Service M

CISOI16-1 PRINCIPLES OF PROGRAMMING

Tanay Jivanbhai Ahir

University of Bedfordshire

Contents

CIS016-1 PRINCIPALES OF PROGRAMING
ASSIGNMENT-2 – INDIVIDUAL PROJECT/CASE STUDY
Service Management System

University of Bedfordshire

INTRODUCTION

The task involves creating a Service Management System (SMS) with JavaFX as its foundation in order to simplify communication between clients and service providers. The SMS provides easy-to-use interfaces for finding services, making appointments, and receiving real-time status updates, all with an emphasis on user experience. Access to customized services and easy navigation are made possible by user-friendly accounts. Tools that help service providers handle reservations, availability, and appointment requests improve operational effectiveness. Users may easily choose services, dates, and hours thanks to the system's dynamic interface, which encourages participation and satisfaction. Modern software development approaches are exemplified by the SMS, which combines JavaFX's flexibility with design principles. Its architecture responds to the changing needs of urban living by embodying resilience, scalability, and responsiveness. This talk explores the ideation, creation, and use of the SMS, highlighting its revolutionary potential to transform service administration and improve user experiences in modern cities. (Gilbert, 2014)

Task Description

Objective:

To design and implement a JavaFX-based Service Management System (SMS) that facilitates seamless interactions between customers and service providers, enhancing user experience and operational efficiency.

Tasks:

1. **User Authentication:**
    - Implement a robust authentication system allowing users to log in as customers or service providers securely.
    - Ensure password encryption and salting for enhanced security.
    - Create error-handling mechanisms for invalid login attempts.
2. **Account Creation:**
    - Develop user-friendly account creation pages for both customers and service providers, capturing essential details such as name, username, password, contact information, and additional details relevant to each user type.
3. **Service Selection:**
    - Design an intuitive interface for customers to browse available services categorized by type and cost.
    - Implement filtering and search functionalities for easy service discovery.
    - Enable customers to view detailed descriptions of services and associated costs before making selections.
4. **Booking Management**:
    - Allow customers to schedule appointments by selecting preferred dates and times, and providing necessary details such as address.

- Develop a booking confirmation mechanism to notify customers of successful bookings.
- Provide options for customers to modify or cancel bookings as needed, with appropriate notifications.

5. **Service Provider Dashboard**:
   - Develop a dashboard for service providers to manage their availability, view incoming bookings, and update their online status.
6. **Alerts and Notifications:**
   - Integrate real-time alerts and notifications to keep users informed of important events such as successful bookings, booking updates, and changes in service provider availability.
   - Utilize push notifications and email alerts for timely communication with users.
7. **Testing and Debugging:**
   - Conduct rigorous testing across various scenarios to identify and address any bugs or issues in the system.
   - Perform unit testing, integration testing, and user acceptance testing to validate system functionality and reliability.

PROJECT PLAN/ SCHEDULE

WEEK	WORK	priority
FIRST Week	Make a blueprint of this assignment and write an overview of the project	High
Second Week	Started making a Service Management system and worked on database connection by using SQLite.	High
Third Week	Started work on my documentation on this project and made Diagrams by using Visual Paradigm.	High
Fourth Week	In the end, I created a video and checked my all work which I completed and I am ready for submission	High

Function Requirements:

The goal of the service management system is to give clients and service providers an intuitive user interface. Clients may sign up, safely log in, and use a variety of services that are divided into

categories for simple browsing. They can view available time slots, set up appointments, and get notifications upon confirmation. Service providers can set service offerings, manage bookings, and update availability through their dashboard. To ensure accuracy and relevance, both users can update their profiles, which include contact details and service details. Error handling techniques and password reset capabilities improve security and usability while guaranteeing a positive user experience. (Gilbert, 2014)

Non-Functional Requirements:

Non-functional requirements include aspects such as usability, performance, security, and reliability in addition to direct functionality. For example, ensuring user friendliness includes intuitive user interfaces and clear instructions to minimize user confusion. Performance considerations include response times and system stability under varying loads. Data security includes measures such as encryption and access control to protect data. Reliability includes availability and error handling to maintain system availability. Scalability requires accommodating increasing user load without compromising performance. All of these non-functional requirements contribute to overall system quality and user satisfaction.

Use Case Diagram

A use case diagram is a graphic representation of how actors, such as users or systems, interact with the functionalities or use cases, that a system offers. It shows how users work with the system to accomplish particular tasks or goals. The use case diagram for this assignment would show how customers and service providers interact with the service management system. Activities like signing in, making reservations, scheduling services, and checking reservations are examples of use cases. Understanding the behavior of the system is aided by the diagram, which gives a general overview of its components and the parties involved.

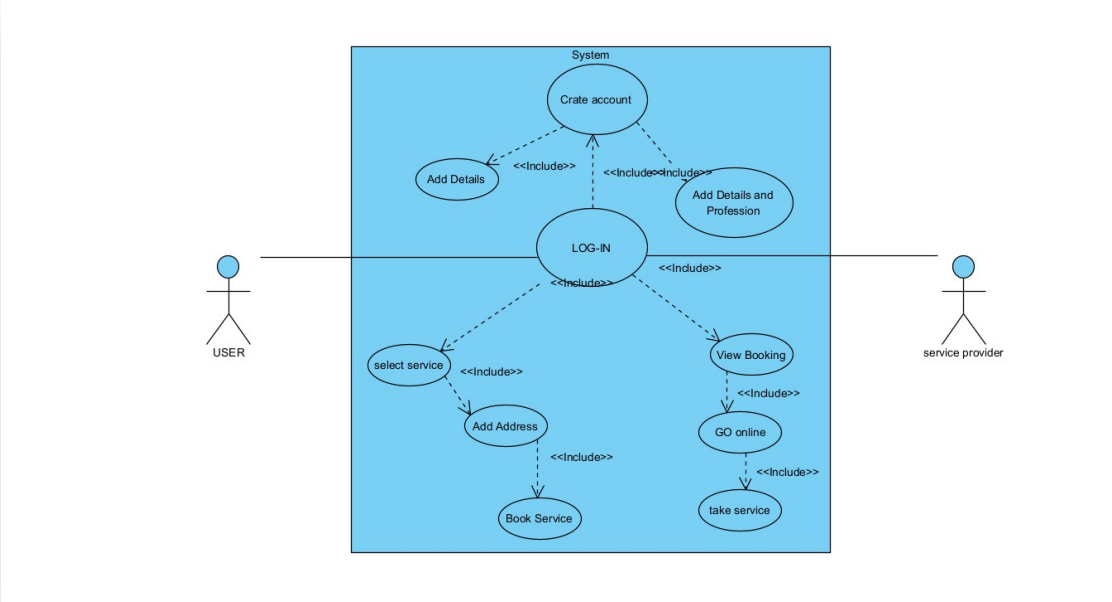

Figure 1: Use Case Diagram

Activity Diagram:

An activity diagram is a graphical representation of the flow of activities within a system. It illustrates the sequence of actions or steps performed to accomplish a particular task. In the context of this assignment, an activity diagram would depict the steps involved in various processes such as logging in, booking services, creating accounts, and viewing bookings. Each step would be represented by a node, connected by arrows to indicate the flow of control. Decision points, parallel activities, and loops can also be represented to capture the complexity of the system's processes. The activity diagram provides a detailed view of the system's workflow, helping to analyze and optimize its processes.

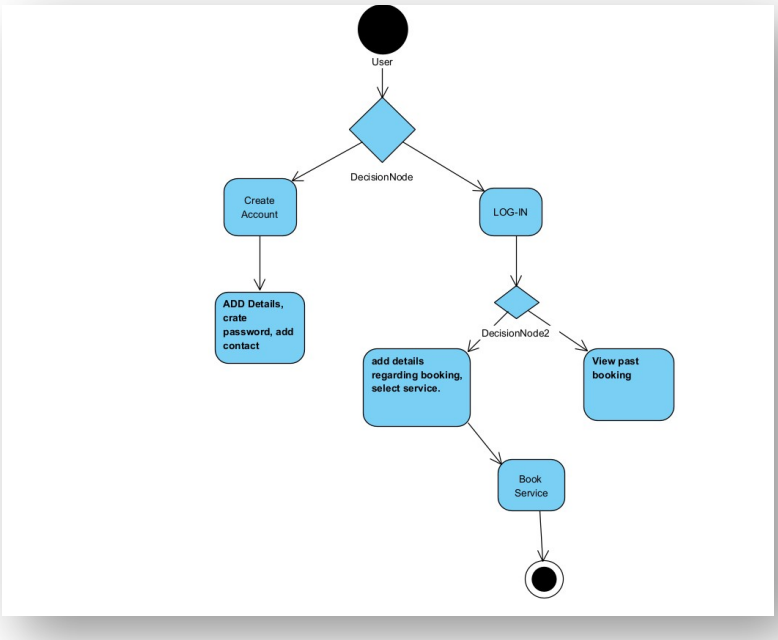

Figure 2: Activity Diagram

ERD Entity relational Diagram:

Entity-Relationship Diagram (ERD) would serve as a vital tool for modeling the database structure. The ERD would depict the various entities, their attributes, and the relationships between them. Here's how the ERD could be structured for this project:

1. **Entities:**
   - User: Represents both customers and service providers. Attributes may include UserID, Username, Password, Name, Email, Phone, and User Type.
   - Service: Describes the types of services offered. Attributes could include Service ID, Service Name, Description, and Price.
   - Booking: Represents the bookings made by customers for services. Attributes may include Booking ID, User ID (foreign key referencing User), Service ID (foreign key referencing Service), Date, Time, and Status.
2. **Relationships:**
   - User-Booking: Many-to-Many relationship, as a user can make multiple bookings, and a booking can involve multiple users (e.g., in the case of a service provider confirming a booking).
   - User-Service: One-to-Many relationship, as a user can offer multiple services, but each service is associated with only one user.
   - Service-Booking: One-to-Many relationship, as a service, can have multiple bookings, but each booking is for only one service.
3. **Cardinalities:**
   - User-Booking: Many-to-Many relationship with optional participation on both sides.
   - User-Service: One-to-Many relationship with mandatory participation from the User side and optional participation from the Service side.
   - Service-Booking: One-to-Many relationship with mandatory participation from the Service side and optional participation from the Booking side.

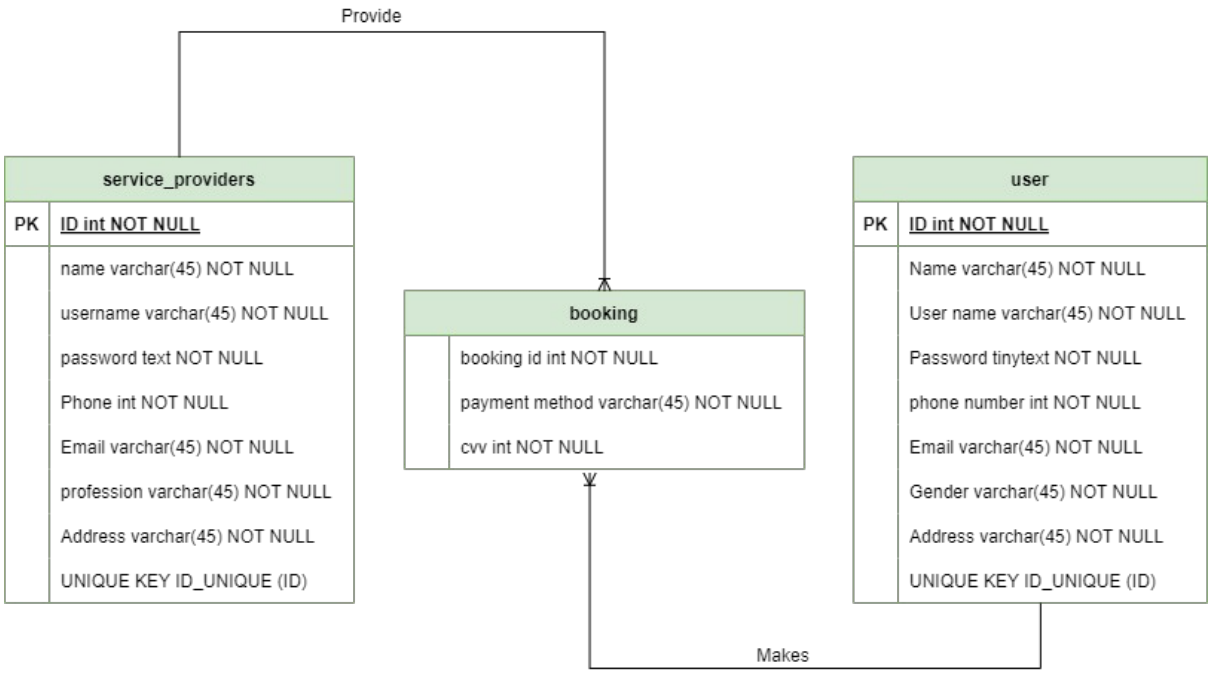

Figure 3: ERD Diagram

Entities:

User Entities:

- User ID, Name, User name, Password, Phone number, Email, Gender, Address.

Service provider Entities:

- ID, Name, User name, Password, Phone, Email, Profession, Address.

Booking Entities:

- Booking ID, Payment Method, CVV.

User:

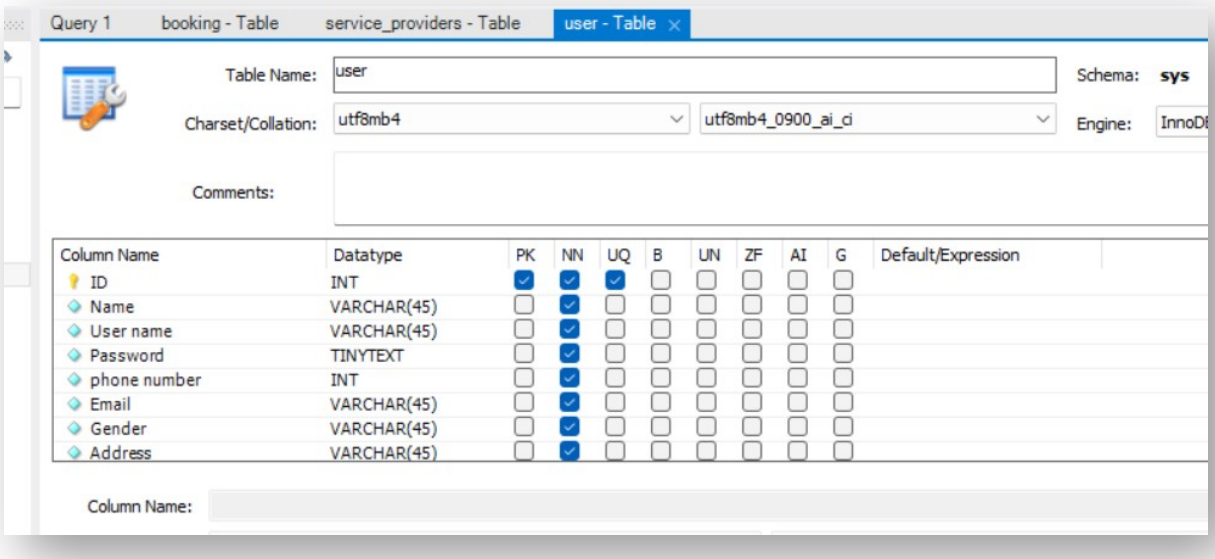

Figure 4: Database table of user

Service provider:

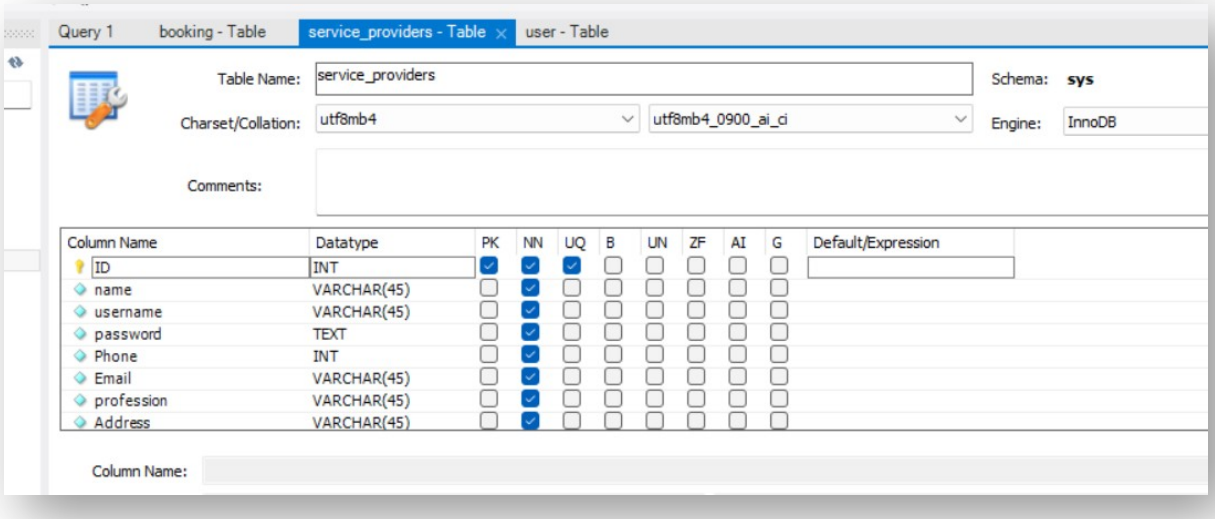

Figure 5: Database Table of Service Providers

Booking:

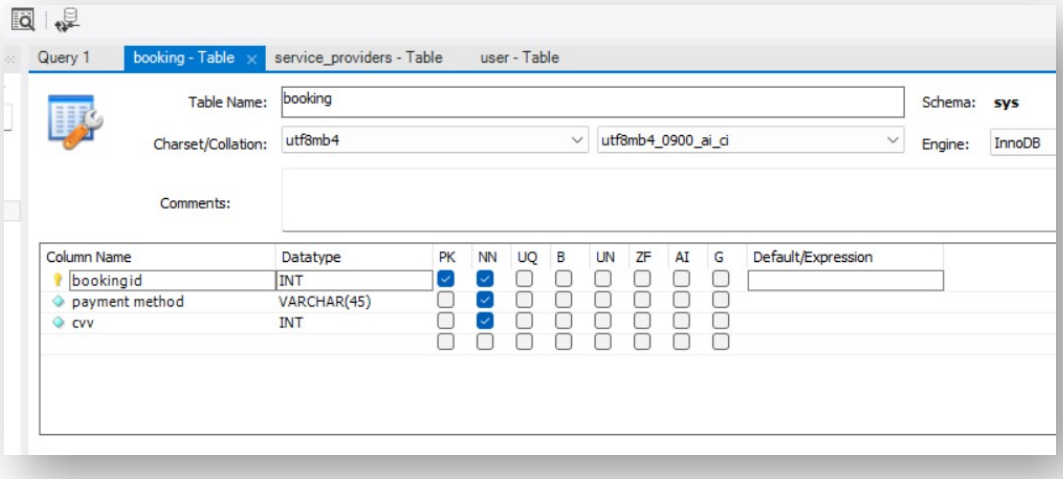

Figure 6: Database Table of Booking

Data Dictionary:

A Data Dictionary for this assignment would serve as a comprehensive reference document detailing the structure and attributes of the data used within the system. It would include a list of all data elements utilized in the system, along with their descriptions, data types, and any constraints or rules associated with them. For example, it would define the attributes of a user account, such as username, password, name, email, and address, specifying their data types (e.g., string, integer) and any validation rules (e.g., minimum length for password). Additionally, it would outline the relationships between different data entities, such as the association between a customer and their

bookings. The Data Dictionary ensures consistency and clarity in data management across the system.

User:

Field name	Data Type	Length	Index	NULL	Validation Rule	Description
User id	Integer (10)	10	PK	NO	-	Customer id
Name	VARCHAR(45)	45	-	NO	-	Insert name as details
user name	VARCHAR(45)	45	-	NO	Must be unique	User for login into the system
Password	Text (15)	15	-	NO	-	User for login into the system
Phone number	INT(15)	15	-	NO	-	Insert for creating an account
E-mail	VARCHAR(45)	15	-	NO	Must be valid	Insert for creating an account
Gender	VARCHAR(45)	45	-	NO	-	Customer's group
Address	VARCHAR(45)	45	-	NO	-	User details

Service provider:

Field name	Data Type	Length	Index	NULL	Validation Rule	Description
Id	Integer (10)	10	PK	NO	Must be unique	Admin's unique ID
Name	Text (15)	15	-	NO	-	Important for registration
Username	Text (15)	15	-	NO	-	Important for registration
password	Text (15)	15	-	NO	-	Use for login
Phone	Integer (15)	15	-	NO	-	Important for registration

E-mail	Text (15)	15	-	NO	Must be valid	Important for registration
profession	VARCHAR(45)	45	-	NO	Not empty	Important for registration
Address	VARCHAR(45)	45	-	NO	-	details

Implementation of Service Management System:

- **Learning Resources Exploration**:
  Start by looking for Java and JavaFX tutorials that are appropriate for beginners on YouTube. Seek out videos that explain fundamental ideas such as loops, variables, and object-oriented programming. Investigate online discussion boards and documentation as well to enhance your education.
- **Practical Application:**
  After grasping the fundamentals, practice coding exercises, and small projects to reinforce your understanding. Experiment with building simple JavaFX applications, such as calculators or to-do lists, to get hands-on experience with the framework.
- **Database Integration:**
  After grasping the fundamentals, practice coding exercises, and small projects to reinforce your understanding. Experiment with building simple JavaFX applications, such as calculators or to-do lists, to get hands-on experience with the framework.
- **User Interface Enhancement:**
  Explore JavaFX to make changes to your application's UI. Try out various styles, layouts, and controls to produce a user interface that is both aesthetically pleasing and intuitive. For a smooth interface, pay close attention to design and usability concepts.
- **Functionality Implementation:**
  Begin by implementing core functionalities like user authentication, service booking, and dashboard features. Break down each feature into smaller tasks and tackle them one at a time, testing as you go to ensure everything works as expected.
- **Testing and Debugging:**
  Thoroughly test each feature of your application to identify and fix any bugs or errors. Use debugging tools and techniques to pinpoint issues and validate the correctness of your code.

TESTING:

First Test:

Purpose:

The user and Service provider can register into the system.

Result:

Account successfully created by User and Service provider.

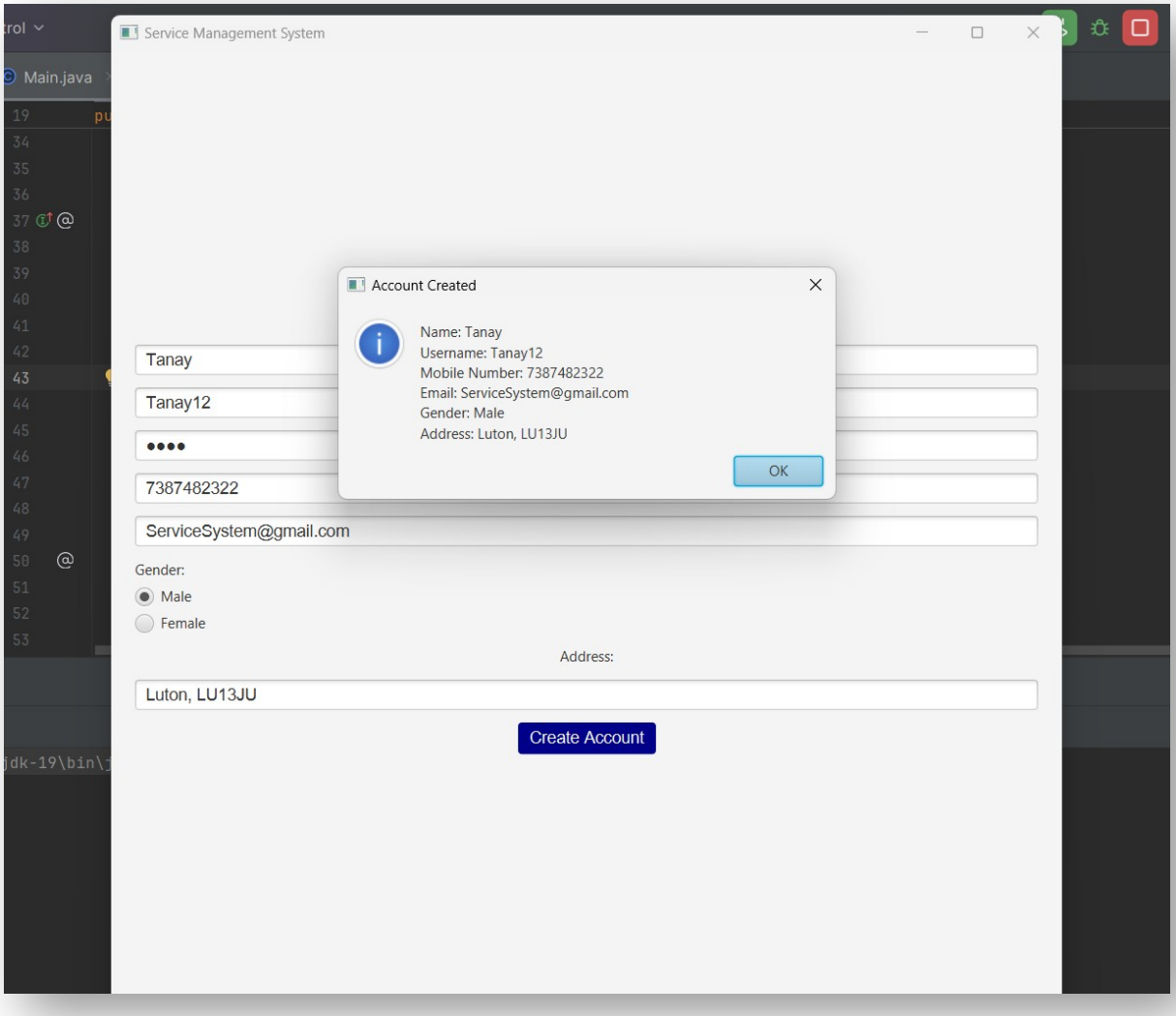

Figure 7: User account registration

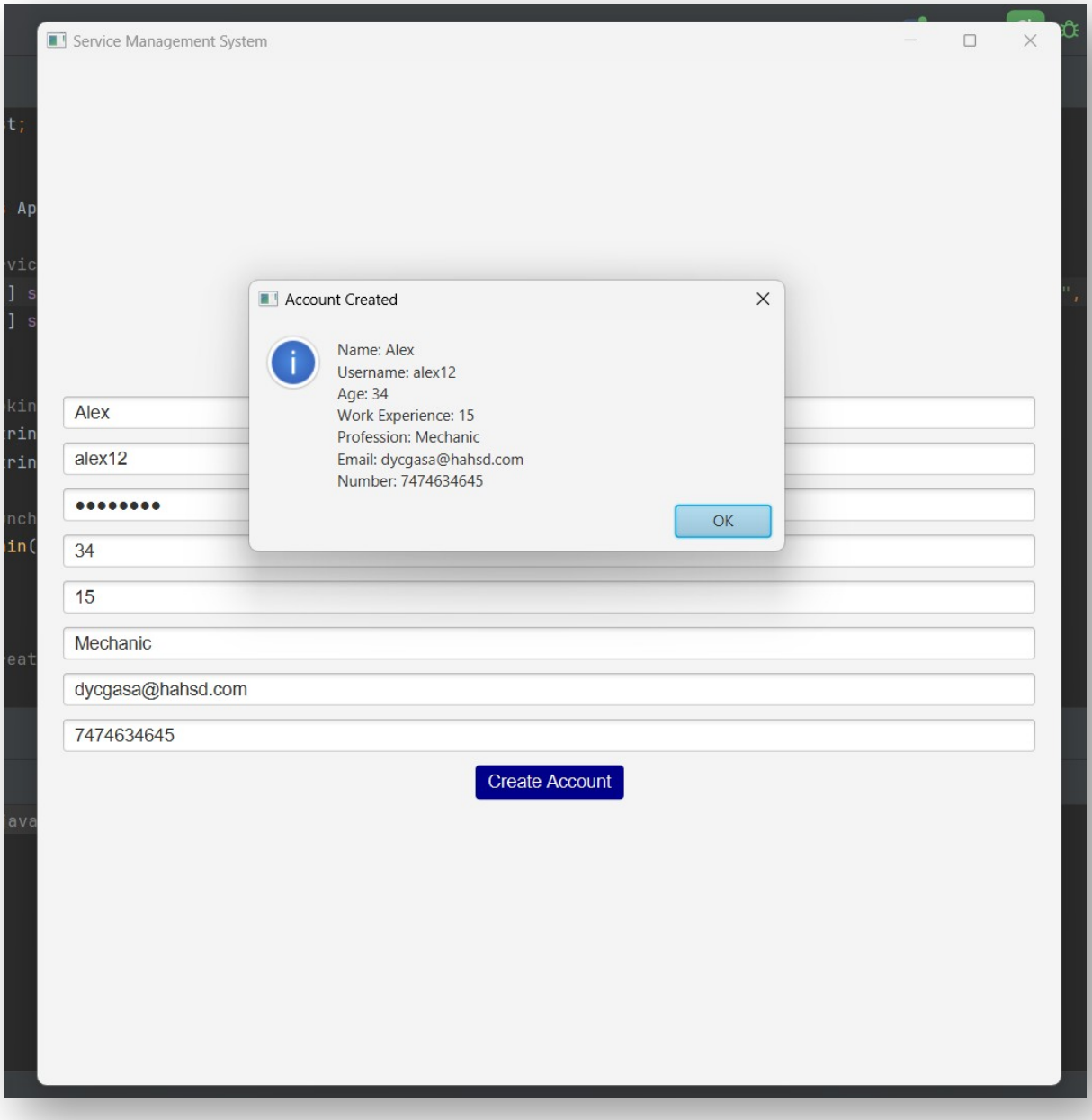

Figure 8: Service provider account registration

Second Test:

Purpose:

The user and Service provider can log in to the system.

Result:

The account was successfully logged in by the User and Service provider.

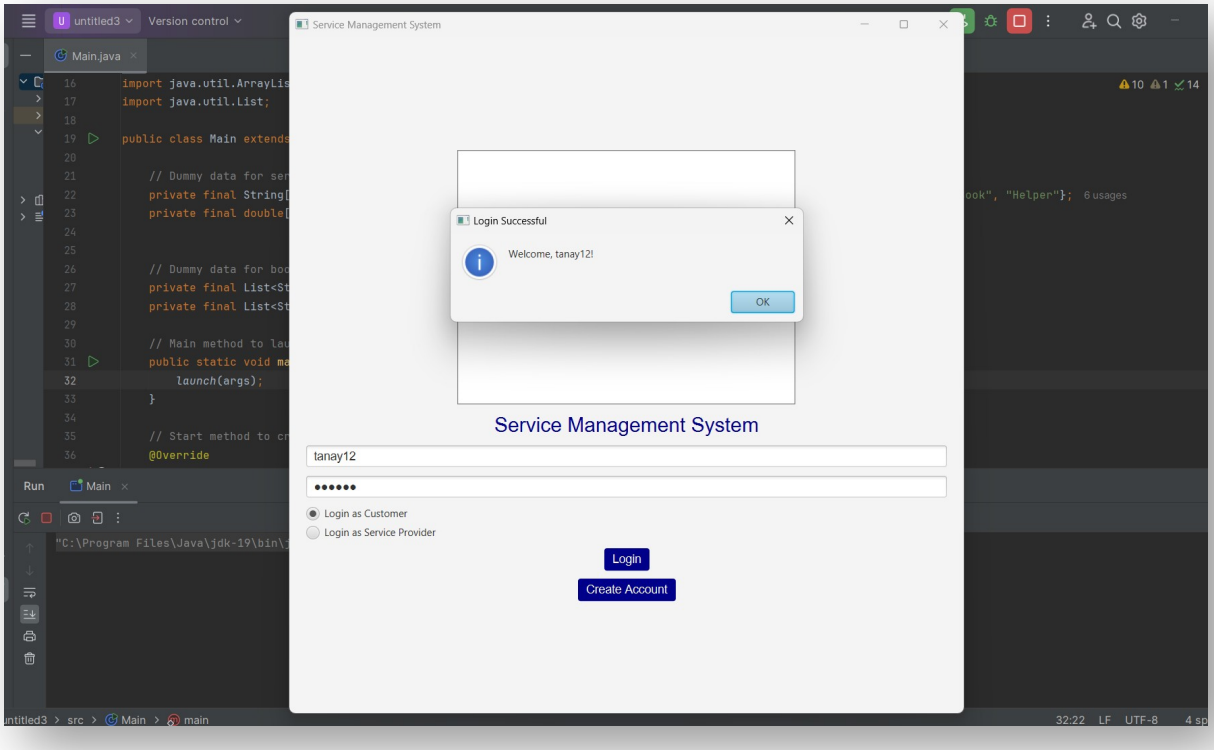

Figure 9: Log-in as a customer.

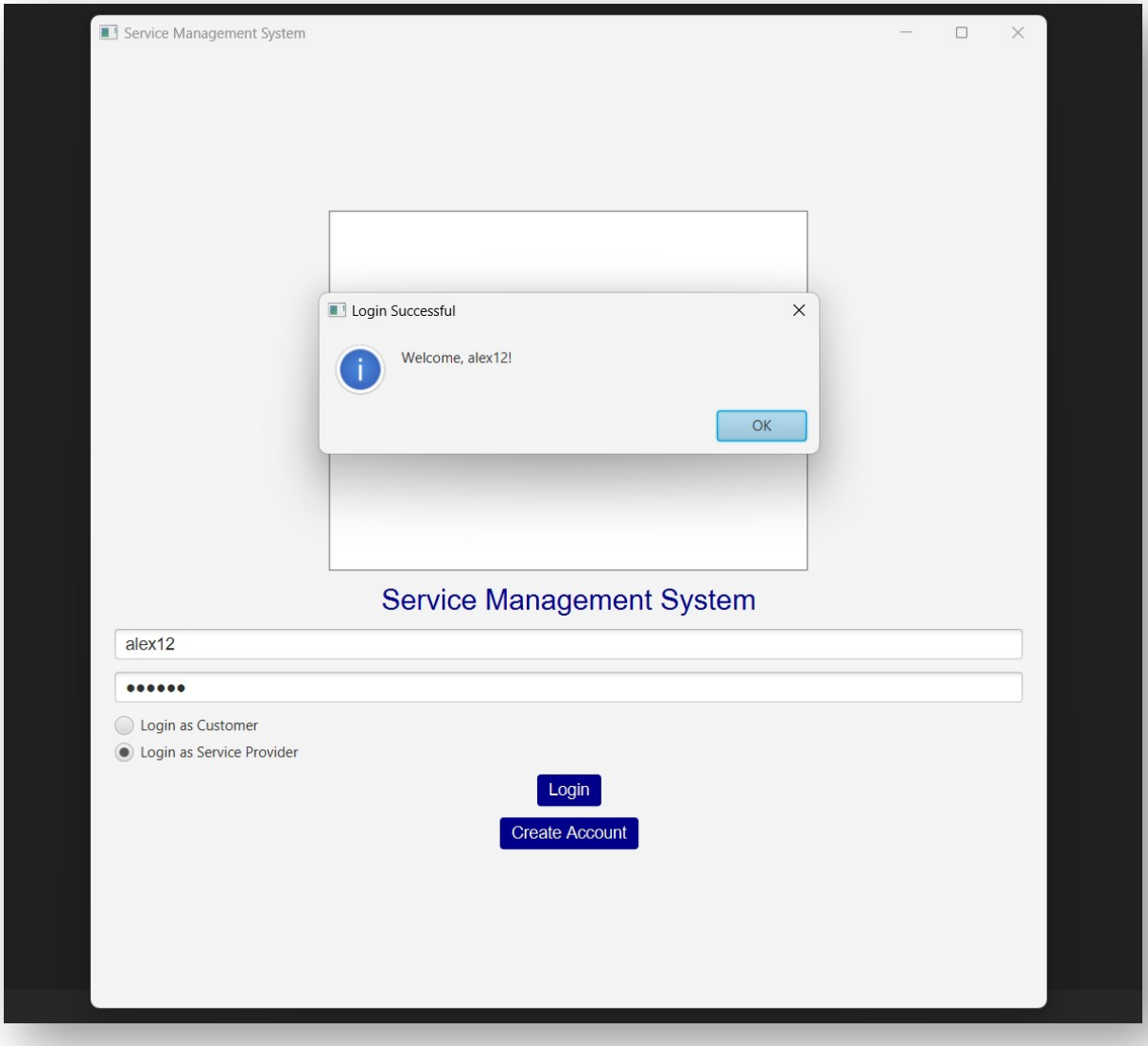

Figure 10: Log-in as a Service provider.

Third Testing:

Purpose:

The user can select services and book services.

Result:

The user gets a pop-up that displays selected services and the cost of services.

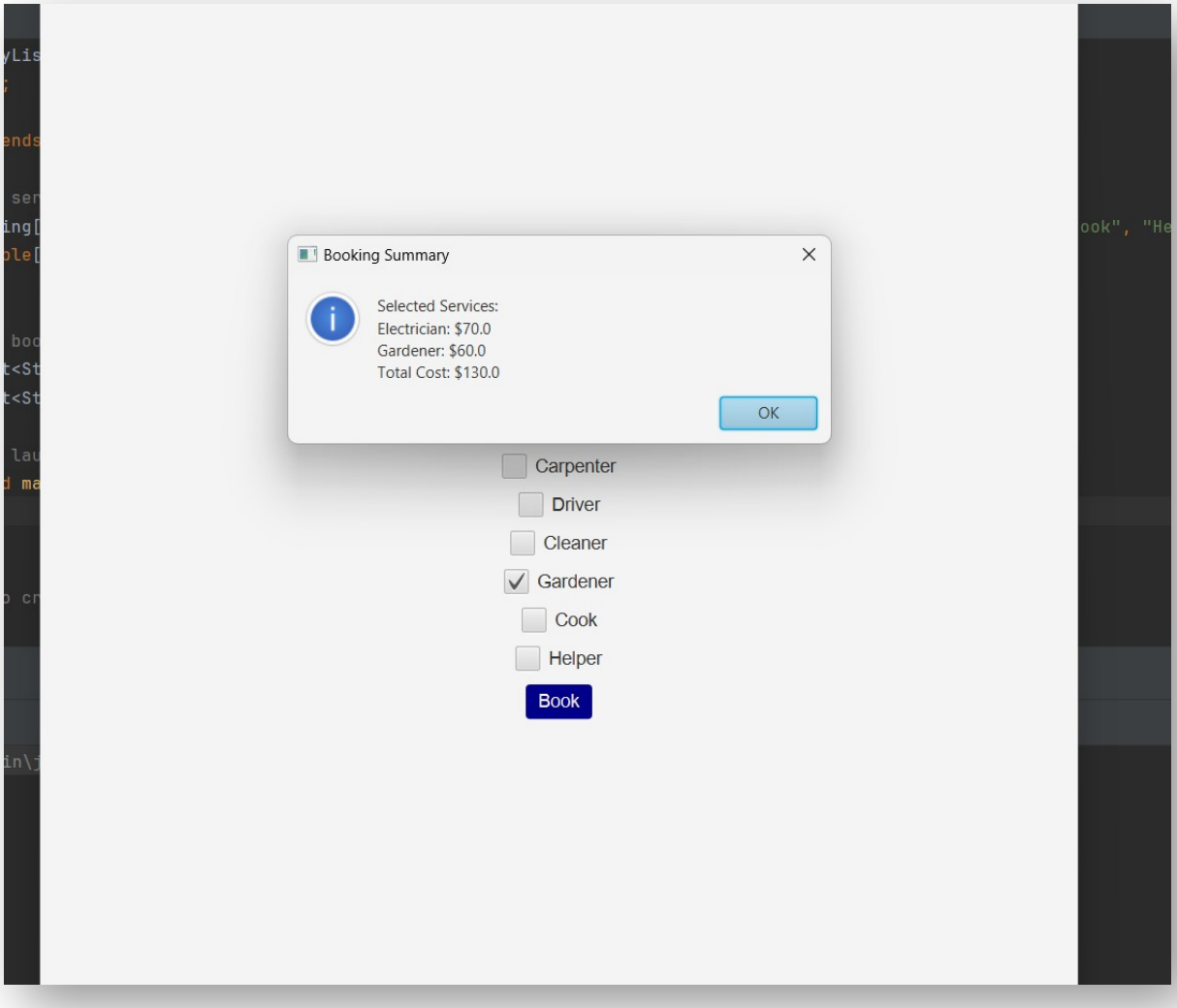

Figure 11: Select service for booking.

Fourth Testing:

Purpose:

The user can select a payment method and enter card details to make a payment.

Result:

The user gets a pop-up of payment successfully.

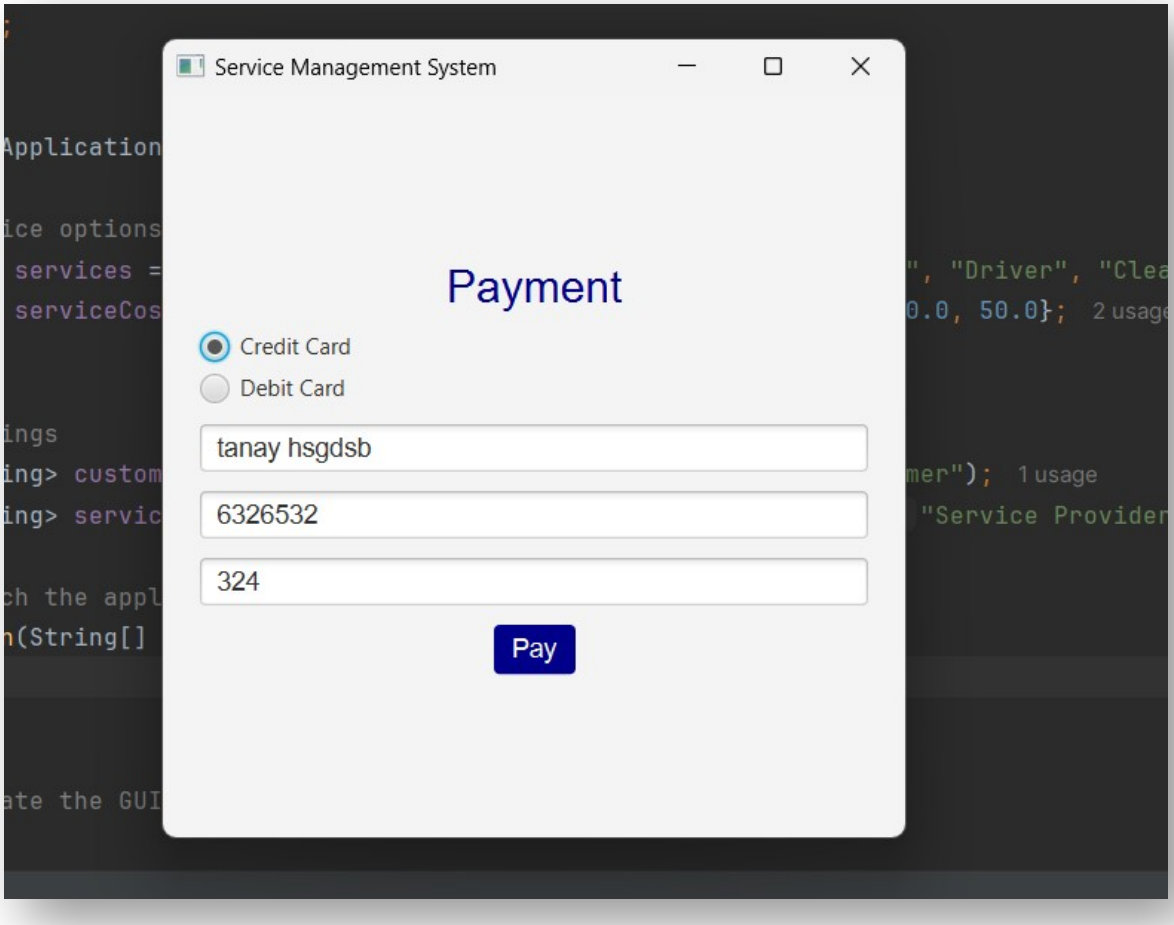

Figure 12: Enter Payment Details.

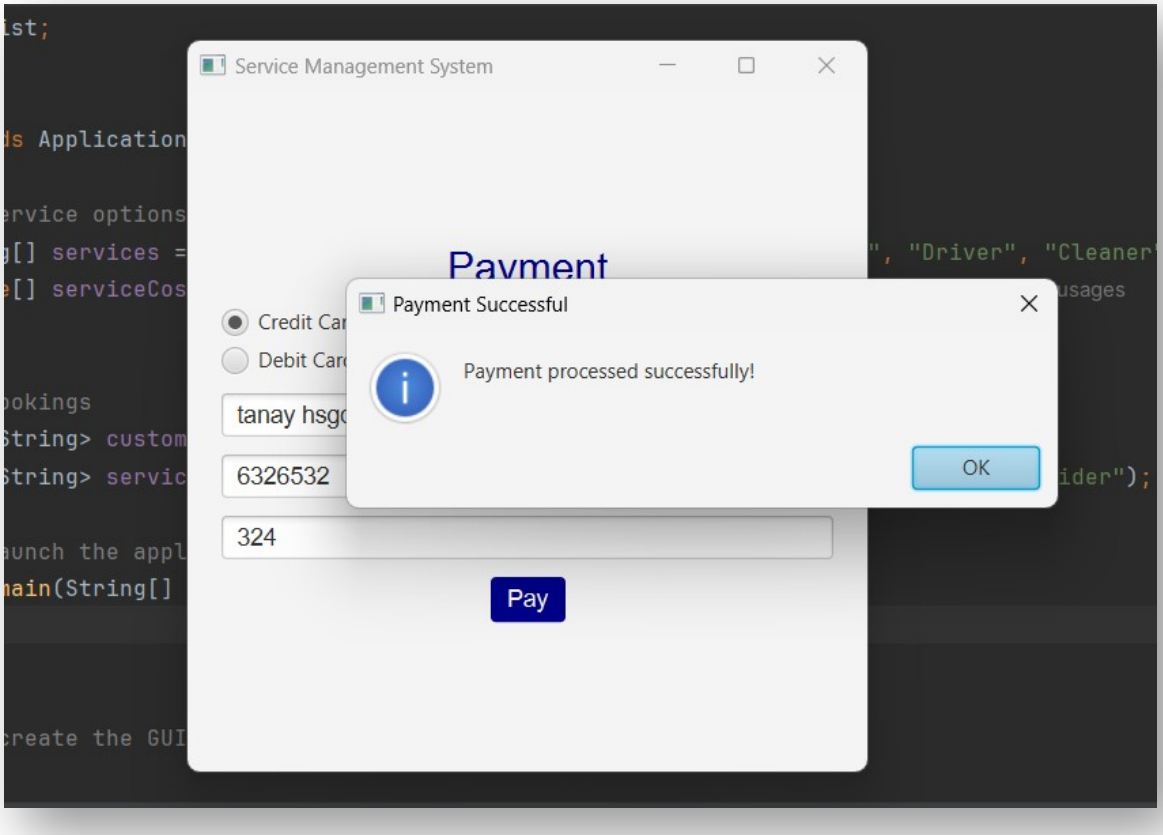

Figure 13: Payment Successful.

Fifth Test:

Purpose:

The user navigates to the last page.

Result:

The user can view the booking or the user can go to the home page.

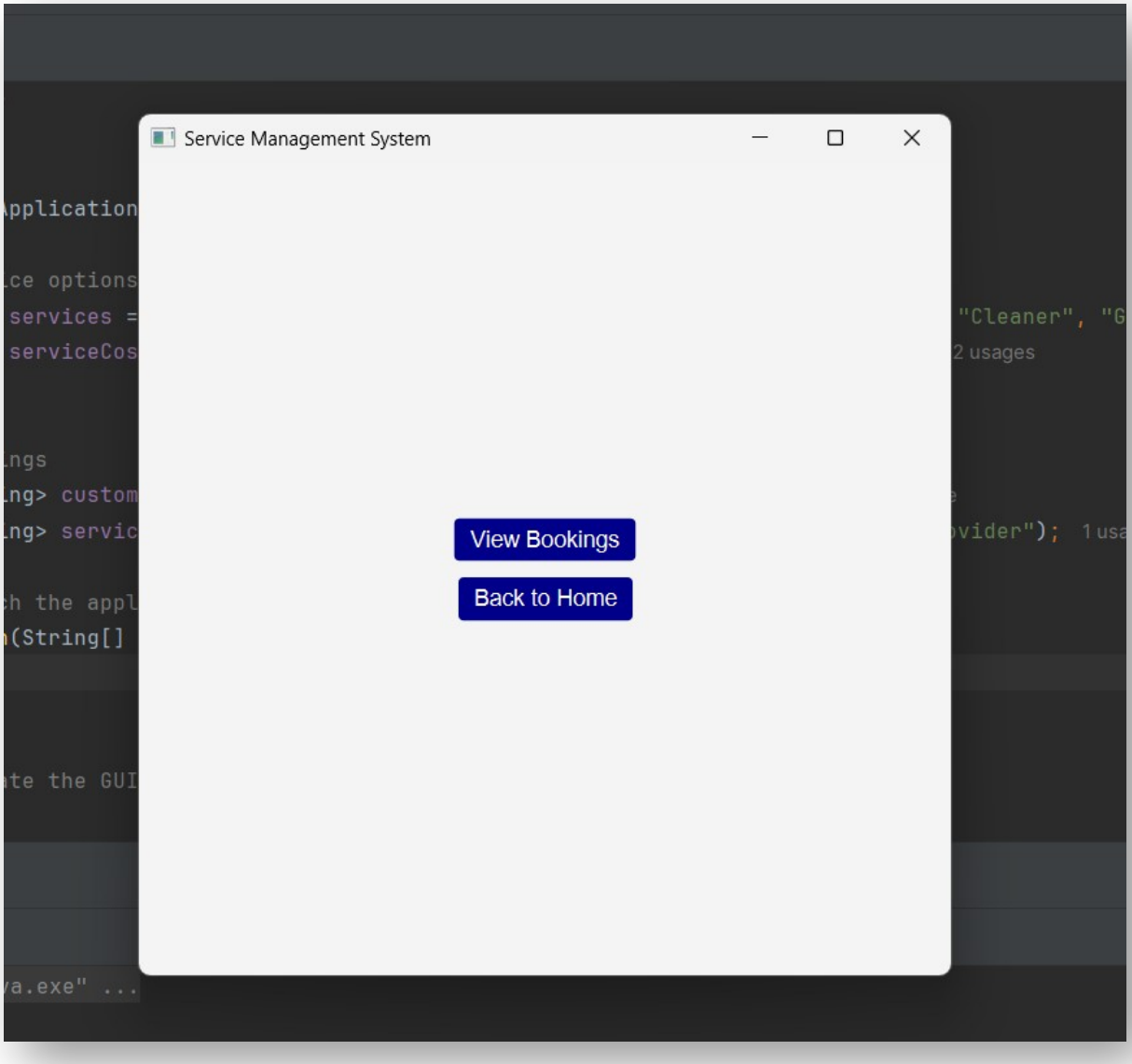

Figure 14: View the booking page.

Conclusions:

To sum up, the process of putting a service management system into place has been gratifying and challenging, with many of possibilities for learning. I developed a strong foundation in programming ideas and graphical user interface development by exploring Java and JavaFX resources. I was able to successfully connect a database backend to store and manage crucial data for the system by utilizing my prior SQL experience.

The process of enhancing the user interface was particularly enriching, as I experimented with different layouts, styles, and controls to create an intuitive and visually appealing experience for

users. Implementing core functionalities, such as user authentication and service booking, required meticulous planning and attention to detail. Testing and debugging played a crucial role in ensuring the reliability and functionality of the application, allowing me to identify and resolve issues effectively.

Incorporating user feedback at every stage of the development process also enabled me to improve the system and make it more responsive to user needs and user-friendly. I do not doubt that the service management system will prove to be a useful tool for both clients and providers, promoting smooth communication and elevating the client experience in general, as I continue to refine and enhance it. My technical skills have improved, but this project has also given me a sense of accomplishment and preparedness for any future software development endeavors.

Reference:

Anon., n.d. [Online]
Available at: https://www.youtube.com/watch?v=FLkOX4Eez6o&list=PL6gx4Cwl9DGBzfXLWLSYVy8EbTdpGbUIG&pp=iAQB
Gilbert, D., 2014. *Creating User Experience with JavaFX.* second ed. Apress: s.n.

https://www.youtube.com/watch?v=FLkOX4Eez6o&list=PL6gx4Cwl9DGBzfXLWLSYVy8EbTdpGbUIG&pp=iAQB

https://www.geeksforgeeks.org/introduction-of-dbms-database-management-system-set-1/?ref=lbp

Appendix:

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.scene.shape.StrokeLineCap;
import javafx.scene.shape.StrokeLineJoin;
import javafx.scene.text.Font;
import javafx.stage.Modality;
import javafx.stage.Stage;

import java.util.ArrayList;
import java.util.List;
import java.sql.Connection;
import java.sql.DriverManager;
```

University of Bedfordshire

```
import java.sql.SQLException;
import java.sql.Statement;


public class Main extends Application {

    // Dummy data for service options and their costs
    private final String[] services = {"Plumber", "Mechanic",
"Electrician", "Carpenter", "Driver", "Cleaner", "Gardener", "Cook",
"Helper"};
    private final double[] serviceCosts = {50.0, 80.0, 70.0, 60.0, 100.0,
40.0, 60.0, 90.0, 50.0};


    // Dummy data for bookings
    private final List<String> customerBookings =
generateDummyBookings("Customer");
    private final List<String> serviceProviderBookings =
generateDummyBookings("Service Provider");

    private static final String URL =
"jdbc:mysql://localhost:3306/service_management";
    private static final String USER = "root";
    private static final String PASSWORD = "Mann7204";

    // JDBC variables for opening, closing and managing connection
    private Connection connection;
    private Statement statement;

    // Main method to launch the application
    public static void main(String[] args) {
        launch(args);
    }

    // Start method to create the GUI
    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Service Management System");

        // Create login page
        VBox loginPage = createLoginPage(primaryStage);

        // Display login page
        Scene scene = new Scene(loginPage, 800, 800);
        primaryStage.setScene(scene);
        primaryStage.show();
        createTables();
    }
    private void createTables() {
        try {
            connection = DriverManager.getConnection(URL, USER, PASSWORD);
            statement = connection.createStatement();

            // Create Customer table
            String createCustomerTableSQL = "CREATE TABLE IF NOT EXISTS
Customer ("
                    + "id INT AUTO_INCREMENT PRIMARY KEY,"
                    + "name VARCHAR(255) NOT NULL,"
                    + "username VARCHAR(50) NOT NULL,"
```

```
                    + "password VARCHAR(50) NOT NULL,"
                    + "mobile_number VARCHAR(15) NOT NULL,"
                    + "email VARCHAR(255) NOT NULL,"
                    + "gender ENUM('Male', 'Female') NOT NULL,"
                    + "address TEXT NOT NULL"
                    + ")";
            statement.execute(createCustomerTableSQL);

            // Create Service_Provider table
            String createServiceProviderTableSQL = "CREATE TABLE IF NOT
EXISTS Service_Provider ("
                    + "id INT AUTO_INCREMENT PRIMARY KEY,"
                    + "name VARCHAR(255) NOT NULL,"
                    + "username VARCHAR(50) NOT NULL,"
                    + "password VARCHAR(50) NOT NULL,"
                    + "age INT NOT NULL,"
                    + "work_experience INT NOT NULL,"
                    + "profession VARCHAR(100) NOT NULL,"
                    + "email VARCHAR(255) NOT NULL,"
                    + "number VARCHAR(15) NOT NULL"
                    + ")";
            statement.execute(createServiceProviderTableSQL);

            // Create Booking table
            String createBookingTableSQL = "CREATE TABLE IF NOT EXISTS
Booking ("
                    + "id INT AUTO_INCREMENT PRIMARY KEY,"
                    + "customer_id INT NOT NULL,"
                    + "service_provider_id INT NOT NULL,"
                    + "service VARCHAR(100) NOT NULL,"
                    + "cost DOUBLE NOT NULL,"
                    + "FOREIGN KEY (customer_id) REFERENCES Customer(id),"
                    + "FOREIGN KEY (service_provider_id) REFERENCES
Service_Provider(id)"
                    + ")";
            statement.execute(createBookingTableSQL);

        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (statement != null) {
                    statement.close();
                }
                if (connection != null) {
                    connection.close();
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }


    // Method to create login page
    private VBox createLoginPage(Stage primaryStage) {
        VBox loginPage = new VBox(10);
        loginPage.setAlignment(Pos.CENTER);
        loginPage.setPadding(new Insets(20));
```

```
        Rectangle loginPageBackground = new Rectangle(400, 300);
        loginPageBackground.setFill(Color.WHITE);
        loginPageBackground.setStroke(Color.GRAY);
        loginPageBackground.setStrokeWidth(1);
        loginPageBackground.setStrokeLineCap(StrokeLineCap.ROUND);
        loginPageBackground.setStrokeLineJoin(StrokeLineJoin.ROUND);
        loginPage.getChildren().add(loginPageBackground);

        Label loginPageLabel = new Label("Service Management System");
        loginPageLabel.setFont(Font.font("Arial", 24));
        loginPageLabel.setTextFill(Color.DARKBLUE);
        loginPage.getChildren().add(loginPageLabel);

        TextField usernameField = new TextField();
        usernameField.setPromptText("Username");
        usernameField.setFont(Font.font("Arial", 14));
        loginPage.getChildren().add(usernameField);

        PasswordField passwordField = new PasswordField();
        passwordField.setPromptText("Password");
        passwordField.setFont(Font.font("Arial", 14));
        loginPage.getChildren().add(passwordField);

        // Create radio buttons for selecting login type
        ToggleGroup loginTypeGroup = new ToggleGroup();
        RadioButton customerRadioButton = new RadioButton("Login as
Customer");
        customerRadioButton.setToggleGroup(loginTypeGroup);
        customerRadioButton.setSelected(true); // Default selection
        RadioButton serviceProviderRadioButton = new RadioButton("Login as
Service Provider");
        serviceProviderRadioButton.setToggleGroup(loginTypeGroup);

        VBox radioButtonsBox = new VBox(5, customerRadioButton,
serviceProviderRadioButton);
        radioButtonsBox.setAlignment(Pos.CENTER_LEFT);
        loginPage.getChildren().add(radioButtonsBox);

        Button loginButton = new Button("Login");
        loginButton.setFont(Font.font("Arial", 14));
        loginButton.setStyle("-fx-background-color: DARKBLUE; -fx-text-
fill: WHITE;");
        loginButton.setOnAction(event -> {
            // Perform login authentication based on selected login type
            String username = usernameField.getText();
            String password = passwordField.getText();
            RadioButton selectedRadioButton = (RadioButton)
loginTypeGroup.getSelectedToggle();
            if (selectedRadioButton != null) {
                String loginType = selectedRadioButton.getText();
                if (loginType.equals("Login as Customer")) {
                    // Authenticate as a customer
                    showAlert(Alert.AlertType.INFORMATION, "Login
Successful", "Welcome, " + username + "!");
                    // Move to service selection page for customer
                    VBox serviceSelectionPage =
createServiceSelectionPage(primaryStage, username, false);
                    Scene scene = new Scene(serviceSelectionPage, 800,
```

```
800);
                    primaryStage.setScene(scene);
                } else if (loginType.equals("Login as Service Provider")) {
                    // Authenticate as a service provider
                    showAlert(Alert.AlertType.INFORMATION, "Login
Successful", "Welcome, " + username + "!");
                    // Move to service provider dashboard
                    VBox serviceProviderDashboard =
createServiceProviderDashboard(primaryStage, username);
                    Scene scene = new Scene(serviceProviderDashboard, 800,
800);
                    primaryStage.setScene(scene);
                }
            }
        });
        loginPage.getChildren().add(loginButton);

        Button createAccountButton = new Button("Create Account");
        createAccountButton.setFont(Font.font("Arial", 14));
        createAccountButton.setStyle("-fx-background-color: DARKBLUE; -fx-
text-fill: WHITE;");
        createAccountButton.setOnAction(event -> {
            // Move to account creation page
            VBox accountCreationPage =
createAccountCreationPage(primaryStage);
            Scene scene = new Scene(accountCreationPage, 800, 800);
            primaryStage.setScene(scene);
        });
        loginPage.getChildren().add(createAccountButton);

        return loginPage;
    }

    // Method to generate dummy booking data
    private List<String> generateDummyBookings(String userType) {
        List<String> bookings = new ArrayList<>();
        for (int i = 1; i <= 3; i++) {
            bookings.add(userType + " Booking " + i);
        }
        return bookings;
    }

    // Method to create customer account creation page
    private VBox createCustomerAccountCreationPage(Stage primaryStage) {
        VBox accountCreationPage = new VBox(10);
        accountCreationPage.setAlignment(Pos.CENTER);
        accountCreationPage.setPadding(new Insets(20));

        Label titleLabel = new Label("Customer Account Creation");
        titleLabel.setFont(Font.font("Arial", 24));
        titleLabel.setTextFill(Color.DARKBLUE);
        accountCreationPage.getChildren().add(titleLabel);

        TextField nameField = new TextField();
        nameField.setPromptText("Name");
        nameField.setFont(Font.font("Arial", 14));
        accountCreationPage.getChildren().add(nameField);

        TextField usernameField = new TextField();
```

```
        usernameField.setPromptText("Username");
        usernameField.setFont(Font.font("Arial", 14));
        accountCreationPage.getChildren().add(usernameField);

        PasswordField passwordField = new PasswordField();
        passwordField.setPromptText("Password");
        passwordField.setFont(Font.font("Arial", 14));
        accountCreationPage.getChildren().add(passwordField);

        TextField mobileNumberField = new TextField();
        mobileNumberField.setPromptText("Mobile Number");
        mobileNumberField.setFont(Font.font("Arial", 14));
        accountCreationPage.getChildren().add(mobileNumberField);

        TextField emailField = new TextField();
        emailField.setPromptText("Email");
        emailField.setFont(Font.font("Arial", 14));
        accountCreationPage.getChildren().add(emailField);

        Label genderLabel = new Label("Gender:");
        ToggleGroup genderGroup = new ToggleGroup();
        RadioButton maleRadioButton = new RadioButton("Male");
        maleRadioButton.setToggleGroup(genderGroup);
        maleRadioButton.setSelected(true); // Default selection
        RadioButton femaleRadioButton = new RadioButton("Female");
        femaleRadioButton.setToggleGroup(genderGroup);
        VBox genderBox = new VBox(5, genderLabel, maleRadioButton,
femaleRadioButton);
        genderBox.setAlignment(Pos.CENTER_LEFT);
        accountCreationPage.getChildren().add(genderBox);

        TextField addressField = new TextField();
        addressField.setPromptText("Address");
        addressField.setFont(Font.font("Arial", 14));
        accountCreationPage.getChildren().addAll(new Label("Address:"),
addressField);

        Button createAccountButton = new Button("Create Account");
        createAccountButton.setFont(Font.font("Arial", 14));
        createAccountButton.setStyle("-fx-background-color: DARKBLUE; -fx-
text-fill: WHITE;");
        createAccountButton.setOnAction(event -> {
            // Perform account creation for customer
            // Retrieve input values
            String name = nameField.getText();
            String username = usernameField.getText();
            String password = passwordField.getText();
            String mobileNumber = mobileNumberField.getText();
            String email = emailField.getText();
            RadioButton selectedGender = (RadioButton)
genderGroup.getSelectedToggle();
            String gender = selectedGender.getText();
            String address = addressField.getText();
            // Implement logic for account creation
            // For demonstration, just display the entered values
            showAlert(Alert.AlertType.INFORMATION, "Account Created",
                    "Name: " + name + "\nUsername: " + username +
                            "\nMobile Number: " + mobileNumber + "\nEmail:
" + email +
```

```
                            "\nGender: " + gender + "\nAddress: " +
address);
            // Move back to login page
            Scene loginScene = new Scene(createLoginPage(primaryStage),
800, 800);
            primaryStage.setScene(loginScene);
        });
        accountCreationPage.getChildren().add(createAccountButton);

        return accountCreationPage;
    }

    // Method to create service provider account creation page
    private VBox createServiceProviderAccountCreationPage(Stage
primaryStage) {
        VBox accountCreationPage = new VBox(10);
        accountCreationPage.setAlignment(Pos.CENTER);
        accountCreationPage.setPadding(new Insets(20));

        Label titleLabel = new Label("Service Provider Account Creation");
        titleLabel.setFont(Font.font("Arial", 24));
        titleLabel.setTextFill(Color.DARKBLUE);
        accountCreationPage.getChildren().add(titleLabel);

        TextField nameField = new TextField();
        nameField.setPromptText("Name");
        nameField.setFont(Font.font("Arial", 14));
        accountCreationPage.getChildren().add(nameField);

        TextField usernameField = new TextField();
        usernameField.setPromptText("Username");
        usernameField.setFont(Font.font("Arial", 14));
        accountCreationPage.getChildren().add(usernameField);

        PasswordField passwordField = new PasswordField();
        passwordField.setPromptText("Password");
        passwordField.setFont(Font.font("Arial", 14));
        accountCreationPage.getChildren().add(passwordField);

        TextField ageField = new TextField();
        ageField.setPromptText("Age");
        ageField.setFont(Font.font("Arial", 14));
        accountCreationPage.getChildren().add(ageField);

        TextField experienceField = new TextField();
        experienceField.setPromptText("Work Experience");
        experienceField.setFont(Font.font("Arial", 14));
        accountCreationPage.getChildren().add(experienceField);

        TextField professionField = new TextField();
        professionField.setPromptText("Profession");
        professionField.setFont(Font.font("Arial", 14));
        accountCreationPage.getChildren().add(professionField);

        TextField emailField = new TextField();
        emailField.setPromptText("Email");
        emailField.setFont(Font.font("Arial", 14));
        accountCreationPage.getChildren().add(emailField);
```

```
        TextField numberField = new TextField();
        numberField.setPromptText("Number");
        numberField.setFont(Font.font("Arial", 14));
        accountCreationPage.getChildren().add(numberField);

        Button createAccountButton = new Button("Create Account");
        createAccountButton.setFont(Font.font("Arial", 14));
        createAccountButton.setStyle("-fx-background-color: DARKBLUE; -fx-
text-fill: WHITE;");
        createAccountButton.setOnAction(event -> {
            // Perform account creation for service provider
            // Retrieve input values
            String name = nameField.getText();
            String username = usernameField.getText();
            String password = passwordField.getText();
            int age = Integer.parseInt(ageField.getText());
            int experience = Integer.parseInt(experienceField.getText());
            String profession = professionField.getText();
            String email = emailField.getText();
            String number = numberField.getText();
            // Implement logic for account creation
            // For demonstration, just display the entered values
            showAlert(Alert.AlertType.INFORMATION, "Account Created",
                    "Name: " + name + "\nUsername: " + username +
                            "\nAge: " + age + "\nWork Experience: " +
experience +
                            "\nProfession: " + profession + "\nEmail: " +
email +
                            "\nNumber: " + number);
            // Move back to login page
            Scene loginScene = new Scene(createLoginPage(primaryStage),
800, 800);
            primaryStage.setScene(loginScene);
        });
        accountCreationPage.getChildren().add(createAccountButton);

        return accountCreationPage;
    }

    // Method to create account creation page
    private VBox createAccountCreationPage(Stage primaryStage) {
        VBox accountCreationPage = new VBox(10);
        accountCreationPage.setAlignment(Pos.CENTER);
        accountCreationPage.setPadding(new Insets(20));

        Label titleLabel = new Label("Account Creation");
        titleLabel.setFont(Font.font("Arial", 24));
        titleLabel.setTextFill(Color.DARKBLUE);
        accountCreationPage.getChildren().add(titleLabel);

        // Create radio buttons for selecting account type
        ToggleGroup accountTypeGroup = new ToggleGroup();
        RadioButton customerRadioButton = new RadioButton("Customer");
        customerRadioButton.setToggleGroup(accountTypeGroup);
        customerRadioButton.setSelected(true); // Default selection
        RadioButton serviceProviderRadioButton = new RadioButton("Service
Provider");
        serviceProviderRadioButton.setToggleGroup(accountTypeGroup);
```

```
        VBox radioButtonsBox = new VBox(5, customerRadioButton,
serviceProviderRadioButton);
        radioButtonsBox.setAlignment(Pos.CENTER_LEFT);
        accountCreationPage.getChildren().add(radioButtonsBox);

        Button createAccountButton = new Button("Next");
        createAccountButton.setFont(Font.font("Arial", 14));
        createAccountButton.setStyle("-fx-background-color: DARKBLUE; -fx-
text-fill: WHITE;");
        createAccountButton.setOnAction(event -> {
            // Move to appropriate account creation page based on selection
            RadioButton selectedRadioButton = (RadioButton)
accountTypeGroup.getSelectedToggle();
            if (selectedRadioButton != null) {
                String accountType = selectedRadioButton.getText();
                if (accountType.equals("Customer")) {
                    // Move to customer account creation page
                    Scene scene = new
Scene(createCustomerAccountCreationPage(primaryStage), 800, 800);
                    primaryStage.setScene(scene);
                } else if (accountType.equals("Service Provider")) {
                    // Move to service provider account creation page
                    Scene scene = new
Scene(createServiceProviderAccountCreationPage(primaryStage), 800, 800);
                    primaryStage.setScene(scene);
                }
            }
        });
        accountCreationPage.getChildren().add(createAccountButton);

        return accountCreationPage;
    }

    // Method to create service selection page for customer
    // Method to create service selection page for customer
    private VBox createServiceSelectionPage(Stage primaryStage, String
username, boolean editMode) {
        VBox serviceSelectionPage = new VBox(10);
        serviceSelectionPage.setAlignment(Pos.CENTER);
        serviceSelectionPage.setPadding(new Insets(20));

        Label titleLabel = new Label("Service Selection");
        titleLabel.setFont(Font.font("Arial", 24));
        titleLabel.setTextFill(Color.DARKBLUE);
        serviceSelectionPage.getChildren().add(titleLabel);

        // Create checkboxes for service selection
        List<CheckBox> serviceCheckboxes = new ArrayList<>();
        for (String service : services) {
            CheckBox checkBox = new CheckBox(service);
            checkBox.setFont(Font.font("Arial", 14));
            serviceCheckboxes.add(checkBox);
            serviceSelectionPage.getChildren().add(checkBox);
        }

        Button bookButton = new Button(editMode ? "Save Changes" : "Book");
        bookButton.setFont(Font.font("Arial", 14));
        bookButton.setStyle("-fx-background-color: DARKBLUE; -fx-text-fill:
WHITE;");
```

```
        bookButton.setOnAction(event -> {
            // Process selected services and calculate total cost
            double totalCost = 0.0;
            StringBuilder selectedServices = new StringBuilder("Selected
Services:\n");
            for (int i = 0; i < services.length; i++) {
                CheckBox checkBox = serviceCheckboxes.get(i);
                if (checkBox.isSelected()) {
                    selectedServices.append(services[i]).append(":
$").append(serviceCosts[i]).append("\n");
                    totalCost += serviceCosts[i];
                }
            }
            // Display selected services and total cost
            showAlert(Alert.AlertType.INFORMATION, "Booking Summary",
selectedServices.toString() +
                    "Total Cost: $" + totalCost);

            // Show payment form
            VBox paymentForm = createPaymentForm(primaryStage);
            Scene paymentScene = new Scene(paymentForm, 400, 400);
            primaryStage.setScene(paymentScene);
        });
        serviceSelectionPage.getChildren().add(bookButton);

        return serviceSelectionPage;
    }

    // Method to create payment form
    private VBox createPaymentForm(Stage primaryStage) {
        VBox paymentForm = new VBox(10);
        paymentForm.setAlignment(Pos.CENTER);
        paymentForm.setPadding(new Insets(20));

        Label titleLabel = new Label("Payment");
        titleLabel.setFont(Font.font("Arial", 24));
        titleLabel.setTextFill(Color.DARKBLUE);
        paymentForm.getChildren().add(titleLabel);

        ToggleGroup paymentMethodGroup = new ToggleGroup();
        RadioButton creditCardRadioButton = new RadioButton("Credit Card");
        creditCardRadioButton.setToggleGroup(paymentMethodGroup);
        creditCardRadioButton.setSelected(true); // Default selection
        RadioButton debitCardRadioButton = new RadioButton("Debit Card");
        debitCardRadioButton.setToggleGroup(paymentMethodGroup);

        VBox paymentMethodBox = new VBox(5, creditCardRadioButton,
debitCardRadioButton);
        paymentMethodBox.setAlignment(Pos.CENTER_LEFT);
        paymentForm.getChildren().add(paymentMethodBox);

        TextField nameField = new TextField();
        nameField.setPromptText("Name on Card");
        nameField.setFont(Font.font("Arial", 14));
        paymentForm.getChildren().add(nameField);

        TextField cardNumberField = new TextField();
        cardNumberField.setPromptText("Card Number");
        cardNumberField.setFont(Font.font("Arial", 14));
```

```
        paymentForm.getChildren().add(cardNumberField);

        TextField cvvField = new TextField();
        cvvField.setPromptText("CVV");
        cvvField.setFont(Font.font("Arial", 14));
        paymentForm.getChildren().add(cvvField);

        Button payButton = new Button("Pay");
        payButton.setFont(Font.font("Arial", 14));
        payButton.setStyle("-fx-background-color: DARKBLUE; -fx-text-fill:
WHITE;");
        payButton.setOnAction(event -> {
            // Process payment and show confirmation
            showAlert(Alert.AlertType.INFORMATION, "Payment Successful",
"Payment processed successfully!");

            // Show options to view bookings or go back to home
            VBox optionsBox = new VBox(10);
            optionsBox.setAlignment(Pos.CENTER);
            optionsBox.setPadding(new Insets(20));

            Button viewBookingButton = new Button("View Bookings");
            viewBookingButton.setFont(Font.font("Arial", 14));
            viewBookingButton.setStyle("-fx-background-color: DARKBLUE; -
fx-text-fill: WHITE;");
            viewBookingButton.setOnAction(e -> {
                // Show customer bookings
                StringBuilder bookings = new StringBuilder("Your Bookings:\
n");
                for (String booking : customerBookings) {
                    bookings.append(booking).append("\n");
                }
                showAlert(Alert.AlertType.INFORMATION, "Your Bookings",
bookings.toString());
            });

            Button backToHomeButton = new Button("Back to Home");
            backToHomeButton.setFont(Font.font("Arial", 14));
            backToHomeButton.setStyle("-fx-background-color: DARKBLUE; -fx-
text-fill: WHITE;");
            backToHomeButton.setOnAction(e -> {
                // Go back to the login page
                Scene loginScene = new Scene(createLoginPage(primaryStage),
800, 800);
                primaryStage.setScene(loginScene);
            });

            optionsBox.getChildren().addAll(viewBookingButton,
backToHomeButton);
            Scene optionsScene = new Scene(optionsBox, 500, 500);
            primaryStage.setScene(optionsScene);
        });
        paymentForm.getChildren().add(payButton);

        return paymentForm;
    }


    // Method to create service provider dashboard
```

University of Bedfordshire

```
    private VBox createServiceProviderDashboard(Stage primaryStage, String
username) {
        VBox dashboard = new VBox(10);
        dashboard.setAlignment(Pos.CENTER);
        dashboard.setPadding(new Insets(20));

        Label titleLabel = new Label("Service Provider Dashboard");
        titleLabel.setFont(Font.font("Arial", 24));
        titleLabel.setTextFill(Color.DARKBLUE);
        dashboard.getChildren().add(titleLabel);

        // Display bookings
        ListView<String> bookingListView = new ListView<>();
        bookingListView.getItems().addAll(serviceProviderBookings);
        dashboard.getChildren().addAll(new Label("Bookings:"),
bookingListView);

        Button editServicesButton = new Button("Edit Services");
        editServicesButton.setFont(Font.font("Arial", 14));
        editServicesButton.setStyle("-fx-background-color: DARKBLUE; -fx-
text-fill: WHITE;");
        editServicesButton.setOnAction(event -> {
            // Move to service selection page in edit mode
            VBox serviceSelectionPage =
createServiceSelectionPage(primaryStage, username, true);
            Scene scene = new Scene(serviceSelectionPage, 800, 800);
            primaryStage.setScene(scene);
        });
        dashboard.getChildren().add(editServicesButton);

        // Take service option
        Button takeServiceButton = new Button("Take Service");
        takeServiceButton.setFont(Font.font("Arial", 14));
        takeServiceButton.setStyle("-fx-background-color: DARKBLUE; -fx-
text-fill: WHITE;");
        takeServiceButton.setOnAction(event -> {
            // Move to service selection page for service provider
            VBox serviceSelectionPage =
createServiceSelectionPageForServiceProvider(primaryStage, username);
            Scene scene = new Scene(serviceSelectionPage, 800, 800);
            primaryStage.setScene(scene);
        });
        dashboard.getChildren().add(takeServiceButton);

        // Back to home button
        Button backToHomeButton = new Button("Back to Home");
        backToHomeButton.setFont(Font.font("Arial", 14));
        backToHomeButton.setStyle("-fx-background-color: DARKBLUE; -fx-
text-fill: WHITE;");
        backToHomeButton.setOnAction(event -> {
            // Go back to the login page
            Scene loginScene = new Scene(createLoginPage(primaryStage),
800, 800);
            primaryStage.setScene(loginScene);
        });
        dashboard.getChildren().add(backToHomeButton);

        return dashboard;
    }
```

University of Bedfordshire

```
    // Method to create service selection page for service provider
    private VBox createServiceSelectionPageForServiceProvider(Stage
primaryStage, String username) {
        VBox serviceSelectionPage = new VBox(10);
        serviceSelectionPage.setAlignment(Pos.CENTER);
        serviceSelectionPage.setPadding(new Insets(20));

        Label titleLabel = new Label("Take Service");
        titleLabel.setFont(Font.font("Arial", 24));
        titleLabel.setTextFill(Color.DARKBLUE);
        serviceSelectionPage.getChildren().add(titleLabel);

        // Create checkboxes for service selection
        List<CheckBox> serviceCheckboxes = new ArrayList<>();
        for (String service : services) {
            CheckBox checkBox = new CheckBox(service);
            checkBox.setFont(Font.font("Arial", 14));
            serviceCheckboxes.add(checkBox);
            serviceSelectionPage.getChildren().add(checkBox);
        }

        Button takeServiceButton = new Button("Take Service");
        takeServiceButton.setFont(Font.font("Arial", 14));
        takeServiceButton.setStyle("-fx-background-color: DARKBLUE; -fx-
text-fill: WHITE;");
        takeServiceButton.setOnAction(event -> {
            // Process selected services and show confirmation
            StringBuilder selectedServices = new StringBuilder("Selected
Services:\n");
            for (int i = 0; i < services.length; i++) {
                CheckBox checkBox = serviceCheckboxes.get(i);
                if (checkBox.isSelected()) {
                    selectedServices.append(services[i]).append("\n");
                }
            }
            showAlert(Alert.AlertType.INFORMATION, "Service Taken",
selectedServices.toString() +
                    "Service taken successfully!");
        });
        serviceSelectionPage.getChildren().add(takeServiceButton);

        // Back to home button
        Button backToHomeButton = new Button("Back to Home");
        backToHomeButton.setFont(Font.font("Arial", 14));
        backToHomeButton.setStyle("-fx-background-color: DARKBLUE; -fx-
text-fill: WHITE;");
        backToHomeButton.setOnAction(event -> {
            // Go back to the login page
            Scene loginScene = new Scene(createLoginPage(primaryStage),
800, 800);
            primaryStage.setScene(loginScene);
        });
        serviceSelectionPage.getChildren().add(backToHomeButton);

        return serviceSelectionPage;
    }

    // Method to display alert dialogs
```

```
    private void showAlert(Alert.AlertType alertType, String title, String
content) {
        Alert alert = new Alert(alertType);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(content);
        alert.showAndWait();
    }

    @Override
    public void stop() {
        // Cleanup resources if needed
    }
}
```