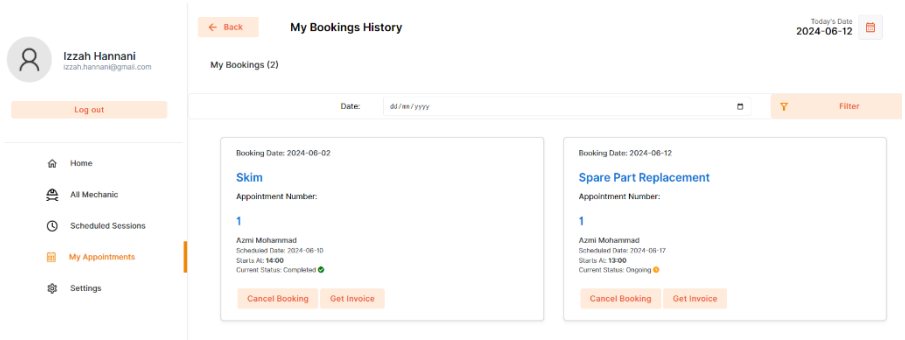

Fig. 30 *Appointment Page for Customer's Interface*

Upon clicking the Pay button on the make payment page, the customer will be redirected to the appointment page as shown in Fig. 30 that consists of the customer's booking history and upcoming booked appointments. In this page, the customer will be able to print or download the invoice to their local device in pdf format.

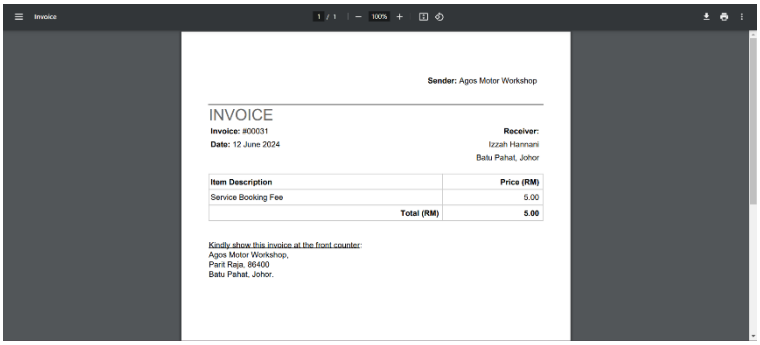

Fig. 31 *Print Customer's Invoice*

Upon clicking on the Get Invoice button on the appointment page, the system will display the invoice for the customer. Fig. 31 shows the invoice interface.

```php
<?php
// Include the TCPDF library
require_once('tcpdf/tcpdf.php');

// Create new PDF document
$pdf = new TCPDF(PDF_PAGE_ORIENTATION, PDF_UNIT, PDF_PAGE_FORMAT, true, 'UTF-8', false);
$pdf->SetCreator(PDF_CREATOR);
$pdf->SetAuthor('Agos Motor Workshop');
$pdf->SetTitle('Invoice');
$pdf->SetSubject('Invoice PDF');
$pdf->SetKeywords('TCPDF, PDF, invoice, receipt');
```

Fig. 32 *Code Segment to initialize TCPDF Library to Generate PDF Invoice*

The code segment shown in Fig. 32 initializes the TCPDF library to create a new PDF document. It sets various metadata for the PDF, including the creator, author, title, subject, and keywords, preparing the document to generate an invoice.

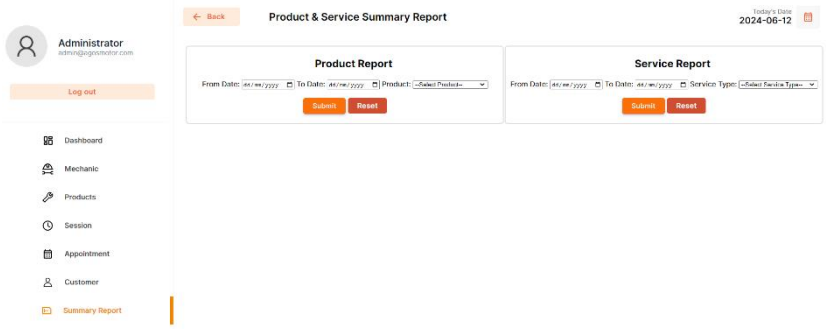

Fig. 33 *Generate Report for Admin's Interface*

Fig. 33 shows the generate report interface. Two types of report can be generated on this page, one of which is product report, and service report. The reports can be filtered by submitting a date range or by selecting a product or service type accordingly.

```php
// Define date range filter
$dateCond = '';
if (!empty($_GET['from']) && !empty($_GET['to'])) {
    $dateCond = "productdate >= '{$_GET['from']}' AND productdate <= '{$_GET['to']}'";
}

// Define product filter
$productCond = '';
if (!empty($_GET['product'])) {
    $productCond = "title='{$_GET['product']}'";
}

// Build SQL query for product report
$sqlProduct = "SELECT title as product, productdate as date, productprice as price, stock as stock FROM product WHERE 1=1";
if ($dateCond) $sqlProduct .= " AND $dateCond";
if ($productCond) $sqlProduct .= " AND $productCond";
$sqlProduct .= " ORDER BY productdate ASC";
$resultProduct = $database->query($sqlProduct);
$arrProduct = $resultProduct->fetch_all(MYSQLI_ASSOC);
```

Fig. 34 *Code Segment to Generate Product Report*

Fig. 34 shows the code segment to generate product report. The code segment defines filters for date range and product title based on GET parameters. It constructs an SQL query to retrieve product details (title, date, price, stock) from the database, applying the filters if provided, and orders the results by product date in ascending order. The query results are then fetched as an associative array.

```php
// Define date range filter for service
$dateCondService = '';
if (!empty($_GET['fromService']) && !empty($_GET['toService'])) {
    $dateCondService = "sessiondate BETWEEN '{$_GET['fromService']}' AND '{$_GET['toService']}'";
}

// Define service type filter
$serviceTypeCond = '';
if (!empty($_GET['serviceType'])) {
    $serviceTypeCond = "session.servicetype = '{$_GET['serviceType']}'";
}

// Build SQL query for service report
$sqlService = "SELECT servicetype.sname as service, session.sessiondate as date, GROUP_CONCAT(customer.custname SEPARATOR ', ') AS customers,
               COUNT(appointment.appoid) as total_appointments
                  FROM session
                  LEFT JOIN appointment ON session.sessionid = appointment.sessionid
                  JOIN servicetype ON session.servicetype = servicetype.id
                  JOIN customer on customer.custid=appointment.custid
                  WHERE 1=1";
if ($dateCondService) $sqlService .= " AND $dateCondService";
if ($serviceTypeCond) $sqlService .= " AND $serviceTypeCond";
$sqlService .= " GROUP BY session.sessionid, servicetype.sname ORDER BY session.sessiondate ASC, session.sessionid, servicetype.sname";
$resultService = $database->query($sqlService);
$arrService = $resultService->fetch_all(MYSQLI_ASSOC);
```

Fig. 35 *Code Segment to Generate Service Report*

The segment code such as shown in Fig. 35 implements the same method to generate a service report. The code segment shown is used to define filters for service date range and service type based on GET parameters. It constructs an SQL query to retrieve service session details, including service type, session date, customer names, total appointments, and maximum customers, applying the filters if provided. The results are grouped by session and service type, ordered by session date and ID, and fetched as an associative array.

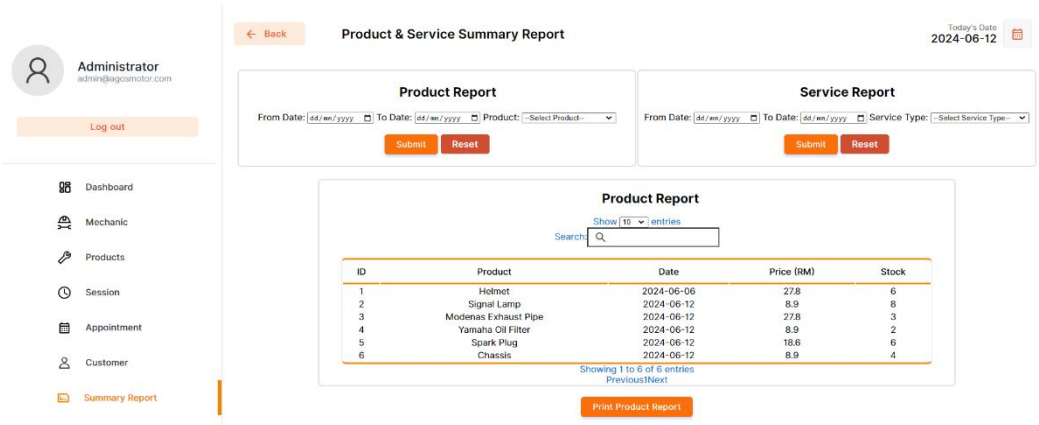

Fig. 36 *Product Report for Admin's Interface.*

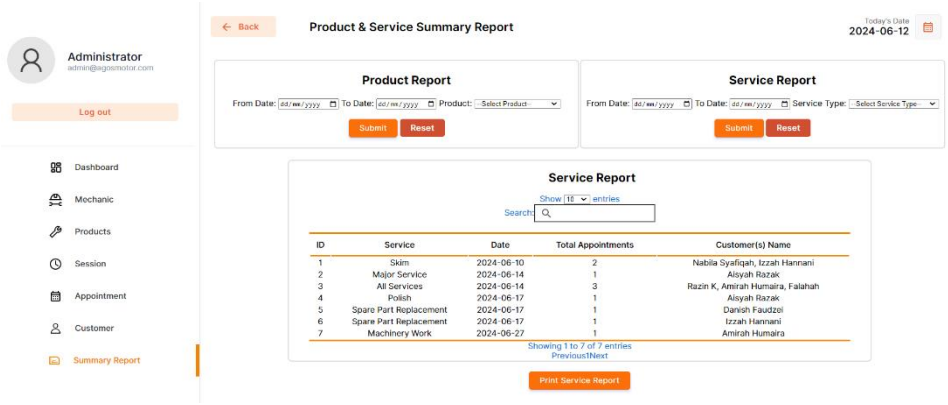

Fig. 37 *Service Report for Admin's Interface*

Both Fig. 36 and Fig. 37 show the product report and service report interface.

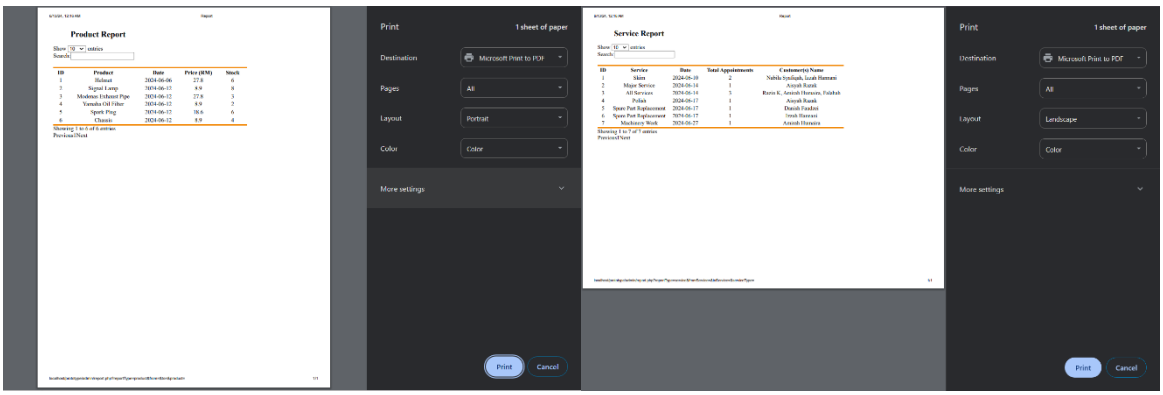

Fig. 38 *(a) Print Product Report Interface; (b) Print Service Report Interface*

The admin can print or save the product report and service report to their local device by clicking on the Print Product Report/Print Service Report button. The interfaces upon clicking on the button are as shown in Fig. 38.

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script src="https://cdn.datatables.net/1.10.25/js/jquery.dataTables.min.js"></script>
<script src="https://cdn.rawgit.com/jasonday/printThis/v1.15.0/printThis.js"></script>

<script>
    $(document).ready(function() {
        $('#productTable').DataTable();
        $('#serviceTable').DataTable();
    });

    function printReport(selector) {
        $(selector).printThis({
            importCSS: true,
            importStyle: true,
            loadCSS: ["../css/main.css", "../css/print.css"],
            pageTitle: "Report",
            removeInline: false,
            printDelay: 333
        });
    }
</script>
```

Fig. 39 *Code Segment to Print Reports*

The segment code shown in Fig. 39 is JavaScript code that includes jQuery, DataTables, and the printThis library for managing and printing tables. It initializes DataTables on elements with IDs productTable and serviceTable for enhanced table functionality and defines a printReport function to print the specified selector's content with custom CSS styles and options.

3.3.2 Database design

A database schema delineates the logical constraints that dictate the organization of data within a relational database. It encompasses details such as table names, fields, data types, and the relationships between these entities. Typically employing visual representations, schemas serve as the fundamental framework for data management practices within an organization. Developers employ a database schema to establish the necessary components and their connections. The database schema extracted from the Class Diagram is listed below.

- Customer (phoneNum, email, custName, custPW, custPointId)
- Appointment (appoId, custId, appoNum, sessionid, appoDate,status)
- Mechanic (mechanicid, mechanicpassword, mechanicemail,mechanictel,mechanicnic)
- Spare Part Product (productId, title, productdate, productprice, stock)

- Invoice (invoiceId, sessionId, invoicedate)
- Owner (aemail, apassword)

4. Result and Discussion

Tests were carried out in this section to assess the functionality of each module. User-Acceptance Testing (UAT) and Functional Testing are utilised to perform testing. User Acceptance Testing is collected through questionnaires that include questions about the system's features to the users. Meanwhile Functional Testing is employed to verify the system's adherence to its functional requirements and specifications. Each function will be evaluated against the corresponding requirements to ensure its output aligns with the end user's expectations. The list of test cases is shown in Table 4.

Table 4 *List of Test Cases*

No.	Test Cases	Description	Status (Pass/Fail)
		TEST_100 Register	
1.	TEST_100_01	The system shall display the register page.	Pass
2.	TEST_100_02	The system should be able to allow new customer to register into the system.	Pass
3.	TEST_100_03	The system shall alert the customer for invalid input.	Pass
4.	TEST_100_04	The system should redirect the customer to the login page after successful registration.	Pass
5.	TEST_100_05	The system shall not allow the customer to register if the information syntax is incorrect.	Pass
6.	TEST_100_06	While exception occurs, the system shall provide the options available to the customer.	Pass
		TEST_200 Login	
1.	TEST_200_01	The system shall display the login page.	Pass
2.	TEST_200_02	The system should allow existing users to login into the system by entering the required information.	Pass
3.	TEST_200_03	The system shall alert the users for invalid input.	Pass
4.	TEST_200_04	The system should redirect the users to the dashboard after successful login.	Pass
5.	TEST_200_05	The system shall not allow the customer to login if the information entered is incorrect.	Pass
		TEST_300 Display Dashboard	
1.	TEST_300_01	The system shall display the system's pages to all users.	Pass
2.	TEST_300_02	The system should redirect the customer, mechanic, and owner to the designated pages if any menu is clicked.	Pass
		TEST_400 Book Service Appointment	
1.	TEST_400_01	The system shall display the booking service appointment page.	Pass
2.	TEST_400_02	The system shall allow the customer to book a service appointment.	Pass
3.	TEST_400_03	The system shall redirect the customer to the payment page after booking an appointment.	Pass
		TEST_500 Manage Service Appointment	
1.	TEST_500_01	The system shall display the manage service appointment page to the mechanic and owner.	Pass
2.	TEST_500_02	The system should allow the mechanic to view the list of service appointments assigned to him.	Pass
3.	TEST_500_03	The system shall allow the owner to add service appointment slots.	Pass
		TEST_600 Update Service Status	
1.	TEST_600_01	The system shall display the service status to the customer and admin.	Pass
2.	TEST_600_02	The system should allow the mechanic to update service status.	Pass
		TEST_700 Manage Spare Parts	
1.	TEST_700_01	The system shall display the manage spare part page to the owner.	Pass
2.	TEST_700_02	The system shall allow the owner to manage spare part products.	Pass
3.	TEST_700_03	The system's database should save the changes in spare part details made by the owner.	Pass

Table 4 *(cont.)*

		TEST_800 Make Payment and Invoice	
1.	TEST_800_01	The system shall display the make payment page to customer.	Pass
2.	TEST_800_02	The system should display the charges.	Pass
3.	TEST_800_03	The system shall allow the customer to checkout by using debit card.	Pass
4.	TEST_800_04	The system shall direct the customer to the payment gateway within five minutes.	Pass
		TEST_900 Generate Report	
1.	TEST_900_01	The system shall display the summary report page to the owner.	Pass
2.	TEST_900_02	The system shall generate the summary report selected by the owner.	Pass
3.	TEST_900_03	The system shall allow the owner to print or save the report into the local device.	Pass

A total of 33 test cases had been conducted to test the booking and management system. The system has passed all 33 test cases successfully. Table 5 shows the overall results of the test cases.

Table 5 *Overall Result of Test Cases*

Test Case ID	Total Test Cases	Total Success	Total Fail
TEST_100	6	6	0
TEST_200	5	5	0
TEST_300	2	2	0
TEST_400	3	3	0
TEST_500	3	3	0
TEST_600	2	2	0
TEST_700	3	3	0
TEST_800	4	4	0
TEST_900	3	3	0
Total	**31**	**31**	**0**

5. Conclusion

The Agos Motor Workshop successfully achieved its primary objectives of designing and implementing a web-based booking and management system using object-oriented approaches. The system allows real-time access to booking information, improving efficiency in scheduling and customer service. However, several limitations were identified during user testing. These include integration issues with Google Calendar, manual input errors for spare parts, lack of service status notifications, and the need for enhanced security measures. Recommendations for future improvements include integrating with Google Calendar for appointment reminders, implementing barcode scanning for inventory accuracy, and enhancing login security with multi-factor authentication (MFA). These upgrades aim to address current system drawbacks and enhance functionality, ensuring better user experience and operational efficiency for the workshop.

Acknowledgement

The author would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The author confirms sole responsibility for the following: study conception and design, data collection, analysis and interpretation of results, and manuscript preparation.

References

[1] Statista. "Automotive Industry in Malaysia - Statistics & Facts." Statista. https://www.statista.com/topics/5040/automotive-industry-in-malaysia/#topicOverview (accessed Aug. 17, 2024)

[2] SEDCO. "What is an appointment booking system?" LinkedIn. https://www.linkedin.com/pulse/what-appointment-booking-system-systems-and-electronic-development (accessed Jul. 15, 2023)

[3] IBM. "What is inventory management and how does it work?" IBM. https://www.ibm.com/topics/inventory-management (accessed Jul. 15, 2023)

[4] Perodua. "UFirst Perodua." Perodua. https://www.ufirst.com.my/Member (accessed Dec. 13, 2023)

[5] SDAC Ford Malaysia. SDAC Ford Malaysia. https://www.sdacford.com.my/home (accessed Dec. 13, 2023)

[6] PROTON. (n.d). PROTON: Inspiring Connection. PROTON. https://www.proton.com/en (accessed Dec. 13, 2023]