# 

**Lab Assignment #4**

**Submitted to: Professor SHAHID BHATTI**

**Submitted by:**

**1:Muhammad Saad Badar (sp24-bse-082)  
  
2:Hafiz Abdullah Aslam (sp24-bse-028)  
  
3:Sohaib Athar (sp24-bse-115)**

**Submitted DATE: 20/12.2024**

Code is as Follows:

**Main Class**

package org.project;

import javafx.application.Application;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.CheckBox;

import javafx.scene.control.Hyperlink;

import javafx.scene.control.Label;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

import javafx.scene.layout.StackPane;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

public class Main extends Application {

    public AccountManager accountManager = new AccountManager();

    public AdminData adminData = new AdminData();

    public AdminDashboard admindashboard = new AdminDashboard();

    Dashboard dashboard = new Dashboard();

    VBox loginForm, signUpForm, title, forgotPasswordForm;

    Stage stage = new Stage();

    @Override

    public void start(Stage stage) {

//        dashboard.showDashboard();

        loginForm = createLoginForm();

        title = titleForm();

        signUpForm = createRegisterForm();

        forgotPasswordForm = createForgotPasswordForm();

        StackPane root = new StackPane();

        StackPane.setAlignment(loginForm, Pos.CENTER\_RIGHT);

        StackPane.setAlignment(signUpForm, Pos.CENTER\_RIGHT);

        StackPane.setAlignment(forgotPasswordForm, Pos.CENTER\_RIGHT);

        StackPane.setAlignment(title, Pos.CENTER\_LEFT);

        loginForm.setMaxSize(300, 200);

        signUpForm.setMaxSize(300, 200);

        title.setMaxSize(300, 600);

        forgotPasswordForm.setMaxSize(300, 200);

        signUpForm.setVisible(false);

        forgotPasswordForm.setVisible(false);

        root.getChildren().addAll(title, loginForm, signUpForm, forgotPasswordForm);

        Scene scene = new Scene(root, 600, 400);

        scene.getStylesheets().add(getClass().getResource("styles.css").toExternalForm());

        stage.setTitle("Cafe Shop Management System");

        stage.setScene(scene);

        stage.setResizable(false);

        stage.show();

    }

    private VBox createLoginForm() {

        VBox login = new VBox(20);

        login.setPadding(new Insets(10, 50, 50, 50));

        login.setAlignment(Pos.CENTER\_LEFT);

        Label loginTitle = new Label("Login Account");

        loginTitle.getStyleClass().add("login-title");

        TextField usernameField = new TextField();

        usernameField.setPromptText("Username");

        usernameField.setPrefHeight(30);

        usernameField.setPrefWidth(200);

        PasswordField passwordField = new PasswordField();

        passwordField.setPromptText("Password");

        passwordField.setPrefHeight(30);

        passwordField.setPrefWidth(200);

        // TextField for showing password dynamically

        TextField passwordTextField = new TextField();

        passwordTextField.setManaged(false);

        passwordTextField.setVisible(false);

        CheckBox showPasswordCheckBox = new CheckBox("Show Password");

        showPasswordCheckBox.setStyle("-fx-text-fill: #000; -fx-font-family: \"Arial\";");

        passwordTextField.textProperty().bindBidirectional(passwordField.textProperty());

        CheckBox adminCheckBox = new CheckBox("Admin");

        showPasswordCheckBox.setStyle("-fx-text-fill: #000; -fx-font-family: \"Arial\";");

        // Toggle visibility on checkbox

        showPasswordCheckBox.setOnAction(e -> {

            if (showPasswordCheckBox.isSelected()) {

                passwordField.setVisible(false);

                passwordField.setManaged(false);

                passwordTextField.setVisible(true);

                passwordTextField.setManaged(true);

            } else {

                passwordTextField.setVisible(false);

                passwordTextField.setManaged(false);

                passwordField.setVisible(true);

                passwordField.setManaged(true);

            }

        });

        Hyperlink forgotPasswordLink = new Hyperlink("Forgot Password?");

        forgotPasswordLink.getStyleClass().add("forgot-password-link");

        forgotPasswordLink.setOnAction(e -> {

            loginForm.setVisible(false);

            forgotPasswordForm.setVisible(true);

        });

        Button loginButton = new Button("Login");

        loginButton.getStyleClass().add("button-login");

        loginButton.setPrefHeight(30);

        loginButton.setPrefWidth(200);

        Label warning = new Label();

        // Event handler for login button

        loginButton.setOnAction(e -> {

            String username = usernameField.getText();

            String password = passwordField.getText();

            if (username.isEmpty() || password.isEmpty()) {

                warning.setText("Please fill in all fields.");

                warning.setStyle("-fx-text-fill: red; -fx-font-family: \"Arial\";");

                return;

            }

            if (adminCheckBox.isSelected()) {

                if (username.equals(adminData.getName()) && password.equals(adminData.getPassword())) {

                    admindashboard.showAdminDashboard();

                    usernameField.clear();

                    passwordField.clear();

                } else {

                    warning.setText("Invalid username or password.");

                    warning.setStyle("-fx-text-fill: red; -fx-font-family: \"Arial\";");

                }

            } else if (accountManager.verify(username, password)) {

                warning.setText("Login successful!");

                warning.setStyle("-fx-text-fill: green; -fx-font-family: \"Arial\";");

                dashboard.showDashboard();

                usernameField.clear();

                passwordField.clear();

                this.stage = stage;

                stage.close();

            } else {

                warning.setText("Invalid username or password.");

                warning.setStyle("-fx-text-fill: red; -fx-font-family: \"Arial\";");

            }

        });

        login.getChildren().addAll(loginTitle, usernameField, passwordField, passwordTextField, showPasswordCheckBox,

                adminCheckBox, forgotPasswordLink, loginButton, warning);

        login.setMargin(loginTitle, new Insets(30, 0, 0, 0));

        return login;

    }

    private VBox createRegisterForm() {

        VBox register = new VBox(20);

        register.setPadding(new Insets(50, 50, 50, 50));

        register.setAlignment(Pos.CENTER\_LEFT);

        Label registerTitle = new Label("Register Account");

        registerTitle.getStyleClass().add("register-title");

        TextField usernameField = new TextField();

        usernameField.setPromptText("Username");

        usernameField.setPrefHeight(30);

        usernameField.setPrefWidth(200);

        TextField emailField = new TextField();

        emailField.setPromptText("Email");

        emailField.setPrefHeight(30);

        emailField.setPrefWidth(200);

        // Password fields

        PasswordField passwordField = new PasswordField();

        passwordField.setPromptText("Password");

        passwordField.setPrefHeight(30);

        passwordField.setPrefWidth(200);

        PasswordField confirmPasswordField = new PasswordField();

        confirmPasswordField.setPromptText("Confirm Password");

        confirmPasswordField.setPrefHeight(30);

        confirmPasswordField.setPrefWidth(200);

        // TextFields for showing passwords dynamically

        TextField passwordTextField = new TextField();

        passwordTextField.setManaged(false);

        passwordTextField.setVisible(false);

        TextField confirmPasswordTextField = new TextField();

        confirmPasswordTextField.setManaged(false);

        confirmPasswordTextField.setVisible(false);

        // Synchronize text properties

        passwordTextField.textProperty().bindBidirectional(passwordField.textProperty());

        confirmPasswordTextField.textProperty().bindBidirectional(confirmPasswordField.textProperty());

        // Checkbox to toggle password visibility

        CheckBox showPasswordCheckBox = new CheckBox("Show Passwords");

        showPasswordCheckBox.setStyle("-fx-text-fill: #000; -fx-font-family: \"Arial\";");

        showPasswordCheckBox.setOnAction(e -> {

            if (showPasswordCheckBox.isSelected()) {

                passwordField.setVisible(false);

                passwordField.setManaged(false);

                confirmPasswordField.setVisible(false);

                confirmPasswordField.setManaged(false);

                passwordTextField.setVisible(true);

                passwordTextField.setManaged(true);

                confirmPasswordTextField.setVisible(true);

                confirmPasswordTextField.setManaged(true);

            } else {

                passwordTextField.setVisible(false);

                passwordTextField.setManaged(false);

                confirmPasswordTextField.setVisible(false);

                confirmPasswordTextField.setManaged(false);

                passwordField.setVisible(true);

                passwordField.setManaged(true);

                confirmPasswordField.setVisible(true);

                confirmPasswordField.setManaged(true);

            }

        });

        Button signupButton = new Button("Signup");

        signupButton.getStyleClass().add("button-login");

        signupButton.setPrefHeight(30);

        signupButton.setPrefWidth(200);

        Label warning = new Label();

        // Event handler for signup button

        signupButton.setOnAction(e -> {

            String username = usernameField.getText();

            String email = emailField.getText();

            String password = passwordField.getText();

            String confirmPassword = confirmPasswordField.getText();

            if (username.isEmpty() || email.isEmpty() || password.isEmpty() || confirmPassword.isEmpty()) {

                warning.setText("Please fill in all fields.");

                warning.setStyle("-fx-text-fill: red; -fx-font-family: \"Arial\";");

                return;

            }

            if (password.length() < 8) {

                warning.setText("Password must be at least 8 characters long.");

                warning.setStyle("-fx-text-fill: red; -fx-font-family: \"Arial\";");

                return;

            }

            if (!password.equals(confirmPassword)) {

                warning.setText("Passwords do not match.");

                warning.setStyle("-fx-text-fill: red; -fx-font-family: \"Arial\";");

                return;

            }

            if (accountManager.register(username, password, email)) {

                warning.setText("Account created successfully!");

                warning.setStyle("-fx-text-fill: green; -fx-font-family: \"Arial\";");

                loginForm.setVisible(true);

                signUpForm.setVisible(false);

            } else {

                warning.setText("Username already exists.");

                warning.setStyle("-fx-text-fill: red; -fx-font-family: \"Arial\";");

            }

        });

        register.getChildren().addAll(registerTitle, usernameField, emailField, passwordField, passwordTextField,

                confirmPasswordField, confirmPasswordTextField, showPasswordCheckBox, signupButton, warning);

        return register;

    }

    private VBox titleForm() {

        VBox switchPane = new VBox(20);

        switchPane.setAlignment(Pos.CENTER);

        switchPane.getStyleClass().add("shift-pane");

        Label logo = new Label("\uD83C\uDF7D");

        logo.setStyle("-fx-font-size: 100; -fx-text-fill: white;");

        Label systemTitle = new Label("Food Hive");

        systemTitle.getStyleClass().add("title");

        Button createAccountButton = new Button("Create new Account");

        createAccountButton.getStyleClass().add("button-create-account");

        createAccountButton.setOnAction(e -> {

            signUpForm.setVisible(true);

            loginForm.setVisible(false);

        });

        Button alreadyAccountButton = new Button("Already have Account");

        alreadyAccountButton.getStyleClass().add("button-create-account");

        alreadyAccountButton.setOnAction(e -> {

            loginForm.setVisible(true);

            signUpForm.setVisible(false);

            stage.close();

        });

        switchPane.getChildren().addAll(logo, systemTitle, createAccountButton, alreadyAccountButton);

        return switchPane;

    }

    private VBox createForgotPasswordForm() {

        VBox forgotPassword = new VBox(10);

        forgotPassword.setPadding(new Insets(10, 50, 50, 50));

        forgotPassword.setAlignment(Pos.CENTER\_LEFT);

        Label forgotPasswordTitle = new Label("Forgot Password");

        forgotPasswordTitle.getStyleClass().add("forgot-password-title");

        Label enterEmailTitle = new Label("Enter your Email.");

        TextField emailField = new TextField();

        emailField.setPrefWidth(200);

        emailField.setPrefHeight(30);

        Button submitButton = new Button("Submit");

        submitButton.getStyleClass().add("button-submit");

        submitButton.setPrefWidth(200);

        submitButton.setPrefHeight(20);

        Label statusLabel = new Label();

        // VBox to hold the new password fields

        VBox resetPasswordForm = new VBox(20);

        resetPasswordForm.setVisible(false);

        resetPasswordForm.setManaged(false);

        TextField newPasswordField = new TextField();

        newPasswordField.setPromptText("New Password");

        newPasswordField.setPrefWidth(200);

        newPasswordField.setPrefHeight(20);

        TextField confirmPasswordField = new TextField();

        confirmPasswordField.setPromptText("Confirm New Password");

        confirmPasswordField.setPrefWidth(200);

        confirmPasswordField.setPrefHeight(20);

        Button savePasswordButton = new Button("Save Password");

        savePasswordButton.getStyleClass().add("button-save");

        savePasswordButton.setPrefWidth(200);

        savePasswordButton.setPrefHeight(20);

        savePasswordButton.setOnAction(e -> {

            String newPassword = newPasswordField.getText();

            String confirmPassword = confirmPasswordField.getText();

            if (newPassword.isEmpty() || confirmPassword.isEmpty()) {

                statusLabel.setText("Please fill in all fields.");

                statusLabel.setStyle("-fx-text-fill: red;");

                return;

            }

            if (newPassword.length() < 8) {

                statusLabel.setText("Password must be at least 8 characters long.");

                statusLabel.setStyle("-fx-text-fill: red;");

                return;

            }

            if (!newPassword.equals(confirmPassword)) {

                statusLabel.setText("Passwords do not match.");

                statusLabel.setStyle("-fx-text-fill: red;");

                return;

            }

            // Update the password in the account manager

            accountManager.updatePassword(emailField.getText(), newPassword);

            // Show success message

            statusLabel.setText("Password saved successfully!");

            statusLabel.setStyle("-fx-text-fill: green;");

            // Hide all forms and go back to the login page

            forgotPassword.setVisible(false);

            resetPasswordForm.setVisible(false);

            loginForm.setVisible(true);

        });

        resetPasswordForm.getChildren().addAll(newPasswordField, confirmPasswordField, savePasswordButton);

        submitButton.setOnAction(e -> {

            String email = emailField.getText();

            if (email.isEmpty()) {

                statusLabel.setText("Please enter an email address.");

                statusLabel.setStyle("-fx-text-fill: red;");

            } else if (!accountManager.isEmailRegistered(email)) { // Check email validity

                statusLabel.setText("Email not registered.");

                statusLabel.setStyle("-fx-text-fill: red;");

            } else {

                statusLabel.setText("Email verified. Enter your new password.");

                statusLabel.setStyle("-fx-text-fill: green;");

                resetPasswordForm.setVisible(true);

                resetPasswordForm.setManaged(true);

            }

        });

        Button backButton = new Button("Back to Login");

        backButton.getStyleClass().add("button-back");

        backButton.setPrefWidth(200);

        backButton.setPrefHeight(20);

        // Event handling for the back button

        backButton.setOnAction(e -> {

            // Reset the visibility and fields when going back

            resetPasswordForm.setVisible(false);

            resetPasswordForm.setManaged(false);

            statusLabel.setText("");

            emailField.clear();

            newPasswordField.clear();

            confirmPasswordField.clear();

            forgotPassword.setVisible(false);

            loginForm.setVisible(true);

        });

        forgotPassword.getChildren().addAll(forgotPasswordTitle, enterEmailTitle, emailField, submitButton, statusLabel,

                resetPasswordForm, backButton);

        forgotPassword.setMargin(forgotPasswordTitle, new Insets(25, 0, 0, 0));

        return forgotPassword;

    }

    public static void main(String[] args) {

        launch(args);

    }

}

**Dashboard Class**

package org.project;

import javafx.collections.ListChangeListener;

import javafx.collections.ObservableList;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Node;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.image.ImageView;

import javafx.scene.layout.\*;

import javafx.stage.Stage;

import java.net.URL;

import java.util.List;

import javafx.animation.FadeTransition;

import javafx.util.Duration;

public class Dashboard {

    Item item = new Item();

    TableView<Item> tableView = new TableView<>();

    // Create three vertical boxes

    static VBox vBox1 = new VBox(20);

    static VBox vBox2 = new VBox(20);

    static VBox vBox3 = new VBox(20);

    static VBox showRestaurant = new VBox(20);

    static VBox categories = new VBox(20);

    ComboBox<String> categoryComboBox = new ComboBox<>();

    public void showDashboard() {

        Stage dashboardStage = new Stage();

        dashboardStage.setTitle("Cafe Shop Management System");

        vBox2.setVisible(true);

        showRestaurant.setVisible(false);

        categories.setVisible(false);

        HBox topHBox = new HBox(20);

        topHBox.getStyleClass().add("top-hbox-style");

        topHBox.setAlignment(Pos.CENTER);

        Label foodHiveLabel = new Label("FOOD HIVE");

        foodHiveLabel.getStyleClass().add("food-hive-label");

        Label logo = new Label("\uD83C\uDF7D");

        logo.setStyle("-fx-font-size: 50; -fx-text-fill: #ffce1b;");

        topHBox.getChildren().addAll(logo, foodHiveLabel);

        vBox1.setMaxWidth(150);

        vBox3.setMaxWidth(300);

        vBox1.getStyleClass().add("vbox1-style");

        vBox2.getStyleClass().add("vbox2-style");

        vBox3.getStyleClass().add("vbox3-style");

        showRestaurant.getStyleClass().add("show-restaurant-style");

        categories.getStyleClass().add("categories-style");

        setupVBox1();

        setupVBox2();

        setupVBox3();

        StackPane mainScreen = new StackPane(vBox2, showRestaurant , categories);

        mainScreen.setPrefSize(800, 600);

        HBox hBox = new HBox(5, vBox1, mainScreen, vBox3);

        hBox.setAlignment(Pos.CENTER);

        hBox.setStyle("-fx-background-color: white;");

        HBox.setHgrow(vBox1, Priority.ALWAYS);

        HBox.setHgrow(vBox2, Priority.ALWAYS);

        HBox.setHgrow(vBox3, Priority.ALWAYS);

        VBox mainVBox = new VBox(10, topHBox, hBox);

        VBox.setVgrow(hBox, Priority.ALWAYS);

        Scene scene = new Scene(mainVBox, 1000, 600);

        scene.getStylesheets().add(getClass().getResource("styles.css").toExternalForm());

        dashboardStage.setScene(scene);

        dashboardStage.setFullScreen(true);

        dashboardStage.show();

    }

    private void setupVBox1() {

        vBox1.setPadding(new Insets(5));

        vBox1.setAlignment(Pos.CENTER);

        Button homeButton = new Button("Home");

        homeButton.getStyleClass().add("home-button");

        homeButton.setOnAction(event -> {

            showRestaurant.setVisible(false);

            categories.setVisible(false);

            vBox2.setVisible(true);

            categoryComboBox.setPromptText("Category");

            categoryComboBox.setValue(null);

            categoryComboBox.getEditor().clear();

        });

        homeButton.setPrefSize(220, 40);

        Button restaurantsButton = new Button("Restaurants");

        restaurantsButton.getStyleClass().add("restaurants-button");

        restaurantsButton.setPrefSize(220, 40);

        restaurantsButton.setOnAction(event -> {

            vBox2.setVisible(false);

            categories.setVisible(false);

            showRestaurant.setVisible(true);

            showRestaurants();

            categoryComboBox.setPromptText("Category");

            categoryComboBox.setValue(null);

            categoryComboBox.getEditor().clear();

        });

        categoryComboBox = new ComboBox<>();

        categoryComboBox.getItems().addAll("Fast Food", "Desi Food", "Sweets");

        categoryComboBox.setPrefSize(220, 40);

        categoryComboBox.setPromptText("Category");

        categoryComboBox.getStyleClass().add("categories-comboBox");

        categoryComboBox.setOnAction(event -> {

            String selectedCategory = categoryComboBox.getValue();

            switch (selectedCategory) {

                case "Fast Food":

                    showRestaurant.setVisible(false);

                    vBox2.setVisible(false);

                    categories.setVisible(true);

                    displayFastFoodItem();

                    break;

                case "Desi Food":

                    showRestaurant.setVisible(false);

                    vBox2.setVisible(false);

                    categories.setVisible(true);

                    displayDesiFoodItem();

                    break;

                case "Sweets":

                    showRestaurant.setVisible(false);

                    vBox2.setVisible(false);

                    categories.setVisible(true);

                    displaySweatsItem();

                    break;

            }

        });

        VBox buttonBox = new VBox(20, homeButton, restaurantsButton,  categoryComboBox );

        buttonBox.setAlignment(Pos.CENTER);

        vBox1.getChildren().add(buttonBox);

    }

    private void setupVBox2() {

        categories.getChildren().clear();

        vBox2.getChildren().clear();

        vBox2.setPadding(new Insets(5));

        vBox2.setAlignment(Pos.CENTER);

        TilePane tilePane = new TilePane();

        tilePane.setPadding(new Insets(5));

        tilePane.setHgap(10);

        tilePane.setVgap(5);

        tilePane.setPrefColumns(3);

        tilePane.setAlignment(Pos.CENTER);

        // Load menu items from the file

        List<MenuItem> menuItems = DataLoader.loadMenuItems("MenuData.txt");

        for (MenuItem menuItem : menuItems) {

            VBox itemBox = new VBox(5);

            itemBox.setAlignment(Pos.CENTER);

            itemBox.setStyle("-fx-border-color: lightgray; -fx-border-width: 1; -fx-padding: 10px;");

            // Loading image from the resource path

            ImageView imageView = null;

            URL resource = getClass().getResource(menuItem.getImagePath());

            if (resource == null) {

                System.out.println("Image not found: " + menuItem.getImagePath());

            } else {

                imageView = new ImageView(resource.toExternalForm());

            }

            if (imageView != null) {

                imageView.setFitWidth(250);

                imageView.setFitHeight(80);

                imageView.setPreserveRatio(true);

            }

            Label itemNameLabel = new Label(menuItem.getName());

            itemNameLabel.setStyle("-fx-font-size: 14; -fx-font-weight: bold; -fx-text-fill: #ffce1b;");

            Label itemPriceLabel = new Label("$" + menuItem.getPrice());

            itemPriceLabel.setStyle("-fx-font-size: 12; -fx-text-fill: white;");

            Label titleLabel = new Label("Select Quantity");

            titleLabel.setStyle("-fx-font-size: 16; -fx-font-weight: bold;");

            HBox counterBox = new HBox(10);

            counterBox.setAlignment(Pos.CENTER);

            Button minusButton = new Button("-");

            minusButton.getStyleClass().add("minus-button");

            Label quantityLabel = new Label("0");

            quantityLabel.setStyle("-fx-font-size: 16; -fx-background-color:transparent; -fx-text-fill:#ffce1b; ");

            Button plusButton = new Button("+");

            plusButton.getStyleClass().add("plus-button");

            counterBox.getChildren().addAll(quantityLabel);

            final int[] quantity = {0}; // Use array to modify in lambda

            minusButton.setOnAction(e -> {

                if (quantity[0] > 0) {

                    quantity[0]--;

                    quantityLabel.setText(String.valueOf(quantity[0]));

                }

            });

            plusButton.setOnAction(e -> {

                quantity[0]++;

                quantityLabel.setText(String.valueOf(quantity[0]));

            });

            Button addToCartButton = new Button("Add to Order");

            addToCartButton.setPrefSize(100, 20);

            addToCartButton.setStyle("-fx-background-color: #ffce1b;  -fx-text-fill: black; -fx-border-radius: 5px;\n"

                    + "    -fx-cursor: hand;");

            addToCartButton.setOnAction(event -> {

                if (quantity[0] > 0) {

                    addItemToTable(menuItem.getName(), quantity[0], menuItem.getPrice() \* quantity[0]);

                    quantity[0] = 0;

                    quantityLabel.setText(String.valueOf(quantity[0]));

                }

            });

            // Create an HBox for name and price to be aligned horizontally

            HBox namePriceBox = new HBox(10);

            namePriceBox.setAlignment(Pos.CENTER);

            namePriceBox.getChildren().addAll(itemNameLabel, itemPriceLabel);

            HBox hBoxcart = new HBox(10);

            hBoxcart.setAlignment(Pos.CENTER);

            hBoxcart.getChildren().addAll(minusButton, addToCartButton, plusButton);

            // Add the HBox (name and price) above the image in the VBox

            itemBox.getChildren().addAll(namePriceBox, imageView, counterBox, hBoxcart);

            tilePane.getChildren().add(itemBox);

        }

        tilePane.widthProperty().addListener((observable, oldValue, newValue) -> {

            double tileWidth = newValue.doubleValue() / 3 - tilePane.getHgap() \* 2;

            for (Node node : tilePane.getChildren()) {

                if (node instanceof VBox) {

                    VBox item = (VBox) node;

                    item.setPrefWidth(tileWidth);

                }

            }

        });

        vBox2.getChildren().add(tilePane);

    }

    private void setupVBox3() {

        vBox3.setPadding(new Insets(10));

        vBox3.setAlignment(Pos.TOP\_CENTER);

        Label orderLabel = new Label("Your Order");

        orderLabel.getStyleClass().add("orderLabel");

        // Table columns

        TableColumn<Item, String> itemNameColumn = new TableColumn<>("Item Name");

        itemNameColumn.setMinWidth(100);

        itemNameColumn.setCellValueFactory(new PropertyValueFactory<>("name"));

        TableColumn<Item, Integer> quantityColumn = new TableColumn<>("Quantity");

        quantityColumn.setMinWidth(40);

        quantityColumn.setCellValueFactory(new PropertyValueFactory<>("quantity"));

        TableColumn<Item, Double> priceColumn = new TableColumn<>("Price");

        priceColumn.setMaxWidth(50);

        priceColumn.setCellValueFactory(new PropertyValueFactory<>("price"));

        // Remove button column

        TableColumn<Item, Void> removeColumn = new TableColumn<>("Remove");

        removeColumn.setMaxWidth(70);

        // Add a "Remove" button to each row

        removeColumn.setCellFactory(param -> new TableCell<>() {

            private final Button removeButton = new Button("\uD83D\uDDD1");

            {

                removeButton.setStyle("-fx-background-color: black; -fx-text-fill: white; -fx-border-radius: 5px;");

                removeButton.setOnAction(event -> {

                    Item item = getTableView().getItems().get(getIndex());

                    tableView.getItems().remove(item);

                });

            }

            @Override

            protected void updateItem(Void item, boolean empty) {

                super.updateItem(item, empty);

                if (empty) {

                    setGraphic(null);

                } else {

                    setGraphic(removeButton);

                }

            }

        });

        // Add columns to the table

        tableView.getColumns().clear();

        tableView.getColumns().addAll(itemNameColumn, quantityColumn, priceColumn, removeColumn);

        // Total price label

        Label totalPriceLabel = new Label();

        totalPriceLabel.setStyle("-fx-font-size: 16; -fx-font-weight: bold; -fx-text-fill: white;");

        Label paymentLabel = new Label("Choose a payment option:");

        Button payButton = new Button("Pay Now");

        payButton.getStyleClass().add("payButton");

        payButton.setPrefSize(220,40);

        payButton.setStyle("-fx-font-size: 16; -fx-font-weight: bold; -fx-text-fill: white;");

        payButton.setOnAction(event -> {

            tableView.getItems().clear();

            totalPriceLabel.setText("Total Price: $0.00");

            // Create a custom dialog to show the checkboxes for payment options

            CheckBox cashCheckBox = new CheckBox("CASH");

            CheckBox cardCheckBox = new CheckBox("CARD");

            // Create a VBox to hold the checkboxes

            VBox vbox = new VBox(cashCheckBox, cardCheckBox);

            vbox.setSpacing(10);

            // Create the dialog with checkboxes

            Alert alert = new Alert(Alert.AlertType.CONFIRMATION);

            alert.setTitle("Payment Method");

            alert.setHeaderText("How would you like to pay?");

            alert.getDialogPane().setContent(vbox);

            // Show the dialog and wait for user response

            alert.showAndWait().ifPresent(response -> {

                if (response == ButtonType.OK) {

                    if (cashCheckBox.isSelected()) {

                        paymentLabel.setText("Give cash to driver.");

                    } else if (cardCheckBox.isSelected()) {

                        TextInputDialog pinDialog = new TextInputDialog();

                        pinDialog.setTitle("Enter PIN");

                        pinDialog.setHeaderText("Please enter your PIN to confirm.");

                        pinDialog.setContentText("PIN:");

                        pinDialog.showAndWait().ifPresent(pin -> {

                            if (pin != null && !pin.isEmpty()) {

                                paymentLabel.setText("Order confirmed. Payment successful.");

                            } else {

                                paymentLabel.setText("Order cancelled.");

                            }

                        });

                    }

                }

            });

        });

        // Update the total price whenever the table changes

        Runnable updateTotalPrice = () -> {

            double total = tableView.getItems().stream()

                    .mapToDouble(Item::getPrice)

                    .sum();

            totalPriceLabel.setText("Total Price: $" + String.format("%.2f", total));

        };

        tableView.getItems().addListener((ListChangeListener<Item>) change -> updateTotalPrice.run());

        updateTotalPrice.run();

        vBox3.getChildren().clear();

        vBox3.getChildren().addAll(orderLabel, tableView, totalPriceLabel, payButton);

    }

    private void addItemToTable(String itemName, int quantity, double itemPrice) {

        ObservableList<Item> items = tableView.getItems();

        for (Item item : items) {

            if (item.getName().equals(itemName)) {

                item.setQuantity(item.getQuantity() + quantity);

                item.setPrice(itemPrice \* item.getQuantity());

                tableView.refresh();

                return;

            }

        }

        items.add(new Item(itemName, quantity, itemPrice));

    }

    private void showRestaurants() {

        categories.getChildren().clear();

        showRestaurant.getChildren().clear();

        showRestaurant.setPadding(new Insets(20));

        // Load restaurant data using DataLoader

        List<Restaurant> restaurants = DataLoader.loadRestaurants();

        // Create a FlowPane for restaurants

        FlowPane flowPane = new FlowPane();

        flowPane.setPadding(new Insets(2, 10, 5, 10));

        flowPane.setHgap(25);

        flowPane.setVgap(25);

        flowPane.setPrefWidth(470);

        flowPane.setAlignment(Pos.CENTER);

        for (Restaurant restaurant : restaurants) {

            // Create a VBox for each restaurant

            VBox restaurantBox = new VBox(5);

            restaurantBox.setPrefSize(180, 280);

            restaurantBox.getStyleClass().add("restaurantBox");

            restaurantBox.setAlignment(Pos.CENTER);

            // Add restaurant name and description

            Button nameLabel = new Button(restaurant.getName());

            nameLabel.setPrefSize(150, 30);

            nameLabel.getStyleClass().add("nameLabel");

            ImageView imageView = new ImageView(restaurant.getImagePath());

            imageView.setFitWidth(230);

            imageView.setFitHeight(80);

            imageView.setPreserveRatio(true);

            Label descriptionLabel = new Label(restaurant.getDescription());

            descriptionLabel.setWrapText(true);

            descriptionLabel.setStyle("-fx-text-fill: #ffce1b; -fx-font-size: 12; -fx-font-weight: bold;");

            // Set action on restaurant button

            nameLabel.setOnMouseClicked(event -> showMenu(restaurant));

            restaurantBox.getChildren().addAll(imageView, descriptionLabel, nameLabel);

            flowPane.getChildren().add(restaurantBox);

        }

        showRestaurant.getChildren().add(flowPane);

    }

    private void showMenu(Restaurant restaurant) {

        // Clear the existing content

        showRestaurant.getChildren().clear();

        // Center-align the menu pane

        showRestaurant.setPadding(new Insets(20));

        showRestaurant.setAlignment(Pos.CENTER);

        // Add a back button

        Button backButton = new Button("Back to Restaurants");

        backButton.getStyleClass().add("backButton");

        backButton.setOnAction(event -> showRestaurants());

        // Get menu items for the selected restaurant

        List<MenuItem> menuItems = restaurant.getMenuItems();

        // Create a TilePane for menu items

        TilePane tilePane = new TilePane();

        tilePane.setPadding(new Insets(5));

        tilePane.setHgap(10);

        tilePane.setVgap(5);

        tilePane.setPrefColumns(3);

        tilePane.setAlignment(Pos.CENTER);

        // Generate menu items

        for (MenuItem menuItem : menuItems) {

            VBox itemBox = new VBox(10);

            itemBox.setPadding(new Insets(10));

            itemBox.setAlignment(Pos.CENTER);

            itemBox.setStyle("-fx-border-color: lightgray; -fx-border-width: 1; -fx-padding: 10px;");

            // Menu item details

            String itemName = menuItem.getName();

            double itemPrice = menuItem.getPrice();

            String imagePath = menuItem.getImagePath();

            // Load image

            ImageView imageView = null;

            URL resource = getClass().getResource(imagePath);

            if (resource != null) {

                imageView = new ImageView(resource.toExternalForm());

                imageView.setFitWidth(150);

                imageView.setFitHeight(100);

                imageView.setPreserveRatio(true);

            }

            Label itemNameLabel = new Label(itemName);

            itemNameLabel.setStyle("-fx-font-size: 14; -fx-font-weight: bold; -fx-text-fill: #ffce1b;");

            Label itemPriceLabel = new Label("$" + itemPrice);

            itemPriceLabel.setStyle("-fx-font-size: 12; -fx-text-fill: white;");

            HBox counterBox = new HBox(10);

            counterBox.setAlignment(Pos.CENTER);

            Button minusButton = new Button("-");

            minusButton.getStyleClass().add("minus-button");

            Label quantityLabel = new Label("0");

            quantityLabel.setStyle("-fx-font-size: 16; -fx-text-fill: #ffce1b;");

            Button plusButton = new Button("+");

            plusButton.getStyleClass().add("plus-button");

            counterBox.getChildren().addAll(quantityLabel);

            final int[] quantity = {0};

            minusButton.setOnAction(e -> {

                if (quantity[0] > 0) {

                    quantity[0]--;

                    quantityLabel.setText(String.valueOf(quantity[0]));

                }

            });

            // Plus button action

            plusButton.setOnAction(e -> {

                quantity[0]++;

                quantityLabel.setText(String.valueOf(quantity[0]));

            });

            // Add to Cart Button

            Button addToCartButton = new Button("Add to Order");

            addToCartButton.setStyle("-fx-background-color: #ffce1b; -fx-text-fill: black;");

            addToCartButton.setOnAction(event -> {

                if (quantity[0] > 0) {

                    addItemToTable(itemName, quantity[0], itemPrice \* quantity[0]);

                    quantity[0] = 0; // Reset quantity after adding to cart

                    quantityLabel.setText(String.valueOf(quantity[0]));

                }

            });

            HBox namePriceBox = new HBox(10);

            namePriceBox.setAlignment(Pos.CENTER);

            namePriceBox.getChildren().addAll(itemNameLabel, itemPriceLabel);

            HBox hBoxCart = new HBox(10);

            hBoxCart.setAlignment(Pos.CENTER);

            hBoxCart.getChildren().addAll(minusButton, addToCartButton, plusButton);

            itemBox.getChildren().addAll(namePriceBox, imageView, counterBox, hBoxCart);

            tilePane.getChildren().add(itemBox);

        }

        showRestaurant.getChildren().addAll(backButton, tilePane);

    }

    private void displayItems(List<MenuItem> menuItems) {

        categories.getChildren().clear();

        categories.setAlignment(Pos.CENTER);

        categories.setPadding(new Insets(5));

        TilePane tilePane = new TilePane();

        tilePane.setPadding(new Insets(5));

        tilePane.setHgap(10);

        tilePane.setVgap(5);

        tilePane.setPrefColumns(3);

        tilePane.setAlignment(Pos.CENTER);

        for (MenuItem menuItem : menuItems) {

            VBox itemBox = createItemBox(menuItem.getName(), menuItem.getPrice(), menuItem.getImagePath());

            tilePane.getChildren().add(itemBox);

        }

        tilePane.widthProperty().addListener((observable, oldValue, newValue) -> {

            double tileWidth = newValue.doubleValue() / 3 - tilePane.getHgap() \* 2;

            for (Node node : tilePane.getChildren()) {

                if (node instanceof VBox) {

                    VBox item = (VBox) node;

                    item.setPrefWidth(tileWidth);

                }

            }

        });

        categories.getChildren().add(tilePane);

    }

    private VBox createItemBox(String itemName, double itemPrice, String imagePath) {

        VBox itemBox = new VBox(5);

        itemBox.setAlignment(Pos.CENTER);

        itemBox.setStyle("-fx-border-color: lightgray; -fx-border-width: 1; -fx-padding: 10px;");

        ImageView imageView = loadImage(imagePath);

        Label itemNameLabel = new Label(itemName);

        itemNameLabel.setStyle("-fx-font-size: 14; -fx-font-weight: bold; -fx-text-fill: #ffce1b;");

        Label itemPriceLabel = new Label("$" + itemPrice);

        itemPriceLabel.setStyle("-fx-font-size: 12; -fx-text-fill: white;");

        Label titleLabel = new Label("Select Quantity");

        titleLabel.setStyle("-fx-font-size: 16; -fx-font-weight: bold;");

        HBox counterBox = createCounterBox();

        Button addToCartButton = createAddToCartButton(itemName, itemPrice, counterBox);

        HBox namePriceBox = new HBox(10);

        namePriceBox.setAlignment(Pos.CENTER);

        namePriceBox.getChildren().addAll(itemNameLabel, itemPriceLabel);

        HBox hBoxcart = new HBox(10);

        hBoxcart.setAlignment(Pos.CENTER);

        hBoxcart.getChildren().addAll(counterBox, addToCartButton);

        itemBox.getChildren().addAll(namePriceBox, imageView, counterBox, hBoxcart);

        return itemBox;

    }

    private ImageView loadImage(String imagePath) {

        ImageView imageView = null;

        URL resource = getClass().getResource(imagePath);

        if (resource == null) {

            System.out.println("Image not found: " + imagePath);

        } else {

            imageView = new ImageView(resource.toExternalForm());

        }

        imageView.setFitWidth(250);

        imageView.setFitHeight(80);

        imageView.setPreserveRatio(true);

        return imageView;

    }

    private HBox createCounterBox() {

        HBox counterBox = new HBox(10);

        counterBox.setAlignment(Pos.CENTER);

        Button minusButton = new Button("-");

        minusButton.getStyleClass().add("minus-button");

        Label quantityLabel = new Label("0");

        quantityLabel.setStyle("-fx-font-size: 16; -fx-background-color:transparent; -fx-text-fill:#ffce1b; ");

        Button plusButton = new Button("+");

        plusButton.getStyleClass().add("plus-button");

        counterBox.getChildren().addAll(minusButton, quantityLabel, plusButton);

        final int[] quantity = {0};

        minusButton.setOnAction(e -> {

            if (quantity[0] > 0) {

                quantity[0]--;

                quantityLabel.setText(String.valueOf(quantity[0]));

            }

        });

        plusButton.setOnAction(e -> {

            quantity[0]++;

            quantityLabel.setText(String.valueOf(quantity[0]));

        });

        return counterBox;

    }

    private Button createAddToCartButton(String itemName, double itemPrice, HBox counterBox) {

        Button addToCartButton = new Button("Add to Order");

        addToCartButton.setPrefSize(100, 20);

        addToCartButton.setStyle("-fx-background-color: #ffce1b;  -fx-text-fill: black; -fx-border-radius: 5px;\n"

                + "    -fx-cursor: hand;");

        addToCartButton.setOnAction(event -> {

            Label quantityLabel = (Label) counterBox.getChildren().get(1); // Get quantity label

            int quantity = Integer.parseInt(quantityLabel.getText());

            if (quantity > 0) {

                addItemToTable(itemName, quantity, itemPrice \* quantity);

                quantityLabel.setText("0");

            }

        });

        return addToCartButton;

    }

    public void displayFastFoodItem() {

        List<MenuItem> fastFoodItems = DataLoader.loadCategoryItems("fast");

        displayItems(fastFoodItems);

    }

    public void displayDesiFoodItem() {

        List<MenuItem> desiFoodItems = DataLoader.loadCategoryItems("desi");

        displayItems(desiFoodItems);

    }

    public void displaySweatsItem() {

        List<MenuItem> sweetsItems = DataLoader.loadCategoryItems("sweets");

        displayItems(sweetsItems);

    }

}

**AccountManager Class**

package org.project;

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

import java.net.URL;

import java.util.List;

import javafx.beans.binding.Bindings;

import javafx.collections.ObservableList;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.image.ImageView;

import javafx.scene.layout.FlowPane;

import javafx.scene.layout.HBox;

import javafx.scene.layout.TilePane;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import static  org.project.Dashboard.categories;

import static org.project.Dashboard.showRestaurant;

public class AdminDashboard {

    TableView<MenuItem> tableView = new TableView<>();

    private static final String MENU\_DATA\_FILE = "MenuData.txt";

    public void showAdminDashboard() {

        Stage adminStage = new Stage();

        adminStage.setTitle("Admin Dashboard - Cafe Shop Management");

        // Define VBox for layout

        VBox vBox1 = new VBox(20);

        VBox vBox2 = new VBox(20);

        VBox vBox3 = new VBox(20);

        vBox1.setMaxWidth(150);

        vBox2.setMaxWidth(500);

        vBox3.setMaxWidth(300);

        // Set styles

        vBox1.getStyleClass().add("vbox1-style");

        vBox2.getStyleClass().add("vbox2-style");

        vBox3.getStyleClass().add("vbox3-style");

        setupVBox1(vBox1);

        setupVBox2(vBox2);

        HBox mainLayout = new HBox(20, vBox1, vBox2, vBox3);

        mainLayout.setAlignment(Pos.CENTER);

        Scene scene = new Scene(mainLayout, 800, 600);

        scene.getStylesheets().add(getClass().getResource("styles.css").toExternalForm());

        adminStage.setScene(scene);

        adminStage.setFullScreen(false);

        adminStage.show();

    }

    // Set up VBox1 for navigation buttons

    private void setupVBox1(VBox vBox1) {

        vBox1.setPadding(new Insets(20));

        vBox1.setAlignment(Pos.TOP\_CENTER);

        Button homeButton = new Button("Home");

        homeButton.setOnAction(event -> {

            // Implement Home functionality (if needed)

        });

        Button signOutButton = new Button("Sign Out");

        signOutButton.setOnAction(event -> {

            // Implement Sign-out functionality to go back to the login form

            // Close the admin dashboard and show the login screen

            Stage stage = (Stage) signOutButton.getScene().getWindow();

            stage.close();  // Close the admin dashboard window

            new Main();  // Implement LoginForm to go back to the login screen

        });

        VBox buttons = new VBox(20, homeButton, signOutButton);

        buttons.setAlignment(Pos.CENTER);

        vBox1.getChildren().add(buttons);

    }

    // Set up VBox2 to display menu items and allow editing

    private void setupVBox2(VBox vBox2) {

        vBox2.setPadding(new Insets(10));

        vBox2.setAlignment(Pos.CENTER);

        // Define the columns for the table

        TableColumn<MenuItem, String> itemNameColumn = new TableColumn<>("Item Name");

        itemNameColumn.setMinWidth(100);

        itemNameColumn.setCellValueFactory(cellData ->

                javafx.beans.binding.Bindings.createStringBinding(() -> cellData.getValue().getName()));

        TableColumn<MenuItem, Double> itemPriceColumn = new TableColumn<>("Price");

        itemPriceColumn.setMinWidth(100);

        itemPriceColumn.setCellValueFactory(cellData ->

                Bindings.createDoubleBinding(() -> cellData.getValue().getPrice()).asObject());

        TableColumn<MenuItem, String> itemImageColumn = new TableColumn<>("Image Path");

        itemImageColumn.setMinWidth(150);

        itemImageColumn.setCellValueFactory(cellData ->

                javafx.beans.binding.Bindings.createStringBinding(() -> cellData.getValue().getImagePath()));

        // Define the action to edit a menu item

        TableColumn<MenuItem, Void> editColumn = new TableColumn<>("Edit");

        editColumn.setMinWidth(100);

        editColumn.setCellFactory(param -> new TableCell<MenuItem, Void>() {

            private final Button editButton = new Button("Edit");

            {

                editButton.setOnAction(event -> {

                    MenuItem menuItem = getTableView().getItems().get(getIndex());

                    showEditDialog(menuItem); // Show edit dialog

                });

            }

            @Override

            protected void updateItem(Void item, boolean empty) {

                super.updateItem(item, empty);

                if (empty) {

                    setGraphic(null);

                } else {

                    setGraphic(editButton);

                }

            }

        });

        // Add the columns to the table

        tableView.getColumns().clear();

        tableView.getColumns().addAll(itemNameColumn, itemPriceColumn, itemImageColumn, editColumn);

        // Load menu items from the file

        List<MenuItem> menuItems = DataLoader.loadMenuItems(MENU\_DATA\_FILE);

        tableView.getItems().setAll(menuItems);

        vBox2.getChildren().clear();

        vBox2.getChildren().add(tableView);

        // Add "Save Changes" button to save changes

        Button saveChangesButton = new Button("Submit Changes");

        saveChangesButton.setOnAction(event -> saveMenuChanges());

        vBox2.getChildren().add(saveChangesButton);

    }

    // Show a dialog to edit a menu item

    private void showEditDialog(MenuItem menuItem) {

        TextInputDialog nameInput = new TextInputDialog(menuItem.getName());

        nameInput.setTitle("Edit Item");

        nameInput.setHeaderText("Edit Item Name");

        nameInput.setContentText("Name:");

        TextInputDialog priceInput = new TextInputDialog(String.valueOf(menuItem.getPrice()));

        priceInput.setTitle("Edit Item");

        priceInput.setHeaderText("Edit Price");

        priceInput.setContentText("Price:");

        TextInputDialog imageInput = new TextInputDialog(menuItem.getImagePath());

        imageInput.setTitle("Edit Item");

        imageInput.setHeaderText("Edit Image Path");

        imageInput.setContentText("Image Path:");

        // Show dialog and wait for the result

        nameInput.showAndWait().ifPresent(newName -> menuItem.setName(newName));

        priceInput.showAndWait().ifPresent(newPrice -> menuItem.setPrice(Double.parseDouble(newPrice)));

        imageInput.showAndWait().ifPresent(newImage -> menuItem.setImagePath(newImage));

    }

    // Save the changes made to the menu items back to the file

    private void saveMenuChanges() {

        try (BufferedWriter writer = new BufferedWriter(new FileWriter(MENU\_DATA\_FILE))) {

            for (MenuItem menuItem : tableView.getItems()) {

                writer.write(menuItem.getName() + "," + menuItem.getPrice() + "," + menuItem.getImagePath() + "\n");

            }

            showInfoDialog("Changes saved successfully!");

        } catch (IOException e) {

            e.printStackTrace();

            showInfoDialog("Failed to save changes!");

        }

    }

    // Show an information dialog to the admin

    private void showInfoDialog(String message) {

        Alert alert = new Alert(Alert.AlertType.INFORMATION);

        alert.setTitle("Info");

        alert.setHeaderText(null);

        alert.setContentText(message);

        alert.showAndWait();

    }

    private void showRestaurants() {

        // Clear the current content

        categories.getChildren().clear();

        showRestaurant.getChildren().clear();

        showRestaurant.setPadding(new Insets(20));

        // Load restaurant data using DataLoader

        List<Restaurant> restaurants = DataLoader.loadRestaurants();

        // Create a FlowPane for restaurants

        FlowPane flowPane = new FlowPane();

        flowPane.setPadding(new Insets(2, 10, 5, 10));

        flowPane.setHgap(25);

        flowPane.setVgap(25);

        flowPane.setPrefWidth(470);

        flowPane.setAlignment(Pos.CENTER);

        for (Restaurant restaurant : restaurants) {

            // Create a VBox for each restaurant

            VBox restaurantBox = new VBox(5);

            restaurantBox.setPrefSize(180, 280);

            restaurantBox.getStyleClass().add("restaurantBox");

            restaurantBox.setAlignment(Pos.CENTER);

            // Add restaurant name and description

            Button nameLabel = new Button(restaurant.getName());

            nameLabel.setPrefSize(150, 30);

            nameLabel.getStyleClass().add("nameLabel");

            // Display restaurant image

            ImageView imageView = new ImageView(restaurant.getImagePath());

            imageView.setFitWidth(230);

            imageView.setFitHeight(80);

            imageView.setPreserveRatio(true);

            // Description of restaurant

            Label descriptionLabel = new Label(restaurant.getDescription());

            descriptionLabel.setWrapText(true);

            descriptionLabel.setStyle("-fx-text-fill: #ffce1b; -fx-font-size: 12; -fx-font-weight: bold;");

            // Action to show menu for the selected restaurant

            nameLabel.setOnMouseClicked(event -> showMenu(restaurant));

            // Add all elements to the restaurant VBox

            restaurantBox.getChildren().addAll(imageView, descriptionLabel, nameLabel);

            flowPane.getChildren().add(restaurantBox);

        }

        // Add the flow pane to the showRestaurant VBox

        showRestaurant.getChildren().add(flowPane);

    }

    private void showMenu(Restaurant restaurant) {

        // Clear the current content in showRestaurant VBox

        showRestaurant.getChildren().clear();

        // Set padding and alignment for showRestaurant

        showRestaurant.setPadding(new Insets(20));

        showRestaurant.setAlignment(Pos.CENTER);

        // Add a back button to navigate back to the restaurant list

        Button backButton = new Button("Back to Restaurants");

        backButton.setStyle("-fx-font-size: 14; -fx-background-color: #ffce1b; -fx-text-fill: black;");

        backButton.setOnAction(event -> showRestaurants());

        // Get the menu items for the selected restaurant

        List<MenuItem> menuItems = restaurant.getMenuItems();

        // Create a TilePane for displaying menu items

        TilePane tilePane = new TilePane();

        tilePane.setPadding(new Insets(5));

        tilePane.setHgap(10);

        tilePane.setVgap(5);

        tilePane.setPrefColumns(3);

        tilePane.setAlignment(Pos.CENTER);

        for (MenuItem menuItem : menuItems) {

            // Create a VBox for each menu item

            VBox itemBox = new VBox(10);

            itemBox.setPadding(new Insets(10));

            itemBox.setAlignment(Pos.CENTER);

            itemBox.setStyle("-fx-border-color: lightgray; -fx-border-width: 1; -fx-padding: 10px;");

            // Menu item name, price, and image path

            String itemName = menuItem.getName();

            double itemPrice = menuItem.getPrice();

            String imagePath = menuItem.getImagePath();

            // Create an ImageView for the menu item

            ImageView imageView = null;

            URL resource = getClass().getResource(imagePath);

            if (resource != null) {

                imageView = new ImageView(resource.toExternalForm());

                imageView.setFitWidth(150);

                imageView.setFitHeight(100);

                imageView.setPreserveRatio(true);

            }

            // Create labels for item name and price

            Label itemNameLabel = new Label(itemName);

            itemNameLabel.setStyle("-fx-font-size: 14; -fx-font-weight: bold; -fx-text-fill: #ffce1b;");

            Label itemPriceLabel = new Label("$" + itemPrice);

            itemPriceLabel.setStyle("-fx-font-size: 12; -fx-text-fill: white;");

            // Create an HBox for the counter buttons

            HBox counterBox = new HBox(10);

            counterBox.setAlignment(Pos.CENTER);

            Button minusButton = new Button("-");

            minusButton.getStyleClass().add("minus-button");

            Label quantityLabel = new Label("0");

            quantityLabel.setStyle("-fx-font-size: 16; -fx-text-fill: #ffce1b;");

            Button plusButton = new Button("+");

            plusButton.getStyleClass().add("plus-button");

            // Add buttons to the counter box

            counterBox.getChildren().addAll(quantityLabel);

            final int[] quantity = {0};

            // Action for minus button

            minusButton.setOnAction(e -> {

                if (quantity[0] > 0) {

                    quantity[0]--;

                    quantityLabel.setText(String.valueOf(quantity[0]));

                }

            });

            // Action for plus button

            plusButton.setOnAction(e -> {

                quantity[0]++;

                quantityLabel.setText(String.valueOf(quantity[0]));

            });

            // Create an "Add to Order" button

            Button addToCartButton = new Button("Add to Order");

            addToCartButton.setStyle("-fx-background-color: #ffce1b; -fx-text-fill: black;");

            addToCartButton.setOnAction(event -> {

                if (quantity[0] > 0) {

                    addItemToTable(itemName, quantity[0], itemPrice \* quantity[0]);

                    quantity[0] = 0; // Reset quantity after adding to cart

                    quantityLabel.setText(String.valueOf(quantity[0]));

                }

            });

            // Create an HBox for the name and price labels

            HBox namePriceBox = new HBox(10);

            namePriceBox.setAlignment(Pos.CENTER);

            namePriceBox.getChildren().addAll(itemNameLabel, itemPriceLabel);

            // Create an HBox for the counter buttons and add-to-cart button

            HBox hBoxCart = new HBox(10);

            hBoxCart.setAlignment(Pos.CENTER);

            hBoxCart.getChildren().addAll(minusButton, addToCartButton, plusButton);

            // Add all elements to the itemBox

            itemBox.getChildren().addAll(namePriceBox, imageView, counterBox, hBoxCart);

            // Add the itemBox to the tilePane

            tilePane.getChildren().add(itemBox);

        }

        // Add the back button and the tilePane to showRestaurant VBox

        showRestaurant.getChildren().addAll(backButton, tilePane);

    }

    private void addItemToTable(String itemName, int quantity, double itemPrice) {

        ObservableList<MenuItem> items = tableView.getItems();

        // Check if the item already exists in the table

//        for (MenuItem item : items) {

//            if (item.getName().equals(itemName)) {

//                item.setQuantity(item.getQuantity() + quantity);

//                item.setPrice(itemPrice \* item.getQuantity());

//                tableView.refresh();

//                return;

//            }

//        }

        // If item doesn't exist, add it to the table as a new item

//        items.add(new Item(itemName, quantity, itemPrice));

    }

}

**Item Class**

package org.project;

public class Item {

    private  String name;

    private int quantity;

    private  double price;

    public Item(String name, int quantity, double price) {

        this.name = name;

        this.quantity = quantity;

        this.price = price;

    }

    public Item() {

    }

    public void setName(String name){

        this.name = name;

    }

    public void setPrice(double  price){

        this.price = price;

    }

    public void setQuantity(int quantity) {

        this.quantity = quantity;

    }

    public String getName() {

        return name;

    }

    public int getQuantity() {

        return quantity;

    }

    public double getPrice() {

        return price;

    }}

**Account Class**

package org.project;

import java.io.Serializable;

public class Account implements Serializable {

    private String username;

    private String password;

    private String email;

    public Account(String username, String password, String email) {

        this.username = username;

        this.password = password;

        this.email = email;

    }

    public String getUsername() {

        return username;

    }

    public String getPassword() {

        return password;

    }

    public void setPassword(String password) {

        this.password = password;

    }

    public String getEmail() {

        return email;

    }

}

**Style. CSS**

.register-title, .login-title , .forgot-password-title {  
 -fx-font-size: 20px;  
 -fx-font-weight: bold;  
 -fx-text-fill: #232323;  
 -fx-font-family: "Times New Roman";  
}  
  
.button-login, .button-register , .button-submit , .button-save ,.button-back{  
 -fx-background-color: #232323;  
 -fx-text-fill: white;  
 -fx-font-size: 14px;  
 -fx-font-weight: bold;  
 -fx-padding: 10px 20px;  
 -fx-border-radius: 5px;  
}  
  
.button-login:hover , .button-submit:hover , .button-save:hover , .button-register:hover ,.button-back:hover {  
 -fx-background-color: #3C3C3C;  
 -fx-text-fill: white;  
}  
  
/\* Hyperlink Styling \*/  
.hyperlink {  
 -fx-font-size: 12px;  
 -fx-text-fill: #232323;  
 -fx-padding: 5px 0;  
}  
  
.hyperlink:hover {  
 -fx-text-fill: #3C3C3C;  
}  
  
.button-create-account, .button-already-account {  
 -fx-background-color: transparent;  
 -fx-text-fill: white;  
 -fx-font-size: 14px;  
 -fx-padding: 10px 20px;  
 -fx-border-radius: 1px;  
 -fx-border-color: white;  
}  
  
.button-create-account:hover, .button-already-account:hover {  
 -fx-background-color: white;  
 -fx-text-fill: black;  
}  
  
.shift-pane {  
 /\*-fx-background-color: #15151e;\*/  
 -fx-background-color: linear-gradient(to bottom ,#3C3C3C , #232323);  
 -fx-padding: 50px;  
}  
  
.title {  
 -fx-font-size: 30px;  
 -fx-font-weight: bold;  
 -fx-text-fill: white;  
}  
  
.food-hive-label ,.orderLabel{  
 -fx-font-family: "Arial Black";  
 -fx-font-size: 20px;  
 -fx-text-fill: #ffce1b;  
 -fx-font-weight: bold;  
 -fx-underline: true;  
}  
  
.top-hbox-style , showRestaurant ,restaurantBox{  
 -fx-background-color: linear-gradient(to bottom ,#3C3C3C , #232323);  
}  
  
  
/\* Button Styles \*/  
.button-style {  
 -fx-pref-width: 150px;  
}  
  
/\* VBox styles \*/  
.vbox-style {  
 -fx-spacing: 20px;  
 -fx-padding: 20px;  
 -fx-alignment: center;  
}  
  
.vbox1-style, .vbox2-style, .vbox3-style{  
 -fx-border-width: 1px;  
 -fx-border-color: white;  
 -fx-background-color: linear-gradient(to bottom ,#3C3C3C , #232323);  
}  
  
/\* TableView Styles \*/  
.table-view {  
 -fx-pref-height: 350px;  
 -fx-max-height: 350px;  
}  
  
/\* Labels in the VBoxes \*/  
.vbox-label {  
 -fx-font-size: 18px;  
 -fx-font-weight: bold;  
}  
  
.restaurants-button, .categories-button, .home-button {  
 -fx-background-color: red;  
 -fx-text-fill: white;  
 -fx-font-size: 12px;  
 -fx-padding: 10px 20px;  
 -fx-border-radius: 5px;  
 -fx-cursor: hand;  
}  
  
/\* Hover effect in JavaFX \*/  
.restaurants-button:hover,  
.categories-button:hover,  
.home-button:hover {  
 -fx-background-color: yellow;  
 -fx-text-fill: #333;  
}  
  
.minus-button , .plus-button{  
 -fx-background-color: transparent;  
 -fx-text-fill: #ffce1b;  
 -fx-font-size: 20px;  
 -fx-padding: 5px 10px;  
 -fx-font-weight: bold;  
}

**Restaurant Class**

package org.project;

import java.util.List;

public class Restaurant {

    private String name;

    private String imagePath;

    private String description;

    private List<MenuItem> menuItems;

    public Restaurant(String name, String imagePath, String description, List<MenuItem> menuItems) {

        this.name = name;

        this.imagePath = imagePath;

        this.description = description;

        this.menuItems = menuItems;

    }

    public Restaurant(String name, String imagePath, String description) {

        this.name = name;

        this.imagePath = imagePath;

        this.description = description;

    }

    public String getName() {

        return name;

    }

    public String getImagePath() {

        return imagePath;

    }

    public String getDescription() {

        return description;

    }

    public void setDescription(String description) {

        this.description = description;

    }

    public void setImagePath(String imagePath) {

        this.imagePath = imagePath;

    }

    public void setMenuItems(List<MenuItem> menuItems) {

        this.menuItems = menuItems;

    }

    public void setName(String name) {

        this.name = name;

    }

    public List<MenuItem> getMenuItems() {

        return menuItems;

    }

}

**DataLoader Class**

package org.project;

import java.io.\*;

import java.util.ArrayList;

import java.util.List;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class DataLoader {

    private static final String IMAGE\_PATH = "/";

    private static final String RESTAURANT\_FILE\_PATH = "RestaurantsData.txt";

    private static final String IMAGE\_PATH2 = "/";

    public static List<MenuItem> loadMenuItems(String filePath) {

        List<MenuItem> menuItems = new ArrayList<>();

        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {

            String line;

            while ((line = reader.readLine()) != null) {

                String[] data = line.split(",");

                if (data.length == 3) {

                    String name = data[0].trim();

                    double price = Double.parseDouble(data[1].trim());

                    String imagePath = IMAGE\_PATH + data[2].trim(); // Concatenate base path

                    menuItems.add(new MenuItem(name, price, imagePath));

                }

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

        return menuItems;

    }

    public static List<Restaurant> loadRestaurants() {

        List<Restaurant> restaurants = new ArrayList<>();

        try (BufferedReader br = new BufferedReader(new FileReader(RESTAURANT\_FILE\_PATH))) {

            String line;

            while ((line = br.readLine()) != null) {

                // Parse the line manually with handling for quoted fields

                String[] parts = parseCSVLine(line);

                if (parts.length < 4) {

                    continue; // Skip malformed lines

                }

                String name = parts[0].trim();

                String imagePath = parts[1].trim();

                String description = parts[2].trim();

                List<MenuItem> menuItems = new ArrayList<>();

                for (int i = 3; i < parts.length; i++) {

                    // Split each menu item by commas

                    String[] itemParts = parts[i].split(",");

                    if (itemParts.length == 3) {

                        String itemName = itemParts[0].trim();

                        double itemPrice = Double.parseDouble(itemParts[1].trim());

                        String itemImage = itemParts[2].trim();

                        menuItems.add(new MenuItem(itemName, itemPrice, itemImage));

                    } else {

                        System.out.println("Invalid menu item format: " + parts[i]);

                    }

                }

                restaurants.add(new Restaurant(name, imagePath, description, menuItems));

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

        return restaurants;

    }

    // Custom method to parse the CSV line considering quoted fields

    private static String[] parseCSVLine(String line) {

        List<String> parts = new ArrayList<>();

        Matcher matcher = Pattern.compile("\"([^\"]\*)\"|([^\",]+)").matcher(line);

        while (matcher.find()) {

            if (matcher.group(1) != null) {

                parts.add(matcher.group(1)); // This is the quoted part

            } else {

                parts.add(matcher.group(2)); // This is the unquoted part

            }

        }

        return parts.toArray(new String[0]);

    }

    public static List<MenuItem> loadCategoryItems(String category) {

        List<MenuItem> menuItems = new ArrayList<>();

        String fileName = category + "\_items.txt"; // Construct the file name based on category

        try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {

            String line;

            while ((line = reader.readLine()) != null) {

                String[] data = line.split(",");

                if (data.length == 3) {

                    String name = data[0].trim();

                    double price = Double.parseDouble(data[1].trim());

                    String imagePath = IMAGE\_PATH2 + data[2].trim(); // Concatenate base path

                    menuItems.add(new MenuItem(name, price, imagePath));

                }

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

        return menuItems;

    }

**AdminDashBoard Class**

package org.project;

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

import java.net.URL;

import java.util.List;

import javafx.beans.binding.Bindings;

import javafx.collections.ObservableList;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.image.ImageView;

import javafx.scene.layout.FlowPane;

import javafx.scene.layout.HBox;

import javafx.scene.layout.TilePane;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import static  org.project.Dashboard.categories;

import static org.project.Dashboard.showRestaurant;

public class AdminDashboard {

    TableView<MenuItem> tableView = new TableView<>();

    private static final String MENU\_DATA\_FILE = "MenuData.txt";

    public void showAdminDashboard() {

        Stage adminStage = new Stage();

        adminStage.setTitle("Admin Dashboard - Cafe Shop Management");

        // Define VBox for layout

        VBox vBox1 = new VBox(20);

        VBox vBox2 = new VBox(20);

        VBox vBox3 = new VBox(20);

        vBox1.setMaxWidth(150);

        vBox2.setMaxWidth(500);

        vBox3.setMaxWidth(300);

        // Set styles

        vBox1.getStyleClass().add("vbox1-style");

        vBox2.getStyleClass().add("vbox2-style");

        vBox3.getStyleClass().add("vbox3-style");

        setupVBox1(vBox1);

        setupVBox2(vBox2);

        HBox mainLayout = new HBox(20, vBox1, vBox2, vBox3);

        mainLayout.setAlignment(Pos.CENTER);

        Scene scene = new Scene(mainLayout, 800, 600);

        scene.getStylesheets().add(getClass().getResource("styles.css").toExternalForm());

        adminStage.setScene(scene);

        adminStage.setFullScreen(false);

        adminStage.show();

    }

    // Set up VBox1 for navigation buttons

    private void setupVBox1(VBox vBox1) {

        vBox1.setPadding(new Insets(20));

        vBox1.setAlignment(Pos.TOP\_CENTER);

        Button homeButton = new Button("Home");

        homeButton.setOnAction(event -> {

            // Implement Home functionality (if needed)

        });

        Button signOutButton = new Button("Sign Out");

        signOutButton.setOnAction(event -> {

            // Implement Sign-out functionality to go back to the login form

            // Close the admin dashboard and show the login screen

            Stage stage = (Stage) signOutButton.getScene().getWindow();

            stage.close();  // Close the admin dashboard window

            new Main();  // Implement LoginForm to go back to the login screen

        });

        VBox buttons = new VBox(20, homeButton, signOutButton);

        buttons.setAlignment(Pos.CENTER);

        vBox1.getChildren().add(buttons);

    }

    // Set up VBox2 to display menu items and allow editing

    private void setupVBox2(VBox vBox2) {

        vBox2.setPadding(new Insets(10));

        vBox2.setAlignment(Pos.CENTER);

        // Define the columns for the table

        TableColumn<MenuItem, String> itemNameColumn = new TableColumn<>("Item Name");

        itemNameColumn.setMinWidth(100);

        itemNameColumn.setCellValueFactory(cellData ->

                javafx.beans.binding.Bindings.createStringBinding(() -> cellData.getValue().getName()));

        TableColumn<MenuItem, Double> itemPriceColumn = new TableColumn<>("Price");

        itemPriceColumn.setMinWidth(100);

        itemPriceColumn.setCellValueFactory(cellData ->

                Bindings.createDoubleBinding(() -> cellData.getValue().getPrice()).asObject());

        TableColumn<MenuItem, String> itemImageColumn = new TableColumn<>("Image Path");

        itemImageColumn.setMinWidth(150);

        itemImageColumn.setCellValueFactory(cellData ->

                javafx.beans.binding.Bindings.createStringBinding(() -> cellData.getValue().getImagePath()));

        // Define the action to edit a menu item

        TableColumn<MenuItem, Void> editColumn = new TableColumn<>("Edit");

        editColumn.setMinWidth(100);

        editColumn.setCellFactory(param -> new TableCell<MenuItem, Void>() {

            private final Button editButton = new Button("Edit");

            {

                editButton.setOnAction(event -> {

                    MenuItem menuItem = getTableView().getItems().get(getIndex());

                    showEditDialog(menuItem); // Show edit dialog

                });

            }

            @Override

            protected void updateItem(Void item, boolean empty) {

                super.updateItem(item, empty);

                if (empty) {

                    setGraphic(null);

                } else {

                    setGraphic(editButton);

                }

            }

        });

        // Add the columns to the table

        tableView.getColumns().clear();

        tableView.getColumns().addAll(itemNameColumn, itemPriceColumn, itemImageColumn, editColumn);

        // Load menu items from the file

        List<MenuItem> menuItems = DataLoader.loadMenuItems(MENU\_DATA\_FILE);

        tableView.getItems().setAll(menuItems);

        vBox2.getChildren().clear();

        vBox2.getChildren().add(tableView);

        // Add "Save Changes" button to save changes

        Button saveChangesButton = new Button("Submit Changes");

        saveChangesButton.setOnAction(event -> saveMenuChanges());

        vBox2.getChildren().add(saveChangesButton);

    }

    // Show a dialog to edit a menu item

    private void showEditDialog(MenuItem menuItem) {

        TextInputDialog nameInput = new TextInputDialog(menuItem.getName());

        nameInput.setTitle("Edit Item");

        nameInput.setHeaderText("Edit Item Name");

        nameInput.setContentText("Name:");

        TextInputDialog priceInput = new TextInputDialog(String.valueOf(menuItem.getPrice()));

        priceInput.setTitle("Edit Item");

        priceInput.setHeaderText("Edit Price");

        priceInput.setContentText("Price:");

        TextInputDialog imageInput = new TextInputDialog(menuItem.getImagePath());

        imageInput.setTitle("Edit Item");

        imageInput.setHeaderText("Edit Image Path");

        imageInput.setContentText("Image Path:");

        // Show dialog and wait for the result

        nameInput.showAndWait().ifPresent(newName -> menuItem.setName(newName));

        priceInput.showAndWait().ifPresent(newPrice -> menuItem.setPrice(Double.parseDouble(newPrice)));

        imageInput.showAndWait().ifPresent(newImage -> menuItem.setImagePath(newImage));

    }

    // Save the changes made to the menu items back to the file

    private void saveMenuChanges() {

        try (BufferedWriter writer = new BufferedWriter(new FileWriter(MENU\_DATA\_FILE))) {

            for (MenuItem menuItem : tableView.getItems()) {

                writer.write(menuItem.getName() + "," + menuItem.getPrice() + "," + menuItem.getImagePath() + "\n");

            }

            showInfoDialog("Changes saved successfully!");

        } catch (IOException e) {

            e.printStackTrace();

            showInfoDialog("Failed to save changes!");

        }

    }

    // Show an information dialog to the admin

    private void showInfoDialog(String message) {

        Alert alert = new Alert(Alert.AlertType.INFORMATION);

        alert.setTitle("Info");

        alert.setHeaderText(null);

        alert.setContentText(message);

        alert.showAndWait();

    }

    private void showRestaurants() {

        // Clear the current content

        categories.getChildren().clear();

        showRestaurant.getChildren().clear();

        showRestaurant.setPadding(new Insets(20));

        // Load restaurant data using DataLoader

        List<Restaurant> restaurants = DataLoader.loadRestaurants();

        // Create a FlowPane for restaurants

        FlowPane flowPane = new FlowPane();

        flowPane.setPadding(new Insets(2, 10, 5, 10));

        flowPane.setHgap(25);

        flowPane.setVgap(25);

        flowPane.setPrefWidth(470);

        flowPane.setAlignment(Pos.CENTER);

        for (Restaurant restaurant : restaurants) {

            // Create a VBox for each restaurant

            VBox restaurantBox = new VBox(5);

            restaurantBox.setPrefSize(180, 280);

            restaurantBox.getStyleClass().add("restaurantBox");

            restaurantBox.setAlignment(Pos.CENTER);

            // Add restaurant name and description

            Button nameLabel = new Button(restaurant.getName());

            nameLabel.setPrefSize(150, 30);

            nameLabel.getStyleClass().add("nameLabel");

            // Display restaurant image

            ImageView imageView = new ImageView(restaurant.getImagePath());

            imageView.setFitWidth(230);

            imageView.setFitHeight(80);

            imageView.setPreserveRatio(true);

            // Description of restaurant

            Label descriptionLabel = new Label(restaurant.getDescription());

            descriptionLabel.setWrapText(true);

            descriptionLabel.setStyle("-fx-text-fill: #ffce1b; -fx-font-size: 12; -fx-font-weight: bold;");

            // Action to show menu for the selected restaurant

            nameLabel.setOnMouseClicked(event -> showMenu(restaurant));

            // Add all elements to the restaurant VBox

            restaurantBox.getChildren().addAll(imageView, descriptionLabel, nameLabel);

            flowPane.getChildren().add(restaurantBox);

        }

        // Add the flow pane to the showRestaurant VBox

        showRestaurant.getChildren().add(flowPane);

    }

    private void showMenu(Restaurant restaurant) {

        // Clear the current content in showRestaurant VBox

        showRestaurant.getChildren().clear();

        // Set padding and alignment for showRestaurant

        showRestaurant.setPadding(new Insets(20));

        showRestaurant.setAlignment(Pos.CENTER);

        // Add a back button to navigate back to the restaurant list

        Button backButton = new Button("Back to Restaurants");

        backButton.setStyle("-fx-font-size: 14; -fx-background-color: #ffce1b; -fx-text-fill: black;");

        backButton.setOnAction(event -> showRestaurants());

        // Get the menu items for the selected restaurant

        List<MenuItem> menuItems = restaurant.getMenuItems();

        // Create a TilePane for displaying menu items

        TilePane tilePane = new TilePane();

        tilePane.setPadding(new Insets(5));

        tilePane.setHgap(10);

        tilePane.setVgap(5);

        tilePane.setPrefColumns(3);

        tilePane.setAlignment(Pos.CENTER);

        for (MenuItem menuItem : menuItems) {

            // Create a VBox for each menu item

            VBox itemBox = new VBox(10);

            itemBox.setPadding(new Insets(10));

            itemBox.setAlignment(Pos.CENTER);

            itemBox.setStyle("-fx-border-color: lightgray; -fx-border-width: 1; -fx-padding: 10px;");

            // Menu item name, price, and image path

            String itemName = menuItem.getName();

            double itemPrice = menuItem.getPrice();

            String imagePath = menuItem.getImagePath();

            // Create an ImageView for the menu item

            ImageView imageView = null;

            URL resource = getClass().getResource(imagePath);

            if (resource != null) {

                imageView = new ImageView(resource.toExternalForm());

                imageView.setFitWidth(150);

                imageView.setFitHeight(100);

                imageView.setPreserveRatio(true);

            }

            // Create labels for item name and price

            Label itemNameLabel = new Label(itemName);

            itemNameLabel.setStyle("-fx-font-size: 14; -fx-font-weight: bold; -fx-text-fill: #ffce1b;");

            Label itemPriceLabel = new Label("$" + itemPrice);

            itemPriceLabel.setStyle("-fx-font-size: 12; -fx-text-fill: white;");

            // Create an HBox for the counter buttons

            HBox counterBox = new HBox(10);

            counterBox.setAlignment(Pos.CENTER);

            Button minusButton = new Button("-");

            minusButton.getStyleClass().add("minus-button");

            Label quantityLabel = new Label("0");

            quantityLabel.setStyle("-fx-font-size: 16; -fx-text-fill: #ffce1b;");

            Button plusButton = new Button("+");

            plusButton.getStyleClass().add("plus-button");

            // Add buttons to the counter box

            counterBox.getChildren().addAll(quantityLabel);

            final int[] quantity = {0};

            // Action for minus button

            minusButton.setOnAction(e -> {

                if (quantity[0] > 0) {

                    quantity[0]--;

                    quantityLabel.setText(String.valueOf(quantity[0]));

                }

            });

            // Action for plus button

            plusButton.setOnAction(e -> {

                quantity[0]++;

                quantityLabel.setText(String.valueOf(quantity[0]));

            });

            // Create an "Add to Order" button

            Button addToCartButton = new Button("Add to Order");

            addToCartButton.setStyle("-fx-background-color: #ffce1b; -fx-text-fill: black;");

            addToCartButton.setOnAction(event -> {

                if (quantity[0] > 0) {

                    addItemToTable(itemName, quantity[0], itemPrice \* quantity[0]);

                    quantity[0] = 0; // Reset quantity after adding to cart

                    quantityLabel.setText(String.valueOf(quantity[0]));

                }

            });

            // Create an HBox for the name and price labels

            HBox namePriceBox = new HBox(10);

            namePriceBox.setAlignment(Pos.CENTER);

            namePriceBox.getChildren().addAll(itemNameLabel, itemPriceLabel);

            // Create an HBox for the counter buttons and add-to-cart button

            HBox hBoxCart = new HBox(10);

            hBoxCart.setAlignment(Pos.CENTER);

            hBoxCart.getChildren().addAll(minusButton, addToCartButton, plusButton);

            // Add all elements to the itemBox

            itemBox.getChildren().addAll(namePriceBox, imageView, counterBox, hBoxCart);

            // Add the itemBox to the tilePane

            tilePane.getChildren().add(itemBox);

        }

        // Add the back button and the tilePane to showRestaurant VBox

        showRestaurant.getChildren().addAll(backButton, tilePane);

    }

    private void addItemToTable(String itemName, int quantity, double itemPrice) {

        ObservableList<MenuItem> items = tableView.getItems();

        // Check if the item already exists in the table

//        for (MenuItem item : items) {

//            if (item.getName().equals(itemName)) {

//                item.setQuantity(item.getQuantity() + quantity);

//                item.setPrice(itemPrice \* item.getQuantity());

//                tableView.refresh();

//                return;

//            }

//        }

        // If item doesn't exist, add it to the table as a new item

//        items.add(new Item(itemName, quantity, itemPrice));

    }

}

}

**MenuItem Class**

package org.project;

public class MenuItem {

    private String name;

    private double price;

    private String imagePath;

    public MenuItem(String name, double price, String imagePath) {

        this.name = name;

        this.price = price;

        this.imagePath = imagePath;

    }

    public String getName() {

        return name;

    }

    public double getPrice() {

        return price;

    }

    public String getImagePath() {

        return imagePath;

    }

    public void setImagePath(String imagePath) {

        this.imagePath = imagePath;

    }

    public void setName(String name) {

        this.name = name;

    }

    public void setPrice(double price) {

        this.price = price;

    }

}