

**Group Members: Fatima Khalid (SP24-BSE-132)**

**Aleesha Batool (SP24-BSE-131)**

**Sara Riaz (SP24-BSE-110)**

**Course name: Object Oriented Programming**

**Submitted to: Muhammad Shahid Bhatti**

**Lab Assignment: 05**

**Due Date: 12-12-2024**

package com.example.hspsm;  
  
import javafx.animation.KeyFrame;  
import javafx.animation.Timeline;  
import javafx.application.Application;  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import javafx.collections.transformation.FilteredList;  
import javafx.concurrent.Task;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.print.PrinterJob;  
import javafx.scene.Scene;  
  
import javafx.scene.chart.\*;  
import javafx.scene.control.\*;  
import javafx.scene.control.Button;  
import javafx.scene.control.Label;  
import javafx.scene.control.TextArea;  
import javafx.scene.control.TextField;  
import javafx.scene.control.cell.PropertyValueFactory;  
import javafx.scene.image.Image;  
import javafx.scene.image.ImageView;  
  
import javafx.scene.layout.\*;  
import javafx.scene.paint.Color;  
import javafx.scene.text.Font;  
import javafx.scene.text.FontWeight;  
import javafx.scene.text.Text;  
import javafx.scene.text.TextAlignment;  
  
import javafx.stage.Stage;  
import javafx.util.Duration;  
  
  
import javax.print.Doc;  
import java.time.LocalDate;  
import java.util.\*;  
  
import java.io.\*;  
import java.util.List;  
  
public class HelloApplication extends Application{  
 public static int *userCount* = 1;  
 @Override  
 public void start(Stage stage) throws IOException {  
 welcomeScreen(stage);  
 stage.show();  
 }  
  
 public void welcomeScreen(Stage stage) {  
 VBox vBox = new VBox();  
 vBox.setAlignment(Pos.*CENTER*);  
 vBox.setSpacing(30);  
 vBox.setPadding(new Insets(20));  
 vBox.setBackground(background());  
  
 Text welcome = new Text("Welcome to Housing Society Plot Management System");  
 welcome.setTextAlignment(TextAlignment.*CENTER*);  
 welcome.setFill(Color.*BLACK*);  
 welcome.setStyle("-fx-font-size: 36px; -fx-font-weight: bold; -fx-font-family: Times New Roman;");  
  
 Label loadingLabel = new Label("Loading....");  
 loadingLabel.setTextFill(Color.*BLACK*);  
 loadingLabel.setFont(new Font("Times New Roman", 36));  
  
 ProgressBar = new ProgressBar(0);  
 progressBar.setPrefWidth(800);  
 progressBar.setStyle("-fx-background-color: #e0e0e0; -fx-accent: #4caf50; -fx-border-color: #2e7d32; -fx-border-radius: 5;");  
  
 vBox.getChildren().addAll(welcome, loadingLabel, progressBar);  
  
 Task<Void> loadingTask = new Task<>() {  
 @Override  
 protected Void call() throws Exception {  
 for(int i=0;i<=100;i++){  
 Thread.*sleep*(10);  
 updateProgress(i, 100);  
 }  
 return null;  
 }  
 };  
  
 progressBar.progressProperty().bind(loadingTask.progressProperty());  
 loadingTask.setOnSucceeded(e-> loginScreen(stage));  
 loadingTask.setOnFailed(e -> {  
 Throwable exception = loadingTask.getException();  
 System.*err*.println("Loading error: " + exception.getMessage());  
 });  
 Scene = new Scene(vBox, 1300, 800);  
 stage.setScene(scene);  
 stage.setTitle("Welcome Screen");  
  
 new Thread(loadingTask).start();  
 }  
 public void loginScreen(Stage stage) {  
 VBox vBox= new VBox();  
 vBox.setAlignment(Pos.*CENTER*);  
 vBox.setSpacing(20);  
 vBox.setPadding(new Insets(20));  
 vBox.setBackground(background());  
 Label loginLabel = new Label ("Login");  
 loginLabel.setFont(new Font("Times New Roman", 60));  
 loginLabel.setTextFill(Color.*BLACK*);  
 loginLabel.setStyle("-fx-font-weight: bold;");  
 loginLabel.setAlignment(Pos.*CENTER*);  
 loginLabel.setPadding(new Insets(10, 20, 50, 150));  
  
 Label signupLabel = new Label("New Buyer? Register Here");  
 signupLabel.setStyle("-fx-text-fill: black; -fx-font-size: 18px; -fx-font-family: Arial; -fx-font-weight: bold;");  
 signupLabel.setOnMouseClicked(e->registerUser(stage));  
 Label usernameLabel = new Label("Username:");  
 Label passwordLabel = new Label("Password:");  
 usernameLabel.setStyle("-fx-text-fill: black; -fx-font-size: 18px; -fx-font-family: Arial; -fx-font-weight: bold;");  
 passwordLabel.setStyle("-fx-text-fill: black; -fx-font-size: 18px; -fx-font-family: Arial; -fx-font-weight: bold;");  
  
  
 TextField usernameField = new TextField();  
 usernameField.setPromptText("Enter your username");  
 usernameField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 usernameField.setPrefWidth(500);  
  
 PasswordField passwordField = new PasswordField();  
 passwordField.setPromptText("Enter your password");  
 passwordField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 passwordField.setPrefWidth(500);  
  
 Button login = new Button("Login");  
 Button admin = new Button("Login as Admin");  
  
 String buttonStyle = "-fx-background-color: #3498db; -fx-text-fill: white; -fx-padding: 12px 30px; -fx-font-size: 16px; -fx-font-family: Arial; -fx-border-radius: 5px;";  
 usernameField.setOnKeyPressed(e->{  
 if(e.getCode().toString().equals("ENTER"))  
 passwordField.requestFocus();  
 });  
  
 passwordField.setOnKeyPressed(e->{  
 if(e.getCode().toString().equals("ENTER"))  
 login.fire();  
 });  
 login.setStyle(buttonStyle);  
 admin.setStyle(buttonStyle);  
 signupLabel.setOnMouseEntered(e->signupLabel.setStyle("-fx-text-fill: blue; -fx-font-size: 18px; -fx-font-family: Arial; -fx-font-weight: bold;"));  
 signupLabel.setOnMouseExited(e->signupLabel.setStyle("-fx-text-fill: black; -fx-font-size: 18px; -fx-font-family: Arial; -fx-font-weight: bold;"));  
 login.setOnMouseEntered(e -> login.setStyle("-fx-background-color: #2980b9; -fx-text-fill: white; -fx-padding: 12px 30px; -fx-font-size: 16px; -fx-font-family: Arial; -fx-border-radius: 5px;"));  
 login.setOnMouseExited(e -> login.setStyle(buttonStyle));  
  
 admin.setOnMouseEntered(e -> admin.setStyle("-fx-background-color: #2980b9; -fx-text-fill: white; -fx-padding: 12px 30px; -fx-font-size: 16px; -fx-font-family: Arial; -fx-border-radius: 5px;"));  
 admin.setOnMouseExited(e -> admin.setStyle(buttonStyle));  
  
 Text invalidMessage = new Text();  
 invalidMessage.setTextAlignment(TextAlignment.*CENTER*);  
 invalidMessage.setFill(Color.*RED*);  
 invalidMessage.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 14));  
 List<User> users = *loadUsers*();  
 login.setOnAction(e -> {  
 boolean isValidUser = false;  
 int buyerId = -1;  
 for(User user: users){  
 if(user.getUsername().equals(usernameField.getText()) &&  
 user.getPassword().equals(passwordField.getText())){  
 isValidUser = true;  
 List<Buyer> buyers = *loadBuyers*();  
 for(Buyer buyer: buyers){  
 if(buyer.getUsername().equals(usernameField.getText()) &&  
 buyer.getPassword().equals(passwordField.getText())){  
 buyerId=buyer.getBuyerId();  
 break;  
 }  
 }  
  
 break;  
 }  
  
 }  
 if (isValidUser) {  
 buyerDashboard(stage, buyerId);  
 } else {  
 invalidMessage.setText("Invalid Username or Password");  
 // Clear the error message after 3 seconds  
 Timeline timeline = new Timeline(new KeyFrame(  
 Duration.*seconds*(3),  
 evt -> invalidMessage.setText("")  
 ));  
 timeline.setCycleCount(1);  
 timeline.play();  
 }  
  
 });  
  
 admin.setOnAction(e -> adminLoginScreen(stage));  
  
 GridPane inputGrid = new GridPane();  
 inputGrid.setAlignment(Pos.*CENTER*);  
 inputGrid.setHgap(10);  
 inputGrid.setVgap(30);  
 inputGrid.add(loginLabel, 0, 0);  
 inputGrid.add(usernameLabel, 0, 1);  
 inputGrid.add(usernameField, 0, 2);  
 inputGrid.add(passwordLabel, 0, 3);  
 inputGrid.add(passwordField, 0, 4);  
  
  
 vBox.getChildren().addAll(inputGrid, invalidMessage, login, admin, signupLabel);  
  
 Scene scene = new Scene(vBox, 1300, 800);  
 stage.setScene(scene);  
 stage.setTitle("Login Screen");  
 }  
 public void adminLoginScreen(Stage stage) {  
 VBox vBox = new VBox();  
 vBox.setAlignment(Pos.*CENTER*);  
 vBox.setSpacing(20);  
 vBox.setPadding(new Insets(20));  
 vBox.setBackground(background());  
 Label loginLabel = new Label ("Login");  
 loginLabel.setFont(new Font("Times New Roman", 60));  
 loginLabel.setTextFill(Color.*BLACK*);  
 loginLabel.setStyle("-fx-font-weight: bold;");  
 loginLabel.setAlignment(Pos.*CENTER*);  
 loginLabel.setPadding(new Insets(10, 20, 50, 150));  
  
 Label usernameLabel = new Label("Username:");  
 Label passwordLabel = new Label("Password:");  
 usernameLabel.setStyle("-fx-text-fill: black; -fx-font-size: 18px; -fx-font-family: Arial; -fx-font-weight: bold;");  
 passwordLabel.setStyle("-fx-text-fill: black; -fx-font-size: 18px; -fx-font-family: Arial; -fx-font-weight: bold;");  
  
 TextField usernameField = new TextField();  
 usernameField.setPromptText("Enter admin username");  
 usernameField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 usernameField.setPrefWidth(500);  
  
 PasswordField passwordField = new PasswordField();  
 passwordField.setPromptText("Enter admin password");  
 passwordField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 passwordField.setPrefWidth(500);  
  
  
 Button login = new Button("Login");  
  
 usernameField.setOnKeyPressed(e->{  
 if(e.getCode().toString().equals("ENTER"))  
 passwordField.requestFocus();  
 });  
 passwordField.setOnKeyPressed(e->{  
 if(e.getCode().toString().equals("ENTER"))  
 login.fire();  
 });  
  
 String buttonStyle = "-fx-background-color: #3498db; -fx-text-fill: white; -fx-padding: 12px 30px; -fx-font-size: 16px; -fx-font-family: Arial; -fx-border-radius: 5px;";  
  
 login.setStyle(buttonStyle);  
 login.setOnMouseEntered(e -> login.setStyle("-fx-background-color: #2980b9; -fx-text-fill: white; -fx-padding: 12px 30px; -fx-font-size: 16px; -fx-font-family: Arial; -fx-border-radius: 5px;"));  
 login.setOnMouseExited(e -> login.setStyle(buttonStyle));  
  
 Text invalidMessage = new Text();  
 invalidMessage.setTextAlignment(TextAlignment.*CENTER*);  
 invalidMessage.setFill(Color.*RED*);  
 invalidMessage.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 14));  
  
 login.setOnAction(e -> {  
 boolean isAdmin = "Admin".equals(usernameField.getText()) && "admin".equals(passwordField.getText());  
 if (isAdmin) {  
 adminDashboardScene(stage);  
 } else {  
 invalidMessage.setText("Invalid Username or Password");  
 // Clear the error message after 3 seconds  
 Timeline timeline = new Timeline(new KeyFrame(  
 Duration.*seconds*(3),  
 evt -> invalidMessage.setText("")  
 ));  
 timeline.setCycleCount(1);  
 timeline.play();  
 }  
 });  
  
 GridPane inputGrid = new GridPane();  
 inputGrid.setAlignment(Pos.*CENTER*);  
 inputGrid.setHgap(10);  
 inputGrid.setVgap(30);  
 inputGrid.add(loginLabel, 0, 0);  
 inputGrid.add(usernameLabel, 0, 1);  
 inputGrid.add(usernameField, 0, 2);  
 inputGrid.add(passwordLabel, 0, 3);  
 inputGrid.add(passwordField, 0, 4);  
  
  
 vBox.getChildren().addAll(inputGrid, invalidMessage, login);  
  
 Scene scene = new Scene(vBox, 1300, 800);  
 stage.setScene(scene);  
 stage.setTitle("Admin Login");  
 }  
 public void registerUser(Stage stage) {  
 VBox vBox = new VBox();  
 vBox.setAlignment(Pos.*CENTER*);  
 vBox.setSpacing(15);  
 vBox.setPadding(new Insets(10));  
 vBox.setBackground(background());  
  
 Label signupLabel = new Label("Sign Up");  
 signupLabel.setFont(Font.*font*("Times New Roman", FontWeight.*BOLD*, 60));  
 signupLabel.setTextFill(Color.*BLACK*);  
 signupLabel.setAlignment(Pos.*CENTER*);  
 signupLabel.setPadding(new Insets(10, 20, 50, 30));  
  
 GridPane inputGrid = new GridPane();  
 inputGrid.setAlignment(Pos.*CENTER*);  
 inputGrid.setHgap(10);  
 inputGrid.setVgap(15);  
 inputGrid.setPadding(new Insets(20, 30, 20, 20));  
 Label username = new Label("Username:");  
 Label password = new Label("Password:");  
 Label email = new Label("Email:");  
 Label phoneNumber = new Label("Phone Number:");  
 Label preferredLocation = new Label("Preferred Location:");  
 Label preferredSize = new Label("Preferred Size:");  
 Label budget = new Label("Budget:");  
  
 username.setStyle("-fx-text-fill: black; -fx-font-size: 20px; -fx-font-family: Arial; -fx-font-weight: bold;");  
 password.setStyle("-fx-text-fill: black; -fx-font-size: 20px; -fx-font-family: Arial; -fx-font-weight: bold;");  
 email.setStyle("-fx-text-fill:black; -fx-font-size: 20px; -fx-font-family: Arial; -fx-font-weight: bold;");  
 phoneNumber.setStyle("-fx-text-fill: black; -fx-font-size: 20px; -fx-font-family: Arial; -fx-font-weight: bold;");  
 preferredLocation.setStyle("-fx-text-fill:black; -fx-font-size: 20px; -fx-font-family: Arial; -fx-font-weight: bold;");  
 preferredSize.setStyle("-fx-text-fill: black; -fx-font-size: 20px; -fx-font-family: Arial; -fx-font-weight: bold;");  
 budget.setStyle("-fx-text-fill: black; -fx-font-size: 20px; -fx-font-family: Arial; -fx-font-weight: bold;");  
  
 TextField usernameField = new TextField();  
 usernameField.setPromptText("Enter your username");  
 usernameField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 usernameField.setPrefWidth(800);  
 PasswordField passwordField = new PasswordField();  
 passwordField.setPromptText("Enter password");  
 passwordField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 passwordField.setPrefWidth(800);  
 TextField emailField = new TextField();  
 emailField.setPromptText("Enter your email");  
 emailField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 emailField.setPrefWidth(800);  
 TextField phoneNumberField = new TextField();  
 phoneNumberField.setPromptText("Enter your phone number");  
 phoneNumberField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 phoneNumberField.setPrefWidth(800);  
 TextField preferredLocationField = new TextField();  
 preferredLocationField.setPromptText("Enter your Preferred Plot Location");  
 preferredLocationField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 preferredLocationField.setPrefWidth(800);  
 TextField preferredSizeField = new TextField();  
 preferredSizeField.setPromptText("Enter your Preferred Plot Size (in Marla)");  
 preferredSizeField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 preferredSizeField.setPrefWidth(800);  
 TextField budgetField = new TextField();  
 budgetField.setPromptText("Enter your Budget");  
 budgetField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 budgetField.setPrefWidth(800);  
  
 Button register = new Button("Register");  
 register.setStyle("-fx-background-color: #1abc9c; -fx-text-fill: black; -fx-padding: 10 20; -fx-font-size: 16px; -fx-font-family: Arial;");  
  
 Button cancel = new Button("Cancel");  
 cancel.setStyle("-fx-background-color: #1abc9c; -fx-text-fill: black; -fx-padding: 10 20; -fx-font-size: 16px; -fx-font-family: Arial;");  
  
 HBox buttonBox = new HBox(cancel, register);  
  
 inputGrid.add(username, 0, 0);  
 inputGrid.add(usernameField, 1,0);  
 inputGrid.add(password, 0, 1);  
 inputGrid.add(passwordField,1,1);  
 inputGrid.add(email,0, 2);  
 inputGrid.add(emailField, 1, 2);  
 inputGrid.add(phoneNumber,0,3);  
 inputGrid.add(phoneNumberField,1,3);  
 inputGrid.add(preferredLocation,0, 4);  
 inputGrid.add(preferredLocationField,1, 4);  
 inputGrid.add(preferredSize,0,5);  
 inputGrid.add(preferredSizeField,1,5);  
 inputGrid.add(budget,0,6);  
 inputGrid.add(budgetField,1,6);  
  
 usernameField.setOnKeyPressed(e->{  
 if(e.getCode().toString().equals("ENTER"))  
 passwordField.requestFocus();  
 });  
 passwordField.setOnKeyPressed(e->{  
 if(e.getCode().toString().equals("ENTER"))  
 emailField.requestFocus();  
 });  
 emailField.setOnKeyPressed(e->{  
 if(e.getCode().toString().equals("ENTER"))  
 phoneNumberField.requestFocus();  
 });  
 phoneNumberField.setOnKeyPressed(e->{  
 if(e.getCode().toString().equals("ENTER"))  
 preferredLocationField.requestFocus();  
 });  
  
 preferredLocationField.setOnKeyPressed(e->{  
 if(e.getCode().toString().equals("ENTER"))  
 preferredSizeField.requestFocus();  
 });  
 preferredSizeField.setOnKeyPressed(e->{  
 if(e.getCode().toString().equals("ENTER"))  
 budgetField.requestFocus();  
 });  
 budgetField.setOnKeyPressed(e->{  
 if(e.getCode().toString().equals("ENTER"))  
 register.fire();  
 });  
  
 buttonBox.setSpacing(15);  
 buttonBox.setAlignment(Pos.*CENTER*);  
 register.setOnAction(e -> {  
 // Input Validation  
 if (usernameField.getText().isEmpty() ||  
 passwordField.getText().isEmpty() ||  
 emailField.getText().isEmpty() ||  
 phoneNumberField.getText().isEmpty() ||  
 preferredLocationField.getText().isEmpty() ||  
 preferredSizeField.getText().isEmpty() ||  
 budgetField.getText().isEmpty()) {  
 showError("All fields are required.", vBox);  
 return;  
 }  
  
 if (!emailField.getText().matches("^[\\w.\_%+-]+@[\\w.-]+\\.[a-zA-Z]{2,6}$")) {  
 showError("Invalid email format.", vBox);  
 return;  
 }  
 try {  
 double preferredPlotSize = Double.*parseDouble*(preferredSizeField.getText());  
 double userBudget = Double.*parseDouble*(budgetField.getText());  
 if (preferredPlotSize <= 0 || userBudget <= 0) {  
 showError("Size and budget must be positive numbers.", vBox);  
 return;  
 }  
 } catch (NumberFormatException ex) {  
 showError("Size and budget must be valid numbers.", vBox);  
 return;  
 }  
 List<User> users = *loadUsers*();  
 List<Buyer> buyers = *loadBuyers*();  
 Buyer buyer = new Buyer(usernameField.getText(), passwordField.getText(), emailField.getText(), phoneNumberField.getText(), preferredLocationField.getText(), Double.*parseDouble*(preferredSizeField.getText()), Double.*parseDouble*(budgetField.getText()));  
 buyers.add(buyer);  
 users.add(buyer);  
 *saveBuyers*(FXCollections.*observableArrayList*(buyers));  
 *saveUsers*(FXCollections.*observableArrayList*(users));  
 int buyerId = buyer.getBuyerId();  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*);  
 alert.setTitle("Registration Successful");  
 alert.setHeaderText(null);  
 alert.setContentText("You have been successfully registered!");  
 alert.showAndWait();  
 buyerDashboard(stage, buyerId);  
 });  
  
 cancel.setOnAction(e->{  
 loginScreen(stage);  
 });  
  
 vBox.getChildren().addAll(signupLabel,inputGrid, buttonBox);  
  
 Scene scene = new Scene(vBox, 1300, 800);  
 stage.setScene(scene);  
 stage.setTitle("Register User");  
 }  
 private void showError(String message, VBox container) {  
 Label errorLabel = new Label(message);  
 errorLabel.setStyle("-fx-text-fill: red; -fx-font-size: 16px; -fx-font-family: Arial;");  
 if (!container.getChildren().contains(errorLabel)) {  
 container.getChildren().add(errorLabel);  
 // Remove error message after 3 seconds  
 Timeline timeline = new Timeline(new KeyFrame(  
 Duration.*seconds*(3),  
 evt -> container.getChildren().remove(errorLabel)  
 ));  
 timeline.setCycleCount(1);  
 timeline.play();  
 }  
 }  
  
 public void adminDashboardScene(Stage stage) {  
  
 VBox vBox = new VBox();  
 vBox.setAlignment(Pos.*CENTER*);  
 vBox.setSpacing(15);  
 vBox.setPadding(new Insets(20));  
 vBox.setBackground(background());  
  
 Label titleLabel = new Label("Admin Dashboard");  
 titleLabel.setFont(new Font("Times New Roman", 60));  
 titleLabel.setPadding(new Insets(20, 30,30, 30));  
 titleLabel.setStyle("-fx-font-weight: bold; -fx-text-fill: black;");  
  
 GridPane functionsGridPane = new GridPane();  
  
 Button manageUsersButton = new Button();  
 Button managePlotsButton = new Button();  
 Button managePaymentsButton = new Button();  
 Button generateReportsButton = new Button();  
  
 VBox userBox = new VBox();  
 VBox plotsBox = new VBox();  
 VBox reportBox = new VBox();  
 VBox paymentBox = new VBox();  
  
 userBox.setSpacing(10);  
 userBox.setAlignment(Pos.*CENTER*);  
 Image userLogo = new Image(this.getClass().getResource("/usersLogo.png").toExternalForm());  
 ImageView userImage = new ImageView(userLogo);  
 Label userText = new Label("Manage Users");  
 userText.setFont(new Font("Times New Roman", 18));  
 userBox.getChildren().addAll(userImage, userText);  
 manageUsersButton.setPrefSize(500, 300);  
 manageUsersButton.setStyle("-fx-background-color: lightblue; -fx-border-color: black; -fx-border-width: 2;");  
 manageUsersButton.setGraphic(userBox);  
  
 plotsBox.setSpacing(10);  
 plotsBox.setAlignment(Pos.*CENTER*);  
 Image plotLogo = new Image(this.getClass().getResource("/plotsLogo.png").toExternalForm());  
 ImageView plotImage = new ImageView(plotLogo);  
 Label plotText = new Label("Manage Plots");  
 plotText.setFont(new Font("Times New Roman", 18));  
 plotsBox.getChildren().addAll(plotImage, plotText);  
 managePlotsButton.setPrefSize(500, 300);  
 managePlotsButton.setStyle("-fx-background-color: lightblue; -fx-border-color: black; -fx-border-width: 2;");  
 managePlotsButton.setGraphic(plotsBox);  
  
 paymentBox.setSpacing(10);  
 paymentBox.setAlignment(Pos.*CENTER*);  
 Image paymentLogo = new Image(this.getClass().getResource("/paymentsLogo.png").toExternalForm());  
 ImageView paymentImage = new ImageView(paymentLogo);  
 Label paymentText = new Label("Manage Payments");  
 paymentText.setFont(new Font("Times New Roman", 18));  
 paymentBox.getChildren().addAll(paymentImage, paymentText);  
 managePaymentsButton.setPrefSize(500, 300);  
 managePaymentsButton.setStyle("-fx-background-color: lightblue; -fx-border-color: black; -fx-border-width: 2;");  
 managePaymentsButton.setGraphic(paymentBox);  
  
 reportBox.setSpacing(10);  
 reportBox.setAlignment(Pos.*CENTER*);  
 Image reportLogo = new Image(this.getClass().getResource("/reportLogo.png").toExternalForm());  
 ImageView reportImage = new ImageView(reportLogo);  
 Label reportText = new Label("Generate Reports");  
 reportText.setFont(new Font("Times New Roman", 18));  
 reportBox.getChildren().addAll(reportImage, reportText);  
 generateReportsButton.setPrefSize(500, 300);  
 generateReportsButton.setStyle("-fx-background-color: lightblue; -fx-border-color: black; -fx-border-width: 2;");  
 generateReportsButton.setGraphic(reportBox);  
  
 functionsGridPane.setHgap(20);  
 functionsGridPane.setVgap(20);  
 functionsGridPane.setAlignment(Pos.*CENTER*);  
  
 functionsGridPane.add(manageUsersButton, 0,0);  
 functionsGridPane.add(managePlotsButton,1,0);  
 functionsGridPane.add(managePaymentsButton, 0, 1);  
 functionsGridPane.add(generateReportsButton,1, 1);  
  
 Button logoutButton = new Button("Logout");  
 logoutButton.setStyle( "-fx-background-color: red ;-fx-text-fill: white;-fx-font-size: 14; -fx-padding: 10 20; -fx-background-radius: 5;");  
 logoutButton.setPrefWidth(300);  
  
 vBox.getChildren().addAll(  
 titleLabel,  
 functionsGridPane,  
 logoutButton  
 );  
  
  
 manageUsersButton.setOnAction(e -> UserManagementScene(stage));  
 managePlotsButton.setOnAction(e -> managePlotsScene(stage));  
 managePaymentsButton.setOnAction(e -> managePaymentsScene(stage));  
 generateReportsButton.setOnAction(e -> generateReportScene(stage));  
  
 logoutButton.setOnAction(e -> {  
 Alert alert = new Alert(Alert.AlertType.*CONFIRMATION*);  
 alert.setTitle("Logout Confirmation");  
 alert.setHeaderText(null);  
 alert.setContentText("Are you sure you want to logout?");  
 Optional<ButtonType> result = alert.showAndWait();  
 if (result.isPresent() && result.get() == ButtonType.*OK*) {  
 loginScreen(stage);  
 }  
 });  
  
 Scene scene = new Scene(vBox, 1300, 800);  
 stage.setScene(scene);  
 stage.setTitle("Admin Dashboard");  
 }  
  
  
 public void UserManagementScene(Stage stage) {  
 VBox vBox = new VBox();  
 vBox.setAlignment(Pos.*CENTER*);  
 vBox.setSpacing(10);  
 vBox.setBackground(background());  
 vBox.setPrefWidth(1300);  
 vBox.setPrefHeight(800);  
  
 Label titleLabel = new Label("Buyers Management");  
 titleLabel.setStyle("-fx-font-weight: bold; -fx-text-fill: black;");  
 titleLabel.setFont(new Font("Times New Roman", 56));  
 titleLabel.setPadding(new Insets(10, 10, 50, 10));  
 titleLabel.setAlignment(Pos.*CENTER*);  
 vBox.getChildren().add(titleLabel);  
  
  
 TableView<User> buyersTable = new TableView<>();  
 buyersTable.setItems(*loadUsers*());  
  
 TableColumn<User, String> idColumn = new TableColumn<>("User ID");  
 idColumn.setCellValueFactory(new PropertyValueFactory<>("userId"));  
  
 TableColumn<User, String> usernameColumn = new TableColumn<>("Username");  
 usernameColumn.setCellValueFactory(new PropertyValueFactory<>("username"));  
  
 TableColumn<User, String> emailColumn = new TableColumn<>("Email");  
 emailColumn.setCellValueFactory(new PropertyValueFactory<>("email"));  
  
 TableColumn<User, String> phoneColumn = new TableColumn<>("Phone Number");  
 phoneColumn.setCellValueFactory(new PropertyValueFactory<>("phoneNumber"));  
  
 TableColumn<User, String> locationColumn = new TableColumn<>("Preferred Location");  
 locationColumn.setCellValueFactory(new PropertyValueFactory<>("preferredLocation"));  
  
 TableColumn<User, Double> sizeColumn = new TableColumn<>("Preferred Size");  
 sizeColumn.setCellValueFactory(new PropertyValueFactory<>("preferredSize"));  
  
 TableColumn<User, Double> budgetColumn = new TableColumn<>("Budget");  
 budgetColumn.setCellValueFactory(new PropertyValueFactory<>("budget"));  
  
 idColumn.setPrefWidth(100);  
 usernameColumn.setPrefWidth(150);  
 emailColumn.setPrefWidth(200);  
 phoneColumn.setPrefWidth(150);  
 locationColumn.setPrefWidth(200);  
 sizeColumn.setPrefWidth(150);  
 budgetColumn.setPrefWidth(200);  
  
 buyersTable.setStyle("-fx-background-color: #f9f9f9; -fx-border-color: #ccc; -fx-border-radius: 5px;");  
  
 idColumn.setStyle("-fx-alignment: CENTER; -fx-font-size: 14px; -fx-font-family: Arial;");  
 usernameColumn.setStyle("-fx-alignment: CENTER\_LEFT; -fx-font-size: 14px; -fx-font-family: Arial;");  
 emailColumn.setStyle("-fx-alignment: CENTER\_LEFT; -fx-font-size: 14px; -fx-font-family: Arial;");  
 phoneColumn.setStyle("-fx-alignment: CENTER; -fx-font-size: 14px; -fx-font-family: Arial;");  
 locationColumn.setStyle("-fx-alignment: CENTER\_LEFT; -fx-font-size: 14px; -fx-font-family: Arial;");  
 sizeColumn.setStyle("-fx-alignment: CENTER\_RIGHT; -fx-font-size: 14px; -fx-font-family: Arial;");  
 budgetColumn.setStyle("-fx-alignment: CENTER\_RIGHT; -fx-font-size: 14px; -fx-font-family: Arial;");  
  
 buyersTable.setPrefWidth(1000);  
 buyersTable.setPrefHeight(500);  
 buyersTable.getColumns().addAll(idColumn, usernameColumn, emailColumn, phoneColumn, locationColumn, sizeColumn, budgetColumn);  
  
  
 Button addButton = new Button("Add User");  
 Button editButton = new Button("Edit User");  
 Button deleteButton = new Button("Delete User");  
  
  
 addButton.setOnAction(e -> {  
 Stage addStage = new Stage();  
 VBox addVBox = new VBox();  
 addVBox.setAlignment(Pos.*CENTER*);  
 addVBox.setSpacing(15);  
 addVBox.setPadding(new Insets(10));  
 addVBox.setBackground(background());  
 GridPane inputGrid = new GridPane();  
 inputGrid.setAlignment(Pos.*CENTER*);  
 inputGrid.setHgap(10);  
 inputGrid.setVgap(15);  
 inputGrid.setPadding(new Insets(20, 30, 20, 20));  
 Label username = new Label("Username:");  
 Label password = new Label("Password:");  
 Label email = new Label("Email:");  
 Label phoneNumber = new Label("Phone Number:");  
 Label preferredLocation = new Label("Preferred Location:");  
 Label preferredSize = new Label("Preferred Size:");  
 Label budget = new Label("Budget:");  
  
 username.setStyle("-fx-text-fill: black; -fx-font-size: 20px; -fx-font-family: Arial; -fx-font-weight: bold;");  
 password.setStyle("-fx-text-fill: black; -fx-font-size: 20px; -fx-font-family: Arial; -fx-font-weight: bold;");  
 email.setStyle("-fx-text-fill:black; -fx-font-size: 20px; -fx-font-family: Arial; -fx-font-weight: bold;");  
 phoneNumber.setStyle("-fx-text-fill: black; -fx-font-size: 20px; -fx-font-family: Arial; -fx-font-weight: bold;");  
 preferredLocation.setStyle("-fx-text-fill:black; -fx-font-size: 20px; -fx-font-family: Arial; -fx-font-weight: bold;");  
 preferredSize.setStyle("-fx-text-fill: black; -fx-font-size: 20px; -fx-font-family: Arial; -fx-font-weight: bold;");  
 budget.setStyle("-fx-text-fill: black; -fx-font-size: 20px; -fx-font-family: Arial; -fx-font-weight: bold;");  
  
 TextField usernameField = new TextField();  
 usernameField.setPromptText("Username");  
 usernameField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 usernameField.setPrefWidth(800);  
 PasswordField passwordField = new PasswordField();  
 passwordField.setPromptText("Password");  
 passwordField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 passwordField.setPrefWidth(800);  
 TextField emailField = new TextField();  
 emailField.setPromptText("Email");  
 emailField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 emailField.setPrefWidth(800);  
 TextField phoneNumberField = new TextField();  
 phoneNumberField.setPromptText("Phone number");  
 phoneNumberField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 phoneNumberField.setPrefWidth(800);  
 TextField preferredLocationField = new TextField();  
 preferredLocationField.setPromptText("Preferred Plot Location");  
 preferredLocationField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 preferredLocationField.setPrefWidth(800);  
 TextField preferredSizeField = new TextField();  
 preferredSizeField.setPromptText("Preferred Plot Size");  
 preferredSizeField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 preferredSizeField.setPrefWidth(800);  
 TextField budgetField = new TextField();  
 budgetField.setPromptText("Budget");  
 budgetField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 budgetField.setPrefWidth(800);  
  
 Button saveButton = new Button("Save");  
 saveButton.setOnAction(event -> {  
 // Validate inputs and save the user  
 String usernameText = usernameField.getText();  
 String emailText = emailField.getText();  
 String phoneText = phoneNumberField.getText();  
 String locationText = preferredLocationField.getText();  
 String sizeText = preferredSizeField.getText();  
 String budgetText = budgetField.getText();  
  
 if (usernameText.isEmpty() || emailText.isEmpty() || phoneText.isEmpty() || locationText.isEmpty() || sizeText.isEmpty() || budgetText.isEmpty()) {  
 Alert alert = new Alert(Alert.AlertType.*WARNING*);  
 alert.setTitle("Missing Fields");  
 alert.setHeaderText("Please fill in all fields.");  
 alert.setContentText("All fields must be completed before saving.");  
 alert.showAndWait();  
 } else {  
 try {  
 double preferredSizeValue = Double.*parseDouble*(sizeText);  
 double budgetValue = Double.*parseDouble*(budgetText);  
 ObservableList<User> buyers = buyersTable.getItems();  
 buyers.add(new Buyer(usernameText, passwordField.getText(), emailText, phoneText, locationText, preferredSizeValue, budgetValue));  
 *saveUsers*(buyers);  
 addStage.close();  
  
 Alert successAlert = new Alert(Alert.AlertType.*INFORMATION*);  
 successAlert.setTitle("User Added");  
 successAlert.setHeaderText("The user has been successfully added.");  
 successAlert.showAndWait();  
 } catch (NumberFormatException ex) {  
 Alert errorAlert = new Alert(Alert.AlertType.*ERROR*);  
 errorAlert.setTitle("Invalid Input");  
 errorAlert.setHeaderText("Please enter valid numbers.");  
 errorAlert.setContentText("Please ensure 'Preferred Size' and 'Budget' are valid numbers.");  
 errorAlert.showAndWait();  
 }  
 }  
 });  
 saveButton.setPrefSize(100, 60);  
 saveButton.setStyle("-fx-font-size: 14px; -fx-text-fill: white; -fx-background-color: green;");  
  
 inputGrid.add(username, 0, 0);  
 inputGrid.add(usernameField, 1,0);  
 inputGrid.add(password, 0, 1);  
 inputGrid.add(passwordField,1,1);  
 inputGrid.add(email,0, 2);  
 inputGrid.add(emailField, 1, 2);  
 inputGrid.add(phoneNumber,0,3);  
 inputGrid.add(phoneNumberField,1,3);  
 inputGrid.add(preferredLocation,0, 4);  
 inputGrid.add(preferredLocationField,1, 4);  
 inputGrid.add(preferredSize,0,5);  
 inputGrid.add(preferredSizeField,1,5);  
 inputGrid.add(budget,0,6);  
 inputGrid.add(budgetField,1,6);  
  
 usernameField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 passwordField.requestFocus();  
 });  
 passwordField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 emailField.requestFocus();  
 });  
 emailField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 phoneNumberField.requestFocus();  
 });  
 phoneNumberField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 preferredLocationField.requestFocus();  
 });  
  
 preferredLocationField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 preferredSizeField.requestFocus();  
 });  
 preferredSizeField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 budgetField.requestFocus();  
 });  
 budgetField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 saveButton.fire();  
 });  
  
  
 addVBox.getChildren().addAll(inputGrid, saveButton);  
 addStage.setScene(new Scene(addVBox, 1300, 800));  
 addStage.setTitle("Add User");  
 addStage.show();  
 });  
  
  
 editButton.setOnAction(e -> {  
 Buyer selectedBuyer = (Buyer) buyersTable.getSelectionModel().getSelectedItem();  
 if (selectedBuyer != null) {  
 Stage editStage = new Stage();  
 VBox editVBox = new VBox();  
 editVBox.setAlignment(Pos.*CENTER*);  
 editVBox.setSpacing(15);  
 editVBox.setPadding(new Insets(10));  
 editVBox.setBackground(background());  
 GridPane inputGrid = new GridPane();  
 inputGrid.setAlignment(Pos.*CENTER*);  
 inputGrid.setHgap(10);  
 inputGrid.setVgap(15);  
 inputGrid.setPadding(new Insets(20, 30, 20, 20));  
  
 Label email = new Label("Email:");  
 Label phoneNumber = new Label("Phone Number:");  
 Label preferredLocation = new Label("Preferred Location:");  
 Label preferredSize = new Label("Preferred Size:");  
 Label budget = new Label("Budget:");  
  
 email.setStyle("-fx-text-fill:black; -fx-font-size: 20px; -fx-font-family: Arial; -fx-font-weight: bold;");  
 phoneNumber.setStyle("-fx-text-fill: black; -fx-font-size: 20px; -fx-font-family: Arial; -fx-font-weight: bold;");  
 preferredLocation.setStyle("-fx-text-fill:black; -fx-font-size: 20px; -fx-font-family: Arial; -fx-font-weight: bold;");  
 preferredSize.setStyle("-fx-text-fill: black; -fx-font-size: 20px; -fx-font-family: Arial; -fx-font-weight: bold;");  
 budget.setStyle("-fx-text-fill: black; -fx-font-size: 20px; -fx-font-family: Arial; -fx-font-weight: bold;");  
  
 TextField emailField = new TextField();  
 emailField.setPromptText("Email");  
 emailField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 emailField.setPrefWidth(800);  
 TextField phoneNumberField = new TextField();  
 phoneNumberField.setPromptText("Phone number");  
 phoneNumberField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 phoneNumberField.setPrefWidth(800);  
 TextField preferredLocationField = new TextField();  
 preferredLocationField.setPromptText("Preferred Plot Location");  
 preferredLocationField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 preferredLocationField.setPrefWidth(800);  
 TextField preferredSizeField = new TextField();  
 preferredSizeField.setPromptText("Preferred Plot Size");  
 preferredSizeField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 preferredSizeField.setPrefWidth(800);  
 TextField budgetField = new TextField();  
 budgetField.setPromptText("Budget");  
 budgetField.setStyle("-fx-font-size: 16px; -fx-padding: 10px; -fx-background-color: #fff; -fx-border-color: black; -fx-border-radius: 5px;");  
 budgetField.setPrefWidth(800);  
  
  
 Button saveButton = new Button("Save Changes");  
 saveButton.setOnAction(event -> {  
 String emailText = emailField.getText();  
 String phoneText = phoneNumberField.getText();  
 String locationText = preferredLocationField.getText();  
 String sizeText = preferredSizeField.getText();  
 String budgetText = budgetField.getText();  
  
 if (emailText.isEmpty() || phoneText.isEmpty() || locationText.isEmpty() || sizeText.isEmpty() || budgetText.isEmpty()) {  
 Alert alert = new Alert(Alert.AlertType.*WARNING*);  
 alert.setTitle("Missing Fields");  
 alert.setHeaderText("Please fill in all fields.");  
 alert.setContentText("All fields must be completed before saving.");  
 alert.showAndWait();  
 } else {  
 try {  
 double preferredSizeValue = Double.*parseDouble*(sizeText);  
 double budgetValue = Double.*parseDouble*(budgetText);  
  
 // Check if any changes were made before saving  
 boolean changesMade = !selectedBuyer.getEmail().equals(emailText) || !selectedBuyer.getPhoneNumber().equals(phoneText) ||  
 !selectedBuyer.getPreferredLocation().equals(locationText) || selectedBuyer.getPreferredSize() != preferredSizeValue ||  
 selectedBuyer.getBudget() != budgetValue;  
  
 if (changesMade) {  
 selectedBuyer.setEmail(emailText);  
 selectedBuyer.setPhoneNumber(phoneText);  
 selectedBuyer.setPreferredLocation(locationText);  
 selectedBuyer.setPreferredSize(preferredSizeValue);  
 selectedBuyer.setBudget(budgetValue);  
 buyersTable.refresh();  
 *saveUsers*(buyersTable.getItems());  
  
 Alert successAlert = new Alert(Alert.AlertType.*INFORMATION*);  
 successAlert.setTitle("Changes Saved");  
 successAlert.setHeaderText("User details have been updated.");  
 successAlert.showAndWait();  
 } else {  
 Alert infoAlert = new Alert(Alert.AlertType.*INFORMATION*);  
 infoAlert.setTitle("No Changes");  
 infoAlert.setHeaderText("No changes were made.");  
 infoAlert.setContentText("The user's details were not modified.");  
 infoAlert.showAndWait();  
 }  
 editStage.close();  
 } catch (NumberFormatException ex) {  
 Alert errorAlert = new Alert(Alert.AlertType.*ERROR*);  
 errorAlert.setTitle("Invalid Input");  
 errorAlert.setHeaderText("Please enter valid numbers.");  
 errorAlert.setContentText("Please ensure 'Preferred Size' and 'Budget' are valid numbers.");  
 errorAlert.showAndWait();  
 }  
 }  
 });  
 inputGrid.add(email,0, 2);  
 inputGrid.add(emailField, 1, 2);  
 inputGrid.add(phoneNumber,0,3);  
 inputGrid.add(phoneNumberField,1,3);  
 inputGrid.add(preferredLocation,0, 4);  
 inputGrid.add(preferredLocationField,1, 4);  
 inputGrid.add(preferredSize,0,5);  
 inputGrid.add(preferredSizeField,1,5);  
 inputGrid.add(budget,0,6);  
 inputGrid.add(budgetField,1,6);  
  
 emailField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 phoneNumberField.requestFocus();  
 });  
 phoneNumberField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 preferredLocationField.requestFocus();  
 });  
  
 preferredLocationField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 preferredSizeField.requestFocus();  
 });  
 preferredSizeField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 budgetField.requestFocus();  
 });  
 budgetField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 saveButton.fire();  
 });  
  
 editVBox.getChildren().addAll(inputGrid, saveButton);  
 saveButton.setPrefSize(100, 60);  
 saveButton.setStyle("-fx-font-size: 14px; -fx-text-fill: white; -fx-background-color: green;");  
 editStage.setScene(new Scene(editVBox, 1300, 800));  
 editStage.setTitle("Edit User");  
 editStage.show();  
 }else {  
 Alert warningAlert = new Alert(Alert.AlertType.*WARNING*);  
 warningAlert.setTitle("No User Selected");  
 warningAlert.setHeaderText(null);  
 warningAlert.setContentText("Please select a user to edit.");  
 warningAlert.showAndWait();  
 }  
 });  
  
  
 deleteButton.setOnAction(e -> {  
 User selectedUser = buyersTable.getSelectionModel().getSelectedItem();  
 if (selectedUser != null) {  
 Alert confirmationAlert = new Alert(Alert.AlertType.*CONFIRMATION*);  
 confirmationAlert.setTitle("Confirm Delete");  
 confirmationAlert.setHeaderText("Are you sure you want to delete this user?");  
 confirmationAlert.setContentText("This action cannot be undone.");  
  
 Optional<ButtonType> result = confirmationAlert.showAndWait();  
 if (result.isPresent() && result.get() == ButtonType.*OK*) {  
 buyersTable.getItems().remove(selectedUser);  
 *saveUsers*(buyersTable.getItems());  
 }  
 } else {  
 Alert warningAlert = new Alert(Alert.AlertType.*WARNING*);  
 warningAlert.setTitle("No User Selected");  
 warningAlert.setHeaderText(null);  
 warningAlert.setContentText("Please select a user to delete.");  
 warningAlert.showAndWait();  
 }  
 });  
  
  
 Button backButton = new Button("Back");  
 backButton.setOnAction(e -> adminDashboardScene(stage));  
  
 addButton.setPrefSize(100, 60);  
 editButton.setPrefSize(100, 60);  
 deleteButton.setPrefSize(100, 60);  
  
 addButton.setStyle("-fx-text-fill: white; -fx-background-color: lightblue; -fx-font-size: 16px;");  
 editButton.setStyle("-fx-text-fill: white; -fx-background-color: lightblue; -fx-font-size: 16px;");  
 deleteButton.setStyle("-fx-text-fill: white; -fx-background-color: lightblue; -fx-font-size: 16px;");  
  
 backButton.setPrefSize(100, 60);  
 backButton.setStyle("-fx-fill-text: white; -fx-background-color: green; -fx-font-size: 16px;");  
  
 HBox buttonBox = new HBox(10, addButton, editButton, deleteButton);  
 buttonBox.setAlignment(Pos.*CENTER*);  
  
 vBox.getChildren().addAll(buyersTable, buttonBox, backButton);  
  
 Scene scene = new Scene(vBox, 1300, 800);  
 stage.setScene(scene);  
 stage.setTitle("User Management");  
 }  
  
 public void managePlotsScene(Stage stage) {  
 VBox vBox = new VBox();  
 vBox.setAlignment(Pos.*CENTER*);  
 vBox.setSpacing(10);  
 vBox.setBackground(background());  
  
 Label titleLabel = new Label("Plots Management Section");  
 titleLabel.setStyle("-fx-font-size: 50px; -fx-font-weight: bold;");  
 titleLabel.setFont(new Font("Times New Roman", 80));  
 titleLabel.setTextFill(Color.*WHITE*);  
 titleLabel.setPadding(new Insets(10, 10, 50, 10));  
 titleLabel.setAlignment(Pos.*CENTER*);  
 vBox.getChildren().add(titleLabel);  
  
  
 TableView<Plot> plotTable = new TableView<>();  
 plotTable.setItems(*loadPlots*());  
  
 TableColumn<Plot, Integer> idColumn = new TableColumn<>("Plot ID");  
 idColumn.setCellValueFactory(new PropertyValueFactory<>("plotId"));  
  
 TableColumn<Plot, String> numberColumn = new TableColumn<>("Plot Number");  
 numberColumn.setCellValueFactory(new PropertyValueFactory<>("plotNumber"));  
  
 TableColumn<Plot, Double> lengthColumn = new TableColumn<>("Length (ft)");  
 lengthColumn.setCellValueFactory(new PropertyValueFactory<>("length"));  
  
 TableColumn<Plot, Double> widthColumn = new TableColumn<>("Width (ft)");  
 widthColumn.setCellValueFactory(new PropertyValueFactory<>("width"));  
  
 TableColumn<Plot, Double> areaColumn = new TableColumn<>("Total Area (Marla)");  
 areaColumn.setCellValueFactory(new PropertyValueFactory<>("totalArea"));  
  
 TableColumn<Plot, String> locationColumn = new TableColumn<>("Location");  
 locationColumn.setCellValueFactory(new PropertyValueFactory<>("location"));  
  
 TableColumn<Plot, String> typeColumn = new TableColumn<>("Plot Type");  
 typeColumn.setCellValueFactory(new PropertyValueFactory<>("plotType"));  
  
 TableColumn<Plot, String> categoryColumn = new TableColumn<>("Plot Category");  
 categoryColumn.setCellValueFactory(new PropertyValueFactory<>("plotCategory"));  
  
 TableColumn<Plot, String> statusColumn = new TableColumn<>("Status");  
 statusColumn.setCellValueFactory(new PropertyValueFactory<>("status"));  
  
 TableColumn<Plot, Double> marlaColumn = new TableColumn<>("Price Per Marla");  
 marlaColumn.setCellValueFactory(new PropertyValueFactory<>("pricePerMarla"));  
  
 TableColumn<Plot, Double> priceColumn = new TableColumn<>("Total Price");  
 priceColumn.setCellValueFactory(new PropertyValueFactory<>("totalPrice"));  
  
 plotTable.getColumns().addAll(idColumn, numberColumn, lengthColumn, widthColumn, areaColumn, locationColumn, typeColumn, categoryColumn, marlaColumn, priceColumn, statusColumn);  
  
  
 Button addButton = new Button("Add Plot");  
 Button editButton = new Button("Edit Plot");  
 Button deleteButton = new Button("Delete Plot");  
  
 addButton.setStyle("-fx-background-color: #3b5998; " +  
 "-fx-text-fill: white; " +  
 "-fx-font-size: 18px; " +  
 "-fx-font-weight: bold; " +  
 "-fx-padding: 10 20; " +  
 "-fx-border-radius: 5; " +  
 "-fx-background-radius: 5;");  
  
 editButton.setStyle("-fx-background-color: #3b5998; " +  
 "-fx-text-fill: white; " +  
 "-fx-font-size: 18px; " +  
 "-fx-font-weight: bold; " +  
 "-fx-padding: 10 20; " +  
 "-fx-border-radius: 5; " +  
 "-fx-background-radius: 5;");  
  
 deleteButton.setStyle("-fx-background-color: #3b5998; " +  
 "-fx-text-fill: white; " +  
 "-fx-font-size: 18px; " +  
 "-fx-font-weight: bold; " +  
 "-fx-padding: 10 20; " +  
 "-fx-border-radius: 5; " +  
 "-fx-background-radius: 5;");  
  
  
 addButton.setOnMouseEntered(e -> addButton.setStyle("-fx-background-color: #5a7ad6; " +  
 "-fx-text-fill: white; " +  
 "-fx-font-size: 18px; " +  
 "-fx-font-weight: bold; " +  
 "-fx-padding: 10 20; " +  
 "-fx-border-radius: 5; " +  
 "-fx-background-radius: 5;"));  
  
 addButton.setOnMouseExited(e -> addButton.setStyle("-fx-background-color: #3b5998; " +  
 "-fx-text-fill: white; " +  
 "-fx-font-size: 18px; " +  
 "-fx-font-weight: bold; " +  
 "-fx-padding: 10 20; " +  
 "-fx-border-radius: 5; " +  
 "-fx-background-radius: 5;"));  
  
  
 editButton.setOnMouseEntered(e -> editButton.setStyle("-fx-background-color: #5a7ad6; " +  
 "-fx-text-fill: white; " +  
 "-fx-font-size: 18px; " +  
 "-fx-font-weight: bold; " +  
 "-fx-padding: 10 20; " +  
 "-fx-border-radius: 5; " +  
 "-fx-background-radius: 5;"));  
  
 editButton.setOnMouseExited(e -> editButton.setStyle("-fx-background-color: #3b5998; " +  
 "-fx-text-fill: white; " +  
 "-fx-font-size: 18px; " +  
 "-fx-font-weight: bold; " +  
 "-fx-padding: 10 20; " +  
 "-fx-border-radius: 5; " +  
 "-fx-background-radius: 5;"));  
  
 deleteButton.setOnMouseEntered(e -> deleteButton.setStyle("-fx-background-color: #5a7ad6; " +  
 "-fx-text-fill: white; " +  
 "-fx-font-size: 18px; " +  
 "-fx-font-weight: bold; " +  
 "-fx-padding: 10 20; " +  
 "-fx-border-radius: 5; " +  
 "-fx-background-radius: 5;"));  
  
 deleteButton.setOnMouseExited(e -> deleteButton.setStyle("-fx-background-color: #3b5998; " +  
 "-fx-text-fill: white; " +  
 "-fx-font-size: 18px; " +  
 "-fx-font-weight: bold; " +  
 "-fx-padding: 10 20; " +  
 "-fx-border-radius: 5; " +  
 "-fx-background-radius: 5;"));  
 addButton.setOnAction(e -> {  
 Stage addStage = new Stage();  
 VBox addVBox = new VBox(10);  
 addVBox.setAlignment(Pos.*CENTER*);  
 addVBox.setSpacing(15);  
 addVBox.setPadding(new Insets(10));  
 addVBox.setBackground(background());  
  
  
 VBox inputBox = new VBox(10);  
 inputBox.setAlignment(Pos.*CENTER*);  
 inputBox.setPadding(new Insets(20));  
 inputBox.setSpacing(15);  
 inputBox.setStyle("-fx-background-color: #ffffff; " +  
 "-fx-border-color: #cccccc; " +  
 "-fx-border-width: 2; " +  
 "-fx-border-radius: 10; " +  
 "-fx-background-radius: 10;");  
  
 GridPane inputGrid = new GridPane();  
 inputGrid.setAlignment(Pos.*CENTER*);  
 inputGrid.setHgap(15);  
 inputGrid.setVgap(20);  
 inputGrid.setPadding(new Insets(20, 30, 20, 20));  
  
 Label plotNumber = new Label("Plot Number:");  
 Label length = new Label("Length:");  
 Label width = new Label("Width:");  
 Label location = new Label("Location:");  
 Label type = new Label("Plot Type:");  
 Label category = new Label("Plot Category:");  
 Label pricePerMarla = new Label("Price Per Marla:");  
 Label status = new Label("Status:");  
  
 String labelStyle = "-fx-font-size: 18px; " +  
 "-fx-font-weight: bold; " +  
 "-fx-text-fill: #2a2a2a;";  
  
 plotNumber.setStyle(labelStyle);  
 length.setStyle(labelStyle);  
 width.setStyle(labelStyle);  
 location.setStyle(labelStyle);  
 type.setStyle(labelStyle);  
 category.setStyle(labelStyle);  
 pricePerMarla.setStyle(labelStyle);  
 status.setStyle(labelStyle);  
  
 TextField numberField = new TextField();  
 numberField.setPromptText("Plot Number");  
  
 TextField lengthField = new TextField();  
 lengthField.setPromptText("Length (ft.)");  
  
 TextField widthField = new TextField();  
 widthField.setPromptText("Width (ft.)");  
  
 TextField locationField = new TextField();  
 locationField.setPromptText("Location");  
  
 TextField typeField = new TextField();  
 typeField.setPromptText("Plot Type (Commercial/ Residential)");  
  
 TextField statusField = new TextField();  
 statusField.setPromptText("Status");  
  
 TextField priceUnitField = new TextField();  
 priceUnitField.setPromptText("Price Per Marla");  
  
 TextField categoryField = new TextField();  
 categoryField.setPromptText("Plot Category (Corner, Park Facing, etc)");  
  
 String textFieldStyle = "-fx-font-size: 14px; " +  
 "-fx-padding: 5 10; " +  
 "-fx-border-color: #cccccc; " +  
 "-fx-border-width: 1; " +  
 "-fx-border-radius: 5; " +  
 "-fx-background-radius: 5;";  
  
 numberField.setStyle(textFieldStyle);  
 lengthField.setStyle(textFieldStyle);  
 widthField.setStyle(textFieldStyle);  
 locationField.setStyle(textFieldStyle);  
 typeField.setStyle(textFieldStyle);  
 statusField.setStyle(textFieldStyle);  
 priceUnitField.setStyle(textFieldStyle);  
 categoryField.setStyle(textFieldStyle);  
  
 Button saveButton = new Button("Save");  
 String buttonStyle = "-fx-background-color: #28a745; " +  
 "-fx-text-fill: white; " +  
 "-fx-font-size: 16px; " +  
 "-fx-font-weight: bold; " +  
 "-fx-padding: 10 20; " +  
 "-fx-border-radius: 5; " +  
 "-fx-background-radius: 5;";  
  
 saveButton.setStyle(buttonStyle);  
  
 saveButton.setOnAction(event -> {  
 try {  
 if (numberField.getText().isEmpty() || lengthField.getText().isEmpty() ||  
 widthField.getText().isEmpty() || locationField.getText().isEmpty() ||  
 typeField.getText().isEmpty() || categoryField.getText().isEmpty() ||  
 priceUnitField.getText().isEmpty() || statusField.getText().isEmpty()) {  
  
 Alert alert = new Alert(Alert.AlertType.*WARNING*);  
 alert.setTitle("Missing Fields");  
 alert.setHeaderText("Please fill in all fields.");  
 alert.setContentText("All fields must be completed before saving.");  
 alert.showAndWait();  
 } else {  
 ObservableList<Plot> plots = *loadPlots*();  
 int plotId = *loadPlots*().size()+1;  
 double area = Double.*parseDouble*(lengthField.getText())\*Double.*parseDouble*(widthField.getText())\*0.00367309;  
 Plot newPlot = new Plot(plotId, numberField.getText(), Double.*parseDouble*(lengthField.getText()),Double.*parseDouble*(widthField.getText()),area,locationField.getText(), typeField.getText(), categoryField.getText(),Double.*parseDouble*(priceUnitField.getText()),Double.*parseDouble*(priceUnitField.getText())\*area, statusField.getText());  
 plots.add(newPlot);  
 *savePlots*(plots);  
 Alert successAlert = new Alert(Alert.AlertType.*INFORMATION*);  
 successAlert.setTitle("Plot Added");  
 successAlert.setHeaderText("The plot has been successfully added.");  
 successAlert.showAndWait();  
 plotTable.setItems(*loadPlots*());  
 addStage.close();  
 }  
 } catch (NumberFormatException ex) {  
 Alert errorAlert = new Alert(Alert.AlertType.*ERROR*);  
 errorAlert.setTitle("Invalid Input");  
 errorAlert.setHeaderText("Please enter valid numbers.");  
 errorAlert.setContentText("Ensure that the numeric fields are filled correctly.");  
 errorAlert.showAndWait();  
 }  
 });  
  
 inputGrid.add(plotNumber, 0, 0);  
 inputGrid.add(numberField, 1, 0);  
 inputGrid.add(length, 0, 1);  
 inputGrid.add(lengthField, 1, 1);  
 inputGrid.add(width, 0, 2);  
 inputGrid.add(widthField, 1, 2);  
 inputGrid.add(location, 0, 3);  
 inputGrid.add(locationField, 1, 3);  
 inputGrid.add(type, 0, 4);  
 inputGrid.add(typeField, 1, 4);  
 inputGrid.add(category, 0, 5);  
 inputGrid.add(categoryField, 1, 5);  
 inputGrid.add(pricePerMarla, 0, 6);  
 inputGrid.add(priceUnitField, 1, 6);  
 inputGrid.add(status, 0, 7);  
 inputGrid.add(statusField, 1, 7);  
 numberField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 lengthField.requestFocus();  
 });  
 lengthField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 widthField.requestFocus();  
 });  
 widthField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 locationField.requestFocus();  
 });  
 locationField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 typeField.requestFocus();  
 });  
  
 typeField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 categoryField.requestFocus();  
 });  
 categoryField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 priceUnitField.requestFocus();  
 });  
  
 priceUnitField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 statusField.requestFocus();  
 });  
  
 statusField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 saveButton.fire();  
 });  
 inputBox.getChildren().addAll(inputGrid, saveButton);  
 addVBox.getChildren().add(inputBox);  
  
 addStage.setScene(new Scene(addVBox, 1300, 800));  
 addStage.setTitle("Add Plot");  
 addStage.show();  
 });  
  
  
  
  
 editButton.setOnAction(e -> {  
 Plot selectedPlot = plotTable.getSelectionModel().getSelectedItem();  
 if (selectedPlot != null) {  
 Stage editStage = new Stage();  
 VBox editVBox = new VBox(10);  
 editVBox.setAlignment(Pos.*CENTER*);  
 editVBox.setSpacing(15);  
 editVBox.setPadding(new Insets(10));  
 editVBox.setBackground(background());  
 GridPane inputGrid = new GridPane();  
 inputGrid.setAlignment(Pos.*CENTER*);  
 inputGrid.setHgap(10);  
 inputGrid.setVgap(15);  
 inputGrid.setPadding(new Insets(20, 30, 20, 20));  
  
 Label plotNumber = new Label("Plot Number: ");  
 Label length = new Label("Length: ");  
 Label width = new Label("Width: ");  
 Label location = new Label("Location: ");  
 Label type = new Label("Plot Type: ");  
 Label category = new Label("Plot Category: ");  
 Label pricePerMarla = new Label("Price Per Marla: ");  
 Label status = new Label("Status: ");  
 TextField numberField = new TextField(selectedPlot.getPlotNumber());  
 numberField.setPromptText("Plot Number");  
  
 TextField lengthField = new TextField(String.*valueOf*(selectedPlot.getLength()));  
 lengthField.setPromptText("Length");  
  
 TextField widthField = new TextField(String.*valueOf*(selectedPlot.getWidth()));  
 widthField.setPromptText("Width");  
  
 TextField locationField = new TextField(selectedPlot.getLocation());  
 locationField.setPromptText("Location");  
  
 TextField typeField = new TextField(selectedPlot.getPlotType());  
 typeField.setPromptText("GPS Coordinates");  
  
 TextField statusField = new TextField(selectedPlot.getStatus());  
 statusField.setPromptText("Status");  
  
 TextField priceUnitField = new TextField(String.*valueOf*(selectedPlot.getPricePerMarla()));  
 priceUnitField.setPromptText("Price Per Marla");  
  
 TextField categoryField = new TextField(selectedPlot.getPlotCategory());  
 categoryField.setPromptText("Development Status");  
  
 Button saveButton = new Button("Save Changes");  
 saveButton.setOnAction(event -> {  
 try {  
 if (numberField.getText().isEmpty() || lengthField.getText().isEmpty() || widthField.getText().isEmpty() ||  
 locationField.getText().isEmpty() || typeField.getText().isEmpty() || categoryField.getText().isEmpty() ||  
 priceUnitField.getText().isEmpty() || statusField.getText().isEmpty()) {  
 Alert alert = new Alert(Alert.AlertType.*WARNING*);  
 alert.setTitle("Missing Fields");  
 alert.setHeaderText("Please fill in all fields.");  
 alert.setContentText("All fields must be completed before saving.");  
 alert.showAndWait();  
 } else {  
 selectedPlot.setPlotNumber(numberField.getText());  
 selectedPlot.setLength(Double.*parseDouble*(lengthField.getText()));  
 selectedPlot.setWidth(Double.*parseDouble*(widthField.getText()));  
 selectedPlot.setLocation(locationField.getText());  
 selectedPlot.setPlotType(typeField.getText());  
 selectedPlot.setStatus(statusField.getText());  
 selectedPlot.setPricePerMarla(Double.*parseDouble*(priceUnitField.getText()));  
 selectedPlot.setTotalArea(selectedPlot.getLength() \* selectedPlot.getWidth() \* 0.00367309);  
 selectedPlot.setTotalPrice(selectedPlot.getTotalArea() \* selectedPlot.getPricePerMarla());  
 selectedPlot.setPlotCategory(categoryField.getText());  
  
 plotTable.refresh();  
 *savePlots*(plotTable.getItems());  
  
 Alert successAlert = new Alert(Alert.AlertType.*INFORMATION*);  
 successAlert.setTitle("Changes Saved");  
 successAlert.setHeaderText("Plot details have been updated.");  
 successAlert.showAndWait();  
  
 editStage.close();  
 }  
 } catch (NumberFormatException ex) {  
 Alert errorAlert = new Alert(Alert.AlertType.*ERROR*);  
 errorAlert.setTitle("Invalid Input");  
 errorAlert.setHeaderText("Please enter valid numbers.");  
 errorAlert.setContentText("Ensure that the numeric fields are filled correctly.");  
 errorAlert.showAndWait();  
 }  
 });  
  
 inputGrid.add(plotNumber, 0, 0);  
 inputGrid.add(numberField, 1,0);  
 inputGrid.add(length, 0, 1);  
 inputGrid.add(lengthField,1,1);  
 inputGrid.add(width,0, 2);  
 inputGrid.add(widthField, 1, 2);  
 inputGrid.add(location,0,3);  
 inputGrid.add(locationField,1,3);  
 inputGrid.add(type,0, 4);  
 inputGrid.add(typeField,1, 4);  
 inputGrid.add(category,0,5);  
 inputGrid.add(categoryField,1,5);  
 inputGrid.add(pricePerMarla,0,6);  
 inputGrid.add(priceUnitField,1,6);  
 inputGrid.add(status,0, 7);  
 inputGrid.add(statusField, 1, 7);  
  
 numberField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 lengthField.requestFocus();  
 });  
 lengthField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 widthField.requestFocus();  
 });  
 widthField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 locationField.requestFocus();  
 });  
 locationField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 typeField.requestFocus();  
 });  
  
 typeField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 categoryField.requestFocus();  
 });  
 categoryField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 priceUnitField.requestFocus();  
 });  
  
 priceUnitField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 statusField.requestFocus();  
 });  
  
 statusField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 saveButton.fire();  
 });  
  
 editVBox.getChildren().addAll(inputGrid, saveButton);  
 editStage.setScene(new Scene(editVBox, 1300, 800));  
 editStage.setTitle("Edit Plot");  
 editStage.show();  
 }  
 });  
  
  
 deleteButton.setOnAction(e -> {  
 Plot selectedPlot = plotTable.getSelectionModel().getSelectedItem();  
 if (selectedPlot != null) {  
 Alert confirmationAlert = new Alert(Alert.AlertType.*CONFIRMATION*);  
 confirmationAlert.setTitle("Delete Plot");  
 confirmationAlert.setHeaderText("Are you sure you want to delete this plot?");  
 confirmationAlert.setContentText("This action cannot be undone.");  
  
 confirmationAlert.showAndWait().ifPresent(response -> {  
 if (response == ButtonType.*OK*) {  
 plotTable.getItems().remove(selectedPlot);  
 *savePlots*(plotTable.getItems());  
  
 Alert successAlert = new Alert(Alert.AlertType.*INFORMATION*);  
 successAlert.setTitle("Plot Deleted");  
 successAlert.setHeaderText("The plot has been deleted successfully.");  
 successAlert.showAndWait();  
 }  
 });  
 }  
 });  
  
  
 Button backButton = new Button("Back");  
 backButton.setOnAction(e -> adminDashboardScene(stage));  
  
  
 HBox buttonBox = new HBox(10, addButton, editButton, deleteButton);  
 buttonBox.setAlignment(Pos.*CENTER*);  
  
 vBox.getChildren().addAll(plotTable, buttonBox, backButton);  
  
 Scene scene = new Scene(vBox, 1300, 800);  
 stage.setScene(scene);  
 stage.setTitle("Plot Management");  
 }  
  
 public void managePaymentsScene(Stage stage){  
 VBox layout = new VBox(10);  
 layout.setPadding(new Insets(10));  
  
 Button backButton = new Button("Back");  
 backButton.setOnAction(e -> adminDashboardScene(stage));  
  
 ObservableList<Payment> payments = *loadPayments*();  
  
 // ListView for displaying payments  
 ListView<Payment> paymentListView = new ListView<>();  
 paymentListView.setItems(payments);  
  
 Button addPaymentButton = new Button("Add Payment");  
 addPaymentButton.setOnAction(e -> {  
  
 Stage addStage = new Stage();  
 VBox addVBox = new VBox(10);  
 addVBox.setAlignment(Pos.*CENTER*);  
 addVBox.setSpacing(15);  
 addVBox.setPadding(new Insets(10));  
 addVBox.setBackground(background());  
 GridPane inputGrid = new GridPane();  
 inputGrid.setAlignment(Pos.*CENTER*);  
 inputGrid.setHgap(10);  
 inputGrid.setVgap(15);  
 inputGrid.setPadding(new Insets(20, 30, 20, 20));  
  
 Label plotId = new Label("Plot ID: ");  
 Label buyerId = new Label("Buyer ID: ");  
 Label amount = new Label("Amount Paid: ");  
 Label paymentMethod = new Label("Payment Method: ");  
  
 TextField plotIdField= new TextField();  
 plotIdField.setPromptText("Plot ID");  
  
 TextField buyerIdField = new TextField();  
 buyerIdField.setPromptText("Buyer ID");  
  
 TextField amountField = new TextField();  
 amountField.setPromptText("Paid Amount");  
  
 TextField methodField = new TextField();  
 methodField.setPromptText("Payment Method");  
  
 Button saveButton = new Button("Save");  
 saveButton.setOnAction(event -> {  
 try {  
 int newId = *loadPayments*().size() + 1;  
 double totalPrice = 0;  
 ObservableList<Plot> plots = *loadPlots*();  
 for(Plot plot: plots){  
 if(plot.getPlotId() == Integer.*parseInt*(plotIdField.getText())){  
 totalPrice = plot.getTotalPrice();  
 break;  
 }  
 }  
 double outstandingBalance = totalPrice - Double.*parseDouble*(amountField.getText());  
 Payment newPayment = new Payment(newId, Integer.*parseInt*(plotIdField.getText()), Integer.*parseInt*(buyerIdField.getText()), Double.*parseDouble*(amountField.getText()), methodField.getText(), outstandingBalance, LocalDate.*now*());  
 payments.add(newPayment);  
 *savePayments*(payments);  
 if(outstandingBalance==0){  
 ObservableList<Document> documents = *loadDocuments*();  
 int documentId = documents.size()+1;  
 Document document = new Document(documentId, Integer.*parseInt*(plotIdField.getText()), Integer.*parseInt*(buyerIdField.getText()), "Ownership", LocalDate.*now*());  
 documents.add(document);  
 *saveDocuments*(documents);  
 }  
  
 // Show success alert  
 Alert successAlert = new Alert(Alert.AlertType.*INFORMATION*);  
 successAlert.setTitle("Payment Added");  
 successAlert.setHeaderText(null);  
 successAlert.setContentText("Payment has been successfully added.");  
 successAlert.showAndWait();  
  
 addStage.close();  
 } catch (NumberFormatException ex) {  
 // Show error alert  
 Alert errorAlert = new Alert(Alert.AlertType.*ERROR*);  
 errorAlert.setTitle("Invalid Input");  
 errorAlert.setHeaderText("Invalid input values");  
 errorAlert.setContentText("Please check your input and try again.");  
 errorAlert.showAndWait();  
 }  
 });  
  
 inputGrid.add(plotId, 0, 0);  
 inputGrid.add(plotIdField, 1,0);  
 inputGrid.add(buyerId, 0, 1);  
 inputGrid.add(buyerIdField,1,1);  
 inputGrid.add(amount,0, 2);  
 inputGrid.add(amountField, 1, 2);  
 inputGrid.add(paymentMethod,0,3);  
 inputGrid.add(methodField,1,3);  
  
 plotIdField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 buyerIdField.requestFocus();  
 });  
 buyerIdField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 amountField.requestFocus();  
 });  
 amountField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 methodField.requestFocus();  
 });  
 methodField.setOnKeyPressed(event->{  
 if(event.getCode().toString().equals("ENTER"))  
 saveButton.fire();  
 });  
  
 addVBox.getChildren().addAll(inputGrid, saveButton);  
 addStage.setScene(new Scene(addVBox, 1300, 800));  
 addStage.setTitle("Add Plot");  
 addStage.show();  
 });  
  
  
 Button removePaymentButton = new Button("Remove Payment");  
 removePaymentButton.setOnAction(e -> {  
 Payment selectedPayment = paymentListView.getSelectionModel().getSelectedItem();  
 if (selectedPayment != null) {  
 payments.remove(selectedPayment);  
 *savePayments*(payments);  
 paymentListView.getItems().setAll(*loadPayments*());  
  
 // Show success alert  
 Alert successAlert = new Alert(Alert.AlertType.*INFORMATION*);  
 successAlert.setTitle("Payment Removed");  
 successAlert.setHeaderText(null);  
 successAlert.setContentText("Payment has been successfully removed.");  
 successAlert.showAndWait();  
 } else {  
 // Show error alert  
 Alert errorAlert = new Alert(Alert.AlertType.*WARNING*);  
 errorAlert.setTitle("No Selection");  
 errorAlert.setHeaderText("No payment selected");  
 errorAlert.setContentText("Please select a payment to remove.");  
 errorAlert.showAndWait();  
 }  
 });  
  
  
 Button updatePaymentButton = new Button("Update Payment");  
 updatePaymentButton.setOnAction(e -> {  
 Payment selectedPayment = paymentListView.getSelectionModel().getSelectedItem();  
 if (selectedPayment != null) {  
 Stage updateStage = new Stage();  
 VBox updateVBox = new VBox(10);  
 updateVBox.setAlignment(Pos.*CENTER*);  
 updateVBox.setSpacing(15);  
 updateVBox.setPadding(new Insets(10));  
 updateVBox.setBackground(background());  
  
 GridPane inputGrid = new GridPane();  
 inputGrid.setAlignment(Pos.*CENTER*);  
 inputGrid.setHgap(10);  
 inputGrid.setVgap(15);  
 inputGrid.setPadding(new Insets(20, 30, 20, 20));  
  
 Label plotIdLabel = new Label("Plot ID: ");  
 Label buyerIdLabel = new Label("Buyer ID: ");  
 Label amountLabel = new Label("Amount Paid: ");  
 Label methodLabel = new Label("Payment Method: ");  
  
 TextField plotIdField = new TextField(String.*valueOf*(selectedPayment.getPlotId()));  
 TextField buyerIdField = new TextField(String.*valueOf*(selectedPayment.getBuyerId()));  
 TextField amountField = new TextField(String.*valueOf*(selectedPayment.getAmountPaid()));  
 TextField methodField = new TextField(selectedPayment.getPaymentMethod());  
  
 Button saveButton = new Button("Save Changes");  
 saveButton.setOnAction(event -> {  
 try {  
 int newPlotId = Integer.*parseInt*(plotIdField.getText());  
 int newBuyerId = Integer.*parseInt*(buyerIdField.getText());  
 double newAmountPaid = Double.*parseDouble*(amountField.getText());  
 String newMethod = methodField.getText();  
  
 // Update the selected payment with new values  
 selectedPayment.setPlotId(newPlotId);  
 selectedPayment.setBuyerId(newBuyerId);  
 selectedPayment.setAmountPaid(newAmountPaid);  
 selectedPayment.setPaymentMethod(newMethod);  
  
 // Recalculate outstanding balance  
 double totalPrice = 0;  
 ObservableList<Plot> plots = *loadPlots*();  
 for (Plot plot : plots) {  
 if (plot.getPlotId() == newPlotId) {  
 totalPrice = plot.getTotalPrice();  
 break;  
 }  
 }  
 selectedPayment.setOutstandingBalance(totalPrice - newAmountPaid);  
  
 // Refresh the ListView  
 paymentListView.refresh();  
  
 // Save the updated list  
 *savePayments*(payments);  
  
 // Show success alert  
 Alert successAlert = new Alert(Alert.AlertType.*INFORMATION*);  
 successAlert.setTitle("Payment Updated");  
 successAlert.setHeaderText(null);  
 successAlert.setContentText("Payment has been successfully updated.");  
 successAlert.showAndWait();  
  
 updateStage.close();  
 } catch (NumberFormatException ex) {  
 // Show error alert  
 Alert errorAlert = new Alert(Alert.AlertType.*ERROR*);  
 errorAlert.setTitle("Invalid Input");  
 errorAlert.setHeaderText("Invalid input values");  
 errorAlert.setContentText("Please check your input and try again.");  
 errorAlert.showAndWait();  
 }  
 });  
  
 inputGrid.add(plotIdLabel, 0, 0);  
 inputGrid.add(plotIdField, 1, 0);  
 inputGrid.add(buyerIdLabel, 0, 1);  
 inputGrid.add(buyerIdField, 1, 1);  
 inputGrid.add(amountLabel, 0, 2);  
 inputGrid.add(amountField, 1, 2);  
 inputGrid.add(methodLabel, 0, 3);  
 inputGrid.add(methodField, 1, 3);  
  
 updateVBox.getChildren().addAll(inputGrid, saveButton);  
 updateStage.setScene(new Scene(updateVBox, 600, 400));  
 updateStage.setTitle("Update Payment");  
 updateStage.show();  
 } else {  
 System.*out*.println("Please select a payment to update.");  
 }  
 });  
  
  
 layout.getChildren().addAll(paymentListView, addPaymentButton, removePaymentButton, updatePaymentButton,backButton);  
  
 Scene scene = new Scene(layout, 1300, 800);  
 stage.setScene(scene);  
 stage.setTitle("Manage Payments");  
 stage.show();  
 }  
  
 public void generateReportScene(Stage stage) {  
 VBox layout = new VBox(10);  
 layout.setPadding(new Insets(10));  
  
 ObservableList<Plot> plots = *loadPlots*();  
  
 int residentialCount = 0;  
 int commercialCount = 0;  
 int availableCount = 0;  
 int reservedCount = 0;  
 int soldCount = 0;  
  
 int cornerCount = 0;  
 int parkFacingCount = 0;  
 int mainBoulevardCount = 0;  
  
 for (Plot plot : plots) {  
 // Count plot types  
 if ("Residential".equalsIgnoreCase(plot.getPlotType())) {  
 residentialCount++;  
 } else if ("Commercial".equalsIgnoreCase(plot.getPlotType())) {  
 commercialCount++;  
 }  
  
 // Count plot statuses  
 if ("Available".equalsIgnoreCase(plot.getStatus())) {  
 availableCount++;  
 } else if ("Reserved".equalsIgnoreCase(plot.getStatus())) {  
 reservedCount++;  
 } else if ("Sold".equalsIgnoreCase(plot.getStatus())) {  
 soldCount++;  
 }  
  
 // Count plot categories  
 if ("Corner".equalsIgnoreCase(plot.getPlotCategory())) {  
 cornerCount++;  
 } else if ("Park-Facing".equalsIgnoreCase(plot.getPlotCategory())) {  
 parkFacingCount++;  
 } else if ("Main Boulevard".equalsIgnoreCase(plot.getPlotCategory())) {  
 mainBoulevardCount++;  
 }  
 }  
  
 // Create report and statistics  
 String report = generateReports();  
 String plotStatistics = analyzePlotStatistics();  
  
 // Create a TextArea for textual reports and make it larger  
 final TextArea reportTextArea = new TextArea(report + "\n\n" + plotStatistics);  
 reportTextArea.setEditable(false);  
 reportTextArea.setWrapText(true);  
  
 // Set preferred width and height for the TextArea  
 reportTextArea.setPrefWidth(1200);  
 reportTextArea.setPrefHeight(400);  
  
 // Create PieCharts for plot types and statuses  
 PieChart plotTypeChart = new PieChart();  
 plotTypeChart.setTitle("Plot Type Distribution");  
 plotTypeChart.getData().addAll(  
 new PieChart.Data("Residential", residentialCount),  
 new PieChart.Data("Commercial", commercialCount)  
 );  
  
 PieChart plotStatusChart = new PieChart();  
 plotStatusChart.setTitle("Plot Status Distribution");  
 plotStatusChart.getData().addAll(  
 new PieChart.Data("Available", availableCount),  
 new PieChart.Data("Reserved", reservedCount),  
 new PieChart.Data("Sold", soldCount)  
 );  
  
 // Create BarChart for plot categories  
 CategoryAxis xAxis = new CategoryAxis();  
 NumberAxis yAxis = new NumberAxis();  
 BarChart<String, Number> plotCategoryChart = new BarChart<>(xAxis, yAxis);  
 plotCategoryChart.setTitle("Plot Category Distribution");  
 xAxis.setLabel("Category");  
 yAxis.setLabel("Count");  
 XYChart.Series<String, Number> series = new XYChart.Series<>();  
 series.setName("Categories");  
 series.getData().addAll(  
 new XYChart.Data<>("Corner", cornerCount),  
 new XYChart.Data<>("Park-Facing", parkFacingCount),  
 new XYChart.Data<>("Main Boulevard", mainBoulevardCount)  
 );  
 plotCategoryChart.getData().add(series);  
  
 // Layout for charts in a single row (HBox)  
 HBox chartsLayout = new HBox(20); // 20px spacing between charts  
 chartsLayout.setAlignment(Pos.*CENTER*);  
 chartsLayout.getChildren().addAll(plotTypeChart, plotStatusChart);  
  
 Button printButton = new Button("Print Report");  
 printButton.setOnAction(e -> {  
 // Print the entire layout (including charts and text)  
 String contentToPrint = reportTextArea.getText();  
 if (!contentToPrint.isEmpty()) {  
 *print*(contentToPrint); // Print the whole scene including charts and text  
 } else {  
 Alert alert = new Alert(Alert.AlertType.*WARNING*, "No report to print!");  
 alert.show();  
 }  
 });  
  
 // Add a Close Button  
 Button closeButton = new Button("Close");  
 closeButton.setOnAction(e -> {  
 adminDashboardScene(stage);  
 });  
  
 // Combine elements in layout  
 layout.getChildren().addAll(  
 reportTextArea,  
 new Label("Plot Type and Status Distribution"),  
 chartsLayout,  
 new Label("Plot Category Distribution"),  
 plotCategoryChart,  
 printButton,  
 closeButton  
 );  
  
 Scene scene = new Scene(layout, 1300, 800);  
 stage.setScene(scene);  
 stage.setTitle("Generate Report");  
 stage.show();  
 }  
  
 private static void print(String content) {  
  
 PrinterJob printerJob = PrinterJob.*createPrinterJob*();  
  
 if (printerJob == null) {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*, "No printers found. Please check your printer setup.");  
 alert.show();  
 return;  
 }  
  
  
 Text printableContent = new Text(content);  
 printableContent.setWrappingWidth(500);  
  
  
 boolean proceed = printerJob.showPrintDialog(null);  
  
 if (proceed) {  
  
 boolean success = printerJob.printPage(printableContent);  
  
 if (success) {  
 printerJob.endJob();  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*, "Printing complete.");  
 alert.show();  
 } else {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*, "Failed to print.");  
 alert.show();  
 }  
 } else {  
  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*, "Printing cancelled.");  
 alert.show();  
 }  
 }  
  
 public void buyerDashboard(Stage stage, int buyerId) {  
 VBox vBox = new VBox();  
 vBox.setAlignment(Pos.*CENTER*);  
 vBox.setSpacing(15);  
 vBox.setPadding(new Insets(20));  
 vBox.setBackground(background());  
  
 Label titleLabel = new Label("Buyer Dashboard");  
 titleLabel.setFont(new Font("Times New Roman", 55));  
 titleLabel.setPadding(new Insets(20, 30,30, 30));  
 titleLabel.setStyle("-fx-font-weight: bold; -fx-text-fill: white;");  
  
 GridPane functionsGridPane = new GridPane();  
 functionsGridPane.setVgap(15);  
 functionsGridPane.setHgap(15);  
 functionsGridPane.setAlignment(Pos.*CENTER*);  
  
 VBox viewPlotsBox = new VBox();  
 VBox requestPlotBox = new VBox();  
 VBox ownershipDetailsBox = new VBox();  
 VBox paymentBox = new VBox();  
 VBox updateBox = new VBox();  
 VBox mapBox = new VBox();  
  
  
 Button viewPlots = new Button();  
 Button requestPlot = new Button();  
 Button ownershipDetails = new Button();  
 Button trackPaymentStatus = new Button();  
 Button updatePreference = new Button();  
 Button viewMap = new Button();  
  
 viewPlotsBox.setSpacing(10);  
 viewPlotsBox.setAlignment(Pos.*CENTER*);  
 Image viewPlotsLogo = new Image(this.getClass().getResource("/plotsLogo.png").toExternalForm());  
 ImageView viewPlotsImage = new ImageView(viewPlotsLogo);  
 Label userText = new Label("View Available Plots");  
 userText.setFont(new Font("Times New Roman", 18));  
 viewPlotsBox.getChildren().addAll(viewPlotsImage, userText);  
 viewPlots.setPrefSize(500, 300);  
 viewPlots.setStyle("-fx-background-color: lightblue; -fx-border-color: black; -fx-border-width: 2;");  
 viewPlots.setGraphic(viewPlotsBox);  
  
 requestPlotBox.setSpacing(10);  
 requestPlotBox.setAlignment(Pos.*CENTER*);  
 Image requestPlotLogo = new Image(this.getClass().getResource("/requestplotLogo.png").toExternalForm());  
 ImageView requestPlotImage = new ImageView(requestPlotLogo);  
 Label requestPlotText = new Label("Request Plot");  
 requestPlotText.setFont(new Font("Times New Roman", 18));  
 requestPlotBox.getChildren().addAll(requestPlotImage, requestPlotText);  
 requestPlot.setPrefSize(500, 300);  
 requestPlot.setStyle("-fx-background-color: lightblue; -fx-border-color: black; -fx-border-width: 2;");  
 requestPlot.setGraphic(requestPlotBox);  
  
 ownershipDetailsBox.setSpacing(10);  
 ownershipDetailsBox.setAlignment(Pos.*CENTER*);  
 Image ownershipDetailsLogo = new Image(this.getClass().getResource("/ownershipDetailLogo.png").toExternalForm());  
 ImageView ownershipDetailsImage = new ImageView(ownershipDetailsLogo);  
 Label ownershipDetailsText = new Label("Ownership Details");  
 ownershipDetailsText.setFont(new Font("Times New Roman", 18));  
 ownershipDetailsBox.getChildren().addAll(ownershipDetailsImage, ownershipDetailsText);  
 ownershipDetails.setPrefSize(500, 300);  
 ownershipDetails.setStyle("-fx-background-color: lightblue; -fx-border-color: black; -fx-border-width: 2;");  
 ownershipDetails.setGraphic(ownershipDetailsBox);  
  
 paymentBox.setSpacing(10);  
 paymentBox.setAlignment(Pos.*CENTER*);  
 Image paymentLogo = new Image(this.getClass().getResource("/paymentsLogo.png").toExternalForm());  
 ImageView paymentImage = new ImageView(paymentLogo);  
 Label paymentText = new Label("Track Payment Status");  
 paymentText.setFont(new Font("Times New Roman", 18));  
 paymentBox.getChildren().addAll(paymentImage, paymentText);  
 trackPaymentStatus.setPrefSize(500, 300);  
 trackPaymentStatus.setStyle("-fx-background-color: lightblue; -fx-border-color: black; -fx-border-width: 2;");  
 trackPaymentStatus.setGraphic(paymentBox);  
  
 updateBox.setSpacing(10);  
 updateBox.setAlignment(Pos.*CENTER*);  
 Image updateLogo = new Image(this.getClass().getResource("/updateLogo.png").toExternalForm());  
 ImageView updateImage = new ImageView(updateLogo);  
 Label updateText = new Label("Update Preferences");  
 updateText.setFont(new Font("Times New Roman", 18));  
 updateBox.getChildren().addAll(updateImage, updateText);  
 updatePreference.setPrefSize(500, 300);  
 updatePreference.setStyle("-fx-background-color: lightblue; -fx-border-color: black; -fx-border-width: 2;");  
 updatePreference.setGraphic(updateBox);  
  
 mapBox.setSpacing(10);  
 mapBox.setAlignment(Pos.*CENTER*);  
 Image mapLogo = new Image(this.getClass().getResource("/viewMapLogo.png").toExternalForm());  
 ImageView mapImage = new ImageView(mapLogo);  
 Label mapText = new Label("View Map");  
 mapText.setFont(new Font("Times New Roman", 18));  
 mapBox.getChildren().addAll(mapImage, mapText);  
 viewMap.setPrefSize(500, 300);  
 viewMap.setStyle("-fx-background-color: lightblue; -fx-border-color: black; -fx-border-width: 2;");  
 viewMap.setGraphic(mapBox);  
  
 Button logout = new Button("Logout");  
 logout.setStyle("-fx-background-color: red; -fx-text-fill: white; -fx-padding: 10 20; -fx-font-size: 14;");  
 logout.setPrefSize(150, 80);  
  
 viewPlots.setOnAction(e -> viewPlots(stage, buyerId));  
 requestPlot.setOnAction(e -> requestPlot(stage, buyerId));  
 ownershipDetails.setOnAction(e -> ownershipDetails(stage, buyerId));  
 trackPaymentStatus.setOnAction(e -> trackPaymentStatus(stage, buyerId));  
 updatePreference.setOnAction(e -> updatePreference(stage, buyerId));  
  
 viewMap.setOnAction(e -> {  
 stage.setScene(ViewMap.*getMainScene*(stage));  
 });  
  
 logout.setOnAction(e -> {  
 loginScreen(stage);  
 });  
  
 functionsGridPane.add(viewPlots, 0,0);  
 functionsGridPane.add(requestPlot,0,1);  
 functionsGridPane.add(ownershipDetails,0,2);  
 functionsGridPane.add(trackPaymentStatus,1,0);  
 functionsGridPane.add(updatePreference,1,1);  
 functionsGridPane.add(viewMap, 1, 2);  
  
 vBox.getChildren().addAll(titleLabel,  
 functionsGridPane,  
 logout  
 );  
  
  
 Scene scene = new Scene(vBox, 1300, 800);  
 stage.setScene(scene);  
 stage.setTitle("Buyer Dashboard");  
 }  
 private void viewPlots(Stage stage, int buyerId) {  
 VBox vBox = new VBox();  
 vBox.setBackground(background());  
 vBox.setSpacing(20);  
 Label titleLabel = new Label("View Available Plots");  
 titleLabel.setAlignment(Pos.*CENTER*);  
 titleLabel.setPadding(new Insets(100, 50, 50, 100));  
  
 titleLabel.setStyle("-fx-font-weight: bold; -fx-text-fill: white;");  
 titleLabel.setFont(new Font("Times New Roman", 56));  
  
  
 vBox.getChildren().add(titleLabel);  
  
  
 TableView<Plot> plotTable = new TableView<>();  
 ObservableList<Plot> plotData = *loadPlots*();  
  
 if (plotData == null || plotData.isEmpty()) {  
 Label noPlotsLabel = new Label("No available plots");  
 noPlotsLabel.setFont(new Font("Times New Roman", 18));  
 vBox.getChildren().addAll(noPlotsLabel);  
 }  
  
  
 Button exit = new Button("Exit");  
 exit.setPrefSize(100, 60);  
 exit.setAlignment(Pos.*CENTER*);  
 exit.setPadding(new Insets(10));  
 exit.setStyle("-fx-font-size: 14px; -fx-text-fill: white; -fx-background-color: green;");  
 exit.setOnAction(e -> buyerDashboard(stage, buyerId));  
 HBox buttonBox = new HBox(exit);  
 buttonBox.setAlignment(Pos.*CENTER*);  
 buttonBox.setPadding(new Insets(20, 0, 20, 0));  
  
  
  
 TableColumn<Plot, Integer> idColumn = new TableColumn<>("Plot ID");  
 idColumn.setCellValueFactory(new PropertyValueFactory<>("plotId"));  
  
 TableColumn<Plot, String> numberColumn = new TableColumn<>("Plot Number");  
 numberColumn.setCellValueFactory(new PropertyValueFactory<>("plotNumber"));  
  
 TableColumn<Plot, Double> lengthColumn = new TableColumn<>("Length (ft)");  
 lengthColumn.setCellValueFactory(new PropertyValueFactory<>("length"));  
  
 TableColumn<Plot, Double> widthColumn = new TableColumn<>("Width (ft)");  
 widthColumn.setCellValueFactory(new PropertyValueFactory<>("width"));  
  
 TableColumn<Plot, Double> areaColumn = new TableColumn<>("Total Area (Marla)");  
 areaColumn.setCellValueFactory(new PropertyValueFactory<>("totalArea"));  
  
 TableColumn<Plot, String> locationColumn = new TableColumn<>("Location");  
 locationColumn.setCellValueFactory(new PropertyValueFactory<>("location"));  
  
 TableColumn<Plot, String> typeColumn = new TableColumn<>("Plot Type");  
 typeColumn.setCellValueFactory(new PropertyValueFactory<>("plotType"));  
  
 TableColumn<Plot, String> categoryColumn = new TableColumn<>("Plot Category");  
 categoryColumn.setCellValueFactory(new PropertyValueFactory<>("plotCategory"));  
  
 TableColumn<Plot, String> statusColumn = new TableColumn<>("Status");  
 statusColumn.setCellValueFactory(new PropertyValueFactory<>("status"));  
  
 TableColumn<Plot, Double> marlaColumn = new TableColumn<>("Price Per Marla");  
 marlaColumn.setCellValueFactory(new PropertyValueFactory<>("pricePerMarla"));  
  
 TableColumn<Plot, Double> priceColumn = new TableColumn<>("Total Price");  
 priceColumn.setCellValueFactory(new PropertyValueFactory<>("totalPrice"));  
  
 plotTable.getColumns().addAll(idColumn, numberColumn, lengthColumn, widthColumn, areaColumn, locationColumn, typeColumn, categoryColumn, marlaColumn, priceColumn, statusColumn);  
  
  
 FilteredList<Plot> filteredPlots = new FilteredList<>(plotData, plot -> "Available".equals(plot.getStatus()));  
 plotTable.setItems(filteredPlots);  
  
 vBox.getChildren().addAll(plotTable, buttonBox);  
  
  
 Scene scene = new Scene(vBox, 1300, 800);  
 stage.setScene(scene);  
 stage.setTitle("View Plots");  
 }  
  
 public void requestPlot(Stage stage, int buyerId) {  
 VBox vBox = new VBox();  
  
 vBox.setBackground(background());  
 vBox.setAlignment(Pos.*CENTER*);  
 vBox.setSpacing(20);  
  
 Label label = new Label("Request a Plot");  
 label.setStyle("-fx-font-size: 36px; -fx-font-weight: bold; -fx-text-fill: white;");  
  
 HBox plotIdBox = new HBox();  
 plotIdBox.setAlignment(Pos.*CENTER*);  
 plotIdBox.setSpacing(10);  
 TextField plotIdField = new TextField();  
 plotIdField.setPromptText("Enter Plot ID");  
 plotIdField.setStyle("-fx-font-size: 16px; -fx-padding: 10; -fx-background-color: white; -fx-text-fill: black;");  
 plotIdField.setPrefWidth(300);  
 plotIdBox.getChildren().add(plotIdField);  
  
  
 Button submitButton = new Button("Submit");  
 submitButton.setStyle("-fx-font-size: 16px; -fx-text-fill: white; -fx-background-color: green; -fx-padding: 10;");  
 submitButton.setOnAction(e -> {  
 String plotIdText = plotIdField.getText();  
  
 if (plotIdText.isEmpty()) {  
 showAlertMessage(Alert.AlertType.*ERROR*, "Error", "Plot ID cannot be empty.");  
 return;  
 }  
  
 int plotId;  
 try {  
 plotId = Integer.*parseInt*(plotIdText);  
 } catch (NumberFormatException ex) {  
 showAlertMessage(Alert.AlertType.*ERROR*, "Error", "Invalid Plot ID. Please enter a valid number.");  
 return;  
 }  
  
  
 ObservableList<Plot> plots = *loadPlots*();  
 boolean plotFound = false;  
 for (Plot plot : plots) {  
 if (plot.getPlotId() == plotId) {  
 if ("Reserved".equals(plot.getStatus())) {  
 showAlertMessage(Alert.AlertType.*WARNING*, "Plot Reserved", "This plot is already reserved.");  
 return;  
 }  
  
 plot.setStatus("Reserved"); // Reserve the plot  
 *savePlots*(plots); // Save the updated list of plots  
 showAlertMessage(Alert.AlertType.*INFORMATION*, "Success", "Plot request submitted successfully!");  
 plotFound = true;  
 break;  
 }  
 }  
  
 if (!plotFound) {  
 showAlertMessage(Alert.AlertType.*ERROR*, "Error", "Plot ID not found.");  
 }  
 });  
  
  
 Button backButton = new Button("Back");  
 backButton.setStyle("-fx-font-size: 16px; -fx-text-fill: white; -fx-background-color: red; -fx-padding: 10;");  
 backButton.setOnAction(e -> {  
 buyerDashboard(stage, buyerId);  
 });  
  
  
 vBox.getChildren().addAll(label, plotIdBox, submitButton, backButton);  
  
 Scene scene = new Scene(vBox, 1300, 800);  
 stage.setScene(scene);  
 stage.setTitle("Request Plot");  
 }  
  
  
 private void showAlertMessage(Alert.AlertType type, String title, String message) {  
 Alert alert = new Alert(type);  
 alert.setTitle(title);  
 alert.setContentText(message);  
 alert.showAndWait();  
 }  
 public void ownershipDetails(Stage stage, int loggedInBuyerId) {  
 VBox vBox = new VBox();  
 vBox.setAlignment(Pos.*CENTER*);  
 vBox.setSpacing(20);  
  
  
 vBox.setBackground(background());  
  
 Label label = new Label("Ownership Details");  
 label.setStyle("-fx-font-size: 36px; -fx-font-weight: bold; -fx-text-fill: white;");  
  
  
 TableView<Document> tableView = new TableView<>();  
  
  
 TableColumn<Document, Integer> documentIdColumn = new TableColumn<>("Document ID");  
 documentIdColumn.setCellValueFactory(new PropertyValueFactory<>("documentId"));  
 documentIdColumn.setPrefWidth(120);  
 TableColumn<Document, Integer> buyerIdColumn = new TableColumn<>("Buyer ID");  
 buyerIdColumn.setCellValueFactory(new PropertyValueFactory<>("buyerId"));  
 buyerIdColumn.setPrefWidth(120);  
  
 TableColumn<Document, Integer> plotIdColumn = new TableColumn<>("Plot ID");  
 plotIdColumn.setCellValueFactory(new PropertyValueFactory<>("plotId"));  
 plotIdColumn.setPrefWidth(120);  
  
 TableColumn<Document, String> documentTypeColumn = new TableColumn<>("Document Type");  
 documentTypeColumn.setCellValueFactory(new PropertyValueFactory<>("documentType"));  
 documentTypeColumn.setPrefWidth(180);  
 TableColumn<Document, LocalDate> uploadDateColumn = new TableColumn<>("Upload Date");  
 uploadDateColumn.setCellValueFactory(new PropertyValueFactory<>("uploadDate"));  
 uploadDateColumn.setPrefWidth(180);  
  
 tableView.getColumns().addAll(documentIdColumn, buyerIdColumn, plotIdColumn, documentTypeColumn, uploadDateColumn);  
  
  
 List<Document> documents = *loadDocuments*();  
 ObservableList<Document> ownershipDocs = FXCollections.*observableArrayList*();  
  
 for (Document doc : documents) {  
 if ("Ownership".equalsIgnoreCase(doc.getDocumentType()) && doc.getBuyerId() == loggedInBuyerId) {  
 ownershipDocs.add(doc);  
 }  
 }  
  
 if (ownershipDocs.isEmpty()) {  
 showAlertMessage(Alert.AlertType.*INFORMATION*, "No Documents", "No ownership documents found for you.");  
 }  
  
 tableView.setItems(ownershipDocs);  
  
  
 Button backButton = new Button("Back");  
 backButton.setStyle("-fx-font-size: 16px; -fx-text-fill: white; -fx-background-color: red; -fx-padding: 10; -fx-pref-width: 200;");  
 backButton.setOnAction(e -> buyerDashboard(stage, loggedInBuyerId));  
  
 vBox.getChildren().addAll(label, tableView, backButton);  
  
  
 Scene scene = new Scene(vBox, 1300, 800);  
 stage.setScene(scene);  
 stage.setTitle("Ownership Details");  
 }  
  
 public void trackPaymentStatus(Stage stage, int buyerId) {  
 VBox vBox = new VBox();  
  
 vBox.setBackground(background());  
 vBox.setAlignment(Pos.*CENTER*);  
 vBox.setSpacing(20);  
  
 Label label = new Label("Track Payment Status");  
 label.setStyle("-fx-font-size: 36px; -fx-font-weight: bold; -fx-text-fill: white;");  
  
 HBox plotIdBox = new HBox();  
 plotIdBox.setAlignment(Pos.*CENTER*);  
 plotIdBox.setSpacing(10);  
 TextField plotIdField = new TextField();  
 plotIdField.setPromptText("Enter Plot ID");  
 plotIdField.setStyle("-fx-font-size: 16px; -fx-padding: 10; -fx-background-color: white; -fx-text-fill: black;");  
 plotIdField.setPrefWidth(300);  
 plotIdBox.getChildren().add(plotIdField);  
 TableView<Payment> tableView = new TableView<>();  
  
 Button searchButton = new Button("Search Payments");  
 searchButton.setStyle("-fx-font-size: 16px; -fx-text-fill: white; -fx-background-color: green; -fx-padding: 10;");  
 searchButton.setOnAction(e -> {  
 String plotIdText = plotIdField.getText();  
 if (plotIdText.isEmpty()) {  
 showAlertMessage(Alert.AlertType.*ERROR*, "Error", "Please enter a Plot ID.");  
 return;  
 }  
  
 try {  
 int plotId = Integer.*parseInt*(plotIdText);  
  
 List<Payment> paymentList = new ArrayList<>();  
 List<Payment> payments = *loadPayments*();  
 for (Payment payment : payments) {  
 if (payment.getPlotId() == plotId) {  
 paymentList.add(payment);  
 }  
 }  
  
 if (paymentList.isEmpty()) {  
 showAlertMessage(Alert.AlertType.*INFORMATION*, "No Payments Found", "No payment records found for the given Plot ID.");  
 } else {  
 ObservableList<Payment> paymentData = FXCollections.*observableArrayList*(paymentList);  
 tableView.setItems(paymentData);  
 }  
 } catch (NumberFormatException ex) {  
 showAlertMessage(Alert.AlertType.*ERROR*, "Invalid Input", "Please enter a valid Plot ID.");  
 }  
 });  
  
 TableColumn<Payment, Integer> paymentIdColumn = new TableColumn<>("Payment ID");  
 paymentIdColumn.setCellValueFactory(new PropertyValueFactory<>("paymentId"));  
 paymentIdColumn.setPrefWidth(120);  
  
 TableColumn<Payment, Integer> buyerIdColumn = new TableColumn<>("Buyer ID");  
 buyerIdColumn.setCellValueFactory(new PropertyValueFactory<>("buyerId"));  
 buyerIdColumn.setPrefWidth(120);  
 TableColumn<Payment, Integer> plotIdColumn = new TableColumn<>("Plot ID");  
 plotIdColumn.setCellValueFactory(new PropertyValueFactory<>("plotId"));  
 plotIdColumn.setPrefWidth(120);  
 TableColumn<Payment, Double> amountPaidColumn = new TableColumn<>("Amount Paid");  
 amountPaidColumn.setCellValueFactory(new PropertyValueFactory<>("amountPaid"));  
 amountPaidColumn.setPrefWidth(150);  
  
 TableColumn<Payment, Double> outstandingBalanceColumn = new TableColumn<>("Outstanding Balance");  
 outstandingBalanceColumn.setCellValueFactory(new PropertyValueFactory<>("outstandingBalance"));  
 outstandingBalanceColumn.setPrefWidth(150);  
  
 TableColumn<Payment, String> paymentMethodColumn = new TableColumn<>("Payment Method");  
 paymentMethodColumn.setCellValueFactory(new PropertyValueFactory<>("paymentMethod"));  
 paymentMethodColumn.setPrefWidth(150);  
 TableColumn<Payment, LocalDate> paymentDateColumn = new TableColumn<>("Payment Date");  
 paymentDateColumn.setCellValueFactory(new PropertyValueFactory<>("paymentDate"));  
 paymentDateColumn.setPrefWidth(200);  
  
 tableView.getColumns().addAll(paymentIdColumn, buyerIdColumn, plotIdColumn, amountPaidColumn, outstandingBalanceColumn, paymentMethodColumn, paymentDateColumn);  
  
 tableView.setStyle("-fx-font-size: 14px; -fx-padding: 10;");  
  
  
 Button backButton = new Button("Back");  
 backButton.setStyle("-fx-font-size: 16px; -fx-text-fill: white; -fx-background-color: red; -fx-padding: 10;");  
 backButton.setOnAction(e -> buyerDashboard(stage, buyerId));  
  
  
 vBox.getChildren().addAll(label, plotIdBox, searchButton, tableView, backButton);  
  
  
 Scene scene = new Scene(vBox, 1300, 800);  
 stage.setScene(scene);  
 stage.setTitle("Track Payment Status");  
 }  
  
  
 public void updatePreference(Stage stage, int buyerId) {  
 VBox vBox = new VBox();  
  
  
 vBox.setBackground(background());  
 vBox.setAlignment(Pos.*CENTER*);  
 vBox.setSpacing(20);  
  
 Label label = new Label("Update Preferences");  
 label.setStyle("-fx-font-size: 36px; -fx-font-weight: bold; -fx-text-fill: white; -fx-effect: dropshadow(gaussian, black, 10, 0.5, 0, 0);");  
  
 HBox locationBox = new HBox();  
 locationBox.setAlignment(Pos.*CENTER*);  
 locationBox.setSpacing(10);  
 TextField preferredLocationField = new TextField();  
 preferredLocationField.setPromptText("Enter preferred location");  
 preferredLocationField.setStyle("-fx-font-size: 16px; -fx-padding: 10; -fx-background-color: white; -fx-text-fill: black;");  
 preferredLocationField.setPrefWidth(300);  
 locationBox.getChildren().add(preferredLocationField);  
  
 HBox sizeBox = new HBox();  
 sizeBox.setAlignment(Pos.*CENTER*);  
 sizeBox.setSpacing(10);  
 TextField preferredSizeField = new TextField();  
 preferredSizeField.setPromptText("Enter preferred size (in Marla)");  
 preferredSizeField.setStyle("-fx-font-size: 16px; -fx-padding: 10; -fx-background-color: white; -fx-text-fill: black;");  
 preferredSizeField.setPrefWidth(300);  
 sizeBox.getChildren().add(preferredSizeField);  
  
 HBox budgetBox = new HBox();  
 budgetBox.setAlignment(Pos.*CENTER*);  
 budgetBox.setSpacing(10);  
 TextField budgetField = new TextField();  
 budgetField.setPromptText("Enter budget");  
 budgetField.setStyle("-fx-font-size: 16px; -fx-padding: 10; -fx-background-color: white; -fx-text-fill: black;");  
 budgetField.setPrefWidth(300);  
 budgetBox.getChildren().add(budgetField);  
  
  
 Button updateButton = new Button("Update");  
 updateButton.setStyle("-fx-font-size: 16px; -fx-text-fill: white; -fx-background-color: green; -fx-padding: 10;");  
 updateButton.setOnAction(e -> {  
  
 String preferredLocation = preferredLocationField.getText();  
 String preferredSizeText = preferredSizeField.getText();  
 String budgetText = budgetField.getText();  
  
  
 if (preferredLocation.isEmpty() || preferredSizeText.isEmpty() || budgetText.isEmpty()) {  
 showAlertMessage(Alert.AlertType.*ERROR*, "Error", "Please fill all the fields.");  
 return;  
 }  
  
 double preferredSize = 0;  
 double budget = 0;  
  
 try {  
 preferredSize = Double.*parseDouble*(preferredSizeText);  
 budget = Double.*parseDouble*(budgetText);  
 } catch (NumberFormatException ex) {  
 showAlertMessage(Alert.AlertType.*ERROR*, "Error", "Please enter valid numbers for size and budget.");  
 return;  
 }  
  
  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*, "Preferences updated successfully!");  
 alert.showAndWait();  
  
 buyerDashboard(stage, buyerId);  
 });  
  
 Button backButton = new Button("Back");  
 backButton.setStyle("-fx-font-size: 16px; -fx-text-fill: white; -fx-background-color: red; -fx-padding: 10;");  
 backButton.setOnAction(e -> buyerDashboard(stage, buyerId));  
  
  
 vBox.getChildren().addAll(label, locationBox, sizeBox, budgetBox, updateButton, backButton);  
  
 Scene scene = new Scene(vBox, 1300, 800);  
 stage.setScene(scene);  
 stage.setTitle("Update Preferences");  
 }  
  
  
 // Load Users as ObservableList  
 public static ObservableList<User> loadUsers() {  
 ObservableList<User> users = FXCollections.*observableArrayList*();  
 try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream("Users.ser"))) {  
 List<User> userList = (List<User>) inputStream.readObject();  
 users.addAll(userList); // Add all items to ObservableList  
 } catch (FileNotFoundException e) {  
 System.*out*.println("Users file not found. Starting with an empty list.");  
 } catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 return users;  
 }  
  
 // Save Users  
 public static void saveUsers(ObservableList<User> users) {  
 try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream("Users.ser"))) {  
 outputStream.writeObject(new ArrayList<>(users)); // Convert ObservableList to ArrayList  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 // Load Buyers as ObservableList  
 public static ObservableList<Buyer> loadBuyers() {  
 ObservableList<Buyer> buyers = FXCollections.*observableArrayList*();  
 try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream("Buyers.ser"))) {  
 List<Buyer> buyerList = (List<Buyer>) inputStream.readObject();  
 buyers.addAll(buyerList);  
 } catch (FileNotFoundException e) {  
 System.*out*.println("Buyers file not found. Starting with an empty list.");  
 } catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 return buyers;  
 }  
  
 // Save Buyers  
 public static void saveBuyers(ObservableList<Buyer> buyers) {  
 try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream("Buyers.ser"))) {  
 outputStream.writeObject(new ArrayList<>(buyers));  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 // Load Plots as ObservableList  
 public static ObservableList<Plot> loadPlots() {  
 ObservableList<Plot> plots = FXCollections.*observableArrayList*();  
 try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream("Plots.ser"))) {  
 List<Plot> plotList = (List<Plot>) inputStream.readObject();  
 plots.addAll(plotList);  
 } catch (FileNotFoundException e) {  
 System.*out*.println("Plots file not found. Starting with an empty list.");  
 } catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 return plots;  
 }  
  
 // Save Plots  
 public static void savePlots(ObservableList<Plot> plots) {  
 try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream("Plots.ser"))) {  
 outputStream.writeObject(new ArrayList<>(plots));  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 // Load Payments as ObservableList  
 public static ObservableList<Payment> loadPayments() {  
 ObservableList<Payment> payments = FXCollections.*observableArrayList*();  
 try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream("Payments.ser"))) {  
 List<Payment> paymentList = (List<Payment>) inputStream.readObject();  
 payments.addAll(paymentList);  
 } catch (FileNotFoundException e) {  
 System.*out*.println("Payments file not found. Starting with an empty list.");  
 } catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 return payments;  
 }  
  
 // Save Payments  
 public static void savePayments(ObservableList<Payment> payments) {  
 try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream("Payments.ser"))) {  
 outputStream.writeObject(new ArrayList<>(payments));  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 // Load Documents as ObservableList  
 public static ObservableList<Document> loadDocuments() {  
 ObservableList<Document> documents = FXCollections.*observableArrayList*();  
 try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream("Documents.ser"))) {  
 List<Document> documentList = (List<Document>) inputStream.readObject();  
 documents.addAll(documentList);  
 } catch (FileNotFoundException e) {  
 System.*out*.println("Documents file not found. Starting with an empty list.");  
 } catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 return documents;  
 }  
  
 // Save Documents  
 public static void saveDocuments(ObservableList<Document> documents) {  
 try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream("Documents.ser"))) {  
 outputStream.writeObject(new ArrayList<>(documents));  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private Background background(){  
 Image image = new Image(this.getClass().getResource("/LandscapeBackground.jpg").toExternalForm());  
 BackgroundImage backgroundImage = new BackgroundImage(image, BackgroundRepeat.*NO\_REPEAT*, BackgroundRepeat.*NO\_REPEAT*,BackgroundPosition.*CENTER*,new BackgroundSize(100, 100, true, true, true, false));  
 return new Background(backgroundImage);  
 }  
 public String generateReports() {  
 List<Plot> plots = *loadPlots*();  
 List<Payment> payments = *loadPayments*();  
  
 int totalPlots = plots.size();  
 int soldPlots = 0;  
 int availablePlots = 0;  
 double totalRevenue = 0;  
 List<Double> soldPlotAreas = new ArrayList<>();  
  
 for (Plot plot : plots) {  
 if ("Sold".equalsIgnoreCase(plot.getStatus())) {  
 soldPlots++;  
 soldPlotAreas.add(plot.getTotalArea());  
 } else if ("Available".equalsIgnoreCase(plot.getStatus())) {  
 availablePlots++;  
 }  
 }  
  
 for (Payment payment : payments) {  
 totalRevenue += payment.getAmountPaid();  
 }  
  
 double popularArea = mostFrequentArea(soldPlotAreas);  
  
 return String.*format*(  
 "--- Report ---\n" +  
 "Total Plots: %d\n" +  
 "Sold Plots: %d (%.2f%%)\n" +  
 "Available Plots: %d (%.2f%%)\n" +  
 "Popular Plot Area: %.2f sq. meters\n" +  
 "Total Revenue: $%.2f\n",  
 totalPlots, soldPlots, (soldPlots \* 100.0) / totalPlots,  
 availablePlots, (availablePlots \* 100.0) / totalPlots,  
 popularArea, totalRevenue  
 );  
 }  
  
 public String analyzePlotStatistics() {  
 List<Plot> plots = *loadPlots*();  
 int totalPlots = plots.size();  
 int soldPlots = 0;  
 int availablePlots = 0;  
 List<Double> soldPlotAreas = new ArrayList<>();  
  
 for (Plot plot : plots) {  
 if ("Sold".equalsIgnoreCase(plot.getStatus())) {  
 soldPlots++;  
 soldPlotAreas.add(plot.getTotalArea());  
 } else if ("Available".equalsIgnoreCase(plot.getStatus())) {  
 availablePlots++;  
 }  
 }  
  
 double popularArea = mostFrequentArea(soldPlotAreas);  
  
 return String.*format*(  
 "--- Plot Statistics ---\n" +  
 "Total Plots: %d\n" +  
 "Sold Plots: %d (%.2f%%)\n" +  
 "Available Plots: %d (%.2f%%)\n" +  
 "Popular Plot Area: %.2f sq. meters",  
 totalPlots, soldPlots, (soldPlots \* 100.0) / totalPlots,  
 availablePlots, (availablePlots \* 100.0) / totalPlots,  
 popularArea  
 );  
 }  
  
 private double mostFrequentArea(List<Double> areas) {  
 double mostFrequent = 0.0;  
 int maxFrequency = 0;  
  
 for (int i = 0; i < areas.size(); i++) {  
 int frequency = 0;  
 for (int j = 0; j < areas.size(); j++) {  
 if (areas.get(i).equals(areas.get(j))) {  
 frequency++;  
 }  
 }  
 if (frequency > maxFrequency) {  
 maxFrequency = frequency;  
 mostFrequent = areas.get(i);  
 }  
 }  
 return mostFrequent;  
 }  
}