
# GROUP ASSIGNMENT

**TECHNOLOGY PARK MALAYSIA**

**CT038–3–2-OODJ-L**

**OBJECT ORIENTED DEVELOPMENT WITH JAVA**

**UC2F1910SE**

**HAND OUT DATE: 04th DECEMBER 2019**
**HAND IN DATE: 17th FEBRUARY 2020**

**WEIGHTAGE: 50%**

---

**INSTURCTION TO CANDIDATES:**

1. **Submit your assignment at the administrative counter**
2. **Student are advised to underpin their answer with the use of references (cited using the Harvard Name System of Referencing)**
3. **Late submission will be awarded zero (0) unless Extenuating Circumstance (EC) are upheld**
4. **Cases of plagiarism will be penalized**
5. **The assignment should be bound in an appropriate style (comb bound or stapled)**
6. **Where the assignment should be submitted in both hardcopy and softcopy, the softcopy of the written assignment and source code (where appropriate) should be on a CD in an envelope / CD cover and attached to the hardcopy.**
7. **You must obtain 50% overall to pass this module.**


# GROUP ASSIGNMENT

**TECHNOLOGY PARK MALAYSIA**

**CT038–3–2-OODJ-L**

**OBJECT ORIENTED DEVELOPMENT WITH JAVA**

**UC2F1910SE**

**HAND OUT DATE: 04th DECEMBER 2019**
**HAND IN DATE: 17th FEBRUARY 2020**

**WEIGHTAGE: 50%**

---

| TP number | Name |
|---|---|
| TP 047472 | Hen Kian Jun |
| TP 048545 | Ng Zheng Jue |

# Table of Contents

# 1.0 Introduction

## 1.1 Brief Introduction

In this assignment, is to design and implement APU Automotive Service Centre (AASC) system. The proposed system is used by centre managers to handle end user registration and help the potential customers to register and book for car servicing appointments. Besides that, the proposed system is also for the centre technicians to check their own individual appointments and to collect payment and provide feedbacks at the end of each appointment. In addition, the proposed system is designed and developed using object-oriented approach covering object orientation concepts and principles which are Abstraction, Encapsulation, Inheritance, and Polymorphism.

## 1.2 Project Assumption

APU Automotive Service Centre (AASC) is a small and medium company, three roles will be included in the proposed system which are Administrator, Centre Manager, and Centre Technician.

Both manager and technician can manage category for the potential customers to make an appointment and manage product for the potential customers to buy if needed. The whole process of making reservation can only be done by technician and manager is responsibility to view only. Administrator has the right clean up the whole database by clicking on reset button.

When the potential customer registers into the proposed system, the potential customer will receive an email regards the registration of APU Automotive Service Centre (AASC) membership. Each potential customer can book only one category at a time. Each potential customer can have only one appointment at one time. When proceeding to the payment, customer can buy a single product with multiple quantity. The product price will be charged accordingly. Each appointment will charge customer extra 6% for service tax. The potential customer is required to provide feedback upon the payment is made. Moreover, an E-receipt will be sent through customer's email based on customer requests.

One technician will have multiple appointments, but the appointment date and time cannot be the exactly same from one technician to another.

## 1.3 Project Scope

This assignment is to build a completely new system for APU Automotive Service Centre (AASC). Allowing the system user to log into the proposed system and help the potential customers to make an appointment for car servicing. Each potential customer will have only one appointment at one time, cannot more than two. When making an appointment for customer, one category must be chosen for the appointment, and along with one technician. The system user is required to check whether the chosen technician is available for the appointment, to prevent one technician is having two exactly the same appointment date and time. When proceeding to the payment, the potential customer can choose the payment method to pay the fees, required to provide a feedback for the further improvement after each appointment.

## 1.4 Aim and Objectives

A completely new booking management system will be developed for APU Automotive Service Centre (AASC) to strengthen the current business processes.

1. To design the use case of the proposed system and provide specification for each use case
2. To illustrate relationship between all class diagrams
3. To explain the object-oriented approach in detailed and how object-oriented approach applies into the proposed system
4. To include user manual with screenshot and detailed explanation

# 2.0 Use Case Diagram

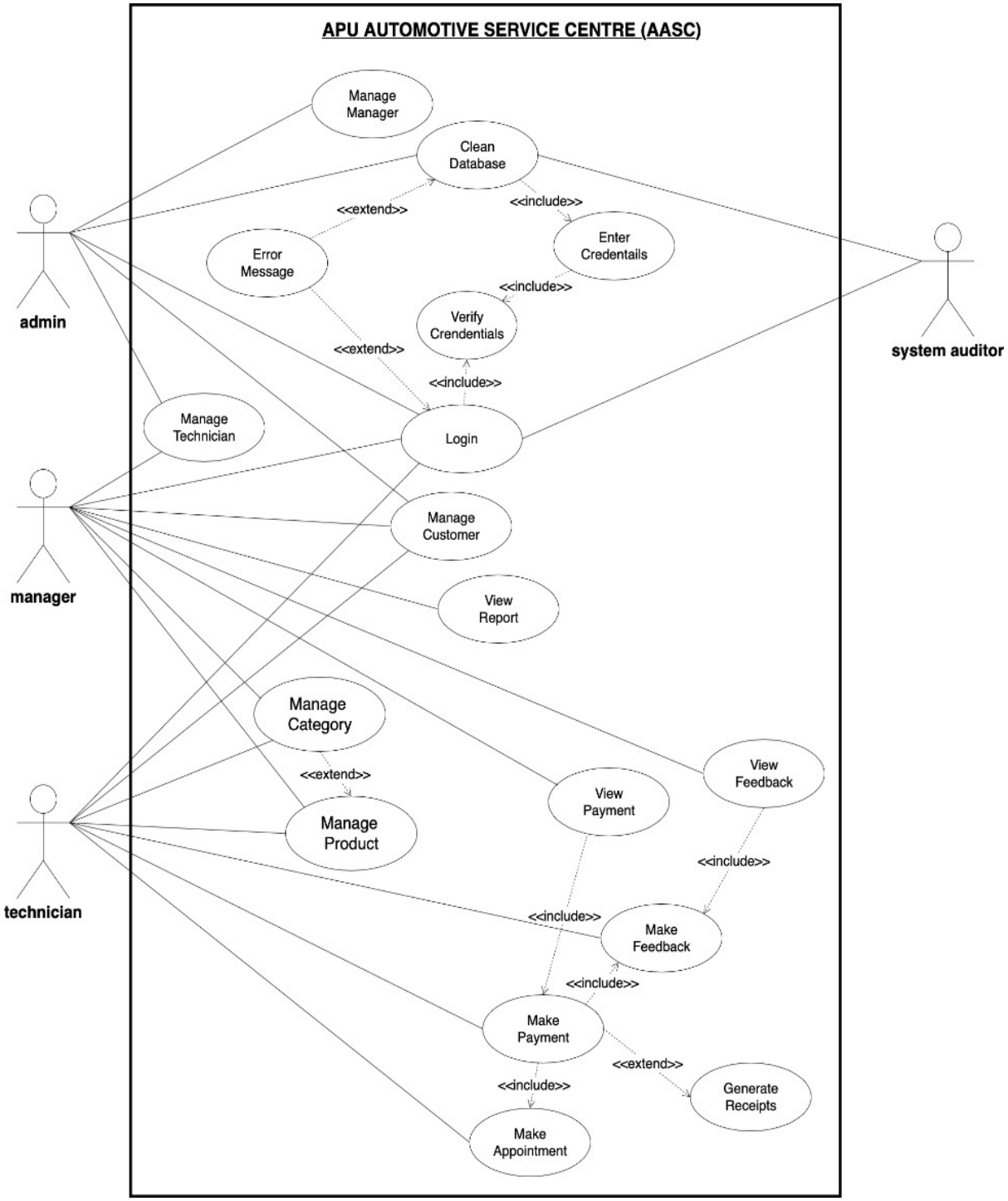



*Figure 1: Use Case Diagram*

## 2.1 Use Case Specification

| **Use Case** | Login |
|---|---|
| **Description** | The user(s) is able to login to the proposed system based on the login credentials |
| **Actor(s)** | Admin, Manager, Technician |
| **Precondition** | The user must be logged out |
| **Basic Flow** | 1. Key in the correct username and password<br>2. The proposed system is able to detect the role automatically<br>3. Logged into the proposed system if the user keyed in the correct login credentials |

| **Use Case** | Manage Customer |
|---|---|
| **Description** | Both manger and technician are able to create, read, and update customers. Admin is able to delete, activate or deactivate customer |
| **Actor(s)** | Admin, Manager, Technician |
| **Precondition** | The user must be logged in |
| **Basic Flow** | 1. Log into the proposed system<br>2. Click on Manage Customer on left navigation tab<br>3. Click on ‘Create’ button to add a new customer<br>4. Click on ‘View’ button to view all customer details |

| **Use Case** | Clean Database |
|---|---|
| **Description** | Admin is able to clean the database which to clear all the tables to be empty |
| **Actor(s)** | Admin |
| **Precondition** | The user must be logged in, and provide admin credentials when cleaning the database |
| **Basic Flow** | 1. Login to the proposed system<br>2. Click on CLEAN button<br>3. Provide admin credentials |

| **Use Case** | Manage Category |
|---|---|
| **Description** | Both manager and technician are able to create, read, update, and activate or deactivate the category |
| **Actor(s)** | Manager, Technician |
| **Precondition** | The user must be logged in |
| **Basic Flow** | 1. Login to the proposed system<br>2. Click on Manage Category on left navigation bar<br>3. Keyed in the category details and click on 'Add' button to create a category<br>4. Click on 'Edit' button to update the category details |

| **Use Case** | Manage Product |
|---|---|
| **Description** | Both manager and technician are able to create, read, update, and activate or deactivate the product |
| **Actor(s)** | Manager, Technician |
| **Precondition** | The user must be logged in and a category must be created before the product is created, because it requires to choose one category for the product |
| **Basic Flow** | 1. Login to the proposed system<br>2. Click on Manage Product on left navigation bar<br>3. Key in the product details and select one category<br>4. Click on 'Add' button to create a product<br>5. Click on 'Edit' button to update the product details |

| **Use Case** | Make Appointment |
|---|---|
| **Description** | Technician is able to create, read, update an appointment for any customer |
| **Actor(s)** | Technician |
| **Precondition** | The user must be logged in and customer name, category name, and technician name must be chosen before the appointment is made. The user must check the availability time of the chosen technician |
| **Basic Flow** | 1. Login to the proposed system<br>2. Click on Manage Appointment on left navigation bar<br>3. Key in the appointment details<br>4. Select one customer<br>5. Select one category<br>6. Select one technician<br>7. Click on 'Check' button next to the technician combo box<br>8. Click on 'Check' button to create an appointment only if the check keyword is appear |

| **Use Case** | Make Payment |
|---|---|
| **Description** | Technician is able to help the potential customer to complete appointment by making the payment |
| **Actor(s)** | Technician |
| **Precondition** | The user must be logged in and an appointment must be created before proceed to the payment |
| **Basic Flow** | 1. Login to the proposed system<br>2. Click on Manage Payment on left navigation bar<br>3. Select one appointment<br>4. Select one product if needed<br>5. Key in the quantity if the potential customer wants to buy the product<br>6. Click on 'Calculate' button to check the latest amount<br>7. Click on 'Proceed' button to complete the payment process |

| **Use Case** | Make Feedback |
|---|---|
| **Description** | Technician is able to create a feedback |
| **Actor(s)** | Technician |
| **Precondition** | The user must be logged in and the customer must be paying the fees for the category and/or product |
| **Basic Flow** | 1. Login to the proposed system<br>2. Make an appointment for a customer<br>3. Complete the appointment by paying the fees<br>4. Choose the payment method and collect the money<br>5. Provide the feedback for the improvement<br>6. Click on 'Submit' to create a feedback |

| **Use Case** | View Feedback |
|---|---|
| **Description** | Manager is able to view a feedback in a bar chart or table form |
| **Actor(s)** | Manager |
| **Precondition** | The user must be logged in and at least payment must be done |
| **Basic Flow** | 1. Login to the proposed system<br>2. Click on View Feedback on left navigation bar<br>3. Click on Category name from combo box to view the bar charts respectively<br>4. Click on 'View' button to view all the feedbacks in a table form |

| **Use Case** | View Payment |
|---|---|
| **Description** | Manager is able to view the payment history in an area chart or table form |
| **Actor(s)** | Manager |
| **Precondition** | The user must be logged in and at least one payment must be done |
| **Basic Flow** | 1. Login to the proposed system<br>2. Click on View Payment on left navigation bar<br>3. Click on 'Past Three Days' from combo box to view the past three days transaction<br>4. Click on 'Past Seven Days' from combo box to view the past seven days transaction<br>5. Click on 'View' button to view all the transactions in a table form |

| **Use Case** | Manage Technician |
|---|---|
| **Description** | Admin is able to create, read, update, and activate or deactivate technician |
| **Actor(s)** | Admin, Manager |
| **Precondition** | The user must be logged in |
| **Basic Flow** | 1. Login to the proposed system<br>2. Click on Manage User on left navigation bar. For manager, click on Manage Technician on left navigation bar<br>3. Select a row and click on 'Update' button to do update<br>4. Click on 'Add' button to create a new user |

| **Use Case** | Manage Manager |
|---|---|
| **Description** | Admin is able to create, read, update, and activate or deactivate manager |
| **Actor(s)** | Admin |
| **Precondition** | The user must be logged in |
| **Basic Flow** | 1. Login to the proposed system<br>2. Click on Manage User on left navigation bar<br>3. Select a row and click on 'Update' button to do update<br>4. Click on 'Add' button to create a new user |

## 2.2 Class Diagram

In this section, all attributes and methods of each class will be separated as a single diagram cannot be read clearly. The figure below shown is the relationships amongst all the classes. The following pages will be showing all the attributes and methods of each class.

The 'manage' keyword represents create, read, update, delete, activate or deactivate.

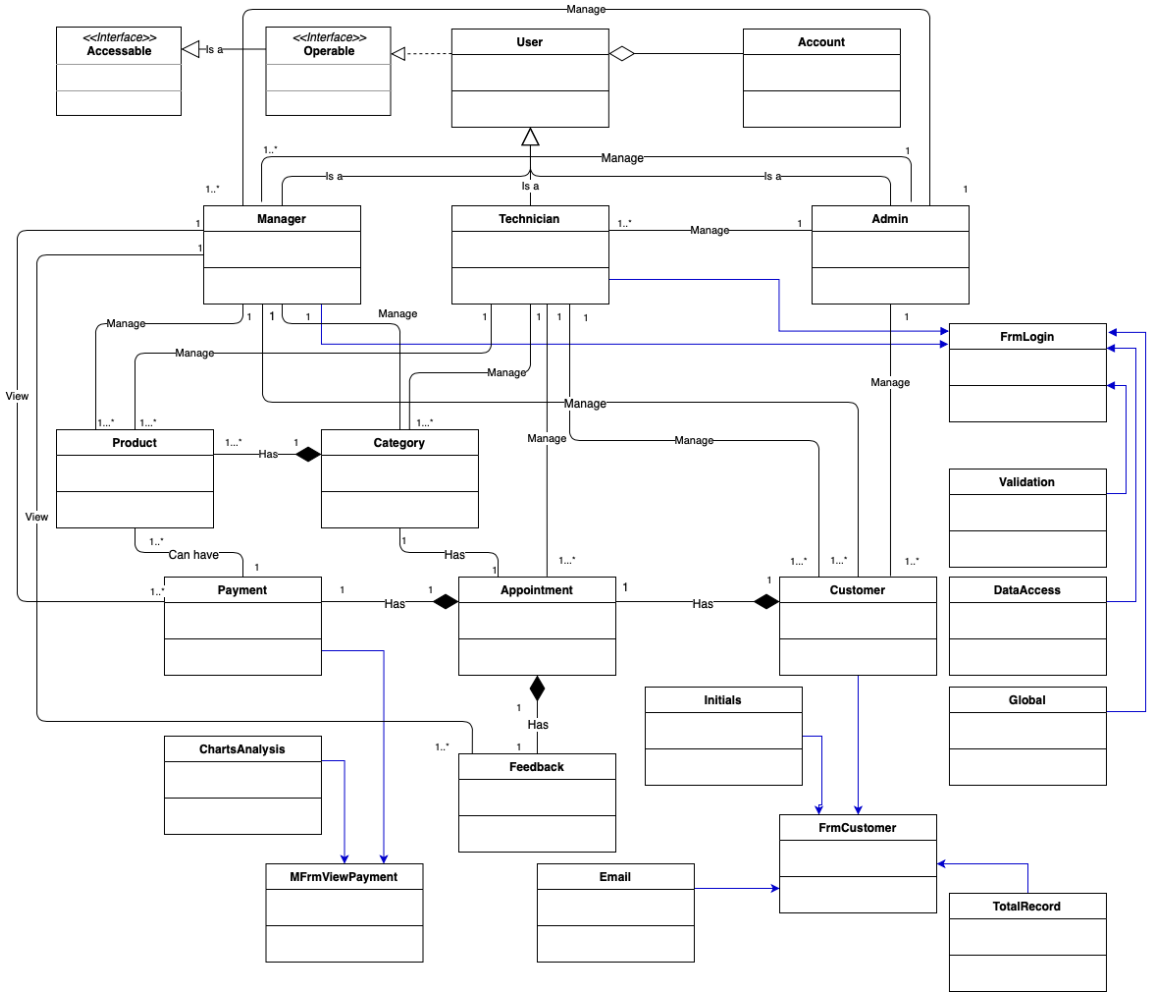



*Figure 2: Class Diagram*

### 2.2.1 User Class

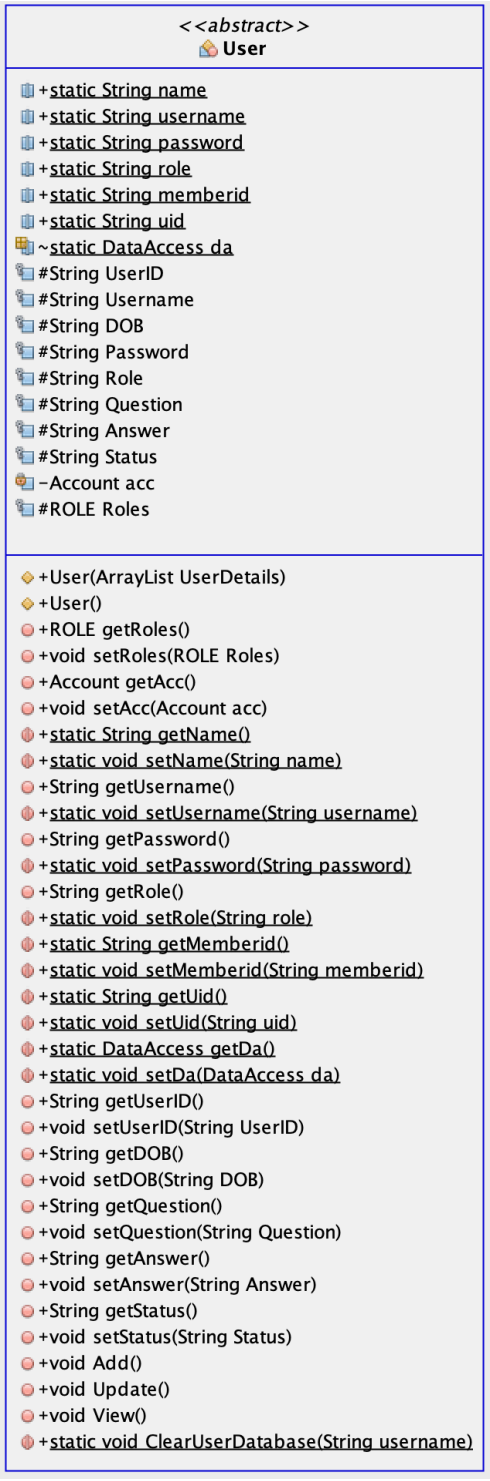



*Figure 3: User Class*

### 2.2.2 Account Class

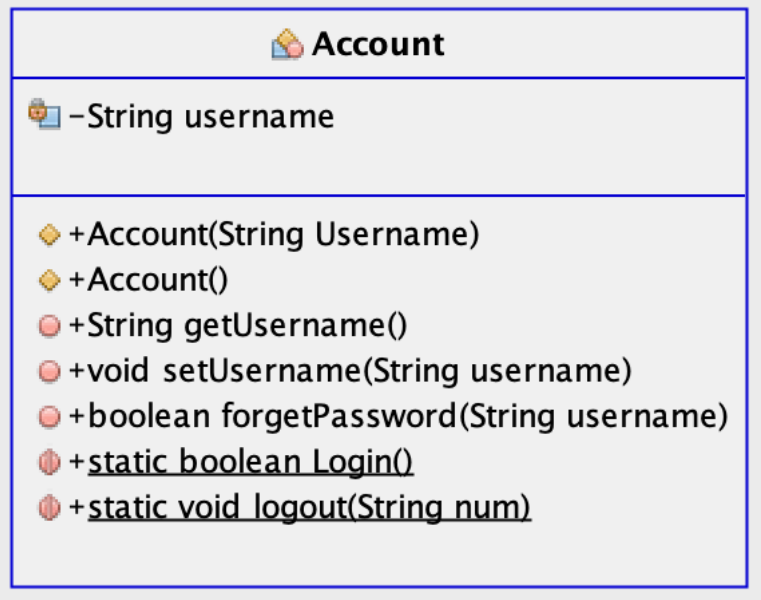



*Figure 4: Account Class*

### 2.2.3 Admin Class

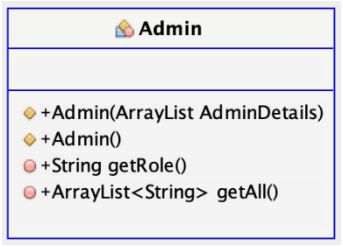



*Figure 5: Admin Class*

### 2.2.4 Manager Class

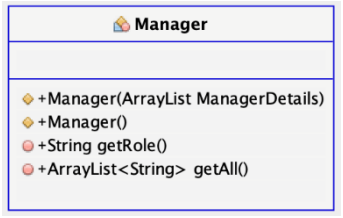



*Figure 6: Manager Class*

### 2.2.5 Technician Class

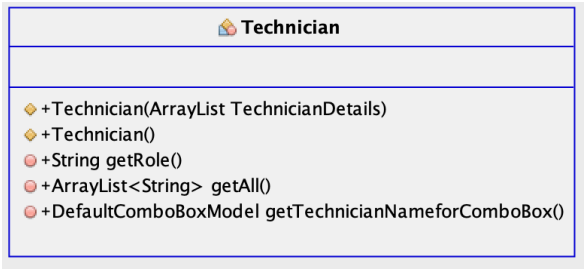



*Figure 7: Technician Class*

### 2.2.6 Category Class

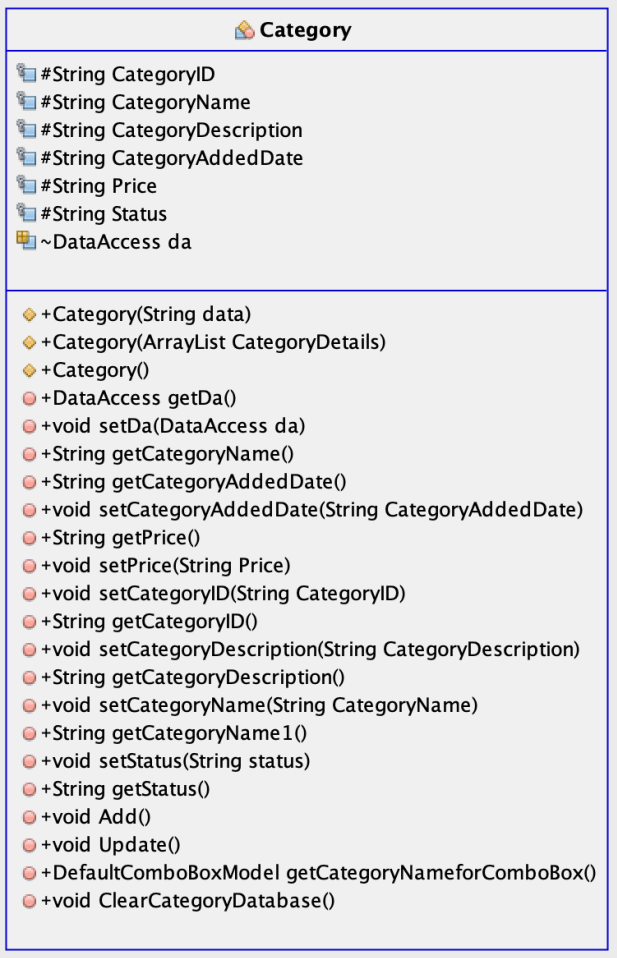



*Figure 8: Category Class*

### 2.2.7 Product Class

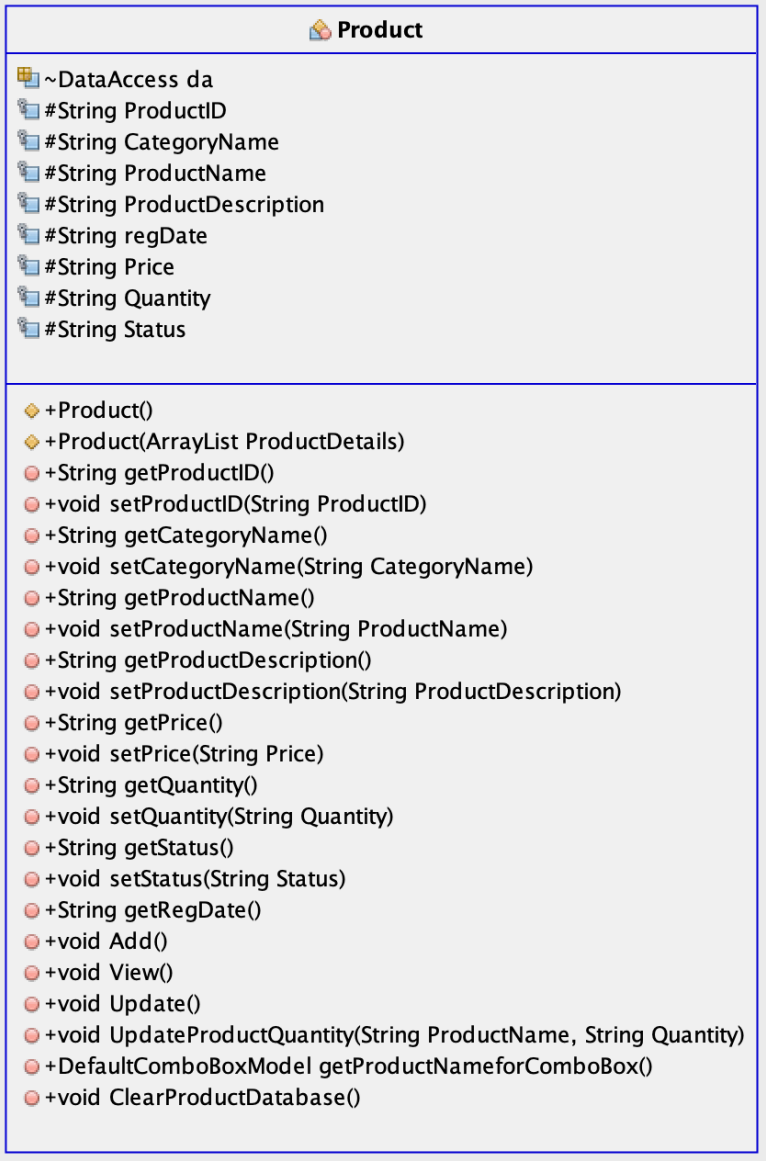



*Figure 9: Product Class*

### 2.2.8 Operable Class

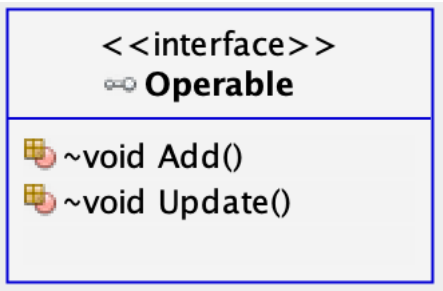



*Figure 10: Operable Class*

### 2.2.9 Accessable Class

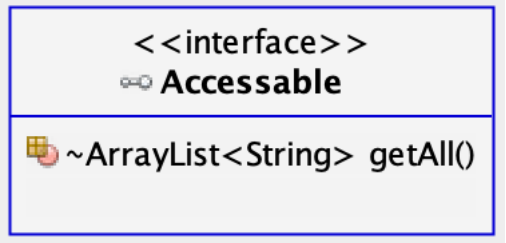



*Figure 11: Accessable Class*

### 2.2.10 ChartsAnalysis Class

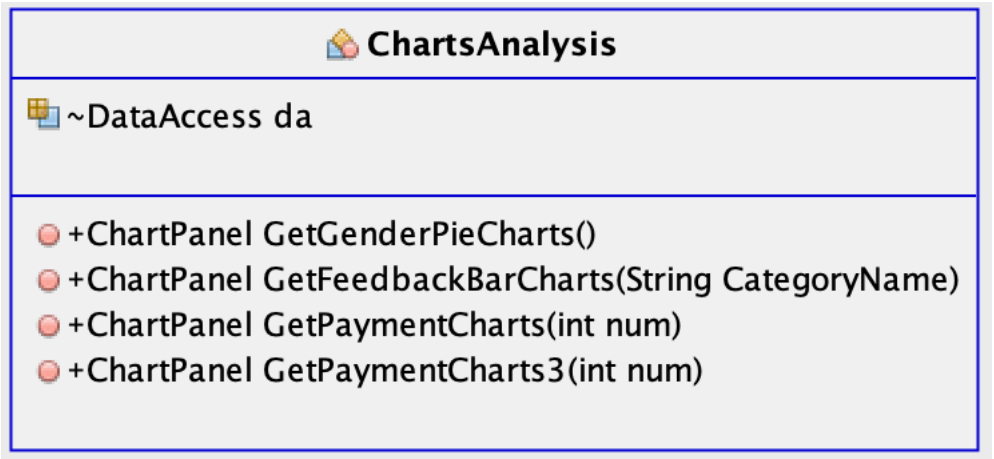



*Figure 12: ChartsAnalysis Class*

### 2.2.11 Initials Class

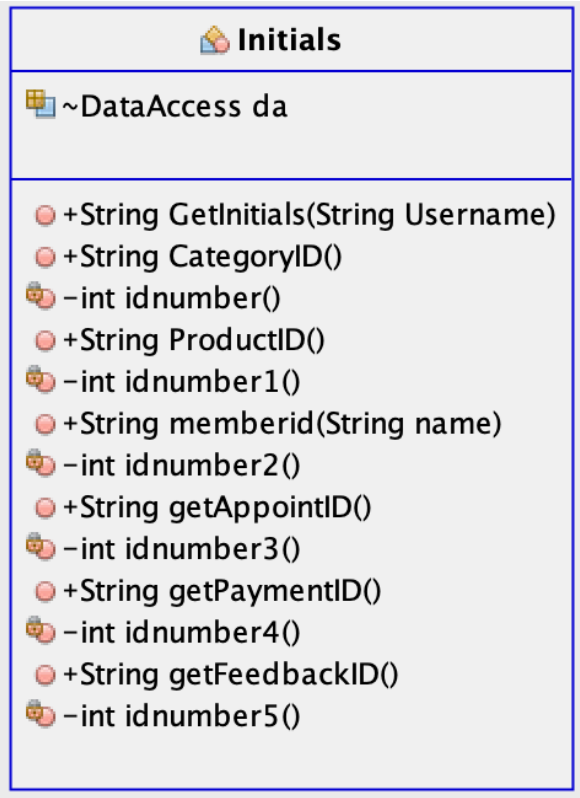



*Figure 13: Initials Class*

### 2.2.12 TotalRecords Class

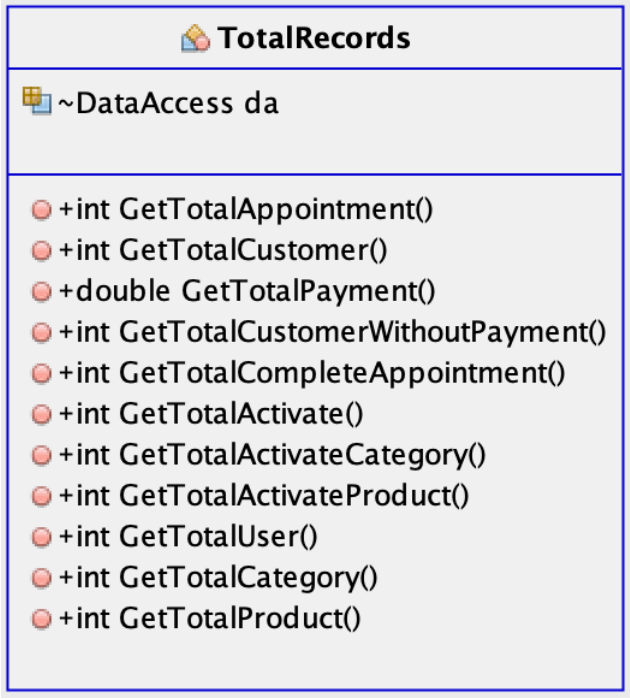



*Figure 14: TotalRecords Class*

### 2.2.13 Validation Class

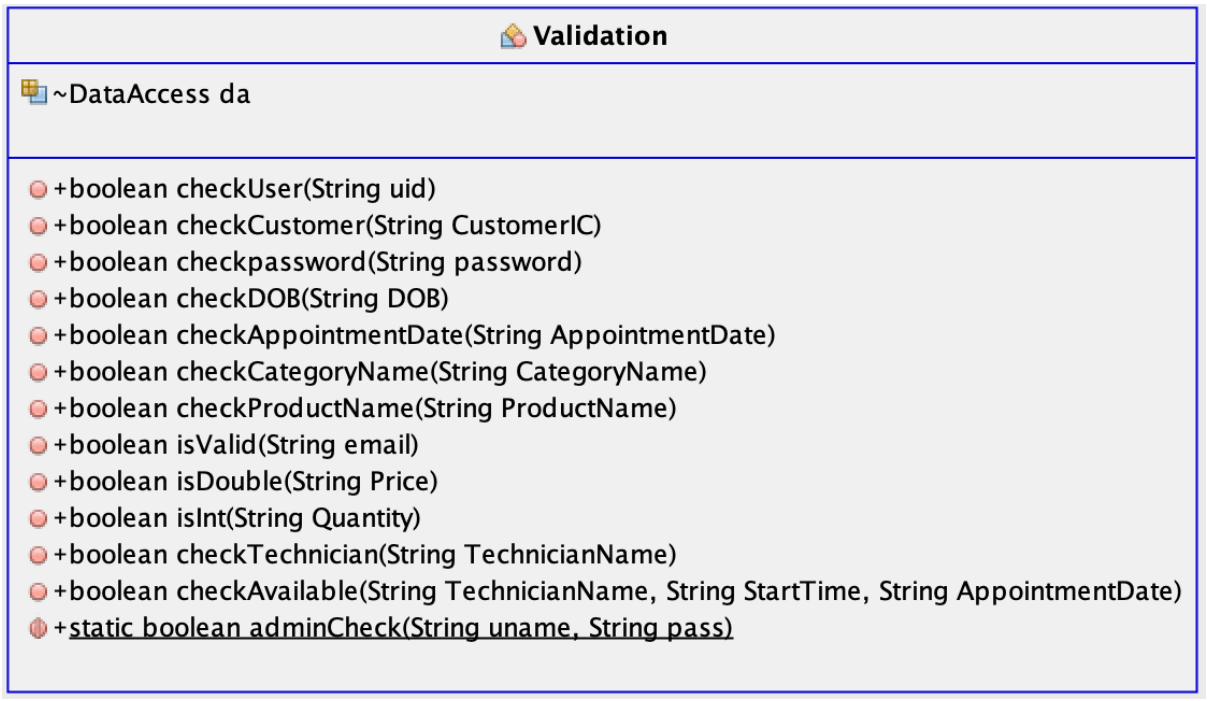



*Figure 15: Validation Class*

### 2.2.14 Appointment Class

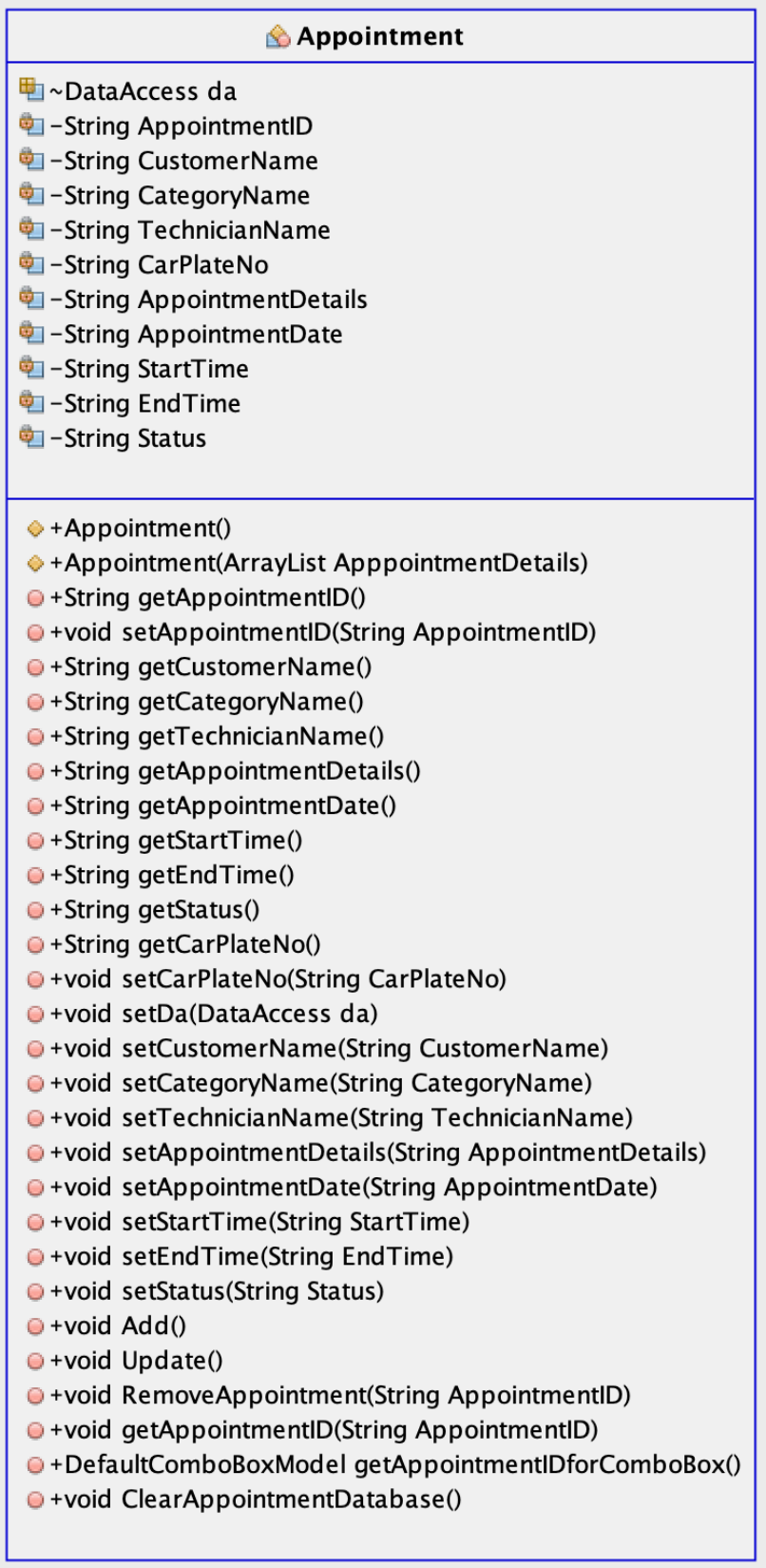



*Figure 16: Appointment Class*

### 2.2.15 Customer Class

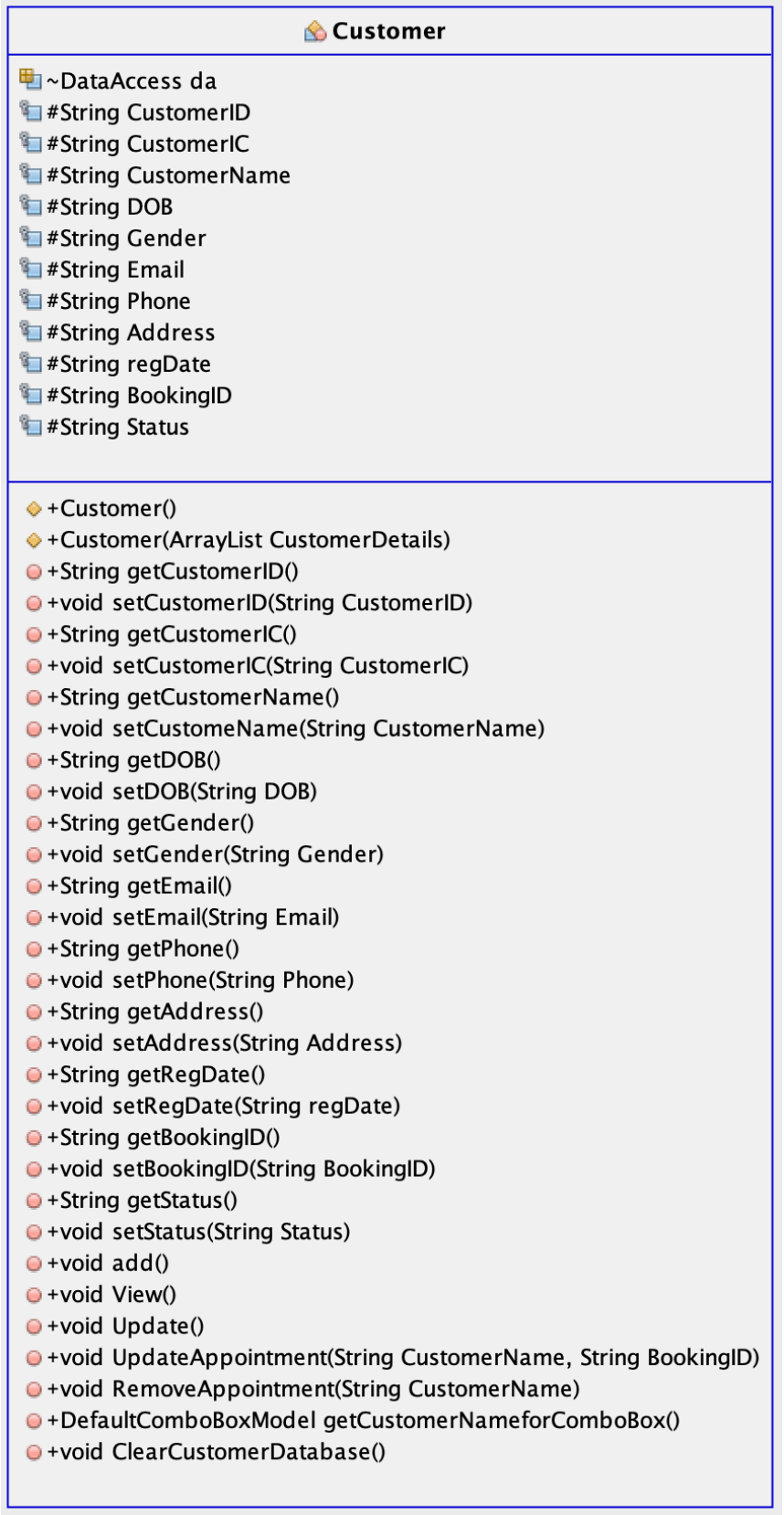



*Figure 17: Customer Class*

### 2.2.16 Feedback Class

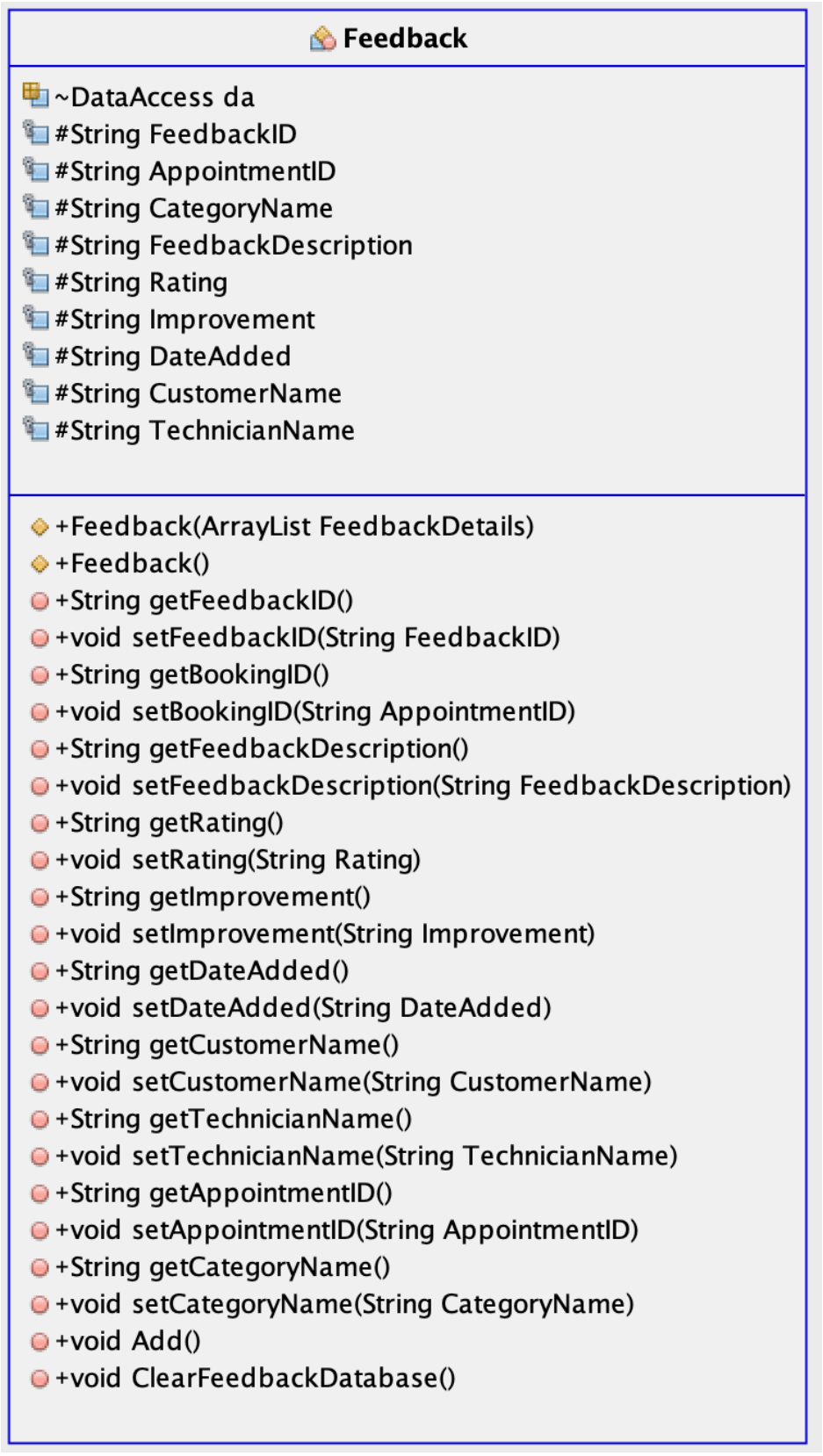



*Figure 18: Feedback Class*

### 2.2.17 Payment Class

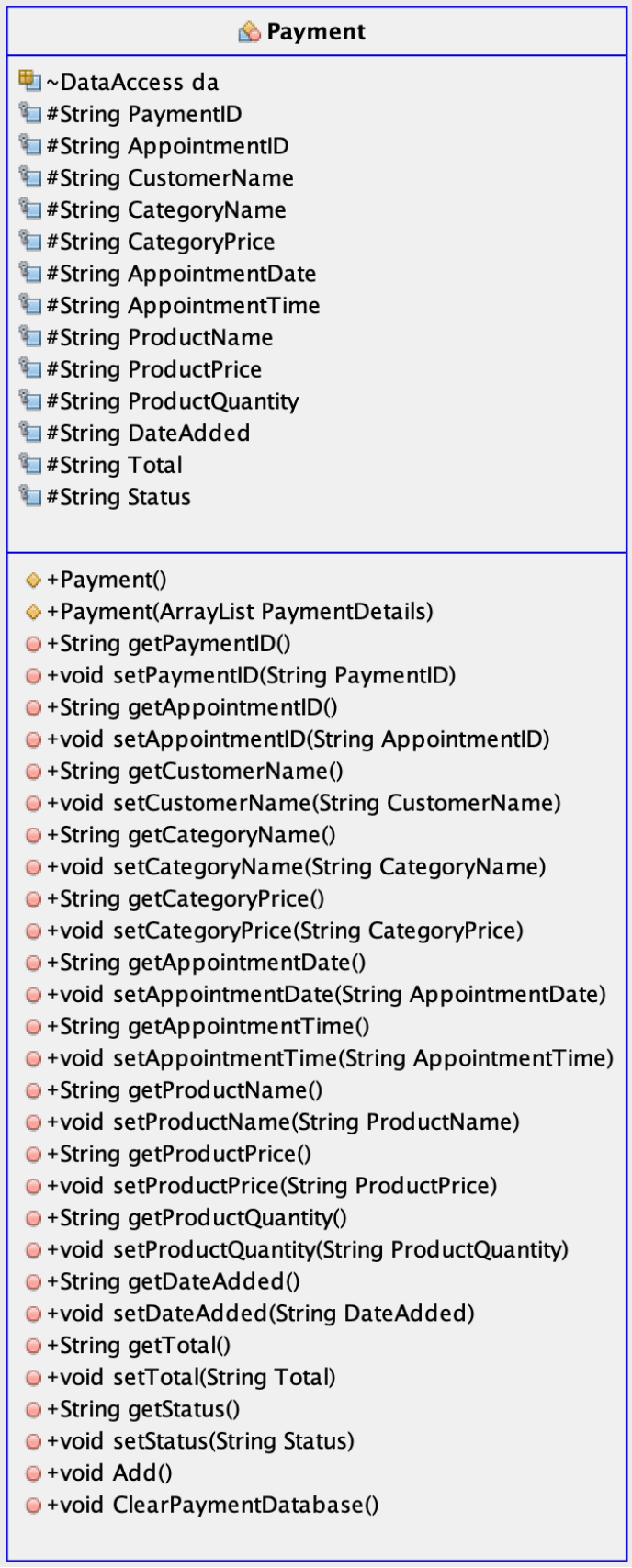



*Figure 19: Payment Class*

# 3.0 Object-Oriented Concepts

Object-oriented concepts are actually the basic main ideas behind the Object-Oriented Programming. Four concepts, Abstraction, Encapsulation, Inheritance, and Polymorphism, are the principle of Object-Oriented concepts (Petkov, 2018).

## 3.1 Abstraction

Abstraction is one of the key concepts of object-oriented programming (OOP) languages. The main goal of Abstraction is to handle the complexity by hiding useless information from the user (Petkov, 2018). Abstraction can be deemed as natural extension of encapsulation, it is only to reveal operations relevant for the other objects. More complex logic on top of the provided Abstraction for the user without understanding or even thinking what the hidden complexity are. Abstraction is a very generic concepts which is not limited to object-oriented programming (Janssen, 2017).

The process of abstraction that hides all the internal implementation details but only shows the essential information from/to the user. Abstraction class can be achieved with two ways, which are Abstraction classes or Interface. Abstract keyword represents a non-access modifier, which can be used for class or method (w3schools.com, 2020). The user would need to know which methods of the objects are available to be called and which input parameters are required to trigger a specific operation. It does not need to understand how the method is implemented and what kinds of operations will be performed to create the expected result (Janssen, 2017).

If the class is set as Abstraction, it means the class is a restricted class, which will cause an error saying that the class cannot be used to instantiate object as the class is Abstraction class. To access the Abstraction class, it would need to instantiate from the child classes (w3schools.com, 2020). Polymorphism concepts will be applied to help to instantiate the class.

Below (Figure 20) is one of the example of Abstraction class that has been implemented into the proposed system. The User class has been set as abstract which means User class cannot be called or instantiated by the object. However, it can be only accessed by its child class by inherited User class.

```
public abstract class User implements Operable {
    public static String name;
    public static String username;
    public static String password;
    public static String role;
    public static String memberid;
    public static String uid;
    static DataAccess da = new DataAccess();

    protected String UserID;
    protected String Username;
    protected String DOB;
    protected String Password;
    protected String Role;
    protected String Question;
    protected String Answer;
    protected String Status;
    private Account acc;
    enum ROLE {
        ADMIN {
            @Override
            public String toString() {
                return "ADMIN";
            }
        },
        MANAGER {
            @Override
            public String toString() {
                return "MANAGER";
            }
        },
        TECHNICIAN {
            @Override
            public String toString() {
                return "TECHNICIAN";
            }
        }
    }
```

*Figure 20: Abstract Class - User*

## 3.2 Encapsulation

Encapsulation is one of the fundamental concepts in object-oriented programming, it describes an idea of bundling all the attributes and methods that work on the attributes within one unit, such as Java class. Encapsulation, as known as information hiding, concept is usually used to hide the internal attributes of an object from outside class. The general idea of this mechanism is very simple. If an attribute is not visible from outside of an object, and the attribute is bundled with methods that provides read or write access to it, so that the specific information can be hidden and control access to the internal attribute of the object (Janssen, 2017). When applying to object-oriented programming language, Getter and Setter methods are mainly involved. An attribute can be read and changed, or if it is invisible at all, or if it is read-only. The purpose of encapsulation is to prevent the variables to be direct accessing or obtaining data by the object. When object instantiated the class, a list of public functions is visible to the object only, because all the attributes are set as private which cannot be accessed by the object. Applying encapsulation, it can use certain methods that are provided, but the attribute of the methods are unchangeable which is fixed and controlled only by its class (Petkov, 2018). Access Modifiers include private, no modifier, protected, and public.

Below (Figure 21) is the one of the example of encapsulation concepts being implemented into the proposed system. In Category Java class, all the attributes are set to be private which is already being encapsulated, which whenever the Category Java class is being called by the object, the object is not visible to object because every attribute in Category Java class has been set the privilege. This is to protect the information that is hold by the attributes.

```
public class Category {
    private String CategoryID;
    private String CategoryName;
    private String CategoryDescription;
    private String CategoryAddedDate;
    private String Price;
    private String Status;
```

*Figure 21: Private Attributes of Category Class*

There are two methods to be included in the encapsulation concepts, that are used for object to use the internal variable, which are Getter and Setter.

In order to access the private variables, Getter and Setter are mainly used for accessing the private variables. In the proposed system, getter methods are applied in all the classes, such as Appointment, Category, Customer, Feedback, Payment, Product, and User, which can be used to show information to the system user in some functionalities such as updating an existing record or adding a new record into the table.

```
public String getCategoryAddedDate() {
    return CategoryAddedDate;
}

public void setCategoryAddedDate(String CategoryAddedDate) {
    this.CategoryAddedDate = CategoryAddedDate;
}

public String getPrice() {
    return Price;
}

public void setPrice(String Price) {
    this.Price = Price;
}
```

*Figure 22: Getter and Setter Methods of Category Class*

Since all the variables in Category class are private, it is not highly recommended to use the private variable directly, Getter method is highly suggested to obtain the data as when the Getter method is being call, the same data will be returned.

```
Category category = new Category();
category.setCategoryID(CategoryID);
category.setCategoryAddedDate(regDate);
category.setCategoryDescription(CategoryDescription);
category.setCategoryName(CategoryName);
category.setPrice(Price);
category.setStatus(Status);
```

*Figure 24: Ways to use Setter Method*

```
public void Add() {
    String line = String.join ("|", getCategoryID(), getCategoryName(), getCategoryDescription(), getCategoryAddedDate(), getPrice(), getStatus());
    da.write(line);
    da.setFileName("Category_activities.txt");
    line = String.join ("|", Global.name , getCategoryName(), "Added");
    da.write(line);
}
```

*Figure 23: Ways to use Getter Method*

## 3.3 Inheritance

In Java, inherit a method or attribute is possible to happen from one class to another. Inheritance has categorized two classes, which are superclass, as known as parent class, and sub class, as known as child class. Superclass is a class that is being inherited by its child class, whereas child class is a class that inherits from its parent class. Extend keyword will be used to inherit a class (w3schools.com, 2020). The advantages of applying inheritance concept is to reuse the code, and the inherited class (child class) can develop with unique features and the rest of the common properties and functionalities can be hold from parent class, to minimize the amount of duplicated code in an application by sharing the command code. Inheritance is to reuse the common logic and extract the unique logic into a separate class (Bhatia, 2017). Inheritance therefore can lead to a better organisation of the code smaller, simpler complication units.

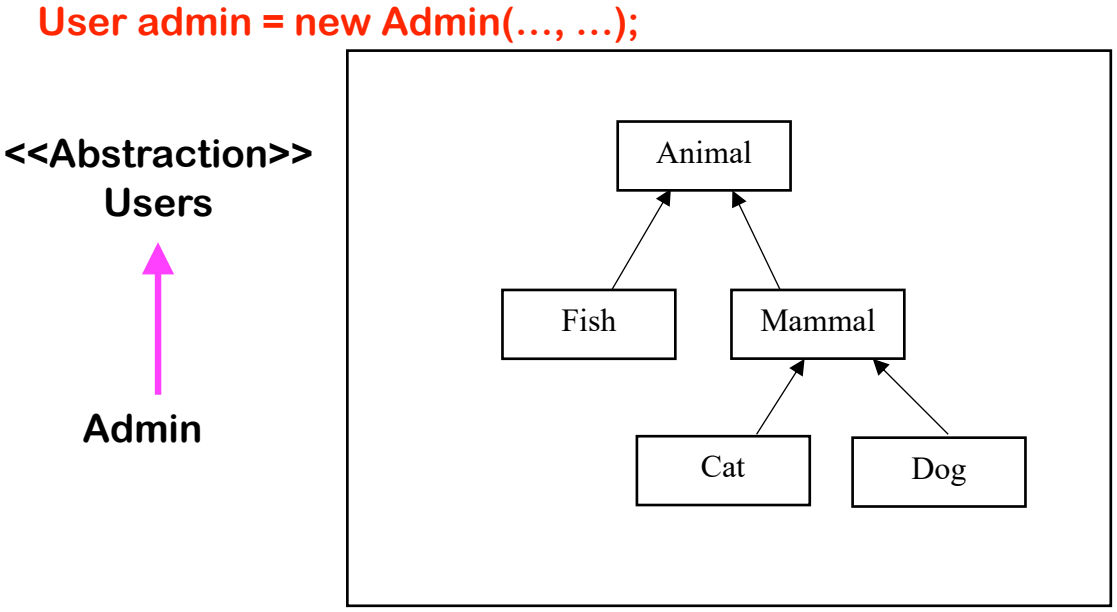



*Figure 25: Example of Inheritance*

Figure (Figure 25) above is one of the example of inheritance applied. Animal class is a parent class of fish and mammal classes. Since fish and mammal classes are inherited from animal class, All attributes and methods will be copied from the animal class. Fish and mammal classes will not need to declare the attributes and methods, it will need to use keyword 'super' to inherited everything from the animal class, but it is possible to add the attributes or methods into that specific class.

Below (Figure 26) is one of the inheritance code that has been implemented into the proposed system. User class is a parent class in the proposed system, which can be inherited by all the system user which include Admin, Manager, and Technician. User is a constructor which whenever being inherited by child class, the child class can use keyword 'super' to inherited this parent class. After inheriting from the parent class, all the attributes will be visible to the child class because the attributes are protected or public.

```
public abstract class User implements Operable {
    public static String name;
    public static String username;
    public static String password;
    public static String role;
    public static String memberid;
    public static String uid;
    static DataAccess da = new DataAccess();

    protected String UserID;
    protected String Username;
    protected String DOB;
    protected String Password;
    protected String Role;
    protected String Question;
    protected String Answer;
    protected String Status;
    private Account acc;

    public User(ArrayList UserDetails) {
        this.UserID = UserDetails.get(0).toString();
        this.Username = UserDetails.get(1).toString();
        this.DOB = UserDetails.get(2).toString();
        this.Password = UserDetails.get(3).toString();
        this.Role = UserDetails.get(4).toString();
        this.Question = UserDetails.get(5).toString();
        this.Answer = UserDetails.get(6).toString();
        this.Status = UserDetails.get(7).toString();
        da.setFileName("user.txt");
    }
```

*Figure 26: User Constructor*

Below (Figure 27, Figure 28, Figure 29) are the three examples of how Admin, Manager, and Technician classes inherit from the User class. Extends keyword is used in Admin, Manager, and Technician, to represent the class is inherited from the User and super keyword is used to initialise the constructor.

```
public class Manager extends User implements Operable {

    public Manager(ArrayList ManagerDetails) {
        super(ManagerDetails);
    }
```

*Figure 27: Manager extends User*

```
public class Technician extends User implements Operable {

    public Technician(ArrayList TechnicianDetails) {
        super(TechnicianDetails);
    }
```

*Figure 28: Technician extends User*

```
public class Admin extends User implements Operable {

    public Admin(ArrayList AdminDetails) {
        super(AdminDetails);
    }
```

*Figure 29: Admin extends User*

## 3.4 Polymorphism

Polymorphism refers “many shapes” in Greeks. Polymorphism means that lets the development team to use the same word but mean to implement to different things in different contexts. Referring back to the Inheritance, inheritance allows the child class to inherit every attribute and method from parent class. Polymorphism uses those methods to perform different tasks. This allows to perform a single action in different ways (w3schools.com, 2020).

Java supports two different type of polymorphisms, which are static or compile time and dynamic. Static or compile time in Java like other object-oriented programming languages, which allows to implement multiple methods that use the same name but a different set of parameters within the same class, that is called method overloading and representing a static form of polymorphism. In terms of dynamic polymorphism, it also creates a form of polymorphism. Both super class and subclass share the same name and parameters but dynamic polymorphism provides different functionality (Janssen, 2020).

According to the different sets of parameters in class, each method has different signature. For example, if a parameter accepts a String and a Long. To access the particular method, it would need to pass a String and a Long. This allows complier to check and identify the method that has been called and bind it to the method call. This kind of calling method can be known as static binding or static polymorphism. Polymorphism allows the programmer to write generic programs is dynamic or late binding. Dynamic binding allows a member function call to be resolved at run time, according to the run-time type of an object reference

Below (Figure 30) is one of the example of Polymorphism concept that has been applied into the proposed system. Since the User is set as an abstraction class, so that cannot be instantiated in the compile time. To instantiated the User class, it would need to instantiated by using its inherited class which is Technician (child) class.

```
ArrayList<String> TechnicianDetails;
TechnicianDetails = new ArrayList<>();
TechnicianDetails.add(UserID);
TechnicianDetails.add(Username);
TechnicianDetails.add(DOB);
TechnicianDetails.add(Password);
TechnicianDetails.add(Role);
TechnicianDetails.add(Question);
TechnicianDetails.add(Answer);
TechnicianDetails.add(Status);

User userTechnician = new Technician(TechnicianDetails);
Validation checkUser = new Validation();
try {
    if(!UserID.isEmpty() && !Username.isEmpty() && !DOB.isEmpty() && !Password.isEmpty() && !CPassword.isEmpty() && !Answer.isEmpty()) {
        if(checkUser.checkUser(UserID)) {
            if(checkUser.checkpassword(Password)) {
                if(Password.compareTo(CPassword) == 0) {
                    userTechnician.Add();
                    JOptionPane.showMessageDialog(rootPane, "One technician is added.", "User Added", JOptionPane.INFORMATION_MESSAGE);
                    ClearFields();
                } else {
                    JOptionPane.showMessageDialog(rootPane, "The password does not match", "Password Mismatch", JOptionPane.WARNING_MESSAGE);
                }
            } else {
                JOptionPane.showMessageDialog(rootPane, "Please provide a password with the length of 8 to 16.", "Password Error", JOptionPane.INFORMATION_MESSAGE)
            }
        } else {
            JOptionPane.showMessageDialog(rootPane, "The username has already existed in the system. Please use another name.", "Error Message", JOptionPane.WARNING
        }
    } else {
        JOptionPane.showMessageDialog(rootPane, "Please make sure that you have entered all the text fields.");
    }
```

*Figure 30: Polymorphism for Technician*

Below (Figure 31) is another of the example of Polymorphism concept that has been applied into the proposed system. Since the User is set as an abstraction class, so that cannot be instantiated in the compile time. To instantiated the User class, it would need to instantiated by using its inherited class which is Manager (child) class.

```
ArrayList<String> UserDetails;
UserDetails = new ArrayList<>();
UserDetails.add(UserID);
UserDetails.add(Username);
UserDetails.add(DOB);
UserDetails.add(Password);
UserDetails.add(Role);
UserDetails.add(Question);
UserDetails.add(Answer);
UserDetails.add(Status);

Validation checkUser = new Validation();
try {
    if (!UserID.isEmpty() && !Username.isEmpty() && !DOB.isEmpty() && !Password.isEmpty() && !CPassword.isEmpty() && !Answer.isEn
        if (checkUser.checkUser(UserID)) {
            if (checkUser.checkpassword(Password)) {
                if (checkUser.checkDOB(DOB)) {
                    if (Password.compareTo(CPassword) == 0) {
                        if (Role.equals("MANAGER")) {
                            User user = new Manager(UserDetails);
                            user.Add();
                            JOptionPane.showMessageDialog(rootPane, "One manager is added.", "User Added", JOptionPane.INFORM
                            ClearFields();
                            new FrmManageUser().setVisible(true);
                            this.dispose();
```

*Figure 31: Polymorphism for Manager*

**Overriding Method**

Overriding is one of the object-oriented methods in providing specific implementation that has been provided by its parent class. It should make sure that the subclass uses the keyword '@Override' with the same name, same parameters, same return type as the superclass. It is importance that to perform an overriding method because when executing the program, it will depend on which method is being tried to invoke by the object, either the parent class or the child class (Singh, 2014).

```
@Override
public String getRole() {
    return ROLE.TECHNICIAN.toString();
}
```

*Figure 32: Overriding Method in Technician Class*

In the proposed system, overriding method has been applied in subclasses of Users, which include Admin, Technician, and Manager. In the superclass (User), a default value has been set in User class, which is ambiguous to be used. The overriding method in Technician class is to override the initial method, change the ambiguous words to a specific role, such as Technician or Manager. Therefore, whenever the technician class is being invoked by the object, the role 'Technician' will be returned and then perform the next task.

**Overloading method**

Method overloading is a feature that allows a single class with more than one method which having the same name, if the argument lists are different. It is similar to constructor overloading in Java, that allows a single with multiple constructors having different argument lists (Singh, 2020). Argument lists mean that the parameters in a method, such as the argument list of a method add(int a) having one parameter is different from the argument list of another method add(int a, int b) having two parameters.

```
public User(ArrayList UserDetails) {
    this.UserID = UserDetails.get(0).toString();
    this.Username = UserDetails.get(1).toString();
    this.DOB = UserDetails.get(2).toString();
    this.Password = UserDetails.get(3).toString();
    this.Role = UserDetails.get(4).toString();
    this.Question = UserDetails.get(5).toString();
    this.Answer = UserDetails.get(6).toString();
    this.Status = UserDetails.get(7).toString();
    da .setFileName("user.txt");
}

//Overloading
public User() { da .setFileName("user.txt"); }
```

*Figure 33: Overloading Methods in User Class*

The figure (Figure 33) above is one of the example of overloading method that has been implemented in the proposed system. The first User constructor accepts an array list parameter (argument list) whereas the second User constructor accepts empty parameter (argument list). Whenever instantiate this class, it can be instantiated by passing empty argument list or passing an array list parameter.

**Interface**

An interface is completely "abstract class" that can be used to group related methods without bodies, interface is another way to achieve abstraction in Java. Java does not allow to a single class to inherit from two different classes, interface therefore to be here to solve the problems. Interface cannot be instantiated variable and create an object, and interface should be declared as empty parentheses. When creating an interface, all specifier are public by default. Implement keyword used to use an interface (Guru99, 2020).

Below (Figure 34) is one of the examples of interface class that has been implemented into the proposed system.

```
import java.util.ArrayList;

/**
 *
 * @author kianjun
 */
public interface Accessable {
    ArrayList<String> getAll();
}
```

*Figure 34: Accessable Interface*

Below (Figure 35) is another example of interface class that has been implemented into the proposed system but the this interface class is inherited from the Accessable Interface.

```
public interface Operable extends Accessable {
    void Add();
    void Update();
}
```

*Figure 35: Operable Interface*

**Object and Class**

Classes and objects in java are very powerful as the class can be called multiple times if needed. In general, all the code will be placed in the java class, that are used to be called by the objects (Ahlawat, 2020). Object are the basic unit in object-oriented programming, objects are instances of a class. Class can be defined as user defined data type but it contains functions in it. Class therefore can be deemed as a blueprint for object.

Below (Figure 36) is one of the examples of class (DataAccess) that has been implemented into the proposed system. The DataAccess class contains several methods, the purpose of the DataAccess class is to retrieve the data from the database, or to create a new record into the database.

```
public class DataAccess {
    private File fileName;

    public void setFileName(String fn) {
        fileName = new File(fn);
    }

    public boolean write(String record) {
        try (PrintWriter write = new PrintWriter(new FileWriter(fileName, true))){
            write.println(record);
            return true;
        } catch (IOException ex) {
            ex.printStackTrace();
            return false;
        }
    }
```

*Figure 36: DataAccess Class*

Below (Figure 37) is the DataAccess is being instantiated in Customer class. After instantiate the DataAccess, the method setFileName therefore can be used by using the handlers.

```
public class Customer {
    DataAccess da = new DataAccess();
    protected String CustomerID;
    protected String CustomerIC;
    protected String CustomerName;
    protected String DOB;
    protected String Gender;
    protected String Email;
    protected String Phone;
    protected String Address;
    protected String regDate;
    protected String BookingID;
    protected String Status;

    public Customer() {
        da.setFileName("Customer.txt");
    }
```

*Figure 37: DataAccess Object*

**Enumeration**

Enumeration refers the purpose of representing a group of constants named in a programming language. Enum keyword is used to create an Enum and separate the constants with a comma. All letters should be in upper letters. Enum can contain both concrete methods and abstract methods (w3schools.com, 2020).

Below (Figure 38) is one of the examples of Enumeration that has been implemented into the proposed system.

```
enum ROLE {
    ADMIN {
        @Override
        public String toString() {
            return "ADMIN";
        }
    },
    MANAGER {
        @Override
        public String toString() {
            return "MANAGER";
        }
    },
    TECHNICIAN {
        @Override
        public String toString() {
            return "TECHNICIAN";
        }
    }
}
```

*Figure 38: Enumeration*

# 4.0 User Manual

## 4.1 General

### 4.1.1 Login Page

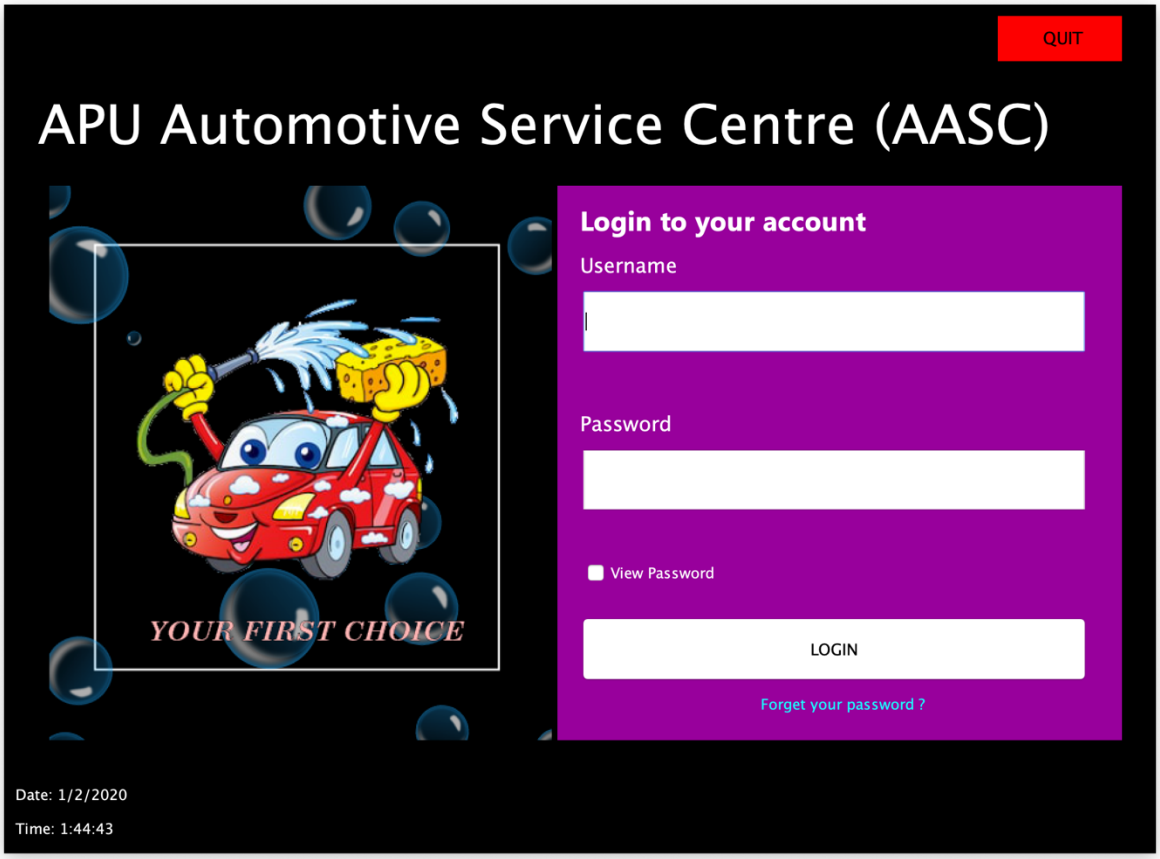



*Figure 39: Login Page*

The figure (Figure 39) above is the login page for all system users which include Administrator, Centre Manager, and Centre Technician to log into the proposed system. The proposed system will identify the login credential automatically and open the dashboard respectively.

The button QUIT on top-right corner is to quit the whole system. The 'Forget your password' below the LOGIN button is a link that is designed for all system users to retrieve the login password by providing the username and correct security answer. The view password checkbox is provided to view the password, by default, it shows the asterisk. The username text field will handle all input except letters. The system will generate the initial letter based on the full name. The user therefore should use the initial letter to login to the system. The initial letter will be given as the username in the process of registration. An error message will be shown to the user if the login credentials does not match with the database.

### 4.1.2 Forget Password Page

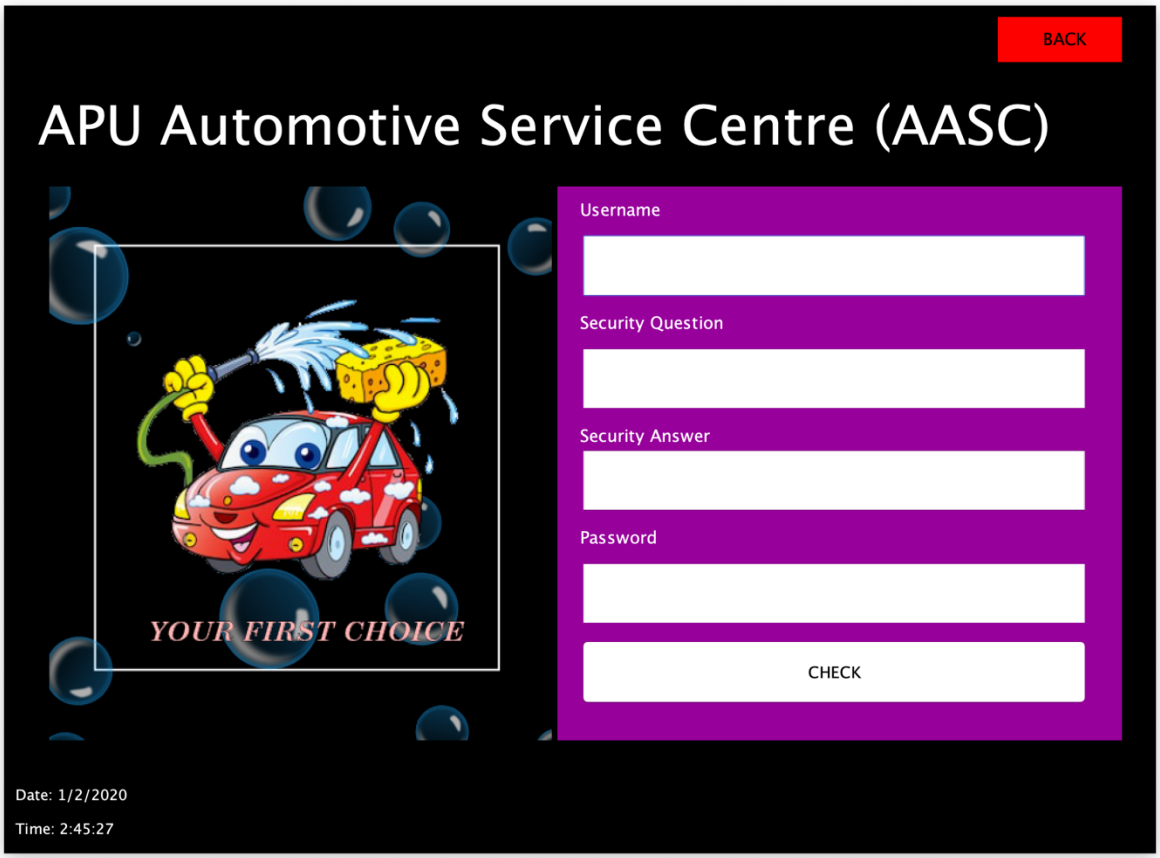



*Figure 40: Forget Password Page*

The figure (Figure 40) above is the forget password page. The top-right side is the BACK button that leads both Centre Manager and Technician to login page. The CHECK button that is designed to check the flat-file database whether the given data are matched. Security question and password text fields will not be able to edit as these two text fields are just for retrieving data from the flat-file database.

Both Centre Manager and Technician would need to provide the login username to retrieve the security question, it will immediately check with the flat-file database when the username text field lost focus. The security question will immediately display in security question text field if the username is correct. Both staff or admin need to provide security answer and clicks on check button to retrieve the password. The password will display in password text field if the security answer is correct.

## 4.2 Manager

### 4.2.1 Dashboard Page - Manager

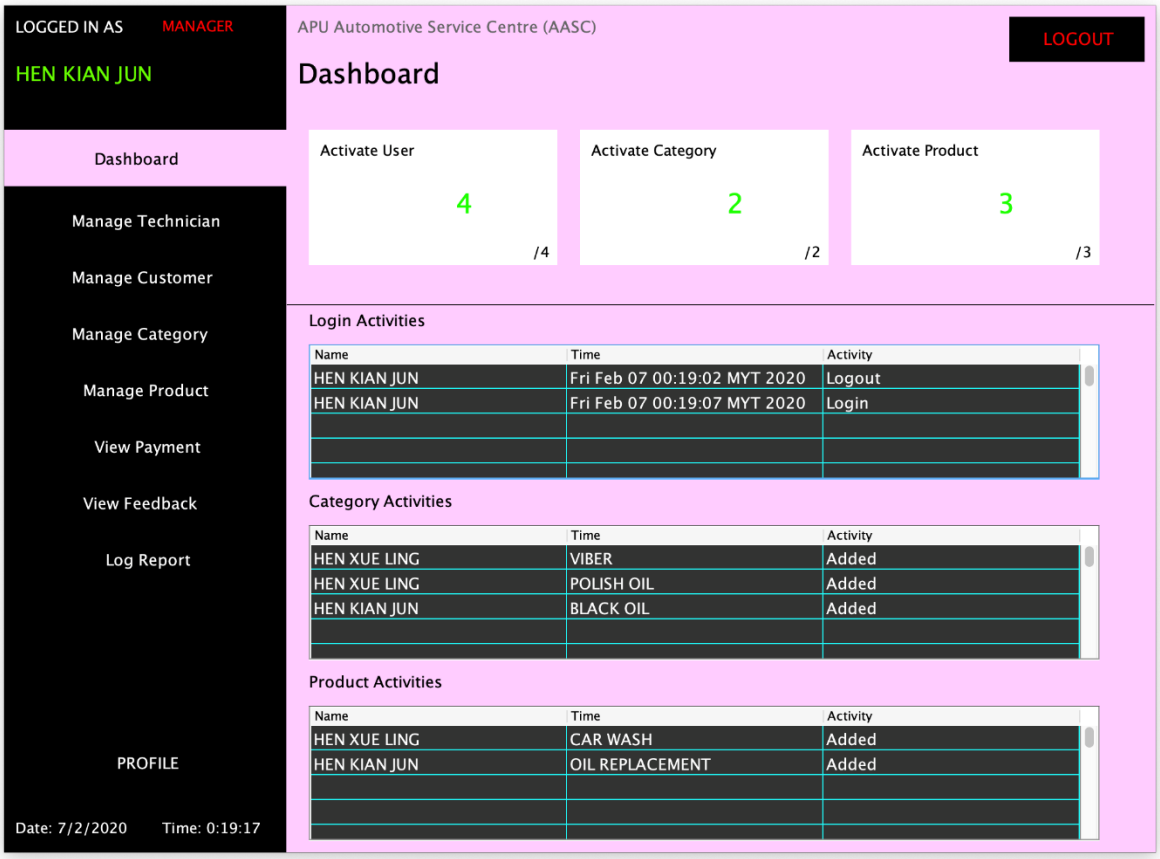



*Figure 41: Dashboard Page - Manager*

The figure (Figure 41) above is dashboard page for manager. There are three panels and three activities include in the dashboard page. Three panels include activate user, activate category, and activate product in the proposed system. The three activities include the login activities, category activities, and eventually product activities. The activities are to keep trace on what technician has done in the proposed system, activities include ‘Added’ or ‘Updated’. The top-right corner is the LOGOUT button which will log the admin out. There are several buttons on left navigation bar, which will lead the manager to the page respectively, which include Manage Technician, Manage Customer, Manage Category, Manage Product, View Payment, View Feedback, Log Report, and eventually profile.

### 4.2.2 Manage Technician Page – Manager

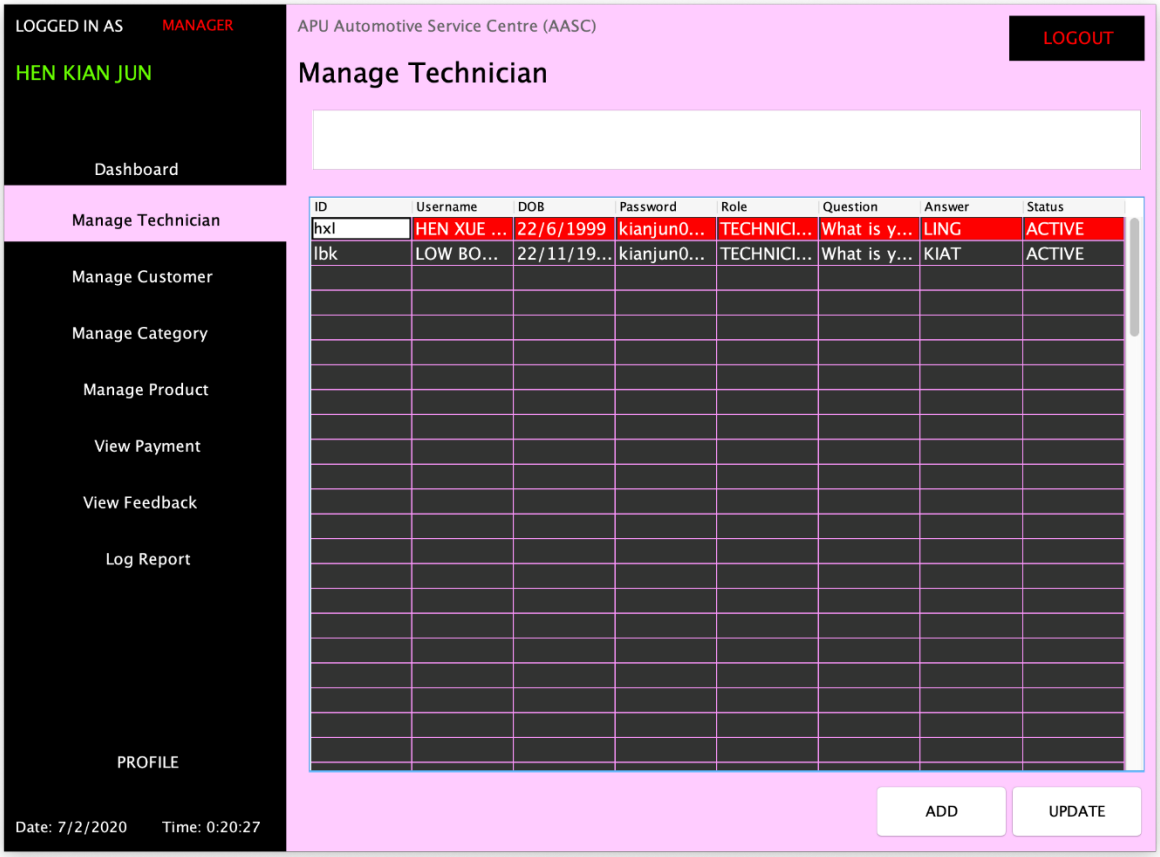



*Figure 42: Manage Technician Page - Manager*

The figure (Figure 42) above is manage technician page for manager. The table is showing all the technician details, however, manage can type on the search bar which is located above the table. The search function is applicable to the technician's name. On the right-bottom, there are two buttons which are ADD and UPDATE, and the top-right button is LOGOUT button which will lead the log manager out. Click on ADD button to create a new technician into the proposed system, and click on UPDATE to modify a particular technician, the manager is required to select a valid record. There are several tabs on left navigation bar, which will lead the manager to the page respectively, which include Dashboard, Manage Customer, Manage Category, Manage Product, View Payment, View Feedback, Log Report, and eventually Profile.

### 4.2.3 Update Technician – Manager

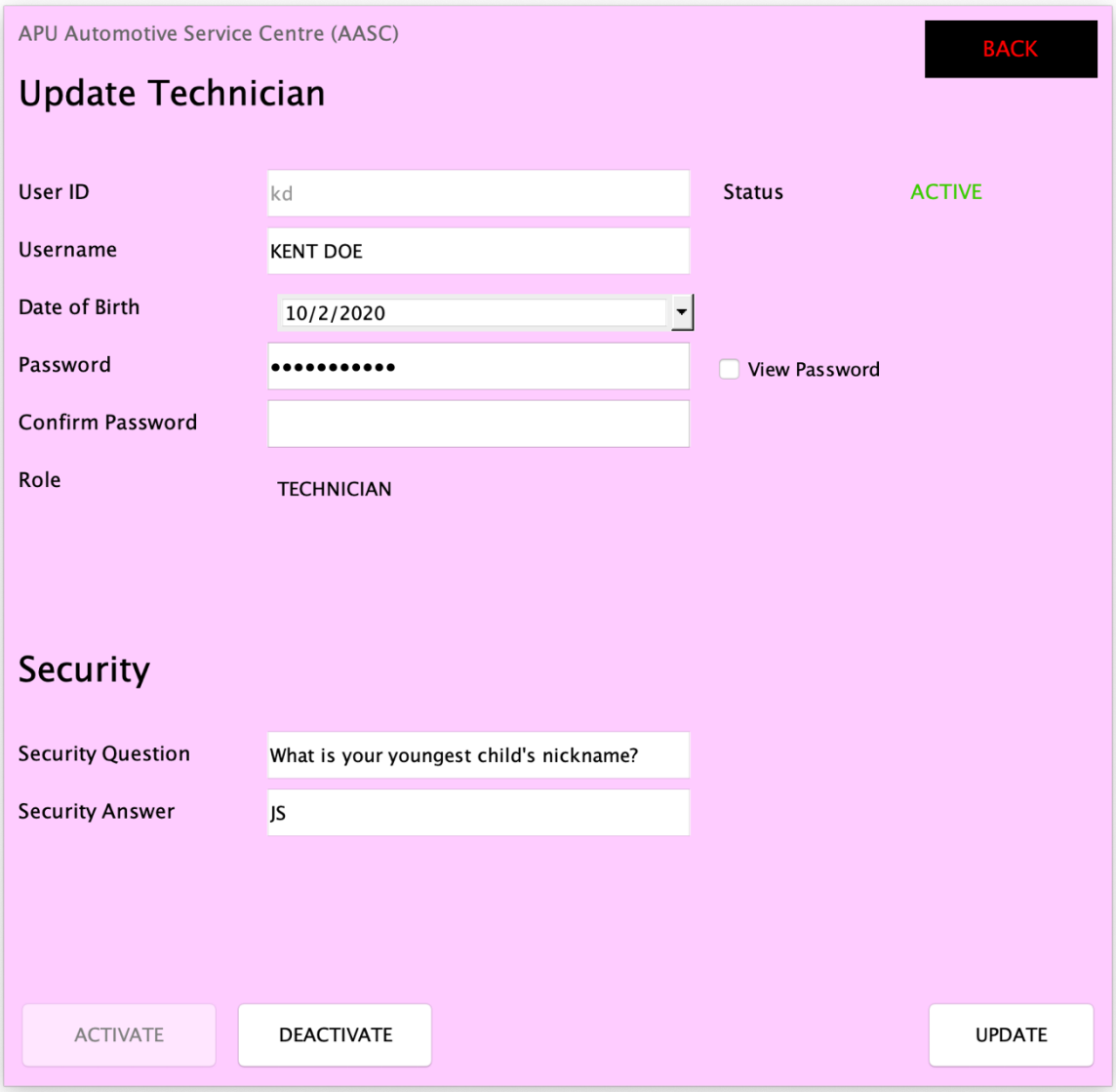



*Figure 43: Update a Technician Page - Manager*

The figure (Figure 43) above is the update technician page for manager. Manager can update technician details. User ID, Username, role, security question, and security answer are fixed. ACTIVATE button will not be able to click because the status of the technician is active, vice versa, DEACTIVATE will not be able to click when the status of the technician is inactive. Manager can click on UPDATE button to update for a specific technician.

### 4.2.4 Add New Customer Page - Manager

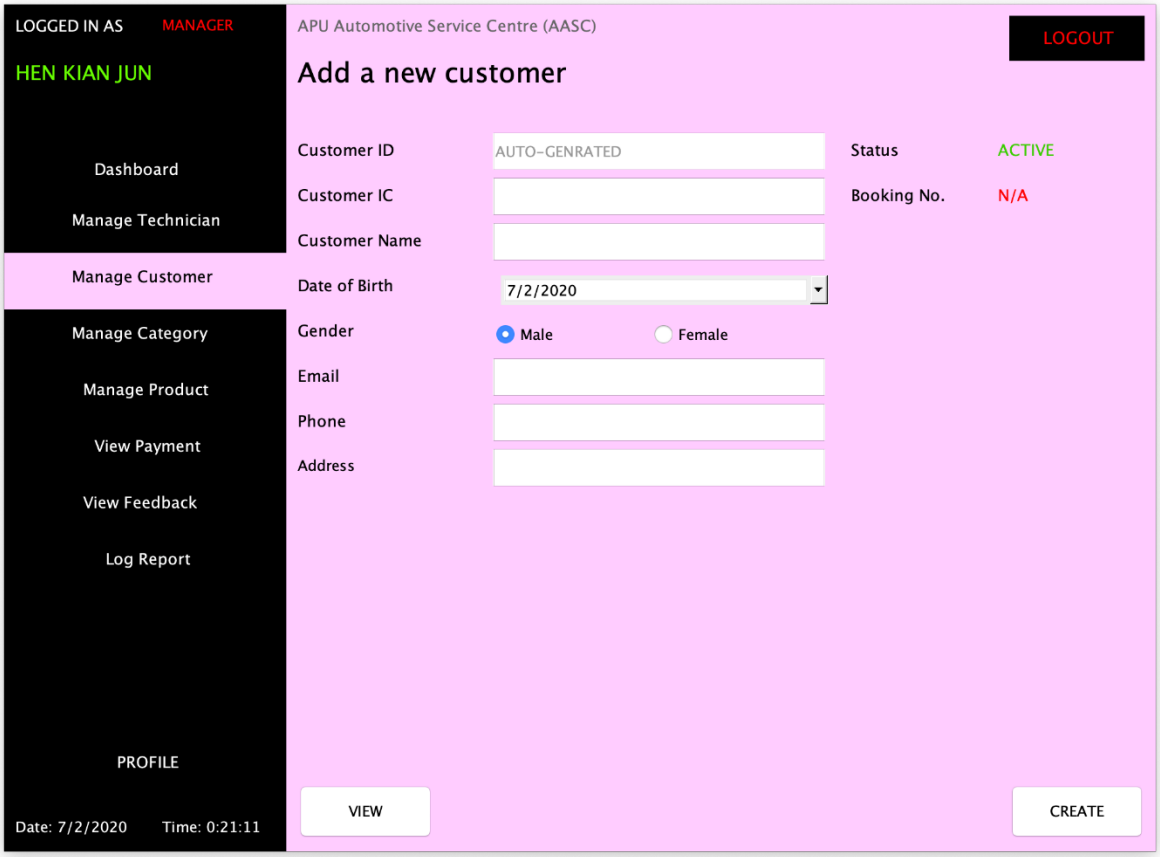



*Figure 44: Add New Customer Page - Manager*

The figure (Figure 44) above is add new customer page for manager. When registering a new customer, seven information of a customer should be provided, which include Customer IC, Customer Name, Date of Birth, Gender, Email, Phone, and Address. Customer ID is auto generated based on the customer name following numbers. Number is based on total of customer in the proposed system. One customer can have one IC only, therefore when customer IC is typed in, it will check customer IC whether it exists in the table. The customer should provide a valid email before registering to the proposed system. An error message box will pop up if any of information of customer is missing. VIEW button will be located at left-bottom which will lead the manager to view all the customer in a table form and CREATE button on right-bottom which allows the manager to create a new customer into the proposed system. On the top-right button is LOGOUT button which will lead the log manager out. There are several buttons on left navigation bar, which will lead the manager to the page respectively, which include Dashboard, Manage Customer, Manage Category, Manage Product, View Payment, View Feedback, Log Report, and eventually profile.

### 4.2.5 Manage Category Page - Manager

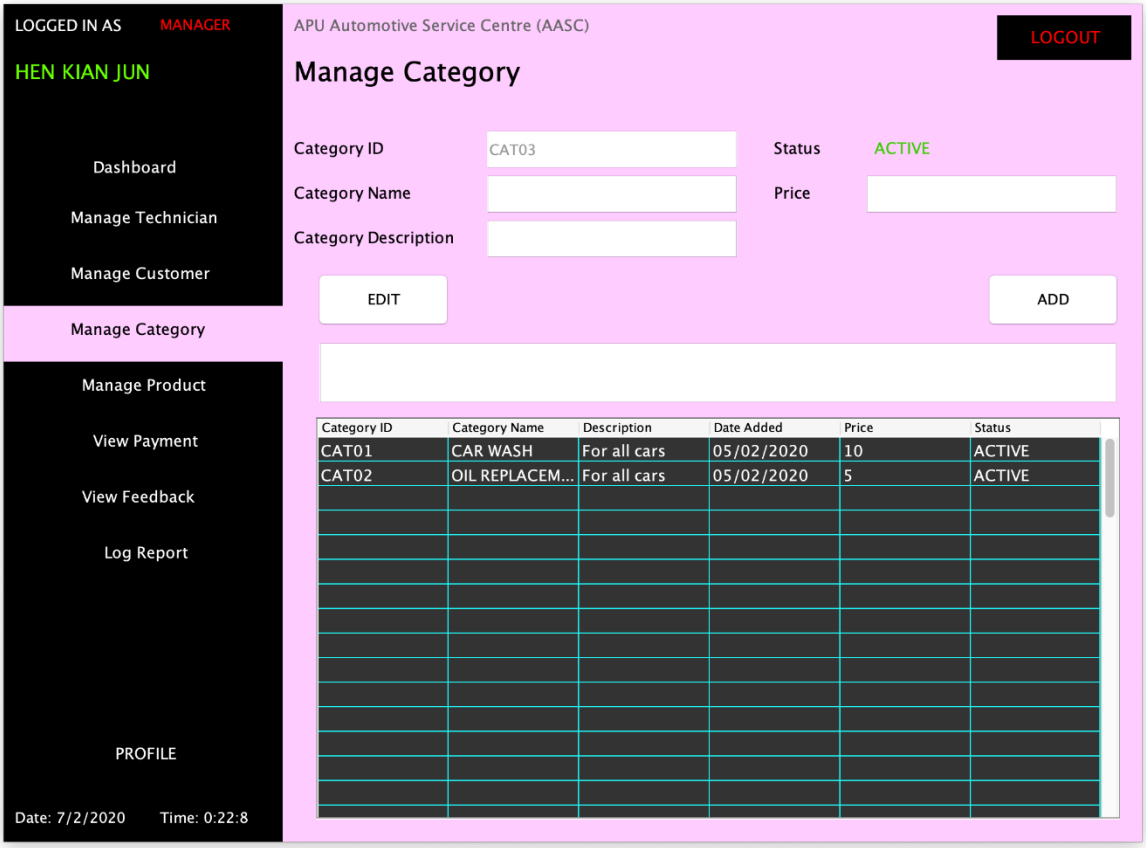



*Figure 45: Manage Category Page - Manager*

This figure (Figure 45) above is manage category page for managing user. This page is to show all the categories in a table form. There are four text fields on the top which need to fill the category information to create a new category, and two buttons which are EDIT and ADD. Besides that, below are a search and view function, blank text field is for manager to filter specific category, and the search is applicable to category name. The table will be filter based on the search field. The top-right button is LOGOUT button which will lead the log manager out. There are several buttons on left navigation bar, which will lead the manager to the page respectively, which include Dashboard, Manage Technician, Manage Customer, Manage Product, View Payment, View Feedback, Log Report, and eventually profile.

### 4.2.6 Manage Product Page – Manager

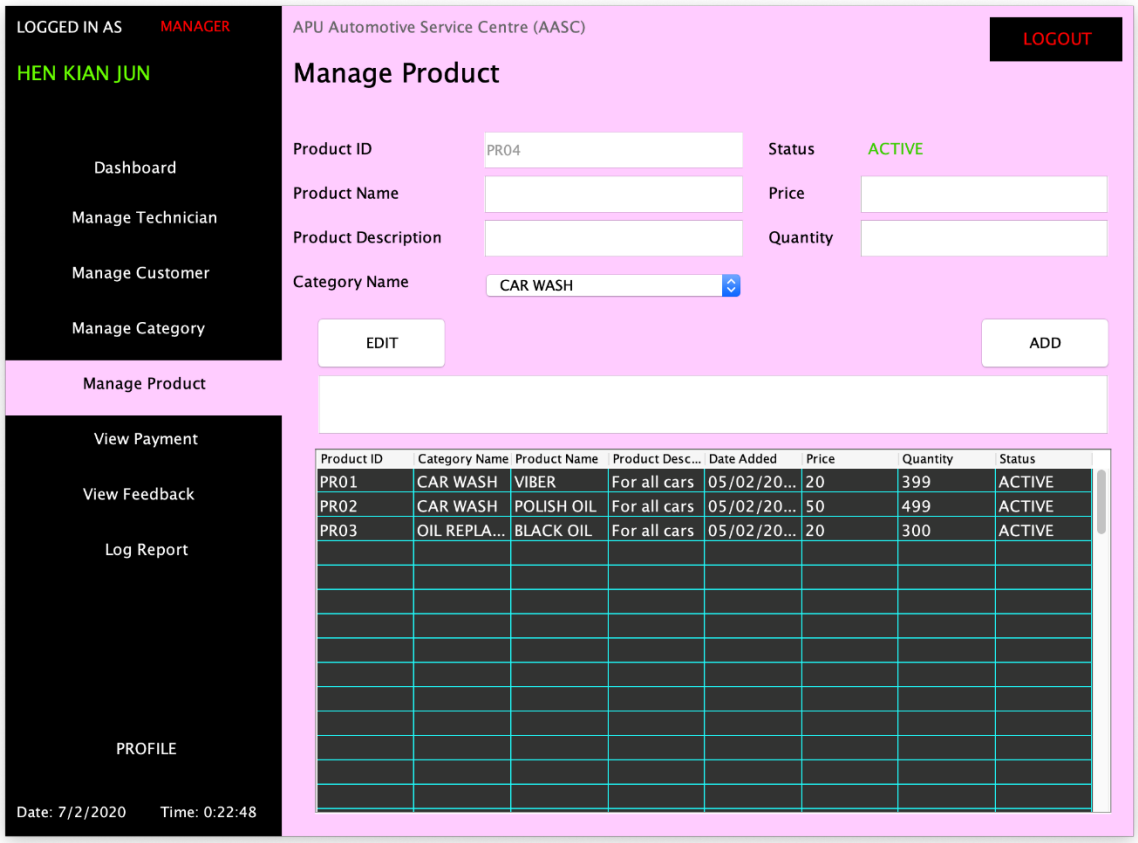



*Figure 46: Manage Product Page - Manager*

The figure (Figure 46) above is managing product page for manager. There are six information required to create a new product, one of the information is combo box. Combo box will be populated based on the active category. Besides that, at the bottom of the page show that is the search function and the table show the product details, details include Product ID, Category Name, Product Name, Product Description, Date Added, Price, Quantity and the Status. Search function is applicable to product name. The top-right button is LOGOUT button which will lead the log manager out. There are several buttons on left navigation bar, which will lead the manager to the page respectively, which include Dashboard, Manage Technician, Manage Customer, Manage Product, View Payment, View Feedback, Log Report, and eventually profile.

### 4.2.7 View Sale Revenue Page - Manager

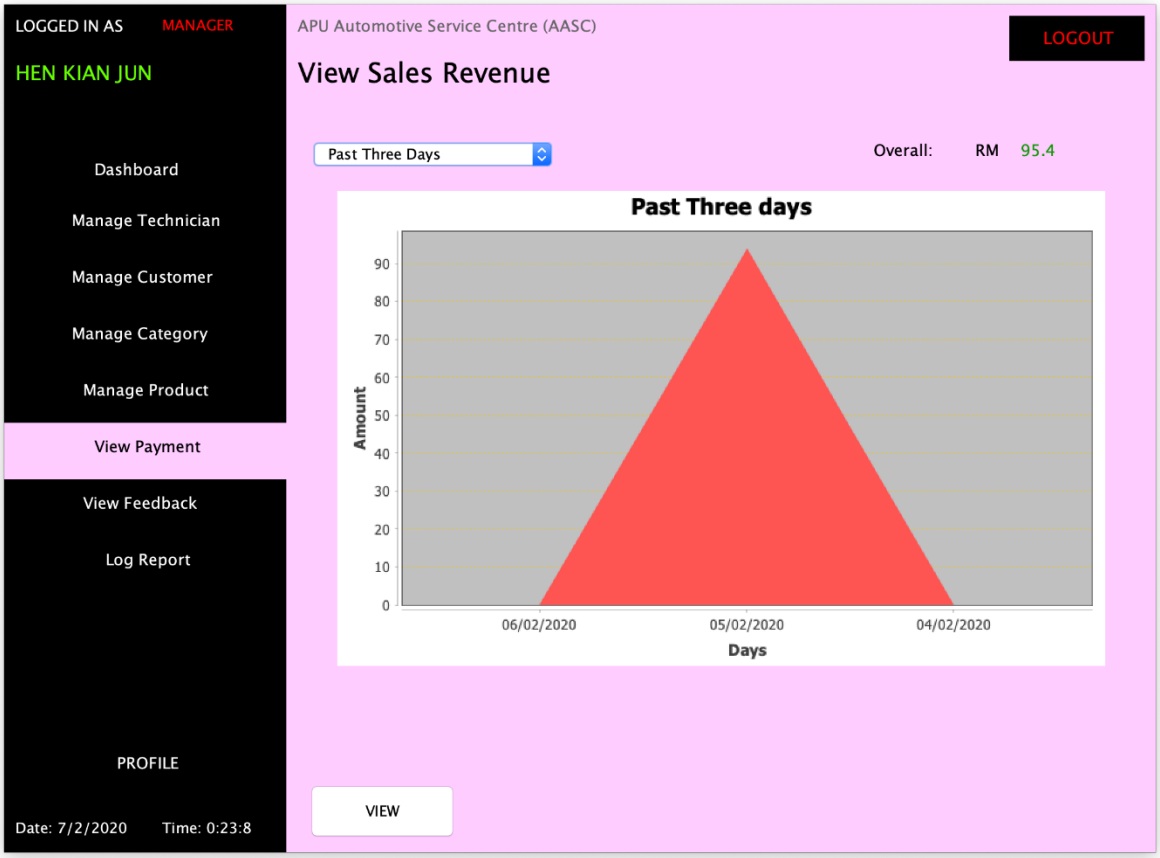



*Figure 47: View Sale Revenue Page - Manager*

The figure (Figure 47) above is view sale revenue page for manager, on the top left corner is the combo box which allows manager to select the past time to generate the area chart. The chart is to analysis the total sale revenue on a specific time duration, which includes three days or seven days. VIEW button will be located at the bottom left, which will show VIEW payment in a table. The top-right button is LOGOUT button which will lead the log manager out. There are several buttons on left navigation bar, which will lead the manager to the page respectively, which include Dashboard, Manage Technician, Manage Customer, Manage Category, Manage Product, View Feedback, Log Report, and eventually profile.

### 4.2.8 View Feedback Page by Charts – Manager

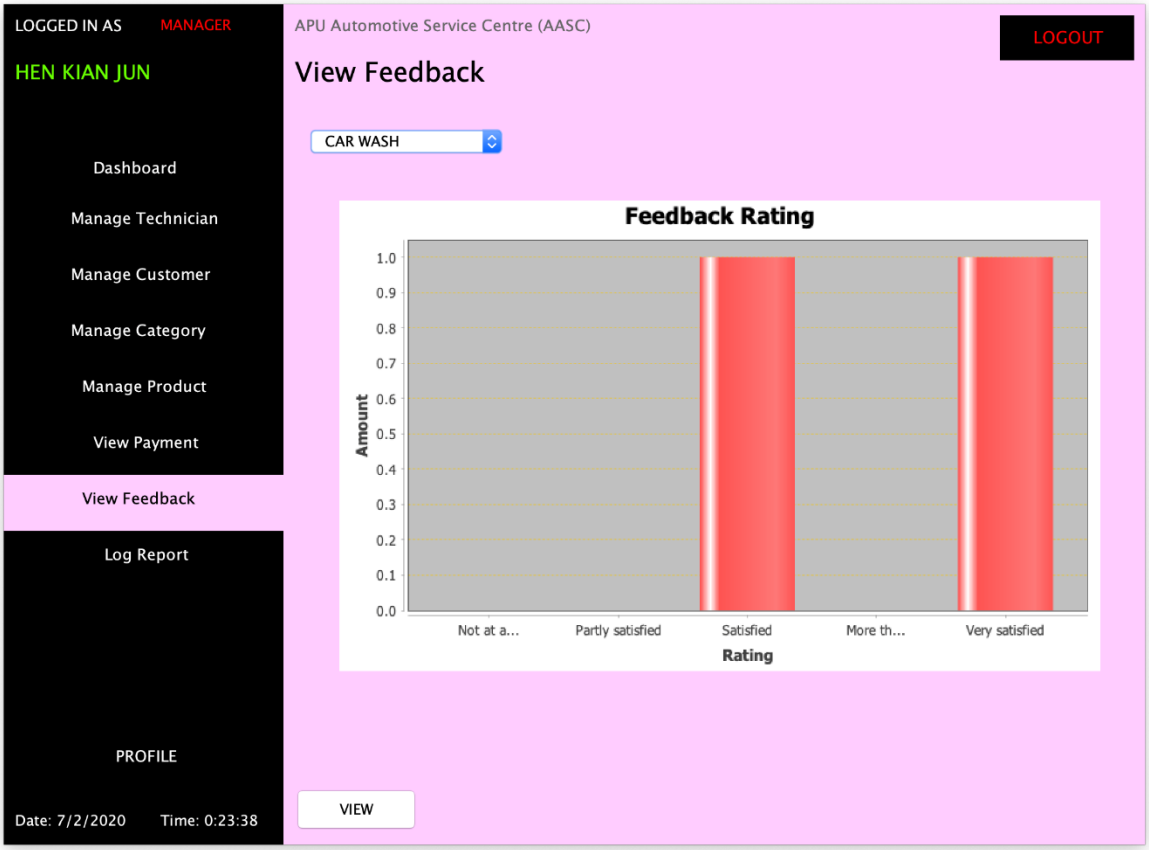



*Figure 48: View Feedback Page by Charts - Manager*

The figure (Figure 48) above is view feedback page by chart for manager, on the top left corner is the category combo box, which allows manager to select the category name. The chart is to analysis the customer satisfaction to specific category. VIEW button will be located at the bottom which is to view all the feedback in a table. The top-right corner is the LOGOUT button which will log the manager out. There are several buttons on left navigation bar, which will lead the manager to the page respectively, which include Dashboard, Manage Technician, Manage Customer, Manage Category, Manage Product, View Payment, Log Report, and eventually profile.

### 4.2.9 View Log Report Page – Manager

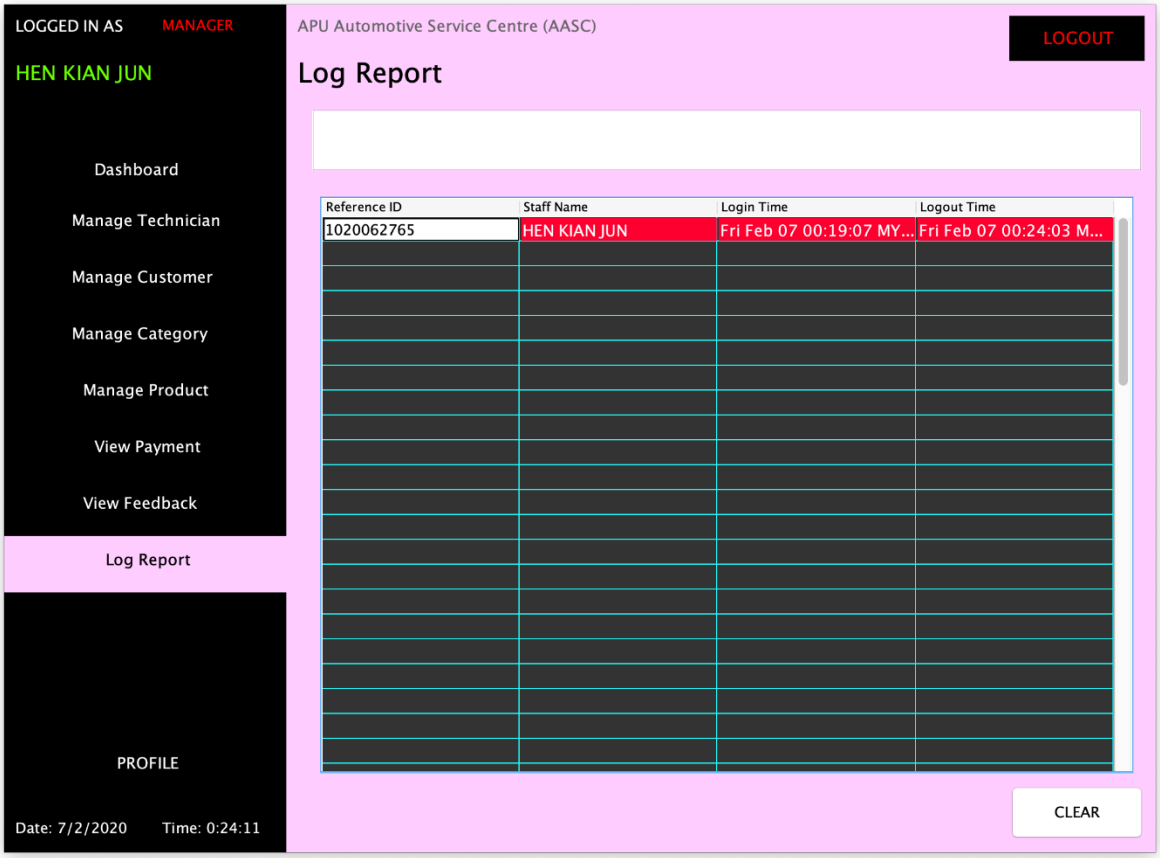



*Figure 49: View Log Report Page - Manager*

The figure (Figure 49) above is view log report in a table for manager, on the top of the page is to search the specific technician name and the table will be filter out the login and logout time. On the right bottom of the page have CLEAR button which to clear the whole log report table. The top-right corner is the LOGOUT button which will log the admin out. There are several buttons on left navigation bar, which will lead the manager to the page respectively, which include Dashboard, Manage Technician, Manage Customer, Manage Category, Manage Product, View Payment, View Feedback, and eventually profile.

### 4.2.10 View Profile Page – Manager

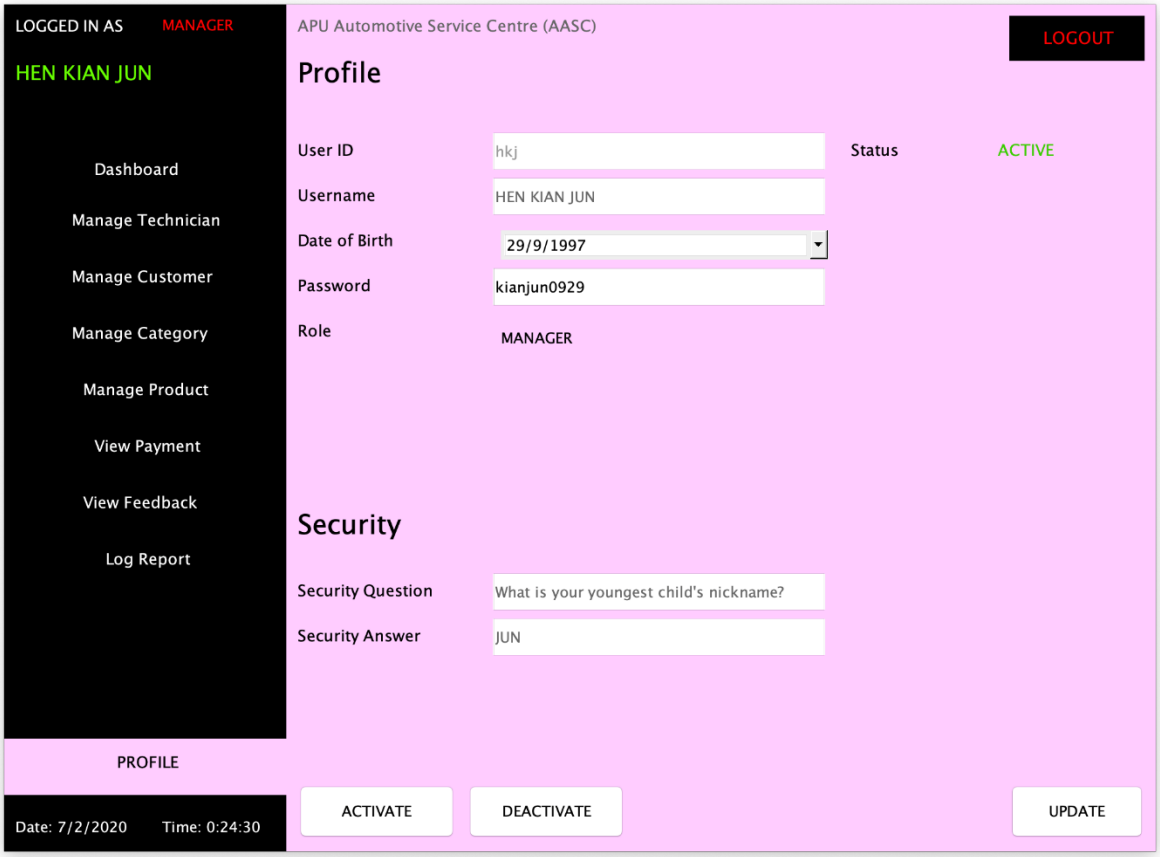



*Figure 50: View Profile Page - Manager*

The figure (Figure 50) above is view profile page for manager. Manager can do update on this page, User ID, Username, Role, security question, security answer, and status are fixed which cannot be changed. Manager can change date of birth and password on this page. There are three buttons on the bottom, which include ACTIVE, DEACTIVE, and UPDATE. The top-right corner is the LOGOUT button which will log the manager out. There are several buttons on left navigation bar, which will lead the manager to the page respectively, which include Dashboard, Manage Technician, Manage Customer, Manage Category, Manage Product, View Payment, View Feedback, and Log Report.

### 4.2.11 Add New Technician Page – Manager

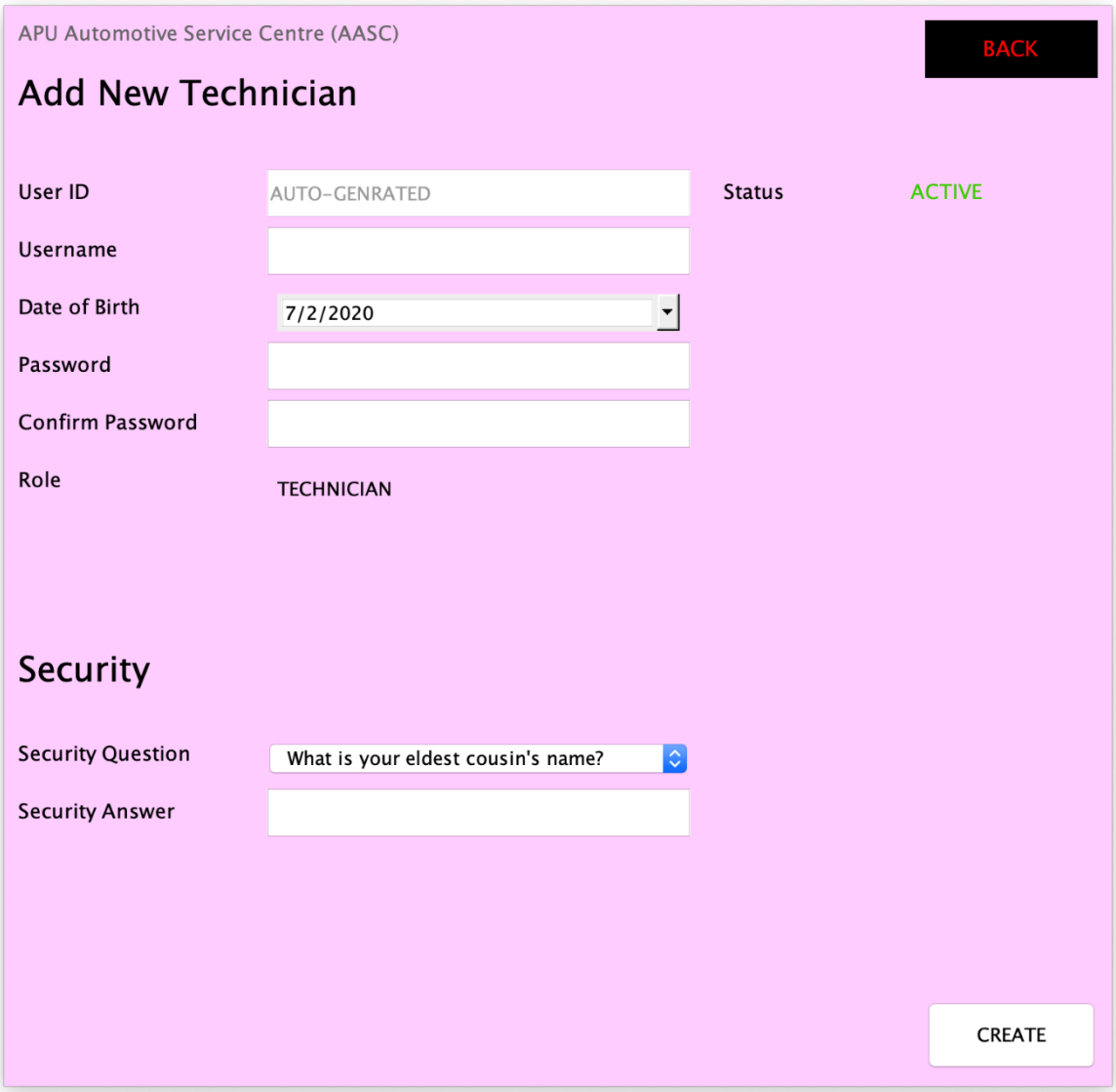



*Figure 51: Add New Technician Page - Manager*

The figure (Figure 51) above is add new technician page for manager, there are two sections in add new technician page which are the profile details and security question, and one button in this page. Manager should fill the technician details, which include username, date of birth, password, confirm password, choose a security question, and security answer. The username is one time only, which cannot be changed in the future. On the right bottom there is one button which is CREATE. The top-right corner is the BACK button which will back to the previous page.

### 4.2.12 View Customer Page - Manager

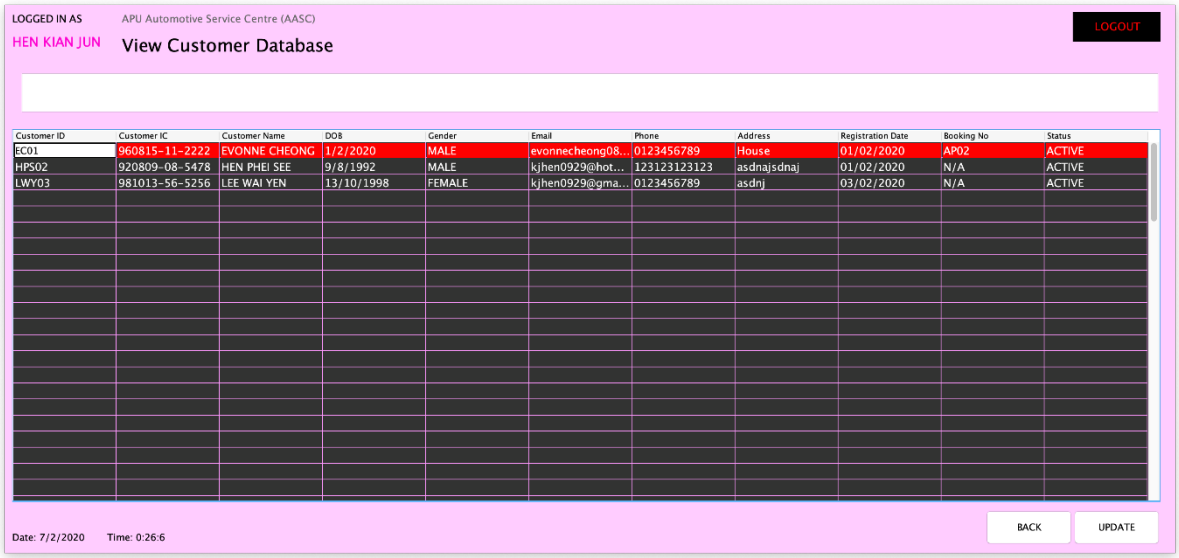



*Figure 52: View Customer Page - Manager*

The figure (Figure 52) above is view customer page for manager, in this page have the search field from the customer table. Search is applicable to customer name. Besides that, there are eleven columns to store into the customer table, which are Customer ID, Customer IC, Customer Name, Date, Category, Date of Birth, Gender, Email, Phone, Address, Registration Date, Booking Number and Status. On the bottom right of this page there are a BACK and UPDATE buttons that convenient user to have some take action in this page. The top-right corner is the LOGOUT button which will log the manager out.

### 4.2.13 Update Customer Page – Manager

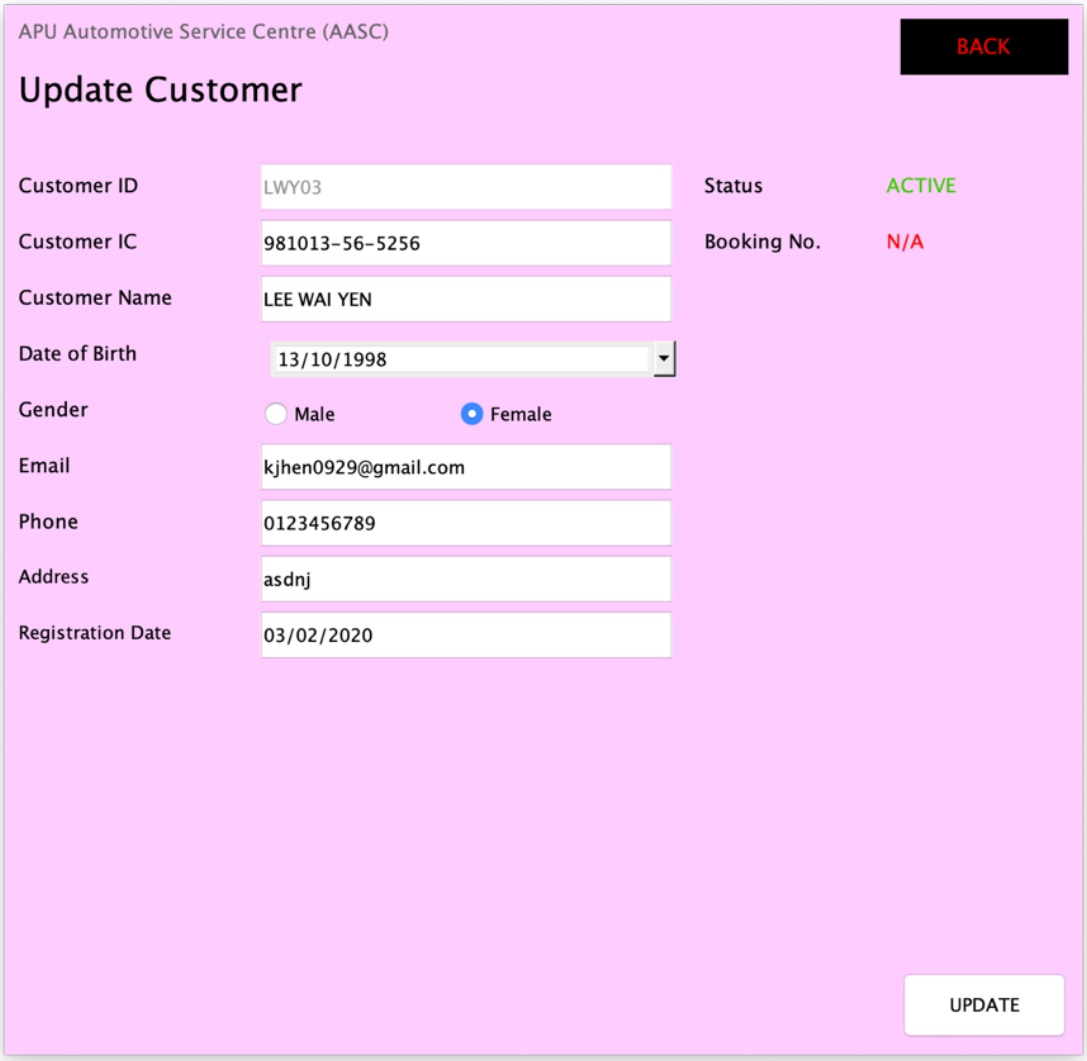



*Figure 53: Update Customer Page - Manager*

The figure (Figure 53) above is the update customer page for manager. Manager can update for the specific customer which include customer IC, customer name, date of birth, gender, email, phone, address. Other information such as Customer ID, registration date, status, and booking No are fixed. On the bottom on this page will have one buttons which is UPDATE. On the top right corner, there is a BACK button for manager back to previous page.

### 4.2.14 Update Category Page – Manager

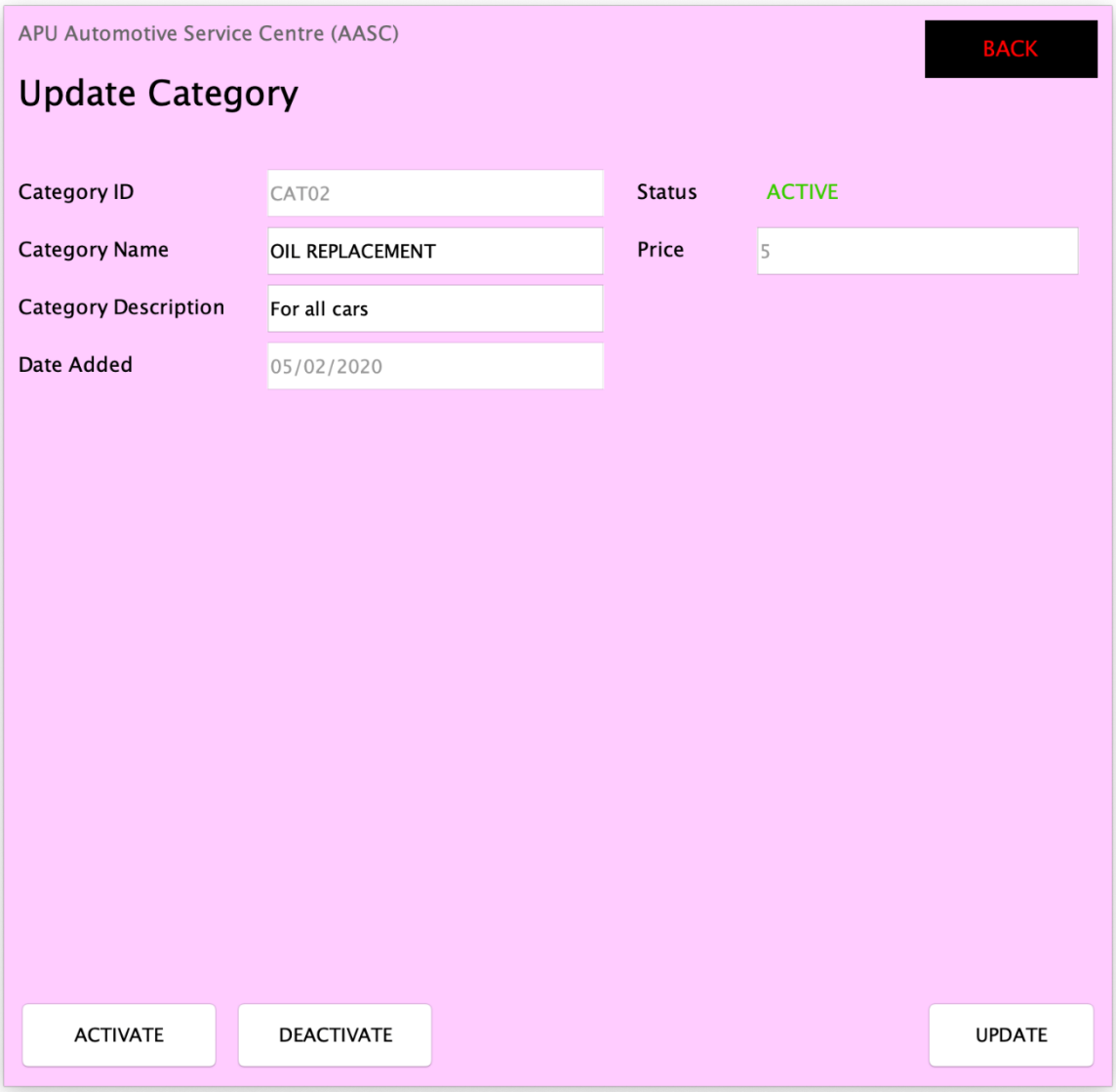



*Figure 54: Update Category Page - Manager*

The figure (Figure 54) above is update category page for manager. Manager can only do update for Category Name, category description, and price. The other information such as category ID, and date added will be fixed. On the bottom of this page contained three buttons which include ACTIVATE, DEACTIVATE and UPDATE. On the top right corner, there is a BACK button for manager back to previous page.

### 4.2.15 Update Product Page - Manager

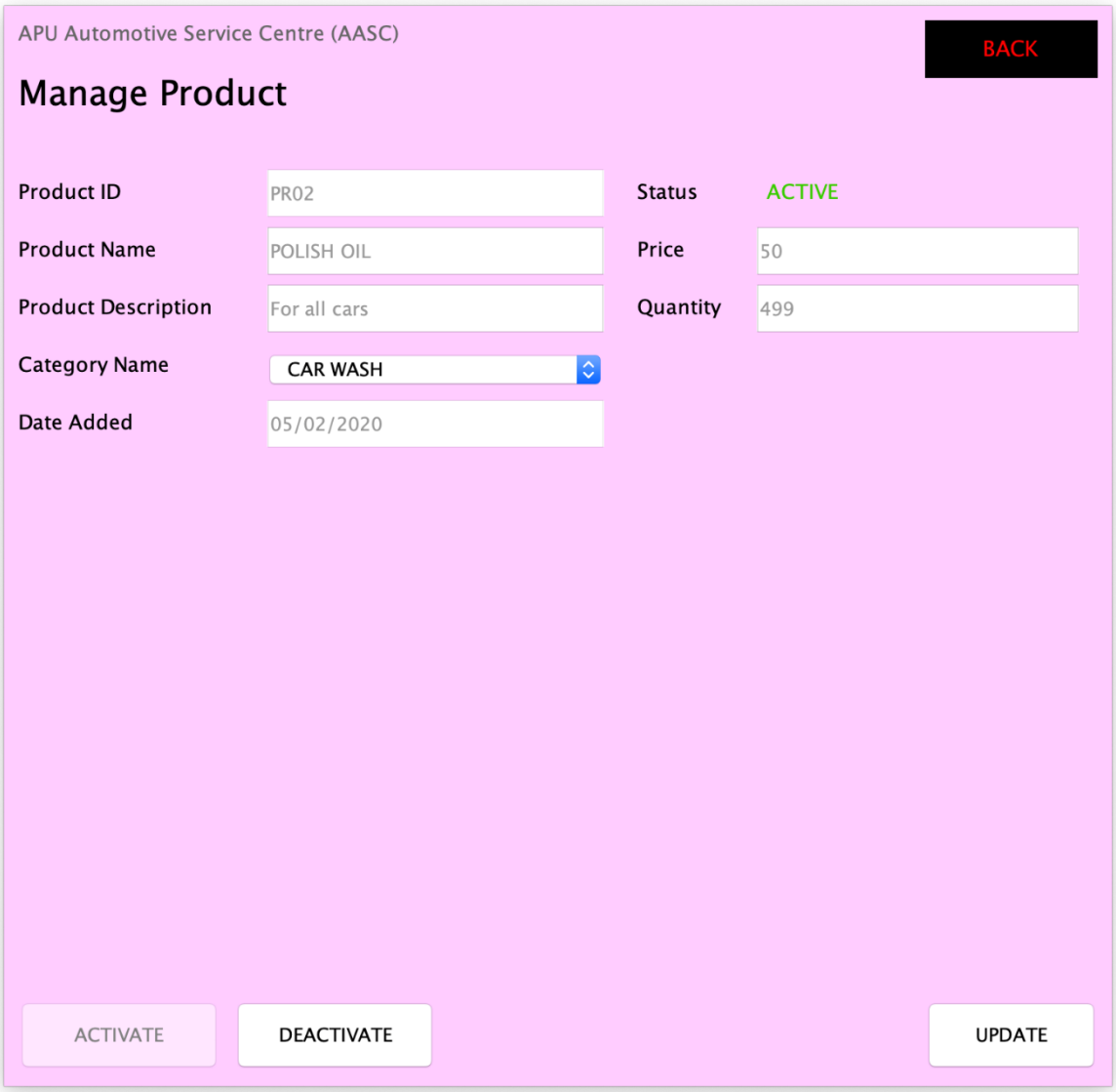



*Figure 55: Update Product Page - Manager*

The figure (Figure 55) above is update product page for manager. Manager can do update for the certain information only, which include product name, product description, category name, price, and quantity. Information, such as product ID, date added, and status, are fixed. However, manager can click ACTIVE or DEACTIVE buttons to update the status of product. Manager can click on UPDATE button to do update. On the top right corner there is a BACK button for manager back to previous page.

### 4.2.16 View All Payment Page – Manager

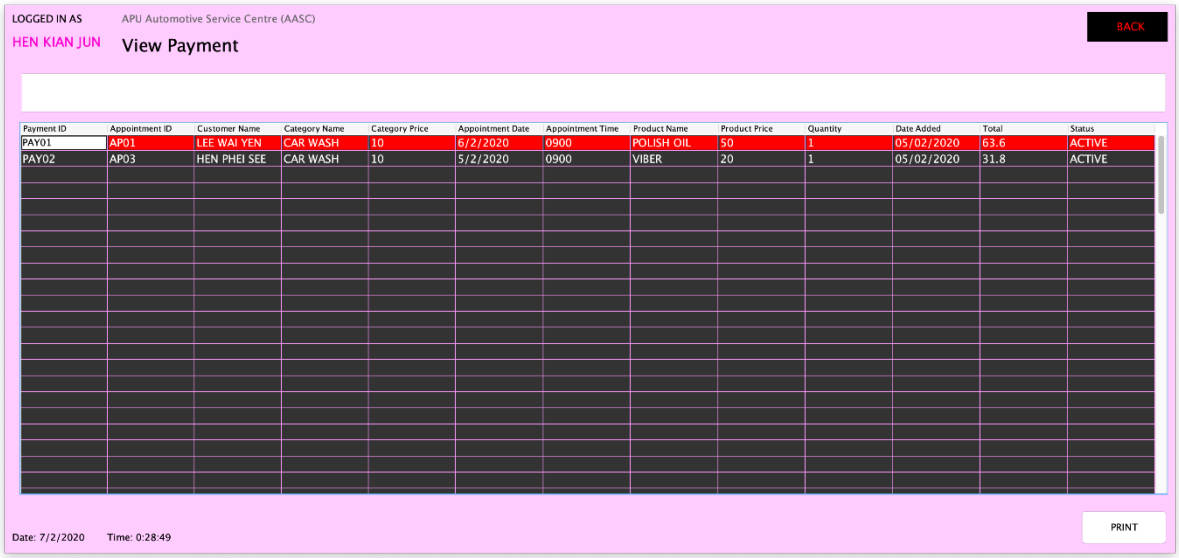



*Figure 56: View All Payment Page - Manager*

The figure (Figure 56) above is view all payment page for manager, manager can view the payment details in a table form. The search function is provided, which allows manager to filter the table. On the right bottom there is a PRINT button to have print function export the table as pdf file. other than printing the pdf, the manager will be asked whether to send through email. On the top right corner there is a BACK button for manager back to previous page.

### 4.2.17 View All Feedback Page – Manager

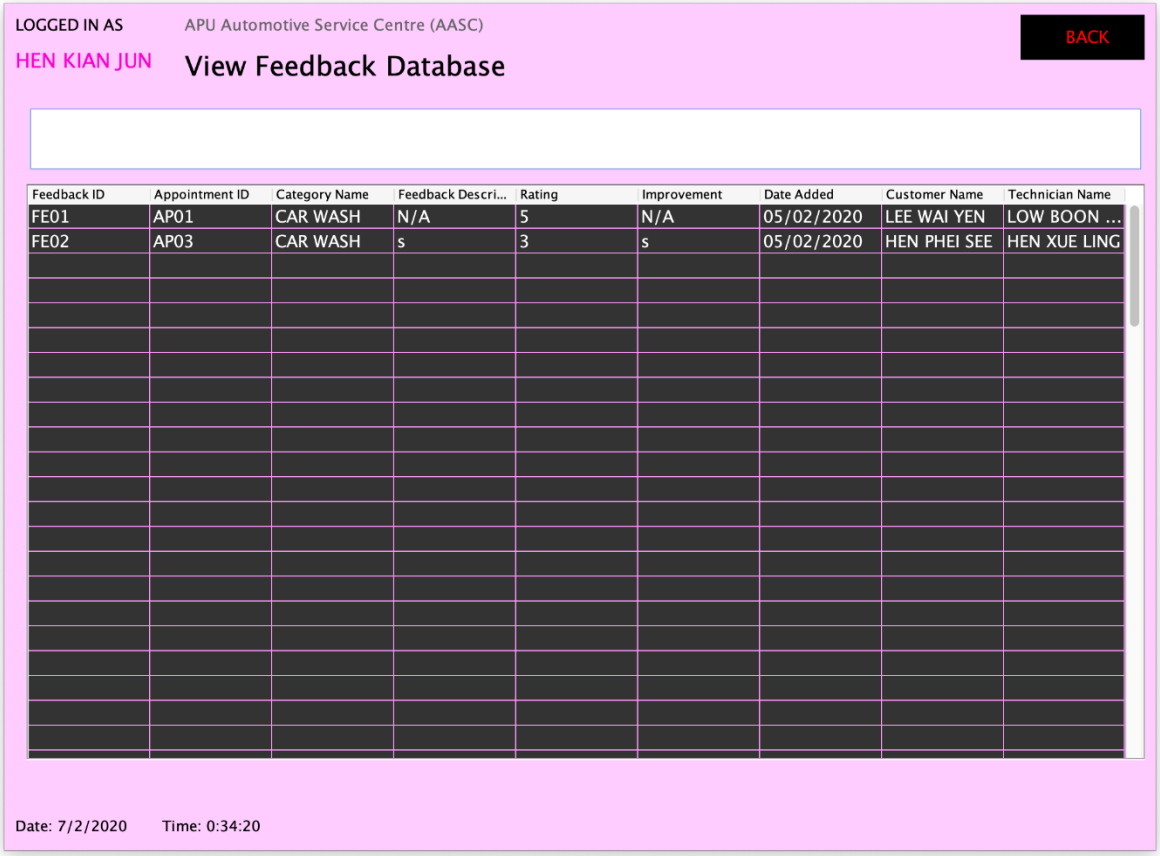



*Figure 57: View All Feedback Page - Manager*

The figure (Figure 57) above is view all feedback page for manager, manager can view the feedback details in specific in table form. The search function is provided, which allows manager to filter the table. On the top right corner, there is a BACK button for manager back to previous page.

## 4.3 Technician

### 4.3.1 Dashboard Page – Technician

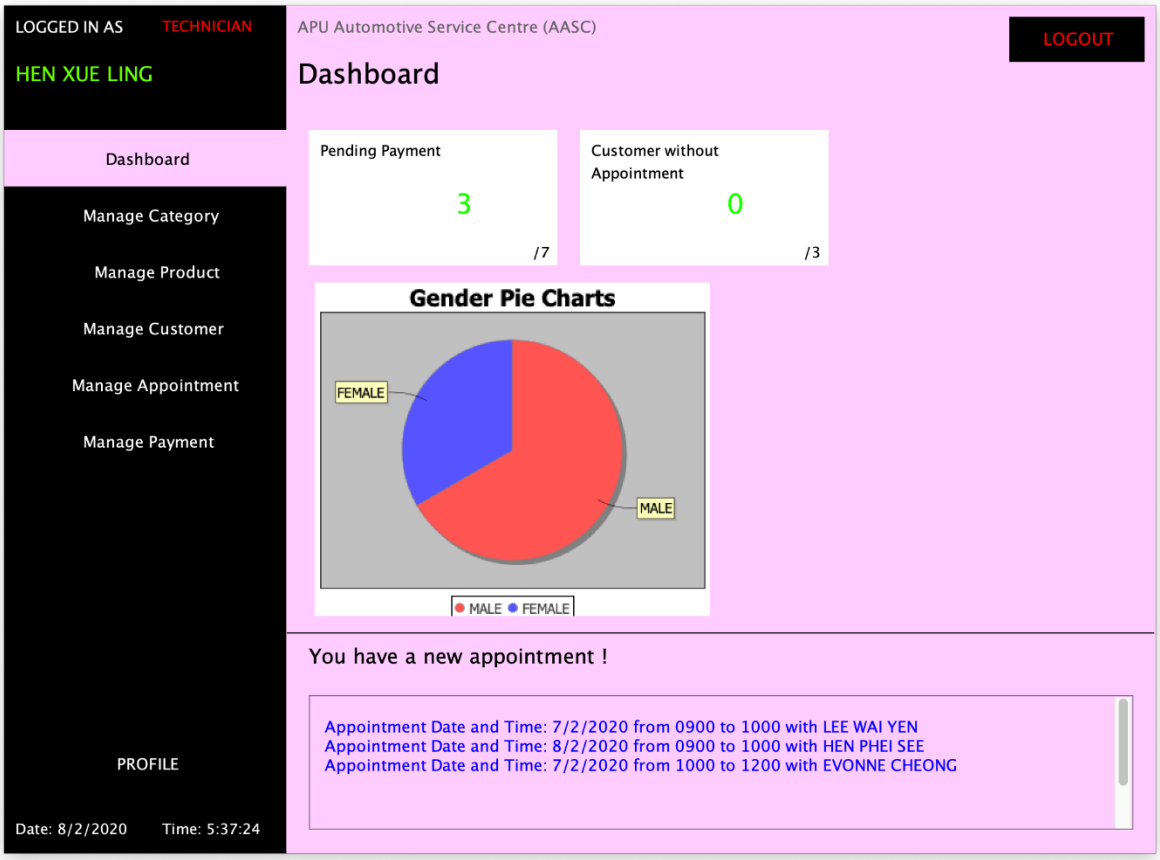



*Figure 58: Dashboard Page - Technician*

The figure (Figure 58) above is dashboard page for technician, there two panels and one generate report by pie chart include in the dashboard page. The panel include pending payment and customer without appointment. On the bottom of the page contain the technician appointment which will display in details date and time. The top-right corner is the LOGOUT button which will log the technician out. There are several buttons on left navigation bar, which will lead the technician to the page respectively, which include Manage Category, Manage Product, Manage Customer, Manage Appointment, Manager Payment, and eventually profile.

### 4.3.2 Manager Category Page - Technician

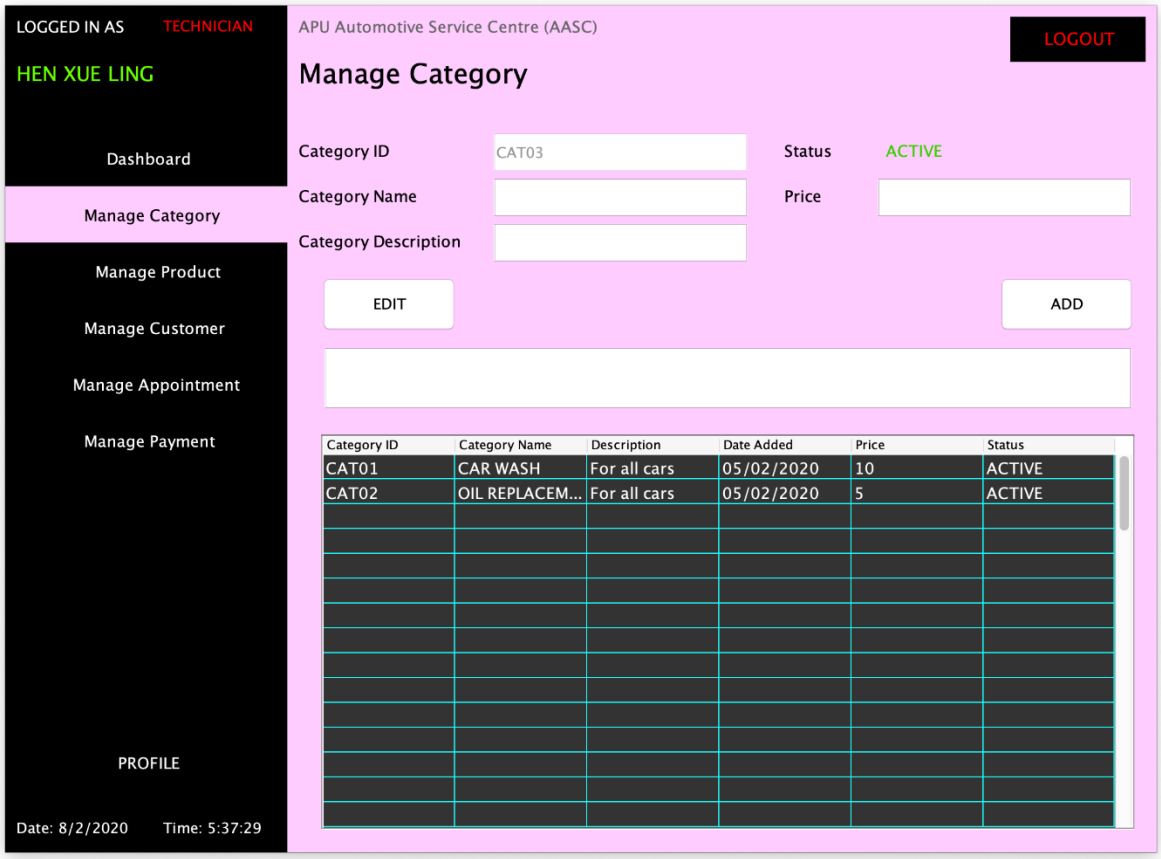



*Figure 59: Manage Category Page - Technician*

The figure (Figure 59) above is manage category page for technician, there are five information text field that technician can fill in as adding category to category table which is Category ID, Category Name, Category Description, Status and Price. Below that, there are two buttons which are EDIT and ADD button, ADD button which is to perform storing the new category details from five text fields. Edit button which is to perform the selected category row that catches to the edit page. On the bottom of the page has table which is to view all the category display as table form. The top-right corner is the LOGOUT button which will log the technician out. There are several buttons on left navigation bar, which will lead the technician to the page respectively, which include Dashboard, Manage Product, Manage Customer, Manage Appointment, Manager Payment, and eventually profile.

### 4.3.3 Update Category Page - Technician

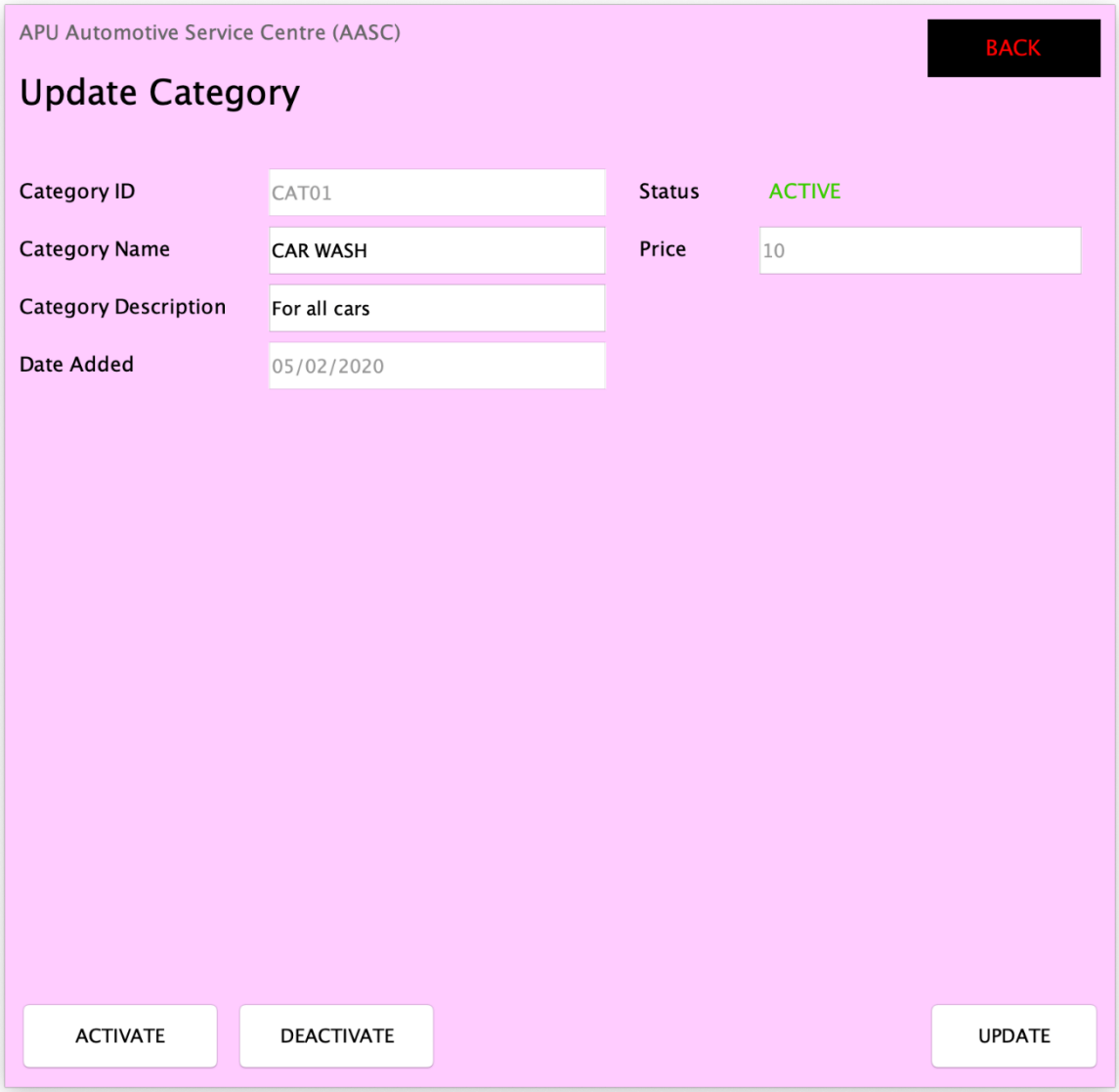



*Figure 60: Update Category Page - Technician*

The figure (Figure 60) above is update category page for technician, there are six criteria that need to be store in the database, which is Category ID, Category Name, Category Description, Date Added, Status and Price. On the bottom of this page contained three buttons which is ACTIVATE, DEACTIVATE and UPDATE. Technician can manage the category status which using activate deactivate and update the detail to the latest. The top-right corner is the BACK button which will back to the previous page.

### 4.3.4 Manage Product Page - Technician

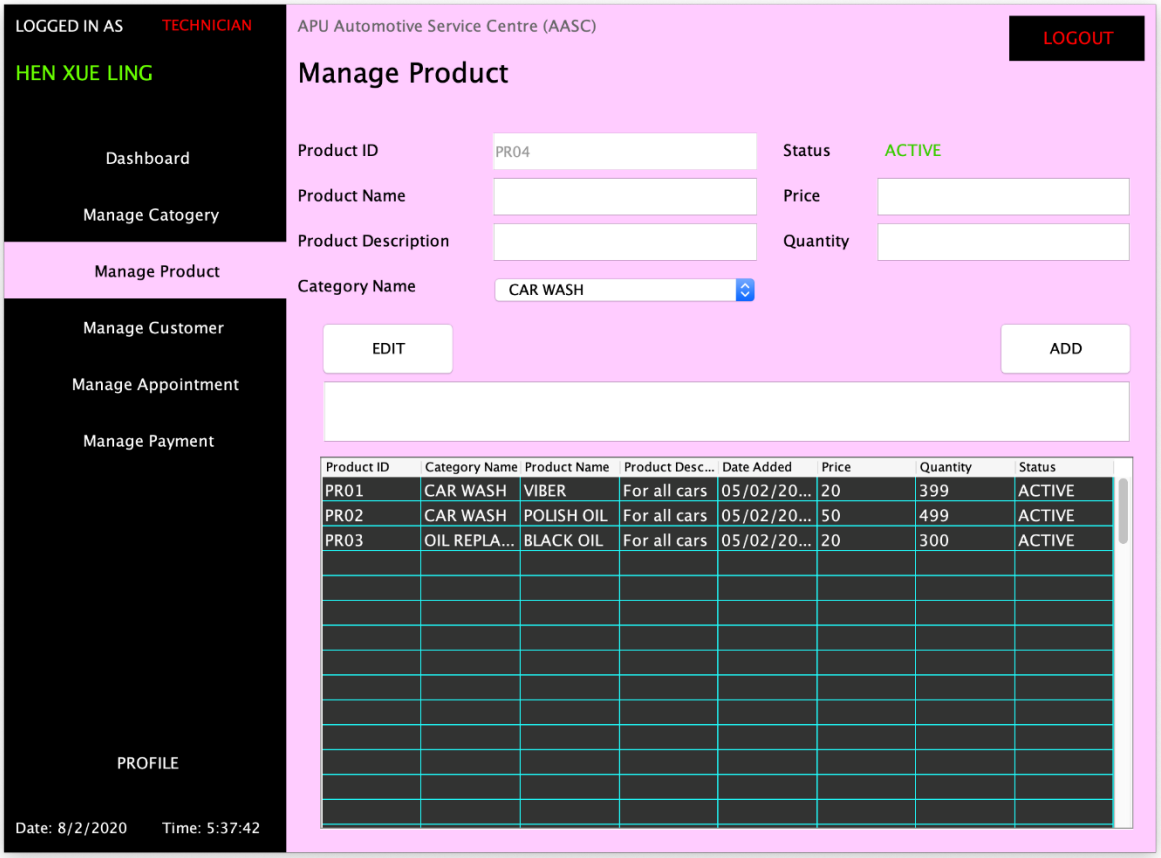



*Figure 61: Manage Product Page - Technician*

The figure (Figure 61) show above is manage product, there are seven criteria that require technician to be store in database, which is Product ID, Product Name, Product Description, Category Name, Status, Price and Quantity. Below that contained the two buttons which is EIDT and ADD. ADD button will get the data from the seven criteria and store it into the database. Edit button which is perform that selected product row that display to the edit page. On the bottom of this page table which is view all the product details. The top-right corner is the LOGOUT button which will log the technician out. There are several buttons on left navigation bar, which will lead the technician to the page respectively, which include Dashboard, Manage Category, Manage Customer, Manage Appointment, Manager Payment, and eventually profile.

### 4.3.5 Update Product Page - Technician

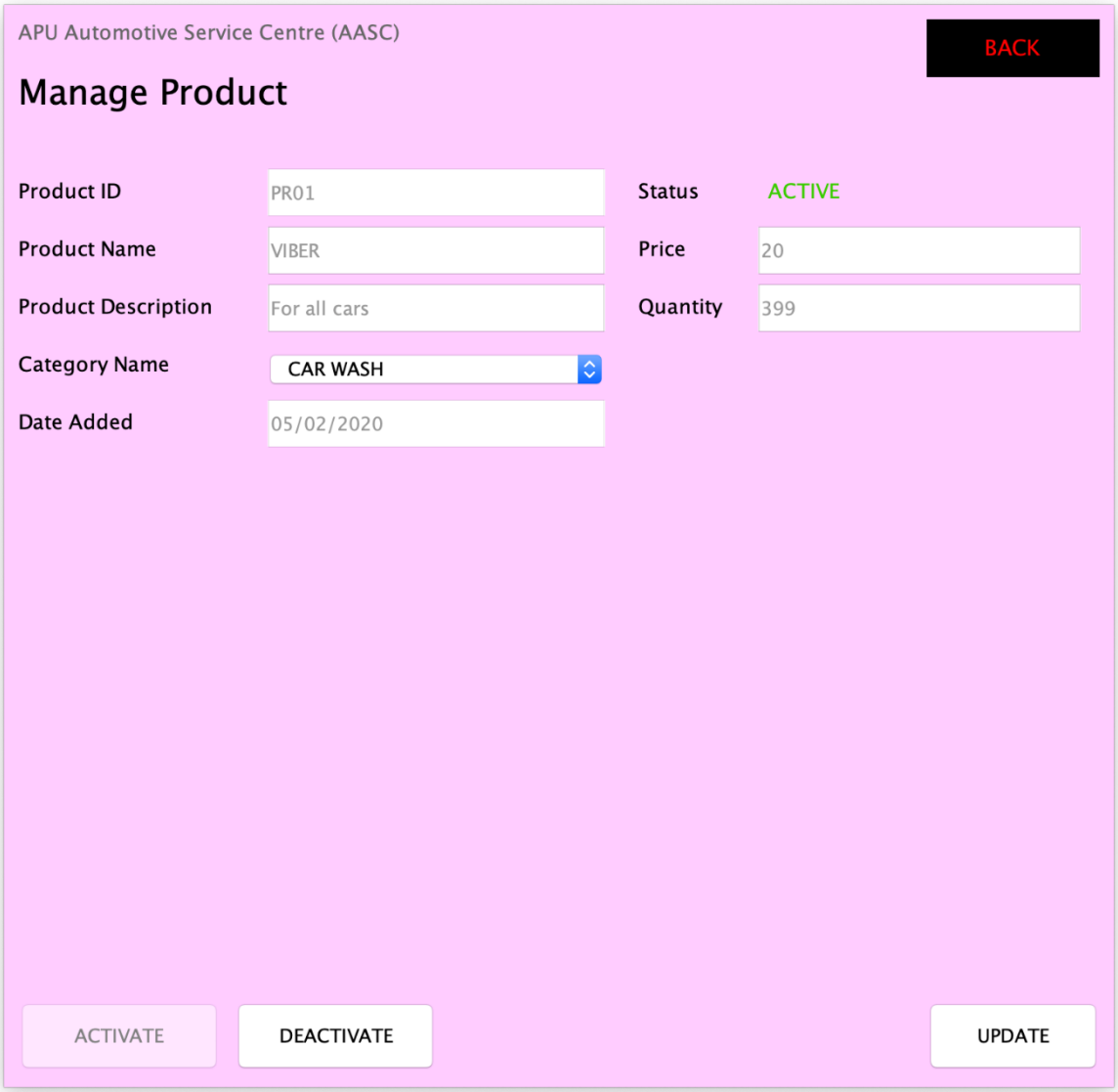



*Figure 62: Update Product Page - Technician*

The figure (Figure 62) above is update product page for technician, there are eight criteria that need to be store in the database, which is Product ID, Product Name, Product Description, Category Name, Date Added, Status, Price and Quantity. On the bottom of this page contain three buttons which is ACTIVATE, DEACTIVATE and UPDATE. Technician can manage the product status and update the details to the latest. The top-right corner is the BACK button which will back to the previous page.

### 4.3.6 Add Customer Page - Technician

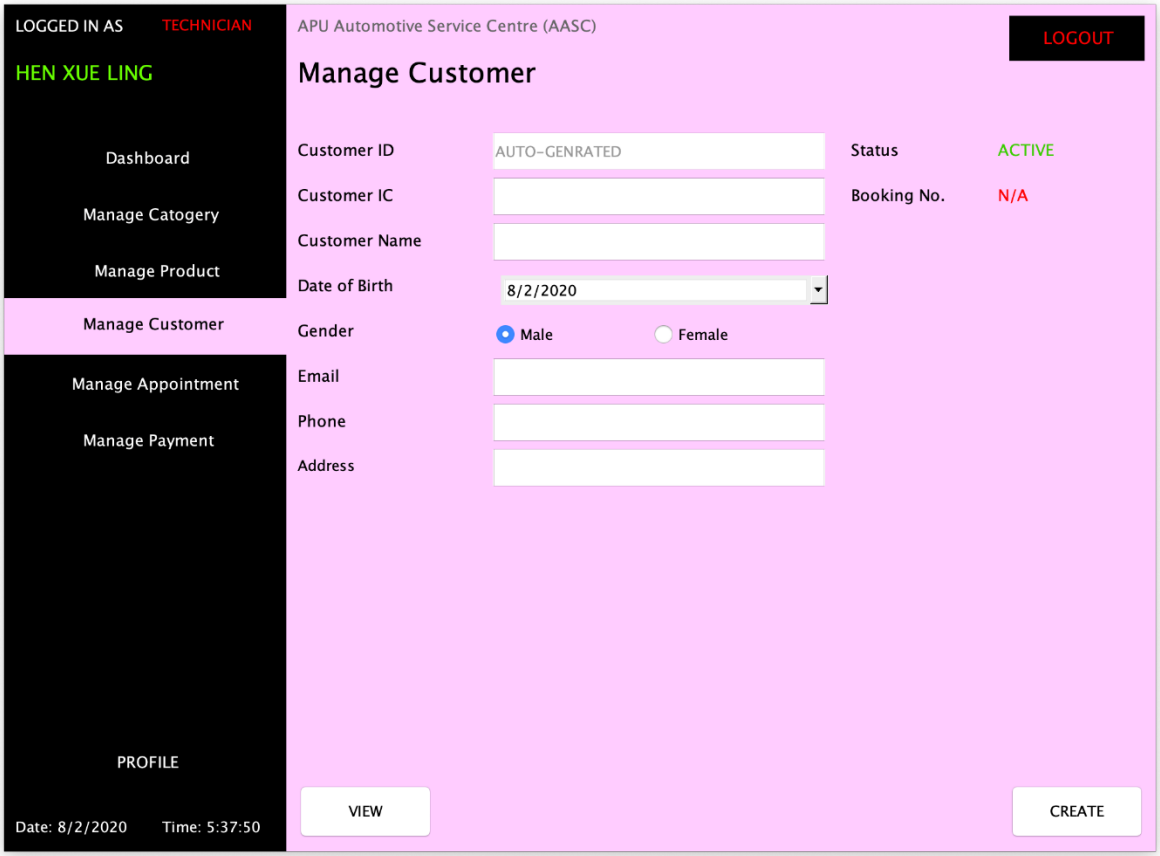



*Figure 63: Add Customer Page - Technician*

The figure (Figure 63) show above is add customer page for technician, there are ten criteria that need to be store in database which is Customer ID, Customer IC, Customer Name, Date of Birth, Gender, Email, Phone, Address Status and automatic generate Booking number. On the bottom of this page contained two button which is VIEW and CREATE. VIEW button performs lead technician to the respective page to view the customer details in table form and create button which is get data from the ten criteria and store to database. The top-right corner is the LOGOUT button which will log the technician out. There are several buttons on left navigation bar, which will lead the technician to the page respectively, which include Dashboard, Manage Category, Manage Product, Manage Appointment, Manager Payment, and eventually profile.

### 4.3.7 View Customer Page - Technician

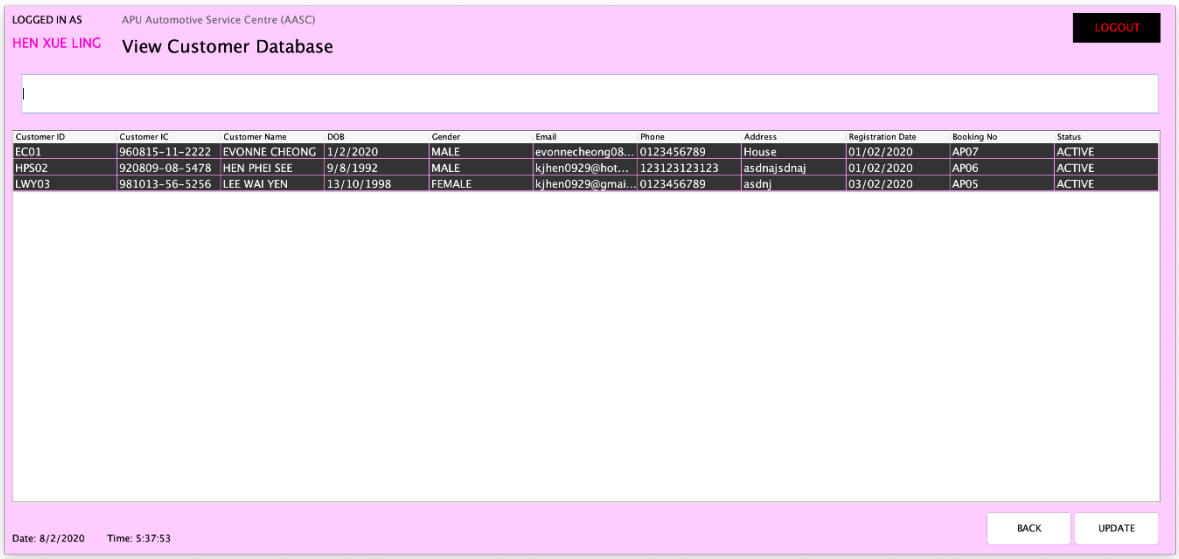



*Figure 64: View Customer Page - Technician*

The figure (Figure 64) above is view customer page for technician, on the top of this page contained the search function which present as blank text filed, below that which is the overall customer database present all the customer details, search based on customer name. On the right bottom of this page contained BACK and UPDATE button, back button will be led technician back to the previous page and update is based on the selected customer row get data to the respective page. The top-right corner is the LOGOUT button which will log the technician out.

### 4.3.8 Update Customer Page – Technician

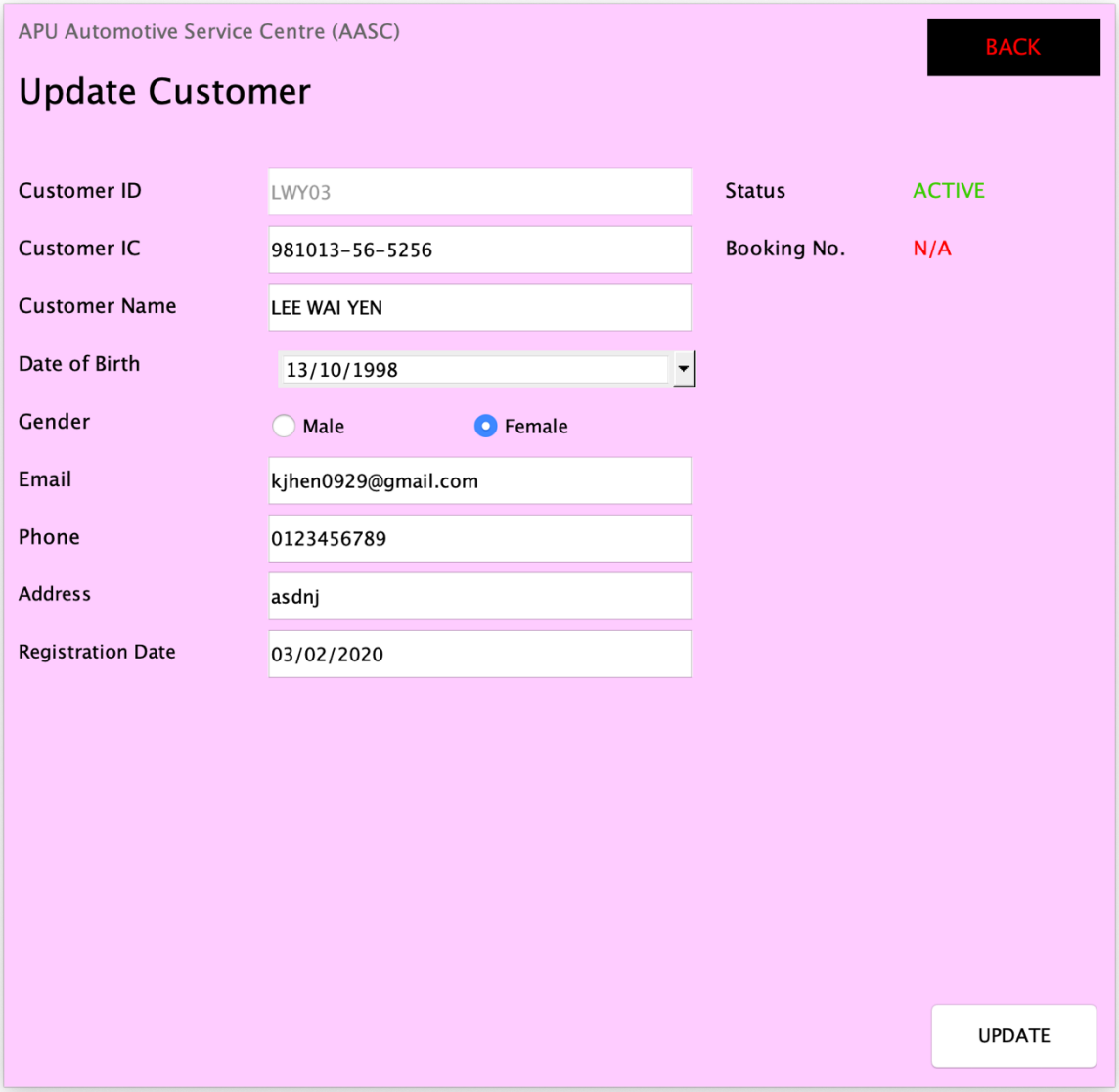



*Figure 65: Update Customer Page – Technician*

The figure (Figure 65) above is update customer page for technician, there are eleven criteria that need to be store in the database, which is Customer ID, Customer IC, Customer Name, Date of Birth, Gender, Email, Phone, Address, Registration Date, Status and Booking number. On the bottom of this page contain one button which is UPDATE. Technician can manage the customer status and update the details to the latest. The top-right corner is the BACK button which will back to the previous page.

### 4.3.9 Add Appointment Page - Technician

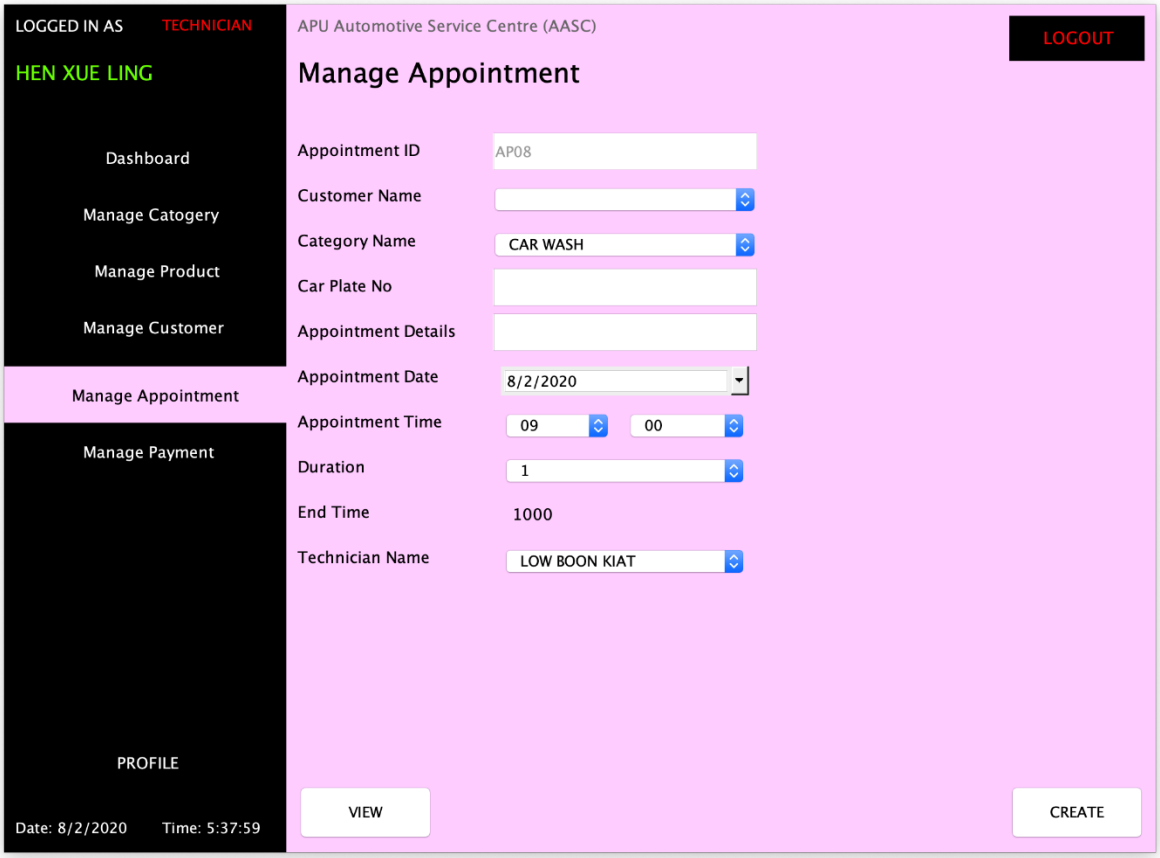



*Figure 66: Add Appointment Page - Technician*

The figure (Figure 66) above is add appointment page for technician, there are ten criteria that need to be store in database, which is Appointment ID, Customer Name, Category Name, Car Plate Number, Appointment Date, Appointment Time, Duration, End Time and Technician Name. There are two buttons located on the bottom of this page VIEW and CREATE buttons. View button will lead to the view appointment tables with a specific detail and create which is add appointment. The top-right corner is the LOGOUT button which will log the technician out. There are several buttons on left navigation bar, which will lead the technician to the page respectively, which include Dashboard, Manage Category, Manage Product, Manage Customer, Manager Payment, and eventually profile.

### 4.3.10 View Appointment Page – Technician

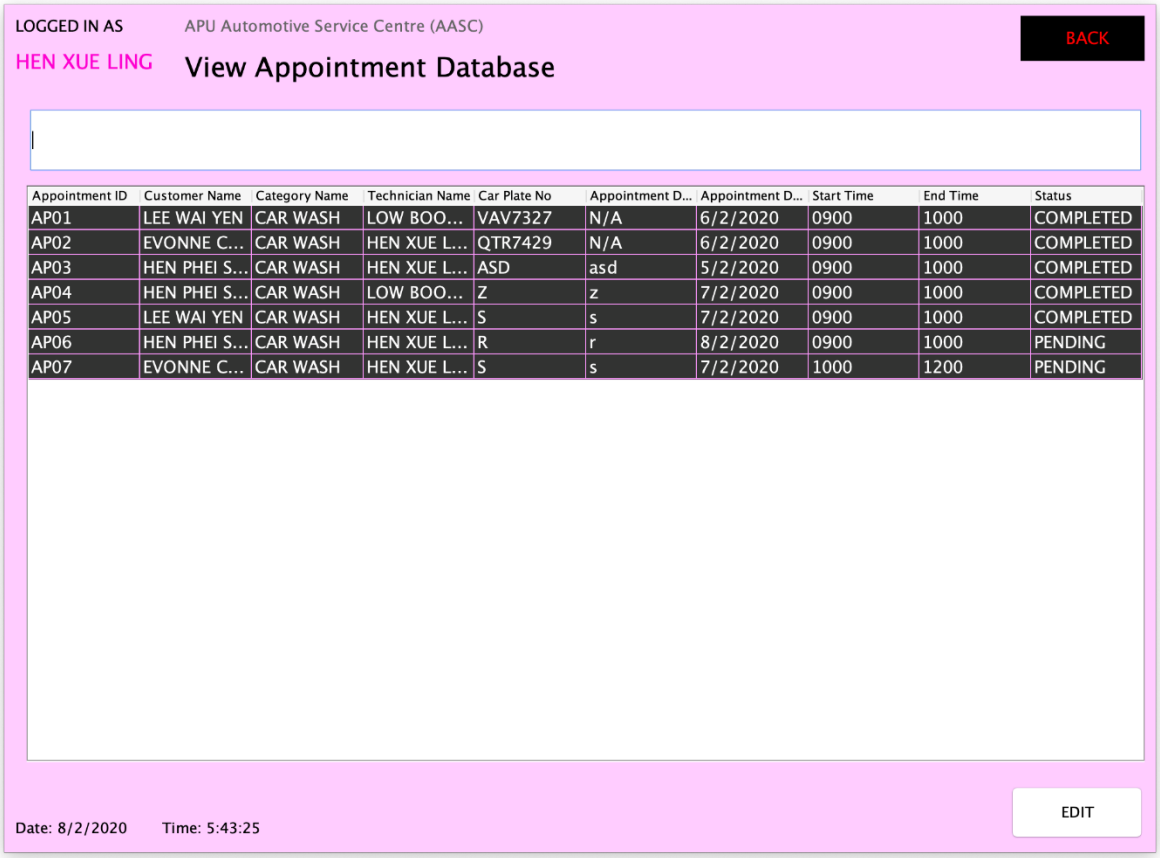



*Figure 67: View Appointment Page - Technician*

The figure (Figure 67) above is view appointment for technician, on the top of this page have search function based on the customer name and filter the table below. Below the search table is the view appointment database, which contained ten criteria such as, Appointment ID, Customer Name, Category Name, Technician Name, Car Plate No, Appointment Description, Appointment Date, Start Time, End Time and Status. On the right bottom of the page is EDIT button can edit the selected appointment row to the respective page. The top-right corner is the BACK button which led the technician back to the previous page.

4.3.11 Update Appointment Page – Technician

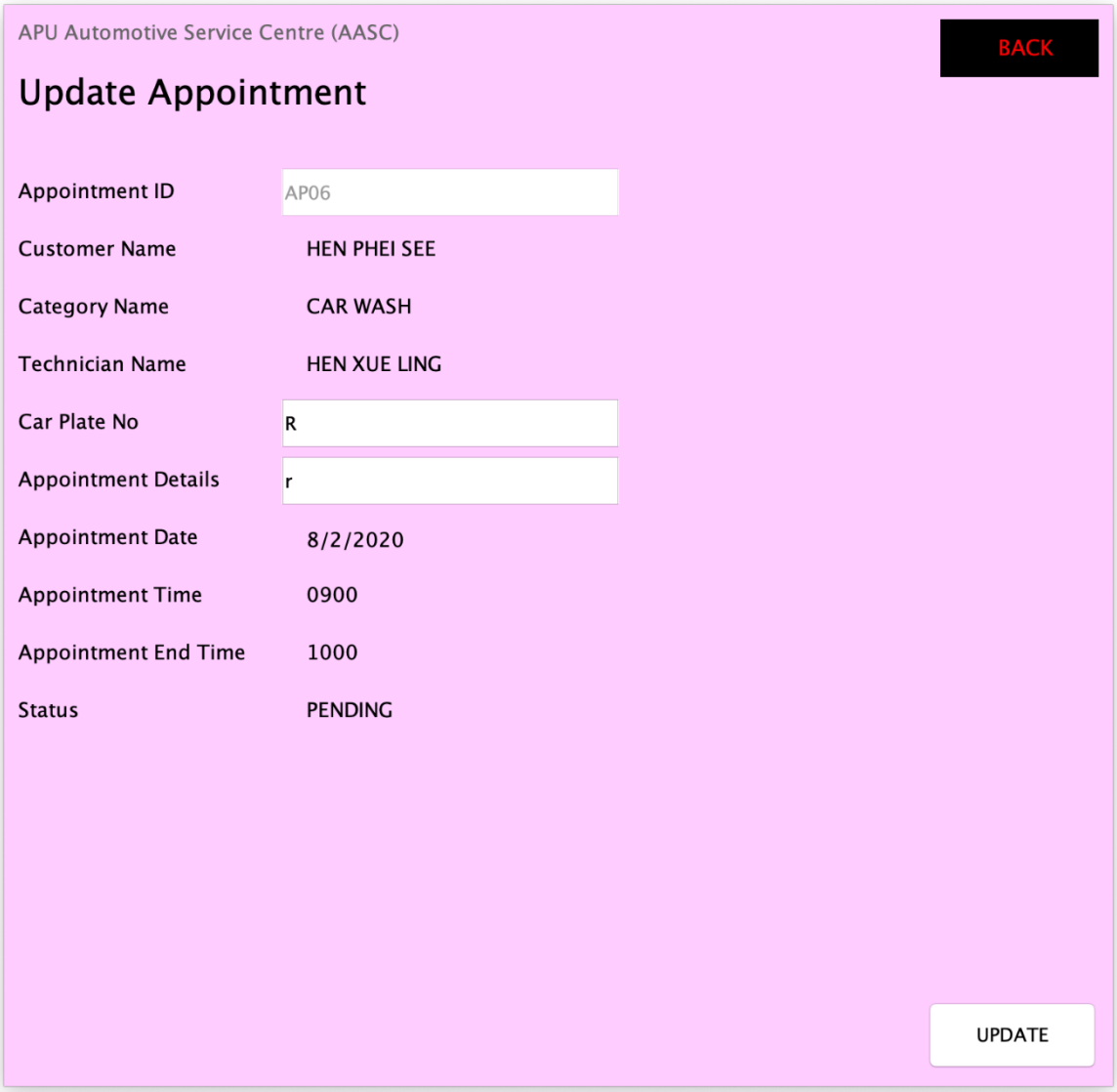



*Figure 68: Update Appointment Page - Technician*

The figure (Figure 68) above is update appointment page for technician, there are ten criteria to be store in database which is Appointment ID, Customer Name, Category Name, Technician Name, Car Plate No, Appointment Description, Appointment Date, Start Time, End Time and Status, only two criteria can be edit which is the Car Plate No and Appointment Details, and once the customer pay for their payment then status only change tod the paid. The top-right corner is the BACK button which led technician back to the previous page.

### 4.3.12 Add Payment Page - Technician

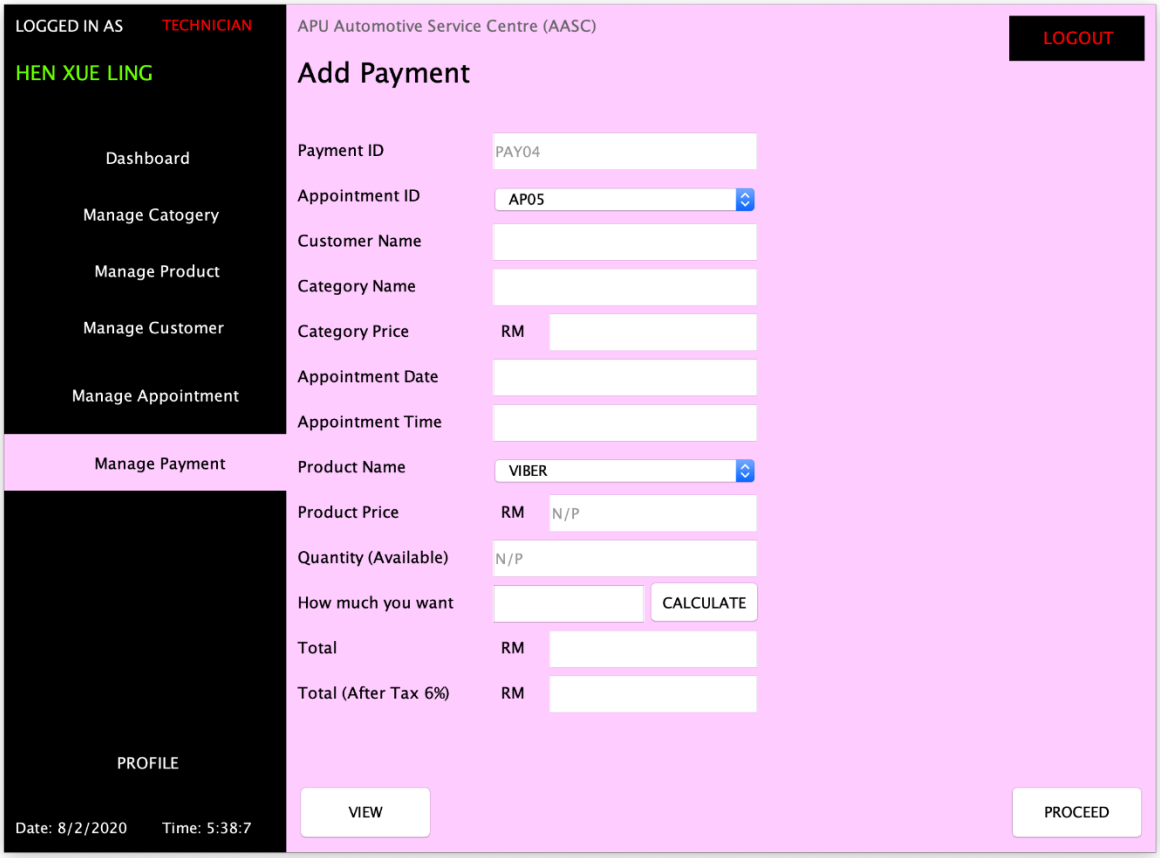



*Figure 69: Add Payment Page - Technician*

The figure (Figure 69) above is add payment page for technician, thirteen criteria that need to be store into database which is Payment ID, Appointment ID, Customer Name, Category Name, Category Price, Appointment Date, Appointment Time, Product Name, Product Price, Quantity, How much you want , Total and Total after Tax (6%). Technician can choose their payment based on the appointment ID. Besides, if there are some product that deactivated, combo box will not display the actual product, and once technician has proceed to the quantity part can be click on CALCULATE button that can be calculate the amount that need to be pay before tax, and the total amount after tax can be auto generating together by AASC system. On the bottom of the page contain two buttons which is VIEW and PROCEED. View is lead technician to the view payment table, and proceed button lead technician to the next payment procedure page. The top-right corner is the LOGOUT button which will log the technician out. There are several buttons on left navigation bar, which will lead the technician to the page respectively, which include Dashboard, Manage Category, Manage Product, Manage Customer, Manager Appointment, and eventually profile.

### 4.3.13 View Payment Page - Technician


*Figure 70: View Payment Page - Technician*

The figure (Figure 70) above is the view payment page for technician, on the top on this page is the search function presented as blank text field, technician can search payment based on customer name. Below that. There is an overall view payment table and shows up thirteen criteria which is Payment ID, Appointment ID, Customer Name, Category Name, Category Price, Appointment Date, Appointment Time, Product Name, Product Price, Product Quantity, Dated Added, Total and Status. Besides, PRINT button which located at the right bottom which can print the selected row to pdf file. The top-right corner is the LOGOUT button which will log the technician out.

4.3.14 Select Payment Page - Technician

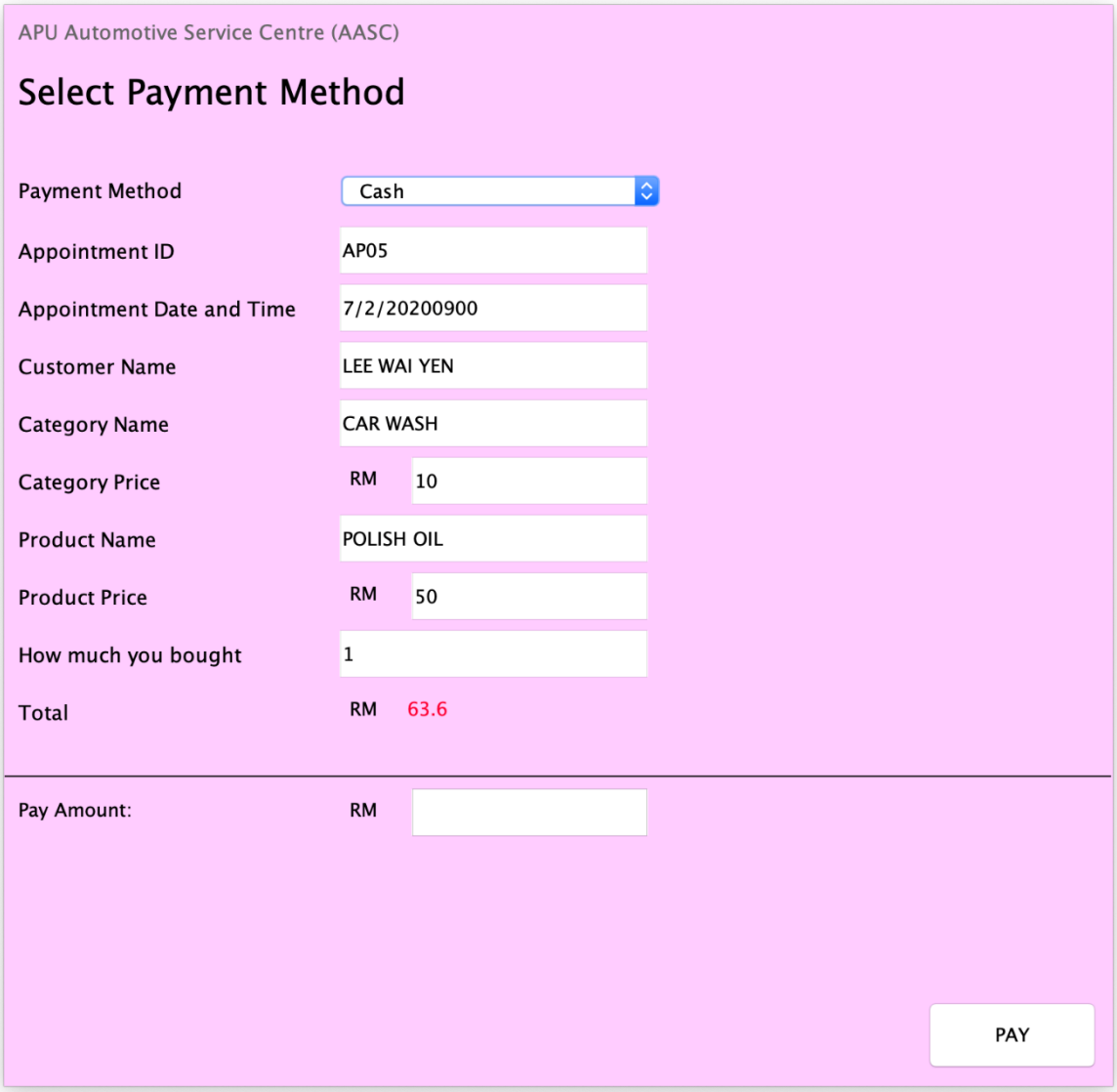



*Figure 71: Select Payment Page - Technician*

The figure (Figure 71) above is select payment page for technician, there are eleven component to be display in this page which contained Payment Method, Appointment ID, Appointment Date and Time, Customer Name, Category Name, Category Price, Product Name, Product Price, How much you bought, Total and Payment Amount. At the bottom of this page is the PAY button, this button can be calculating the pay amount is that greater than the total. If yes can be proceed, if no will be show up the alert message to technician.

### 4.3.15 Provide Feedback Page – Technician

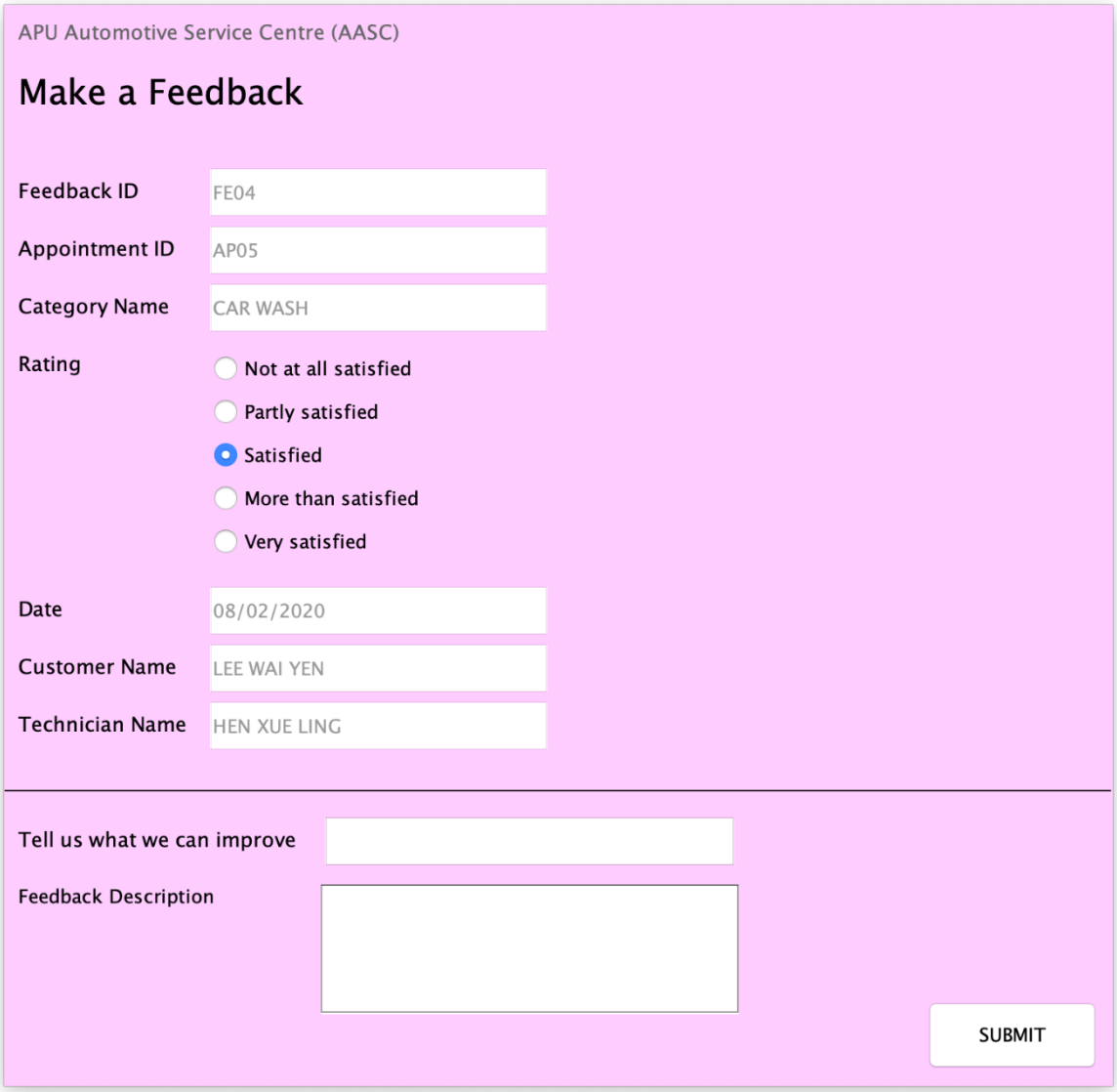



*Figure 72: Provide Feedback Page - Technician*

The figure (Figure 72) above is provide feedback page for technician, there are nine criteria that require to be store in database which is Feedback ID, Appointment ID, Category Name, Rating, Date, Customer Name, Technician Name, tell us what we can improve and Feedback Description. Feedback ID which is auto generating by AASC system, and Appointment ID, Category Name, Date, Customer Name, Technician Name will be catch data from the database that displayed in the text field. The result from rating will be affected the dashboard pie chart which is the report. On the right bottom of this page contained the SUBMIT button which is the save function. This page is compulsory to be submit after payment.

### 4.3.16 Profile Page – Technician

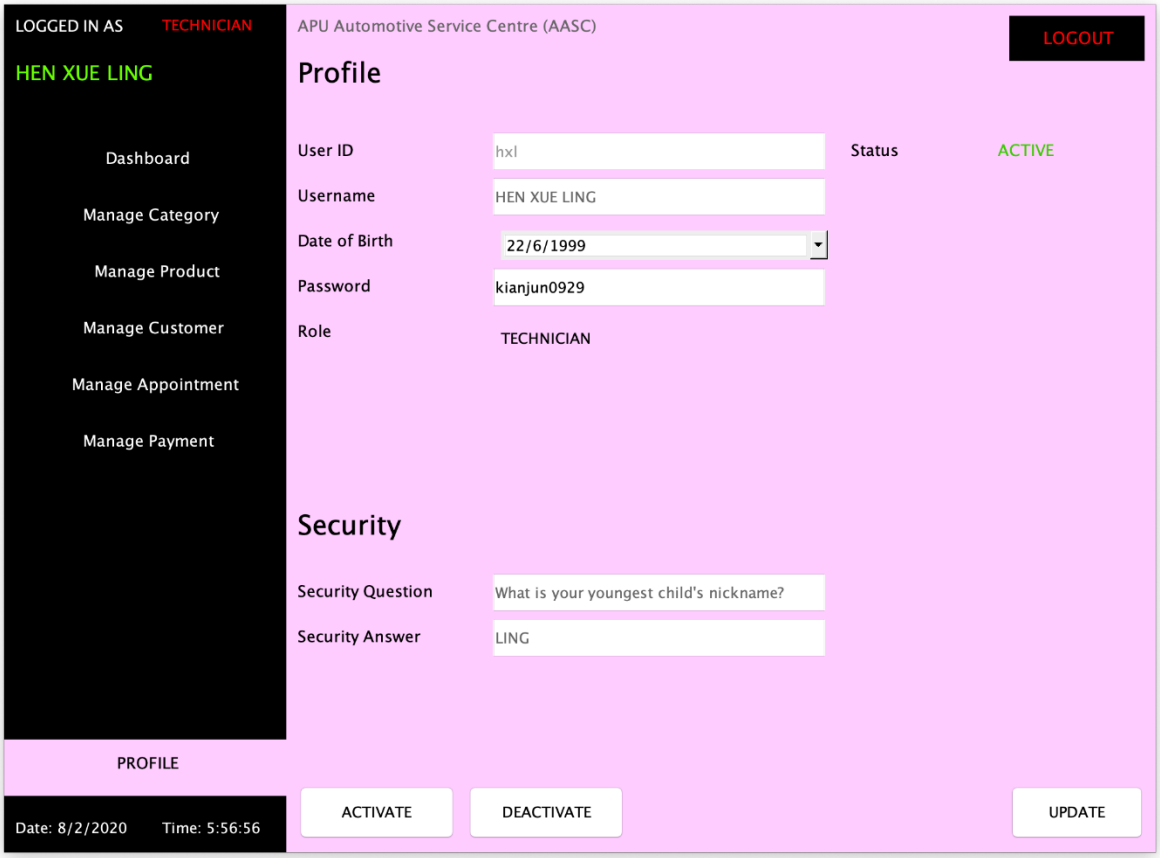



*Figure 73: Profile Page - Technician*

The figure (Figure 73) above is profile page for technician, there are eight criteria that need to be store in database, which is the User ID, Username, Date of Birth, Password, Role, Security Question and Security Answer. There is only one details cannot be change which is the personal role. On the bottom of the page contain three buttons which is ACTIVATE, DEACTIVATE and UPDATE, technician status is based on the activate and deactivate buttons, and update button can which is the save button. The top-right corner is the LOGOUT button which will log the technician out. There are several buttons on left navigation bar, which will lead the technician to the page respectively, which include Dashboard, Manage Category, Manage Product, Manage Customer Manager Appointment and Manage Payment.

## 4.4 Administrator

### 4.4.1 Dashboard Page – Administrator

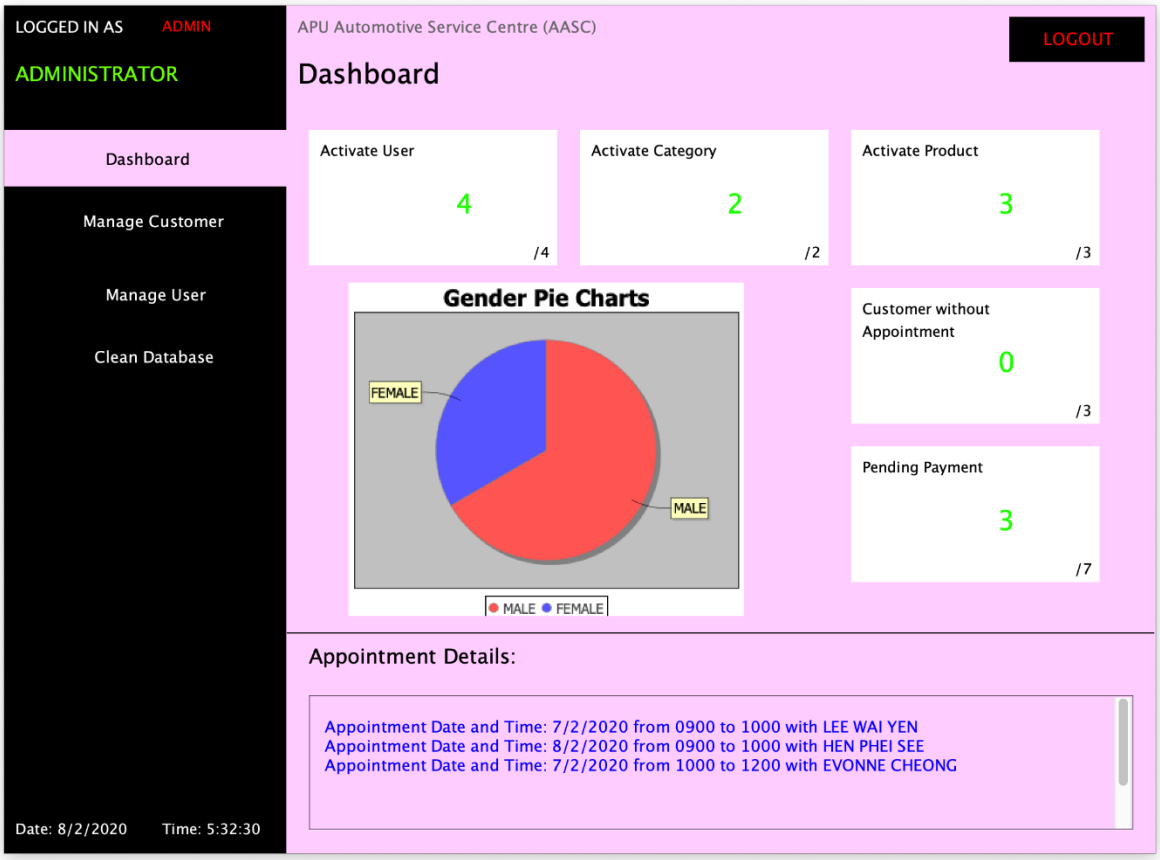



*Figure 74: Dashboard Page - Administrator*

The figure (Figure 74) above is dashboard page for administrative. There are five panels in the dashboard page. Five panels include activate user, activate category, and activate product, customer without appointment and pending payment. There also a pie chart that shows gender report. On the bottom of this page that contained the appointment details which show all technician appointment details. The top-right corner is the LOGOUT button which will log the admin out. There are several buttons on left navigation bar, which will lead the admin to the page respectively, which include Dashboard, Manager Customer, Manage User and Clean Database.

### 4.4.2 View Customer Page - Administrator

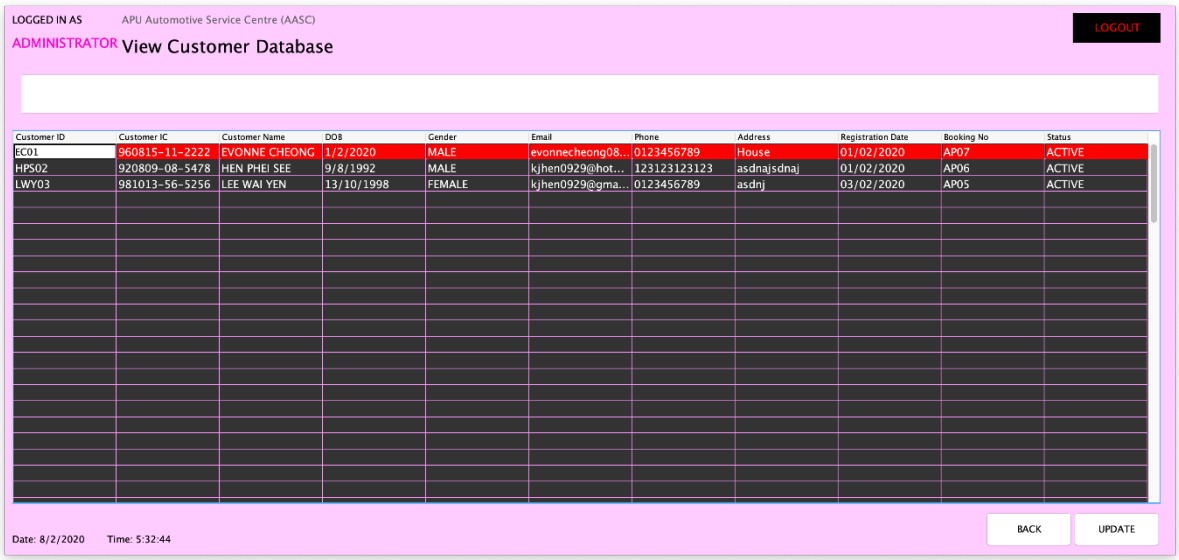



*Figure 75: View Customer Page - Administrator*

The figure (Figure 75) above is view customer page for administrative. On the top of this page, there are a search blank field which let admin to search the customer database. On the bottom of this page there are two buttons which is BACK and UPDATE button. Back button led admin to the previous page. Update button is based on the selected row click on UPDATE button will be catch all the row data. The top-right corner is the LOGOUT button which will log the admin out.

### 4.4.3 Update Customer Page - Administrator

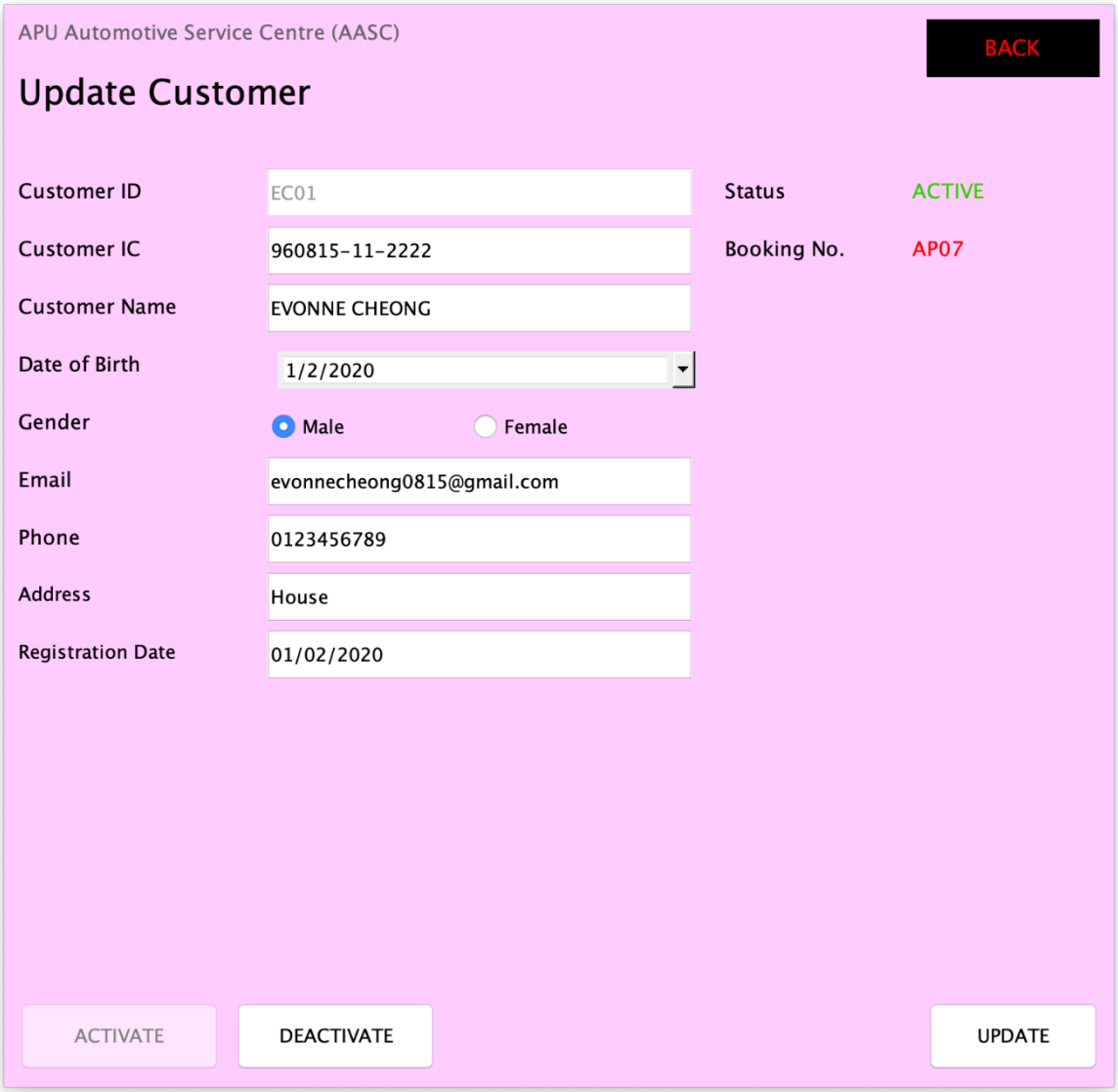



*Figure 76: Update Customer Page - Administrator*

The figure (Figure 76) above is update customer page for administrative, there are eleven criteria need to get from customer. Which is Customer ID, Customer IC, Customer Name, Date of Birth, Gender, Email, Phone, and Address, Registration Date, Status and Booking Number. Customer ID and booking number is auto generating by system. On the top right corner there are BACK button to led admin to the previous page. On the bottom of this page contained three buttons which is ACTIVATE DEACTIVATE and UPDATE.

### 4.4.4 Manage User Page - Administrator

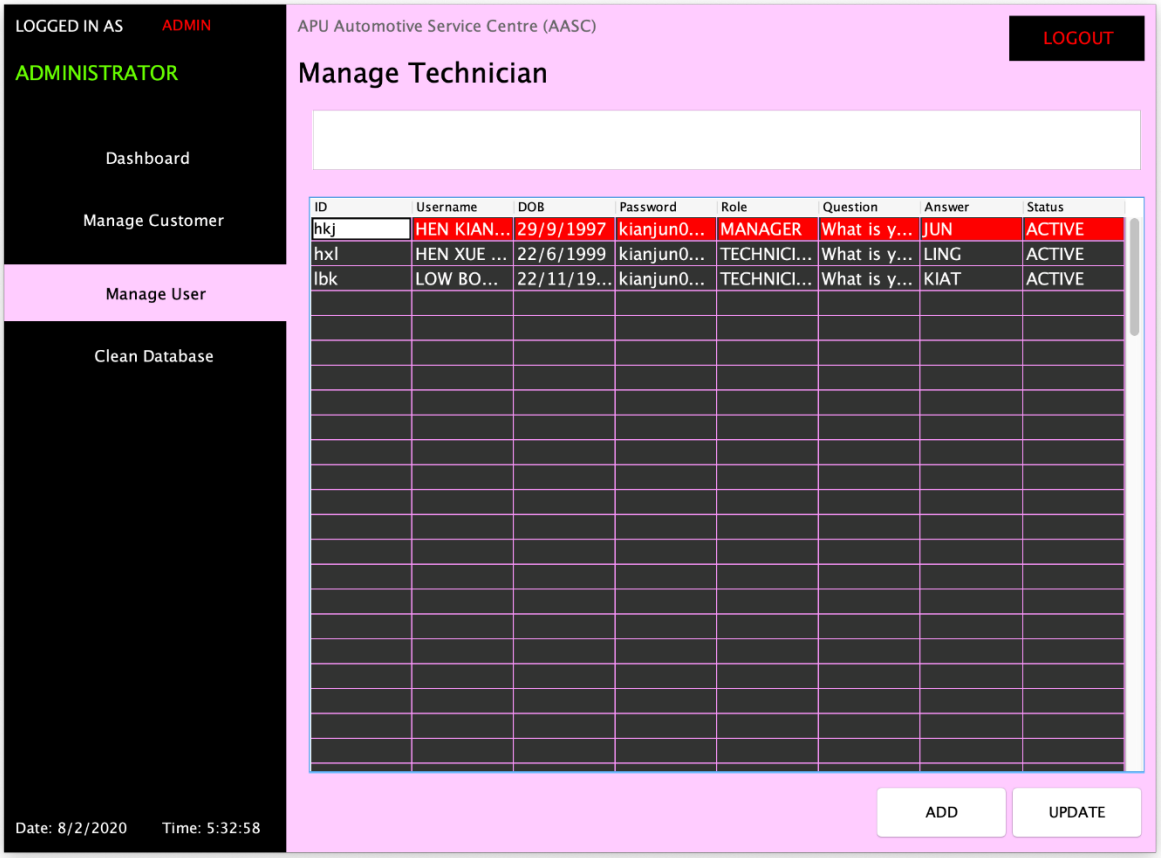



*Figure 77: Manage User Page - Administrator*

The figure (Figure 77) above is manage user page for administrative. On the top of this page there is a search field presented as blank text field. There only can be search by the technician username. In the table, there are eight criteria that stored in database which display as table perform. Below the table there are two buttons which is ADD and UPDATE. Admin can add manager and technician. Update button that perform that selected row data which catch the data to the respectively page. The top-right corner is the LOGOUT button which will log the admin out. There are several buttons on left navigation bar, which will lead the admin to the page respectively, which include Dashboard, Manager Customer and Clean Database.

### 4.4.5 Add User Page - Administrator

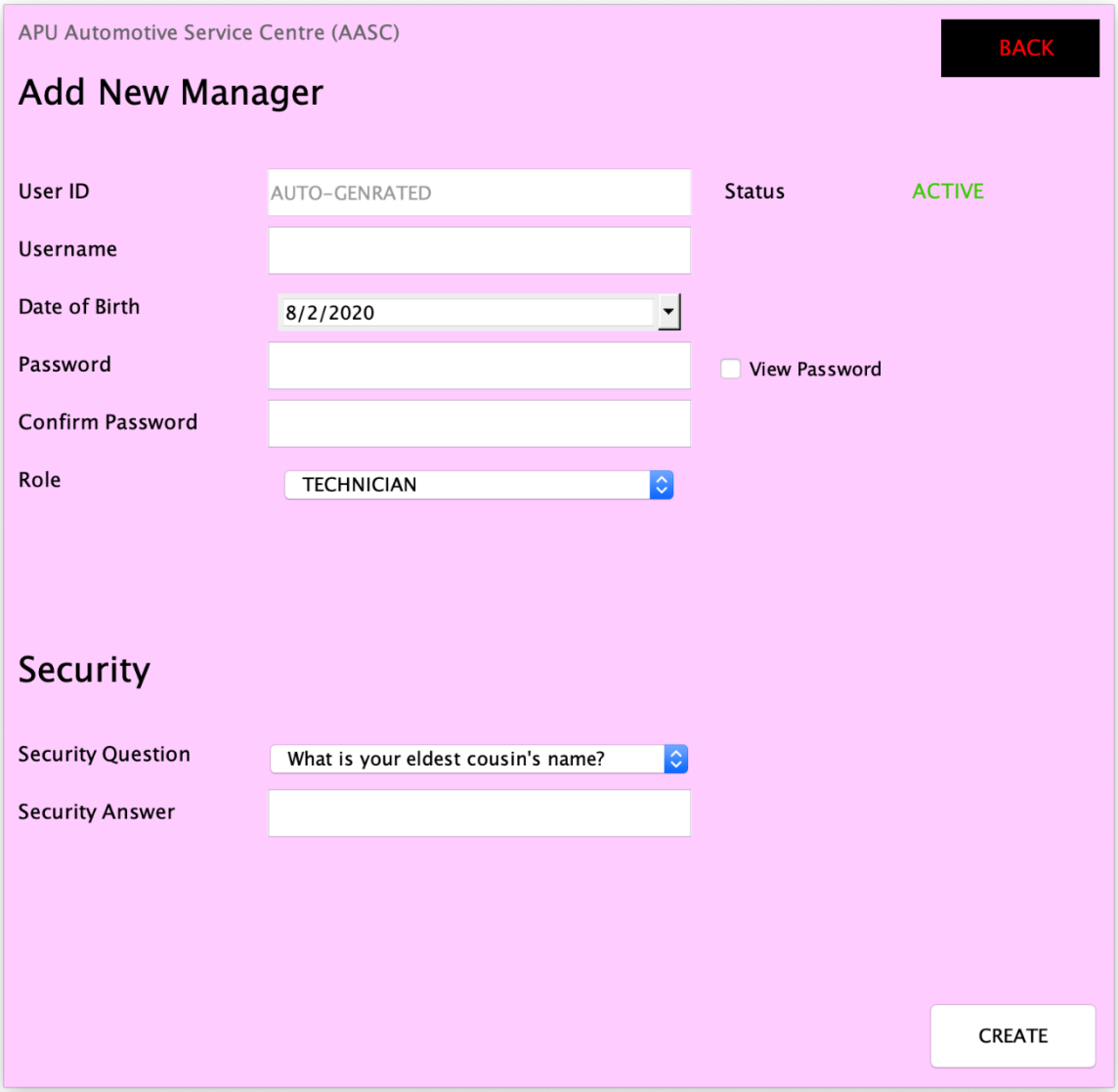



*Figure 78: Add User Page - Administrator*

The figure (Figure 78) above is add user page for administrative, there are nine components need to be store in the database, which is User ID, Username, Date of Birth, Password, Confirm Password, Role, Security Question, Security Answer and Status. User ID is based on the username and made by auto generating. On the right bottom which is CREATE button. On the top right corner there are BACK button to led admin to the previous page.

4.4.6 Update User Page - Administrator

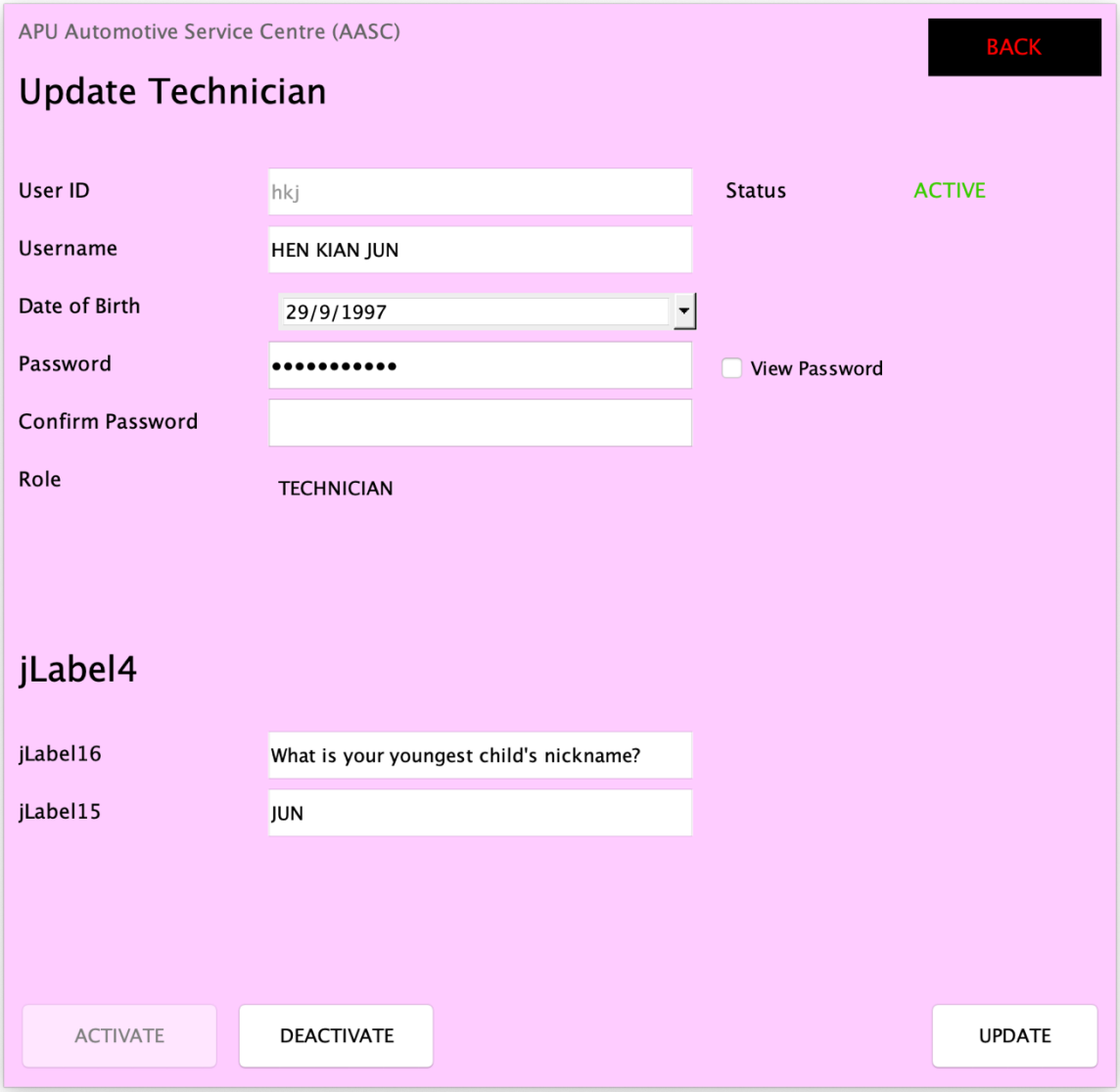



*Figure 79: Update User Page - Administrative*

The figure (Figure 79) above is update user page for administrative. There are nine criteria need to catch data from database and pasted into the specific text filed, criteria included User ID, Username, Date of Birth, Password, Confirm Password, Role, Security Question and Security Answer. There are three buttons located at bottom of the page which is ACTIVATE, DEACTIVATE and UPDATE. The reason why ACTIVATE button not available is because the current technician is in activated situation.

### 4.4.7 Clean Database Page - Administrator

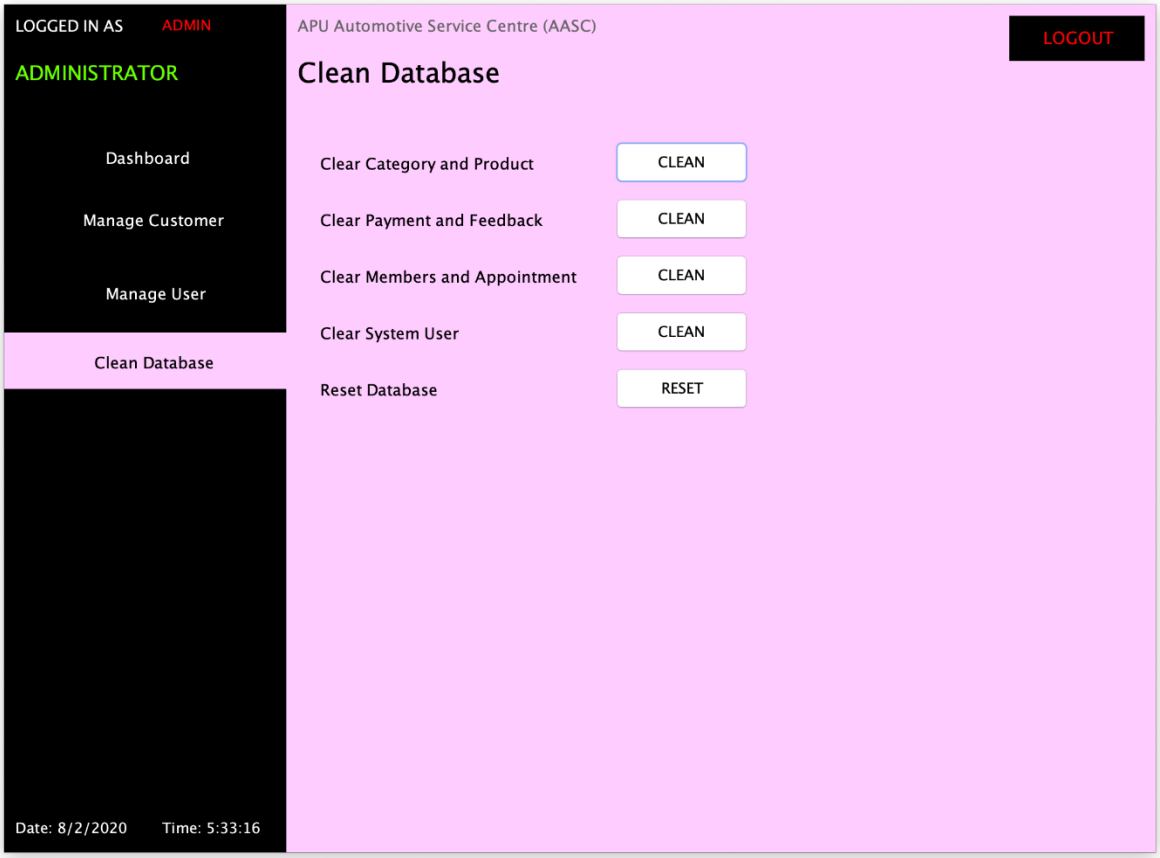



*Figure 80: Clean Database Page - Administrator*

The figure (Figure 80) above is clean database page for administrative, there are four CLEAN buttons and one reset button, which perform the several databases can clear function which is category and product, payment and feedback, members and appointment, system user, and reset database which used reset button. The top-right corner is the LOGOUT button which will log the admin out. There are several buttons on left navigation bar, which will lead the admin to the page respectively, which include Dashboard, Manager Customer and Manage User.

# 5.0 Additional Features

I. Auto-generated ID

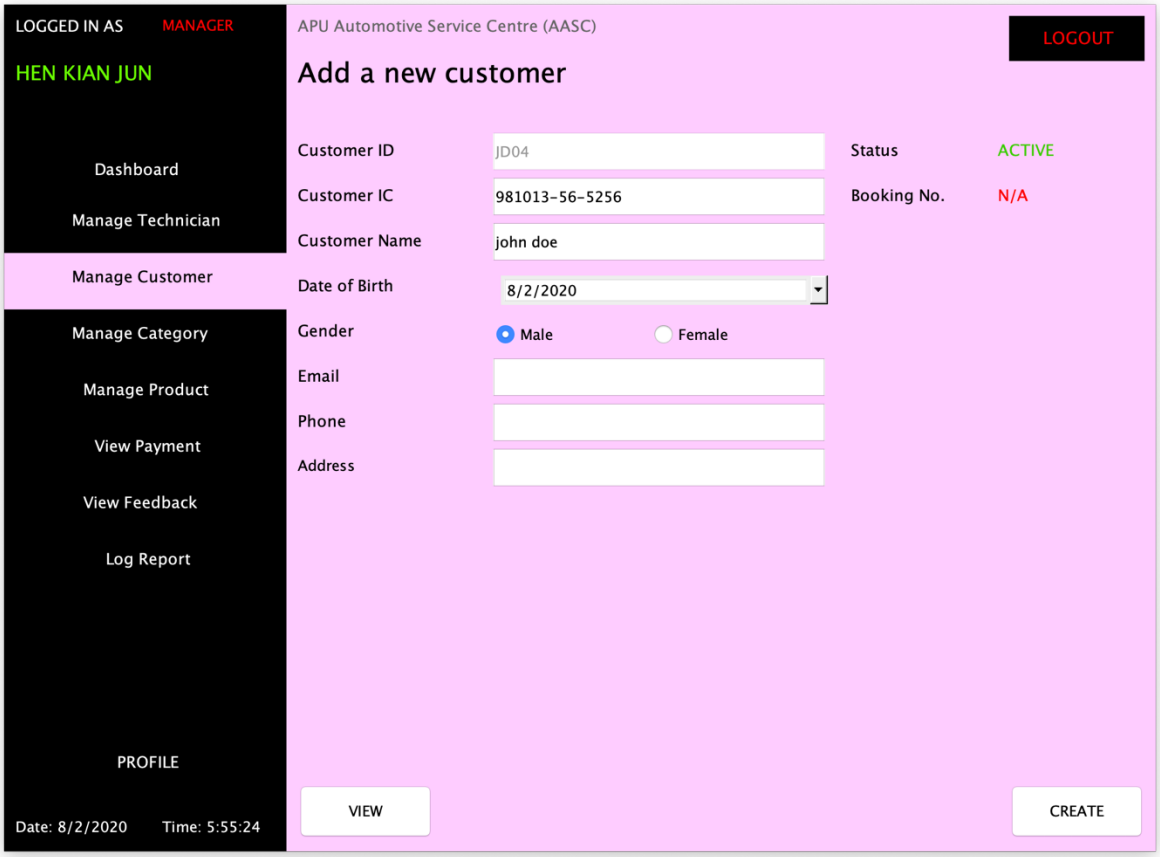



*Figure 81: Auto-generated ID*

Auto-generated ID (Figure 81) for system user (manager and technician), customer, category, product, appointment, payment, and feedback is one of the additional features that is implemented into the proposed system. For system user, username will be auto-generated according to the initial letters of the user's full name. For customer, the customer ID will be generated from the initial letters of the customer name and followed by the numbers, the numbers are according to the number of records in the system. Auto-generated ID for category, product, appointment, payment, feedback is based on the number of the records in the database. Below is one of the examples of auto-generating an ID for the system user, customer, category, product, appointment, payment, and feedback. All auto-generated ID is placed in Initials Class.

```
public String memberid(String name) {
    String username;
    String initials="";
    String[] parts = name.split(" ");
    char initial;
    for (String part : parts) {
            initial = part.charAt(0);
            initials= initials+initial;
    }
    username = initials.toUpperCase()+000+idnumber2();
    return username;
}

private int idnumber2 () {
    int num = 1;
    da.setFileName("Customer.txt");
    ArrayList<String> data = da.readAll();
    for(String rec: data ) {
        String[] split = rec.split("\\|");
        num++;
    }
    return num;
}
```

*Figure 82: Auto-generated ID sample code*

II. Password Retrieval

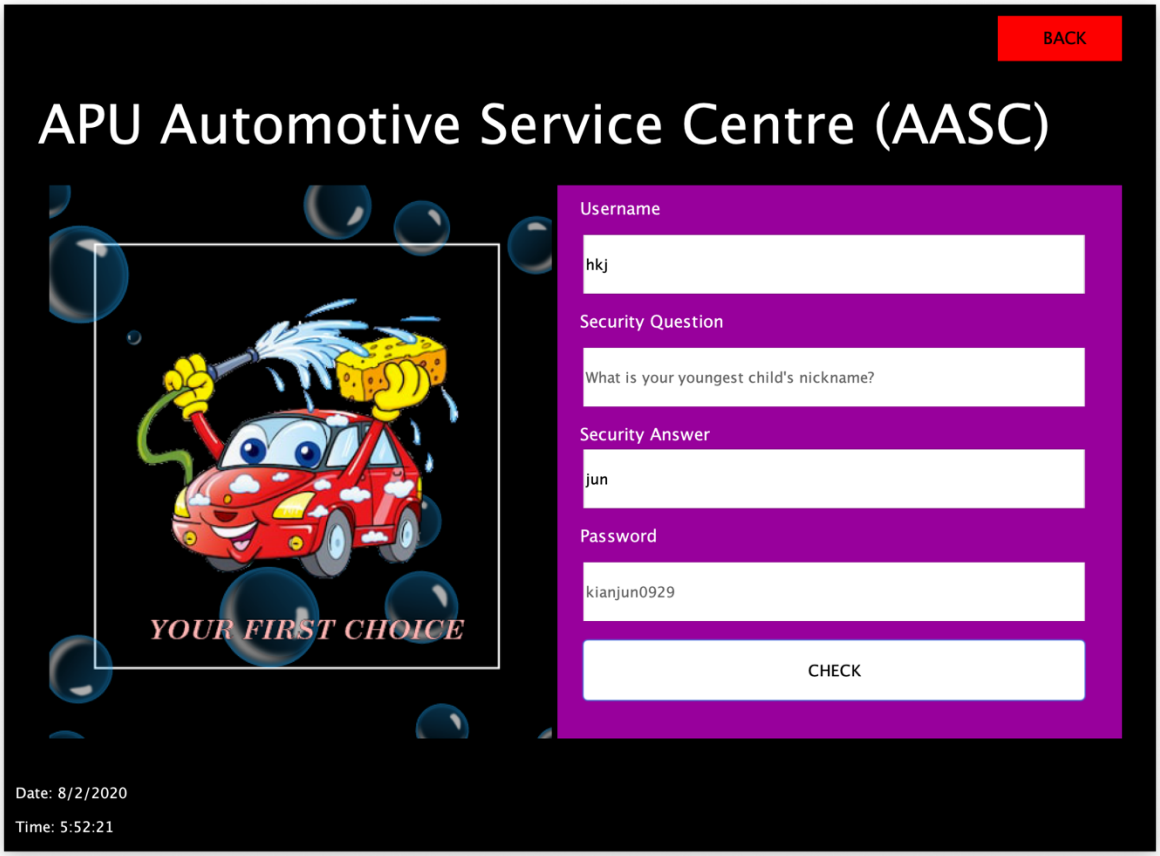



*Figure 83: Password Retrieval*

The figure (Figure 83) above is the password retrieval for system user (manager and technician). The user must key in the username which is auto generated in the process of registering into the proposed system. After providing the username, the proposed system will check with the database if the username matches, it will be filling the security question if the username is matched, or it will prompt a message to the users that to enter a correct username. The user should provide security answer and click on CHECK button. Security answer should be provided by the user to retrieve the password. If the username and security answer are correct, the password will be returned in password text fields.

```
private void txtUsernameFocusLost(java.awt.event.FocusEvent evt) {

    String Username = txtUsername.getText();
    Admin admin = new Admin();
    admin.setAcc(new Account(Username));

    Account account = new Account();
    if (account.forgetPassword(Username)) {
        txtQuestion.setText(Global.question);
    } else {
        JOptionPane.showMessageDialog(rootPane, "Please enter a valid username.");
    }
}
```

*Figure 84: Password Retrieval*

```
public boolean forgetPassword(String username) {
    boolean success = false;
    da.setFileName("user.txt");
    ArrayList<String> data = da.readAll();
    for (String rec : data) {
        String[] split = rec.split("\\|");
        if (split[0].equals(username)) {
            //if the given username is matched in the system, assign the question, answer, and password to globa
            Global.question = split[5];
            Global.answer = split[6];
            Global.password = split[3];
            success = true;
            break;
        }
    }
    return success;
}
```

*Figure 85: Forget Password sample code*

III. Database Cleansing

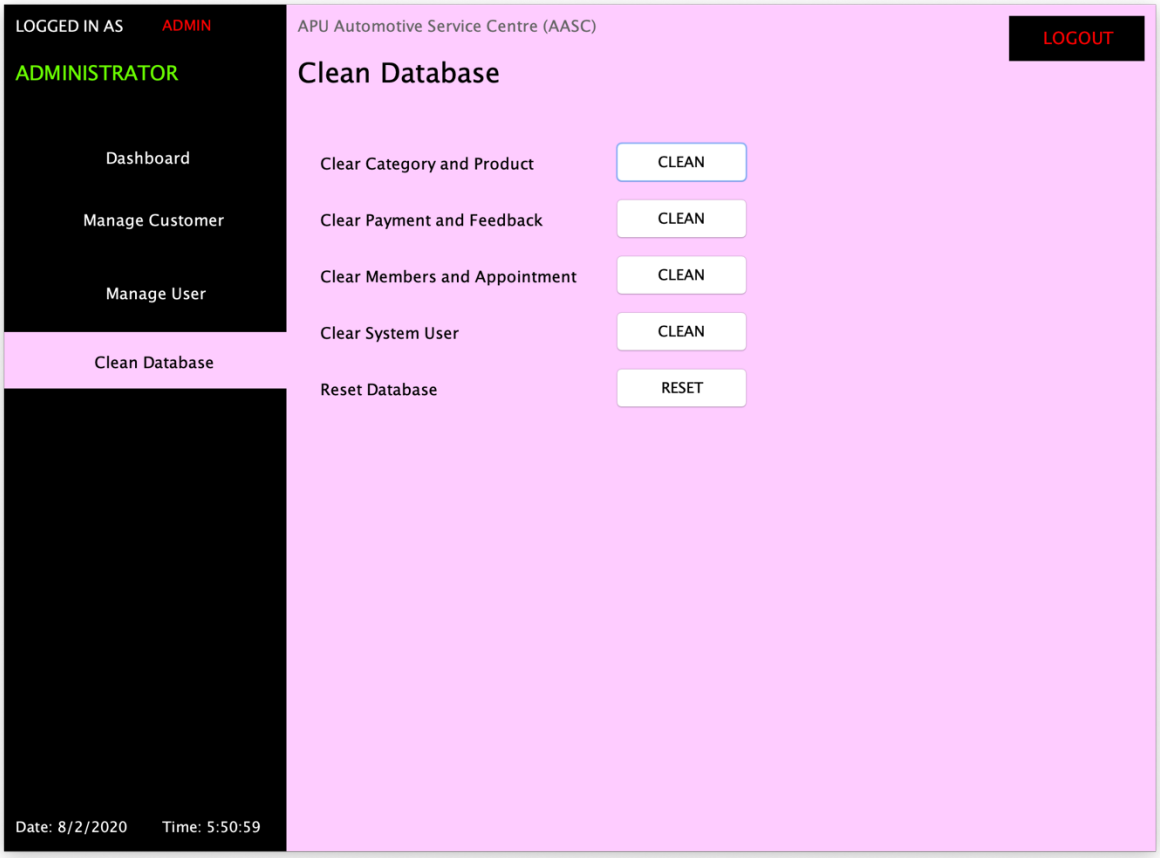



*Figure 86: Database Cleansing*

The figure (Figure 86) above is the database cleansing page for admin. Five buttons that are available for admin. Each button will perform cleansing database action. The first button is to clean category and product table, the second button is to clean payment and feedback table, the third button is to clean member and appointment table, the fourth button is to clean system user table, whereas the last button is to reset the whole database. When any of five buttons is being clicked, a message box will prompt to admin and the admin would need to provide an authorization account to clean the specific table or database.

```
public void Delete(){
    if(this.fileName.exists()) {
        this.fileName.delete();
        try {
            this.fileName.createNewFile();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

*Figure 87: Clean Database sample code (DataAccess class)*

```
public void ClearProductDatabase() {
    da.Delete();
    da.setFileName("Product_activities.txt");
    da.Delete();
}
```

*Figure 88: Clean Database sample code (Product class)*

IV. Bar Charts

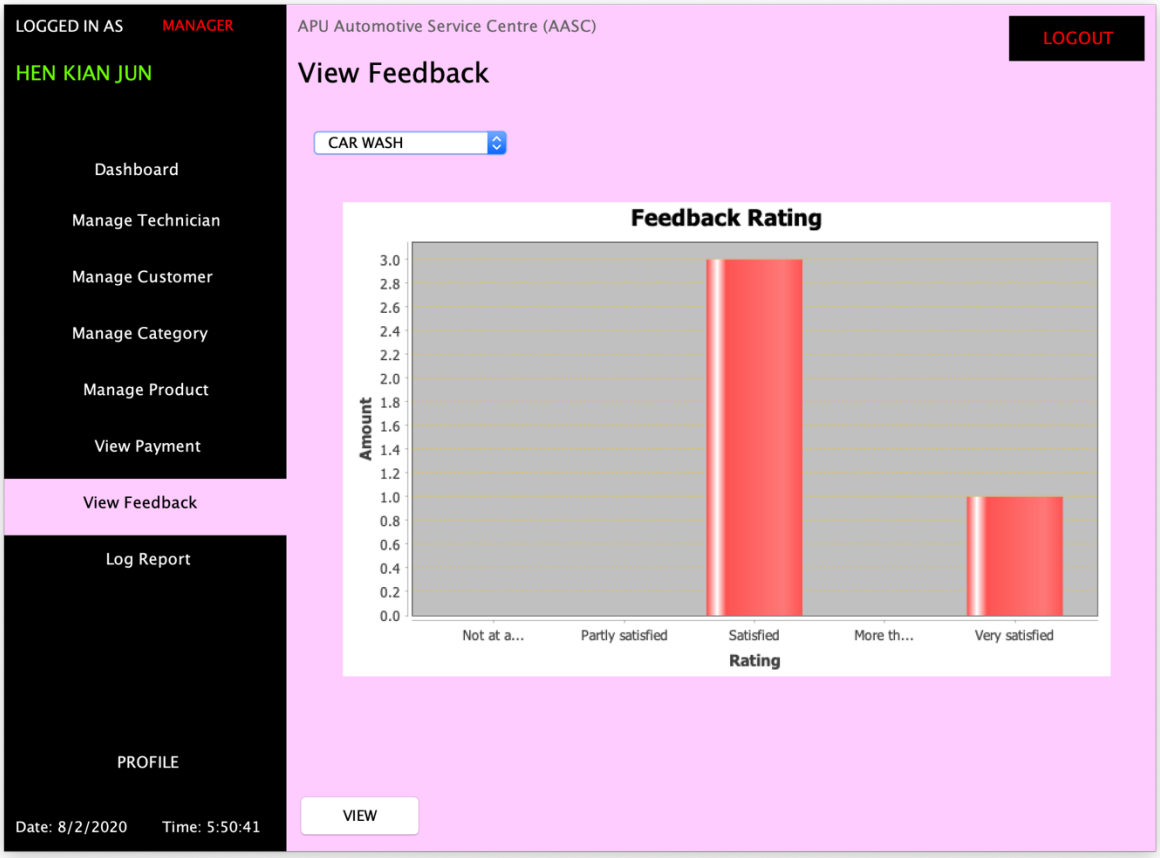



*Figure 89: Bar Charts*

The figure (Figure 89) above is the Feedback page for manager. Manager can view the feedback in a bar chart form by selecting the existing category which is available. By plotting the data in a graphic form, will bring more understanding to the manager.

```
public ChartPanel GetFeedbackBarCharts(String CategoryName) {
    da.setFileName("Feedback.txt");
    ArrayList<String> data = da.readAll();
    int ones = 0;
    int twos = 0;
    int threes = 0;
    int fours = 0;
    int fives = 0;
    for (String rec : data) {
        String[] split = rec.split("\\|");
        if (split[2].equals(CategoryName)) {
            int rate = Integer.parseInt(split[4]);
            switch (rate) {
                case 1:
                    ones++;
                    break;
                case 2:
                    twos++;
                    break;
                case 3:
                    threes++;
                    break;
                case 4:
                    fours++;
                    break;
                default:
                    fives++;
                    break;
            }
        }
    }
    DefaultCategoryDataset barchartdata = new DefaultCategoryDataset();
    barchartdata.setValue(ones, "Rating", "Not at all satisfied");
    barchartdata.setValue(twos, "Rating", "Partly satisfied");
    barchartdata.setValue(threes, "Rating", "Satisfied");
    barchartdata.setValue(fours, "Rating", "More than satisfied");
    barchartdata.setValue(fives, "Rating", "Very satisfied");
    JFreeChart barchart = ChartFactory.createBarChart("Feedback Rating", "Rating", "Amount", barchartdata, PlotOrientation.VERTICAL, false, true, false);
    CategoryPlot barchart1 = barchart.getCategoryPlot();
    barchart1.setRangeGridlinePaint(Color.ORANGE);
    ChartPanel barPanel1 = new ChartPanel(barchart);
    return barPanel1;
}
```

*Figure 90: Bar Charts sample code*

V. Area Charts

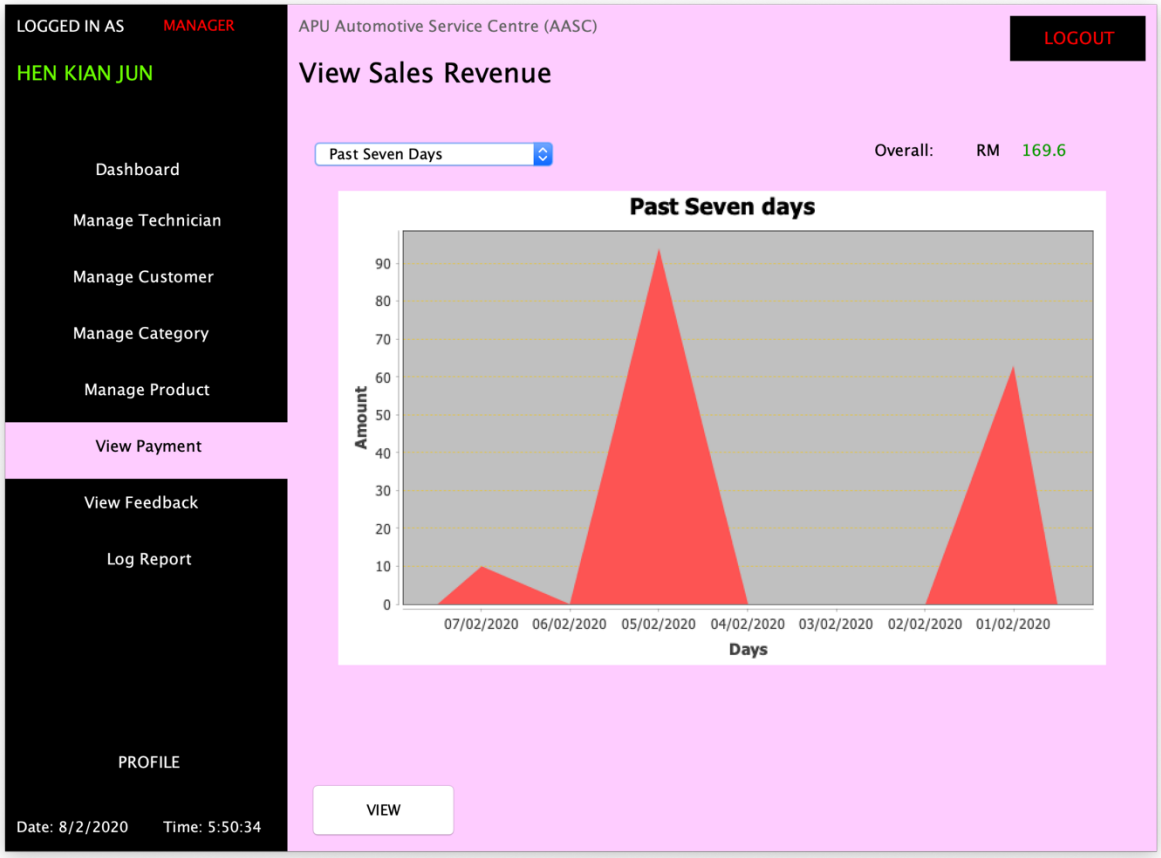



*Figure 91: Area Charts*

The figure (Figure 91) above is View Sales Revenue report page for manager. Manager can view the sale revenue in an area chart by selecting either three days ago or seven days ago. Plotting the value to the area chart, it allows manager to keep trace on the previous transaction easily.

```
public ChartPanel GetPaymentCharts3(int num) {
    String pattern = "dd/MM/yyyy";
    String[] date = new String[num];
    Calendar c = Calendar.getInstance();
    for (int i = 0; i < date.length; i++) {
        c.add(Calendar.DAY_OF_MONTH, -1);
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);
        date[i] = sdf.format(c.getTime());
    }

    da.setFileName("Payment.txt");
    ArrayList<String> data = da.readAll();
    int dayone = 0, daytwo = 0, daythree = 0;
    for (String rec : data) {
        String[] split = rec.split("\\|");
        double total = Double.parseDouble(split[11]);
        if (split[10].equals(date[0])) {
            dayone += total;
        } else if (split[10].equals(date[1])) {
            daytwo += total;
        } else if (split[10].equals(date[2])) {
            daythree += total;
        }
    }
    DefaultCategoryDataset barchartdata = new DefaultCategoryDataset();
    barchartdata.setValue(dayone, "Amount", date[0]);
    barchartdata.setValue(daytwo, "Amount", date[1]);
    barchartdata.setValue(daythree, "Amount", date[2]);
    JFreeChart barchart = ChartFactory.createAreaChart("Past Three days", "Days", "Amount", barchartdata, PlotOrientation.VERTICAL, false, true, false);
    CategoryPlot barchart1 = barchart.getCategoryPlot();
    barchart1.setRangeGridlinePaint(Color.ORANGE);
    ChartPanel barPanel1 = new ChartPanel(barchart);
    return barPanel1;
}
```

*Figure 92: Area Chart sample code*

## VI. Auto-Emailing

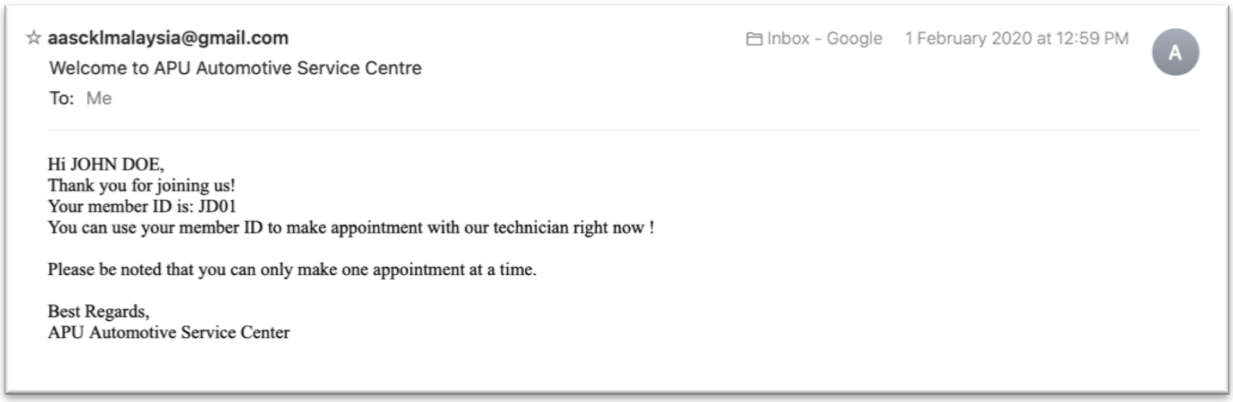



*Figure 93: Auto-Emailing*

The figure (Figure 93) above is the customer receives an auto-emailing page. A customer will be receiving an email when registering into the proposed system.

```
public static void sendEmail(String recipient, ArrayList CustomerDetails) throws Exception {
    System.out.println("Preparing to send email");
    Properties properties = new Properties();
    properties.put("mail.smtp.auth", "true");
    properties.put("mail.smtp.starttls.enable", "true");
    properties.put("mail.smtp.host", "smtp.gmail.com");
    properties.put("mail.smtp.port", "587");

    //AASCKLMalaysia@gmail.com
    String myAccountEmail = "AASCKLMalaysia@gmail.com";
    String password = "kianjun0929";


    Session session = Session.getInstance(properties, new Authenticator() {
        @Override
        protected PasswordAuthentication getPasswordAuthentication() {
            return new PasswordAuthentication(myAccountEmail, password);
        }
    });
    Message message = prepareMessage(session, myAccountEmail, recipient, CustomerDetails);

    Transport.send(message);
    System.out.println("Email Sent");
}
```

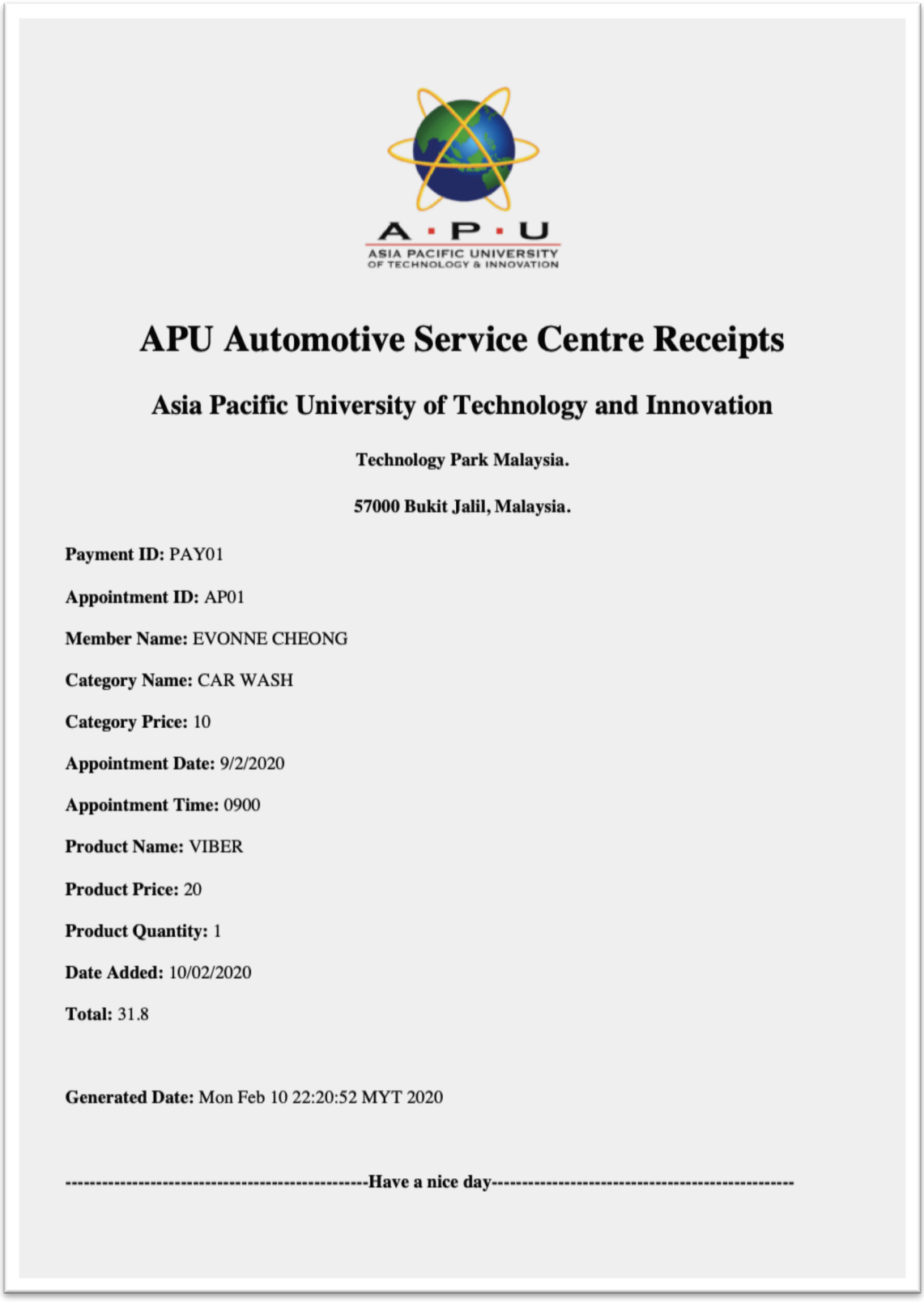
*Figure 94: Send Email sample code*

```
private static Message prepareMessage(Session session, String myAccountEmail, String recipient, ArrayList CustomerDetails) {
    Message message = new MimeMessage(session);
    try {
        message.setFrom(new InternetAddress(myAccountEmail));
        message.setRecipient(Message.RecipientType.TO, new InternetAddress(recipient));
        message.setSubject("Welcome to APU Automotive Service Centre");
        message.setText("Hi " + CustomerDetails.get(2) + ",\nThank you for joining us!\nYour member ID is: " + CustomerDetails.get(0) +" \nYou

        return message;
    } catch (Exception ex) {
        Logger.getLogger(MailUtil.class.getName()).log(Level.SEVERE, null, ex);
    }
    return null;
}
```

*Figure 95: Prepare Message sample code*

VII. Receipts generation



**APU Automotive Service Centre Receipts**

**Asia Pacific University of Technology and Innovation**

**Technology Park Malaysia.**

**57000 Bukit Jalil, Malaysia.**

**Payment ID:** PAY01

**Appointment ID:** AP01

**Member Name:** EVONNE CHEONG

**Category Name:** CAR WASH

**Category Price:** 10

**Appointment Date:** 9/2/2020

**Appointment Time:** 0900

**Product Name:** VIBER

**Product Price:** 20

**Product Quantity:** 1

**Date Added:** 10/02/2020

**Total:** 31.8

**Generated Date:** Mon Feb 10 22:20:52 MYT 2020

----------**Have a nice day**----------

*Figure 96: Receipts Generation*

The figure (Figure 96) above is the sample of receipts for customer. The receipt is generated based on the customer's request. Receipt will be generated from the payment page in Manager or Technician.

```
content.append("<center><img src=\"
content.append("<center><h1>APU Automotive Service Centre Receipts</h1></center>");
content.append("<center><h2>Asia Pacific University of Technology and Innovation<br></h2></center>");
content.append("<center><h4>Technology Park Malaysia.<br></h4></center>");
content.append("<center><h4>57000 Bukit Jalil, Malaysia.<br></h4></center>");
content.append("<p><b>Payment ID: </b>").append(PaymentID).append("</p>");
content.append("<p><b>Appointment ID: </b>").append(AppointmentID).append("</p>");
content.append("<p><b>Member Name: </b>").append(CustomerName).append("</p>");
content.append("<p><b>Category Name: </b>").append(CategoryName).append("</p>");
content.append("<p><b>Category Price: </b>").append(CategoryPrice).append("</p>");
content.append("<p><b>Appointment Date: </b>").append(AppointmentDate).append("</p>");
content.append("<p><b>Appointment Time: </b>").append(AppointmentTime).append("</p>");
content.append("<p><b>Product Name: </b>").append(ProductName).append("</p>");
content.append("<p><b>Product Price: </b>").append(ProductPrice).append("</p>");
content.append("<p><b>Product Quantity: </b>").append(ProductQuantity).append("</p>");
content.append("<p><b>Date Added: </b>").append(DateAdded).append("</p>");
content.append("<p><b>Total: </b>").append(Total).append("</p>");
content.append("<br>");
content.append("<p><b>Generated Date: </b>").append(new Date().toString()).append("</p>");
content.append("<br>");
content.append("<p><b>--------------------------------------------------Have a nice day-------------
```

*Figure 97: Receipts Generation sample code*

## VIII. Pie Charts

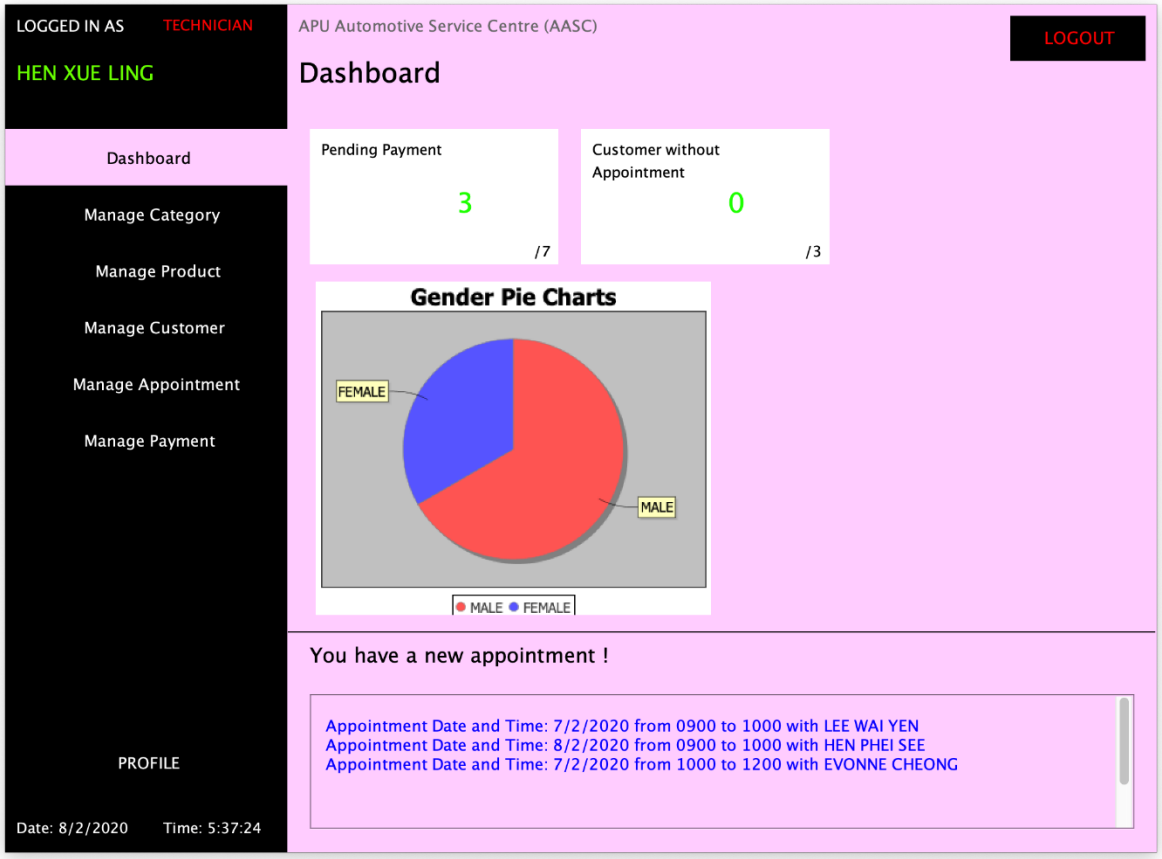



*Figure 98: Gender Pie Charts*

The figure (Figure 98) above is the gender pie charts in Technician and Admin dashboard. Pie charts provides a better visualisation overall record.

```
public ChartPanel GetGenderPieCharts() {
    da.setFileName("Customer.txt");
    ArrayList<String> data = da.readAll();
    int numOfMales = 0, numOfFemales = 0;
    for (String rec : data) {
        String[] split = rec.split("\\|");
        if (split[10].equals("ACTIVE")) {
            if (split[4].equals("MALE")) {
                numOfMales++;
            } else {
                numOfFemales++;
            }
        }

    }
    DefaultPieDataset pieDataSet = new DefaultPieDataset();
    pieDataSet.setValue("MALE", new Integer(numOfMales));
    pieDataSet.setValue("FEMALE", new Integer(numOfFemales));
    JFreeChart chart = ChartFactory.createPieChart("Gender Pie Charts", pieDataSet, true, true, true);
    PiePlot P = (PiePlot) chart.getPlot();
    ChartPanel frame = new ChartPanel(chart);
    frame.setPreferredSize(new java.awt.Dimension(350, 300));
    return frame;

}
```

*Figure 99: Gender Pie Charts sample code*

## IX. Emailing Receipts

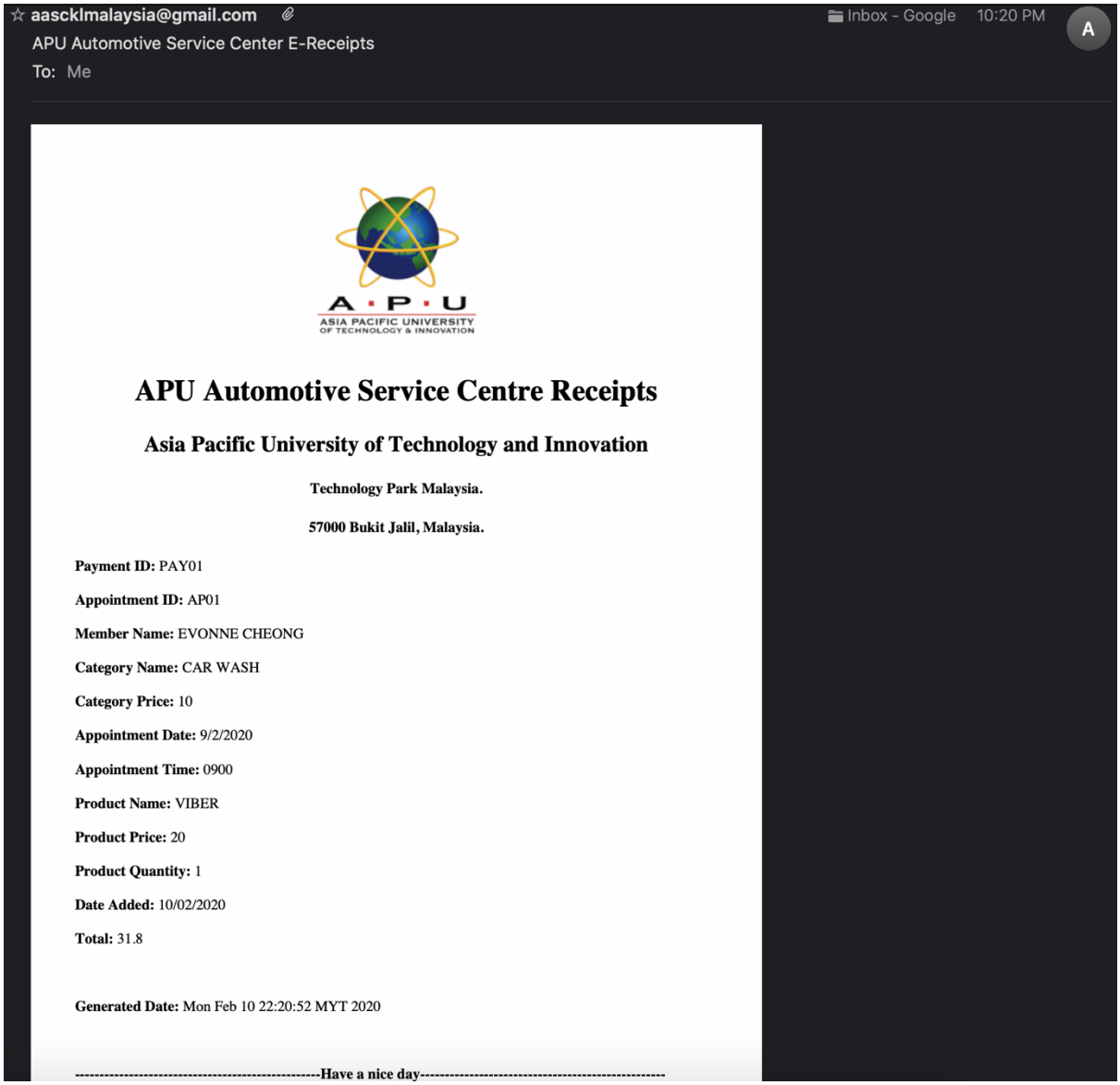



*Figure 100: Email Receipts*

The figure (Figure 100) above is the example of customer receiving an email receipt from APU Automotive Service Centre. Customer can request to send the receipts through Email.

```
public static void sendReceipt(String recipient) throws Exception {
    System.out.println("Preparing to send email");
    Properties properties = new Properties();
    properties.put("mail.smtp.auth", "true");
    properties.put("mail.smtp.starttls.enable", "true");
    properties.put("mail.smtp.host", "smtp.gmail.com");
    properties.put("mail.smtp.port", "587");

    String myAccountEmail = "AASCKLMalaysia@gmail.com";
    String password = "kianjun0929";


    Session session = Session.getInstance(properties, new Authenticator() {
        @Override
        protected PasswordAuthentication getPasswordAuthentication() {
            return new PasswordAuthentication(myAccountEmail, password);
        }
    });
    Message message = prepareReceipt(session, myAccountEmail, recipient);

    Transport.send(message);
    System.out.println("Email Sent");
}
```

*Figure 102: Send Receipts sample code*

```
private static Message prepareReceipt(Session session, String myAccountEmail, String recipient) {
    Message message = new MimeMessage(session);
    try {
        message.setFrom(new InternetAddress(myAccountEmail));
        message.setRecipient(Message.RecipientType.TO, new InternetAddress(recipient));
        message.setSubject("APU Automotive Service Center E-Receipts");
        MimeBodyPart messageBodyPart = new MimeBodyPart();
        Multipart multipart = new MimeMultipart();
        String file = "receipt.pdf";
        String fileName = "receipt.pdf";
        DataSource source = new FileDataSource(file);
        messageBodyPart.setDataHandler(new DataHandler(source));
        messageBodyPart.setFileName(fileName);
        multipart.addBodyPart(messageBodyPart);
        message.setContent(multipart);
        return message;
    } catch (Exception ex) {
        Logger.getLogger(MailUtil.class.getName()).log(Level.SEVERE, null, ex);
    }

    return null;
}
```

*Figure 101: Receipt Preparation sample code*

# 6.0 Conclusion

In this assignment, four object-oriented concepts have been applied in developing APU Automotive Service Centre project. Object-oriented programming provides more efficiency and effective for the programming to build the system that sharing takes place among the class in terms of security and time spent. The proposed system is more secured and validated since the Encapsulation concept is applied to the process of data hiding. Due to the four object-oriented programming, if problems happened in the program, problem can be easily tracked as it is only referenced to the specific class only. Moreover, the reusability of the code allows development team to read and understand the code easily. This is important to apply inheritance concept for features modifying as inheritance allows to change the particular class or method. As a conclusion, object-oriented programming is beneficial to software development team in building APU Automotive Service Centre project in term of modularity, reusability and flexibility.

# List of References

Ahlawat, A., 2020. *Object Oriented Programming.* [Online]
Available at: https://www.studytonight.com/cpp/cpp-and-oops-concepts.php
[Accessed 13 February 2020].

Beal, V., 2017. *polymorphism.* [Online]
Available at: https://www.webopedia.com/TERM/P/polymorphism.html
[Accessed 05 Febuary 2020].

Bhatia, V., 2017. *What are advantages and disadvantages of using inheritance?.* [Online]
Available at: https://practice.geeksforgeeks.org/problems/what-are-advantages-and-disadvantages-of-using-inheritance
[Accessed 06 Febuary 2020].

Guru99, 2020. *What is Interface in Java with Example.* [Online]
Available at: https://www.guru99.com/java-interface.html
[Accessed 13 February 2020].

Janssen, T., 2017. *OOP Concept for Beginners: What Is Abstraction?.* [Online]
Available at: https://dzone.com/articles/oop-concept-for-beginners-what-is-abstraction
[Accessed 06 Febuary 2020].

Janssen, T., 2017. *OOP Concept for Beginners: What is Encapsulation.* [Online]
Available at: https://stackify.com/oop-concept-for-beginners-what-is-encapsulation/
[Accessed 05 Febuary 2020].

Janssen, T., 2020. *OOP Concepts for Beginners: What is Polymorphism.* [Online]
Available at: https://stackify.com/oop-concept-polymorphism/
[Accessed 12 February 2020].

Petkov, A., 2018. *How to explain object-oriented programming concepts to a 6-year-old.* [Online]
Available at: https://www.freecodecamp.org/news/object-oriented-programming-concepts-21bb035f7260/
[Accessed 06 Febuary 2020].

Singh, C., 2014. *Method overriding in java with example.* [Online]
Available at: https://beginnersbook.com/2014/01/method-overriding-in-java-with-example/
[Accessed 12 February 2020].

Singh, C., 2017. *Inheritance in Java Programming with examples.* [Online]
Available at: https://beginnersbook.com/2013/03/inheritance-in-java/
[Accessed 05 Febuary 2020].

Singh, C., 2020. *Method Overloading in Java with examples.* [Online]
Available at: https://beginnersbook.com/2013/05/method-overloading/
[Accessed 12 February 2020].

Stackify, 2017. *What Are OOP Concepts in Java? The Four Main OOP Concepts in Java, How They Work, Examples, and More.* [Online]
Available at: https://stackify.com/oops-concepts-in-java/
[Accessed 05 Febuary 2020].

w3schools.com, 2020. *Java Abstraction.* [Online]
Available at: https://www.w3schools.com/java/java_abstract.asp
[Accessed 13 February 2020].

w3schools.com, 2020. *Java Enums.* [Online]
Available at: https://www.w3schools.com/java/java_enums.asp
[Accessed 13 February 2020].

w3schools.com, 2020. *Java Inheritance (Subclass and Superclass).* [Online]
Available at: https://www.w3schools.com/java/java_inheritance.asp
[Accessed 13 February 2020].

w3schools.com, 2020. *Java Polymorphism.* [Online]
Available at: https://www.w3schools.com/java/java_polymorphism.asp
[Accessed 12 February 2020].

**List of Figures**