# Project Report

## Parking Management System

**Team Members:**

1. Syed Mohammad Saadaan Hassan (SP22-BCS-003)
2. Jawad Hassan (SP22-BCS-132)

# ABSTRACT IDEA

We are developing a "Parking Management System" that will help the Parking Managers easily manage the vehicle parking and calculate the parking price for different vehicles. The main idea is that it will tell the manager how many parking spaces are available and where to park the vehicle.

## Description:

The project will consist of a login page, Main Screen and Files where the vehicles data will be stored.

- The main screen will be consisting of sections such as "park vehicle, move vehicle, show vehicles, show parking slots, and show parked vehicles history etc."
- There will also be a section to add users who can use the system. Users can be of two types "Admin and Controller." The admin can access all the sections of the system, set the prices for parking, and add other users to the system while the controller will only be able to access the parking section of the system. He will not be able to set the price for parking.

This is the abstract idea and sketch for the system. There may be changes in the final system.

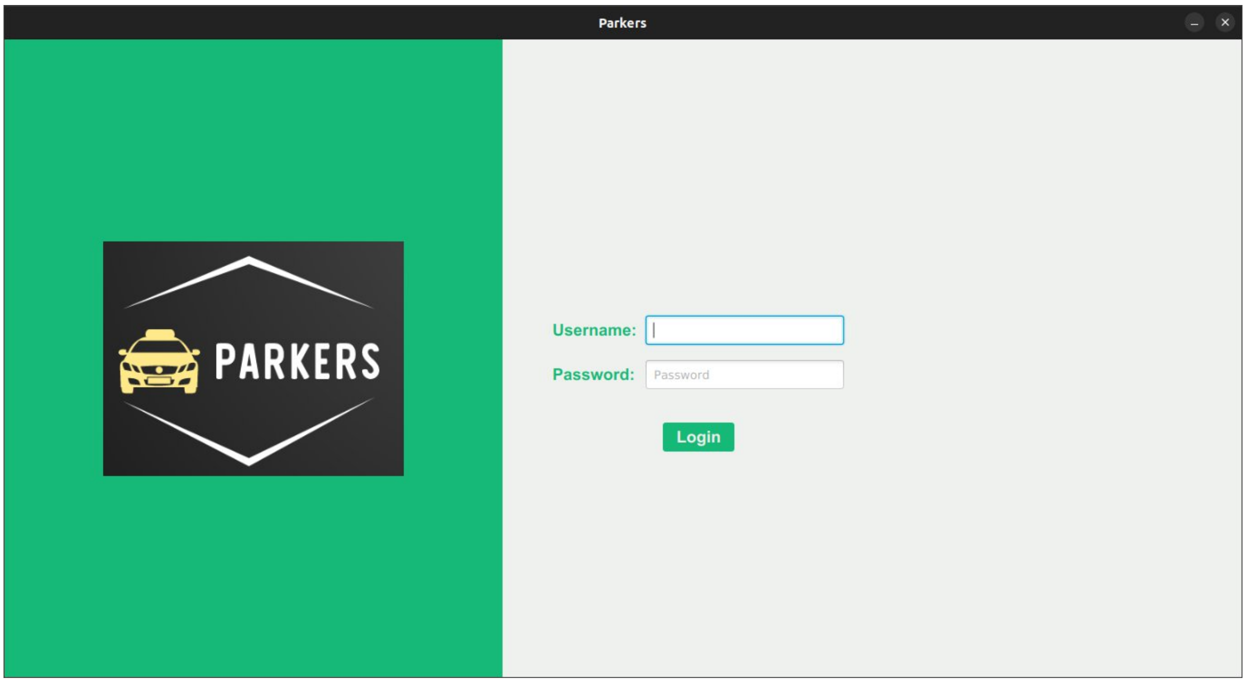
## Team Members:

1. Syed Mohammad Saadaan Hassan (SP22-BCS-003)
2. Jawad Hassan (SP22-BCS-132)

# Implementation

## Login Section:

The program starts with a login section where the user has to enter his user name and password.

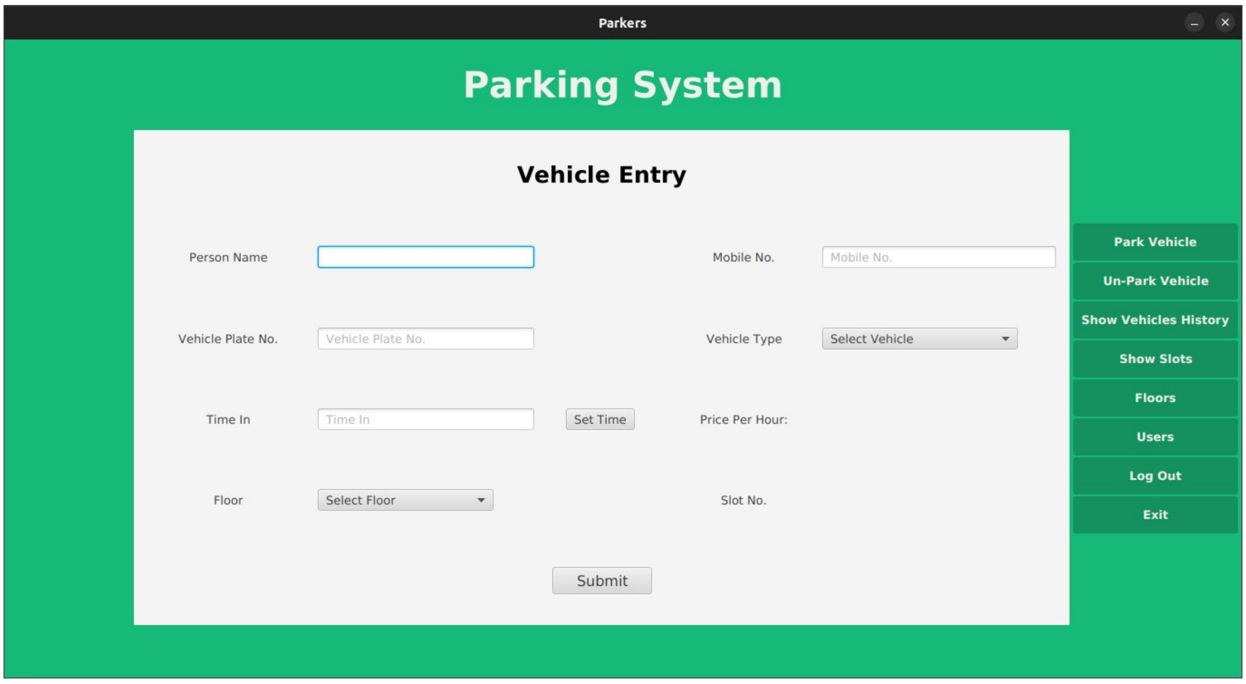


This program contains two types of users:

- Admin
- Controller

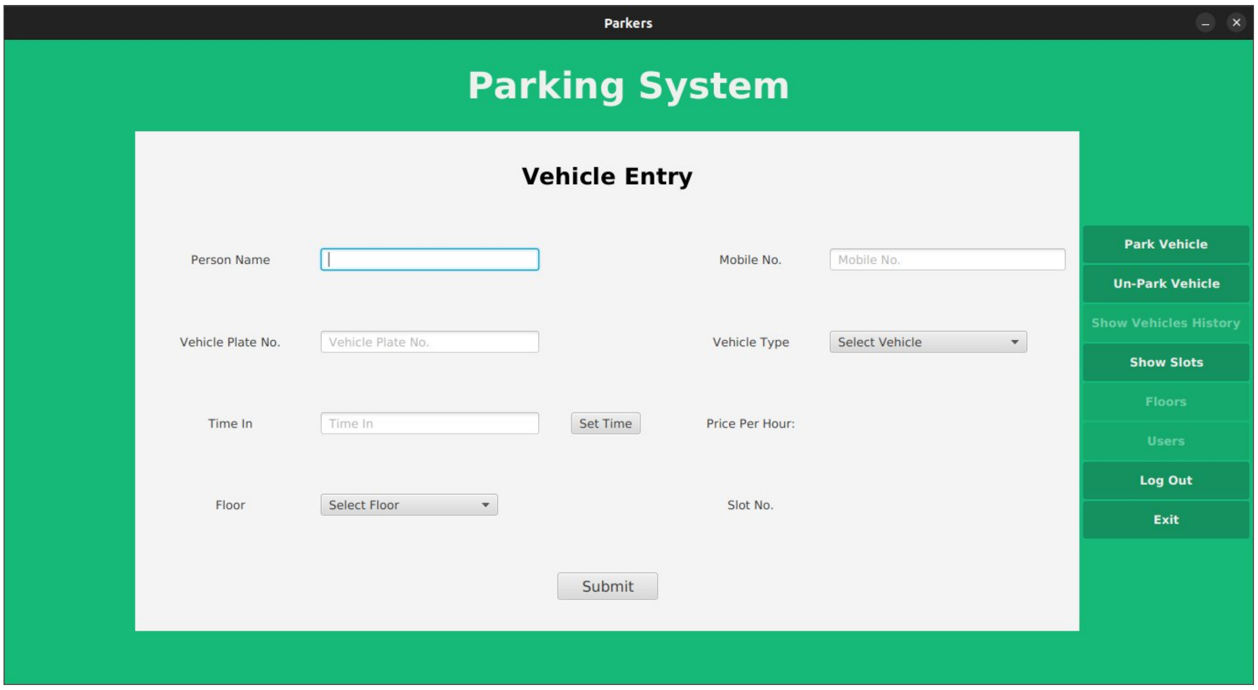
The Admin can access all the section / Areas of the program whereas the Controller can only access the Vehicle's Enter, Vehicle's Unpark and Slots Area. The Admin has the Full control of the program. This Program automatically detects if the logged in user is an Admin or a Controller. On the basis of their role, the program disable and enable the sections for the logged-in user.

If the logged-in user is **Admin** then this window is shown to him.



If the logged-in user is Controller then these options are disabled for him:

- Show Vehicles History
- Floors
- Users



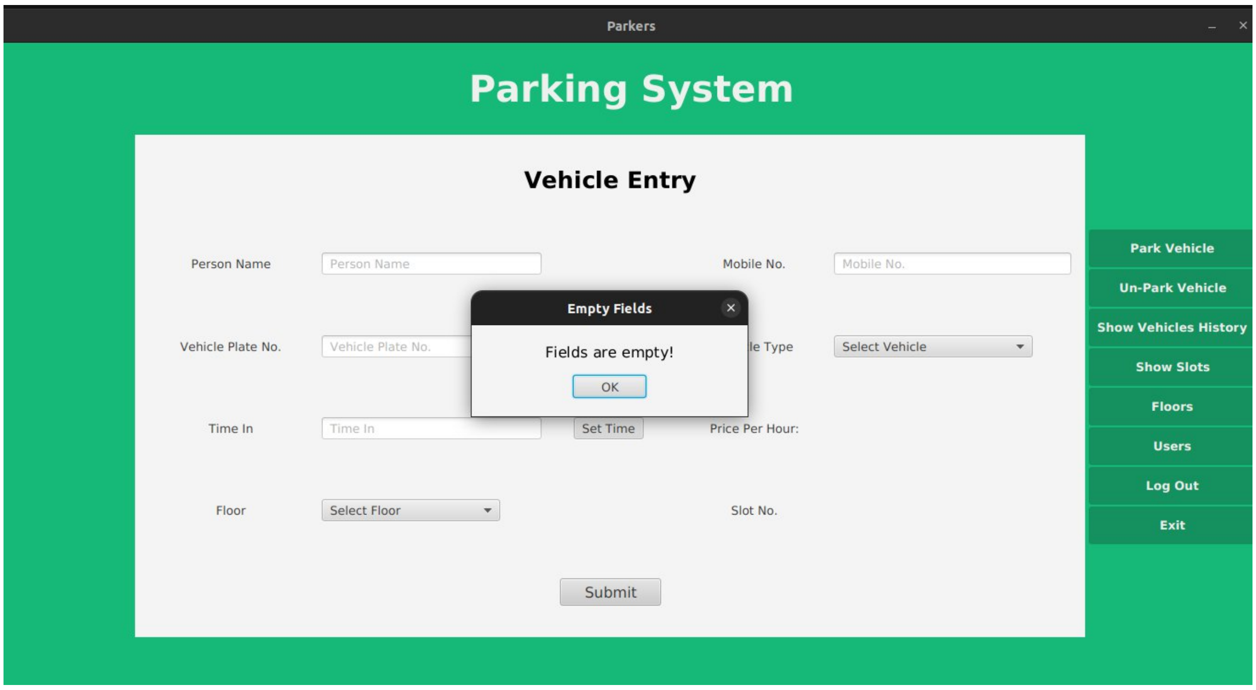
# Parking Section

## Vehicle Entry Area:

This area allow controller/admin to enter the vehicles data which are coming for parking. Here the following details are entered in the system:

- Customer’s name
- Customer’s mobile number
- Vehicle number
- Vehicle type
- Floor / Plot
- Time In

As soon as Floor is selected, the Slot number, which is not reserved is assigned by the computer. The Controller has to enter all the details. If any field is ledt empty then a pop-up appears saying “Fields are empty”.


Parkers

Parking System

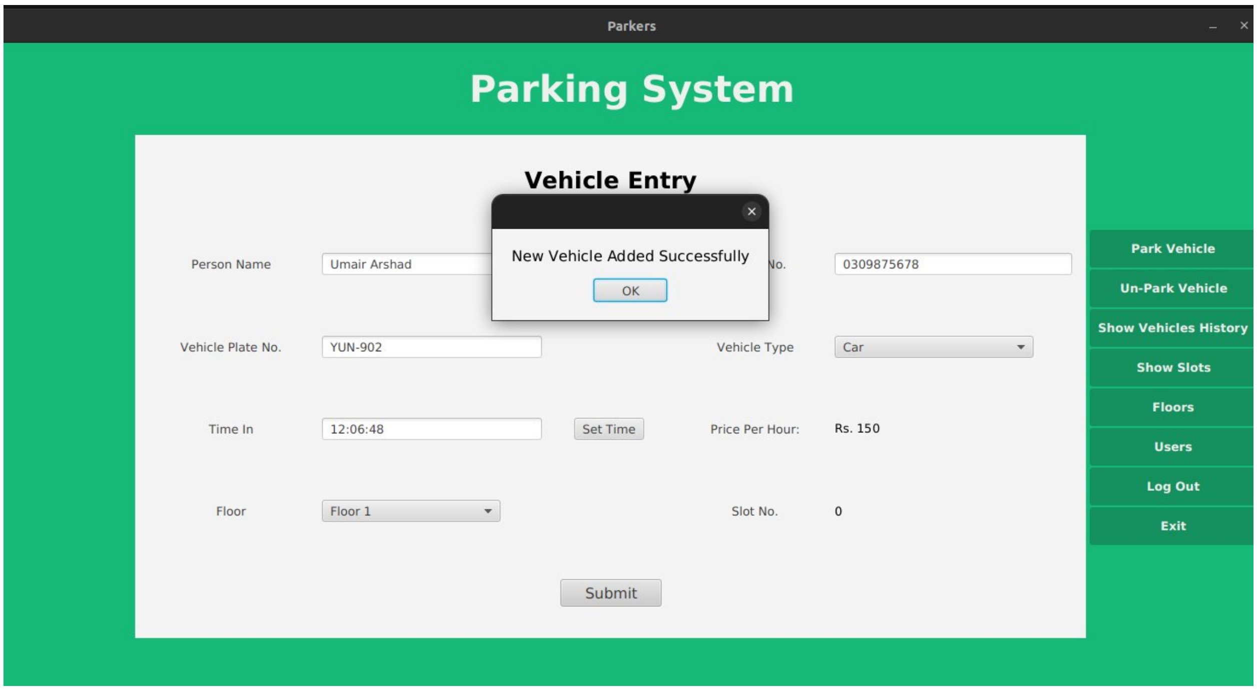
Vehicle Entry

| | | | |
|---|---|---|---|
| Person Name | Person Name | Mobile No. | Mobile No. |
| Vehicle Plate No. | Vehicle Plate No. | le Type | Select Vehicle |
| Time In | Time In | Price Per Hour: | |
| Floor | Select Floor | Slot No. | |

Empty Fields

Fields are empty!

If all the details are entered then that vehicle is entered as parked.


Parkers

Parking System

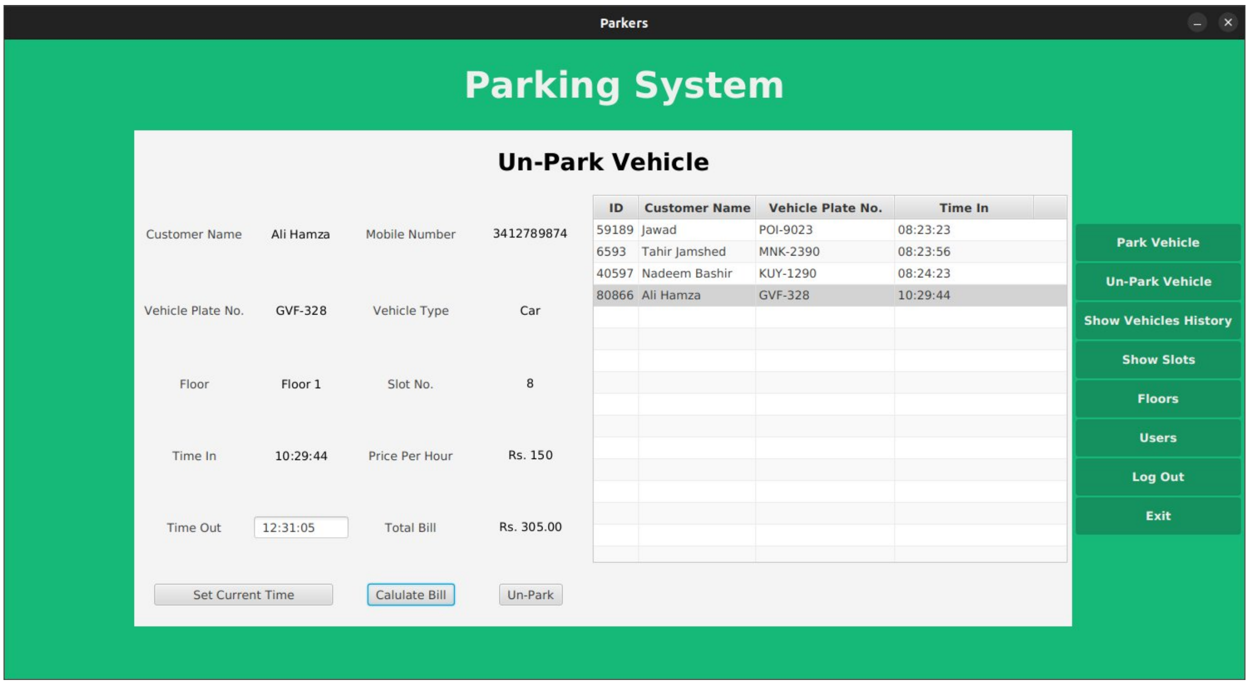
Vehicle Entry

| | | | |
|---|---|---|---|
| Person Name | Umair Arshad | No. | 0309875678 |
| Vehicle Plate No. | YUN-902 | Vehicle Type | Car |
| Time In | 12:06:48 | Price Per Hour: | Rs. 150 |
| Floor | Floor 1 | Slot No. | 0 |

New Vehicle Added Successfully

## Vehicle Unpark:

Here the controller / admin can unpark the vehicle from the assigned slot and floor. To unpark a vehicle, the controller / admin simply select the vehicle from the table and set the Time Out of the vehicle. Then they can calculate the Bill of parking and unpark the vehicle. The vehicle can only be unpark if the bill has been paid.



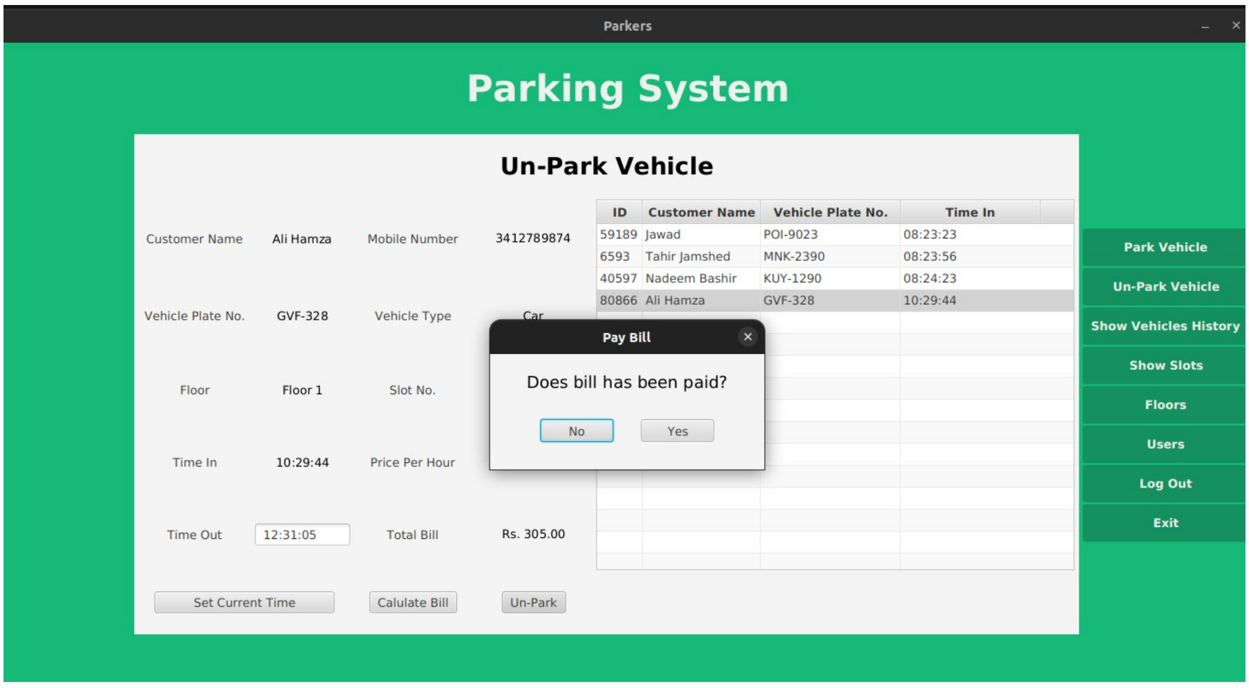
Parkers

Parking System

Un-Park Vehicle

| | | | |
|---|---|---|---|
| Customer Name | Ali Hamza | Mobile Number | 3412789874 |
| Vehicle Plate No. | GVF-328 | Vehicle Type | Car |
| Floor | Floor 1 | Slot No. | 8 |
| Time In | 10:29:44 | Price Per Hour | Rs. 150 |
| Time Out | 12:31:05 | Total Bill | Rs. 305.00 |

| ID | Customer Name | Vehicle Plate No. | Time In |
|---|---|---|---|
| 59189 | Jawad | POI-9023 | 08:23:23 |
| 6593 | Tahir Jamshed | MNK-2390 | 08:23:56 |
| 40597 | Nadeem Bashir | KUY-1290 | 08:24:23 |
| 80866 | Ali Hamza | GVF-328 | 10:29:44 |



Parkers

Parking System

Un-Park Vehicle

| | | | |
|---|---|---|---|
| Customer Name | Ali Hamza | Mobile Number | 3412789874 |
| Vehicle Plate No. | GVF-328 | Vehicle Type | Car |
| Floor | Floor 1 | Slot No. | |
| Time In | 10:29:44 | Price Per Hour | |
| Time Out | 12:31:05 | Total Bill | Rs. 305.00 |

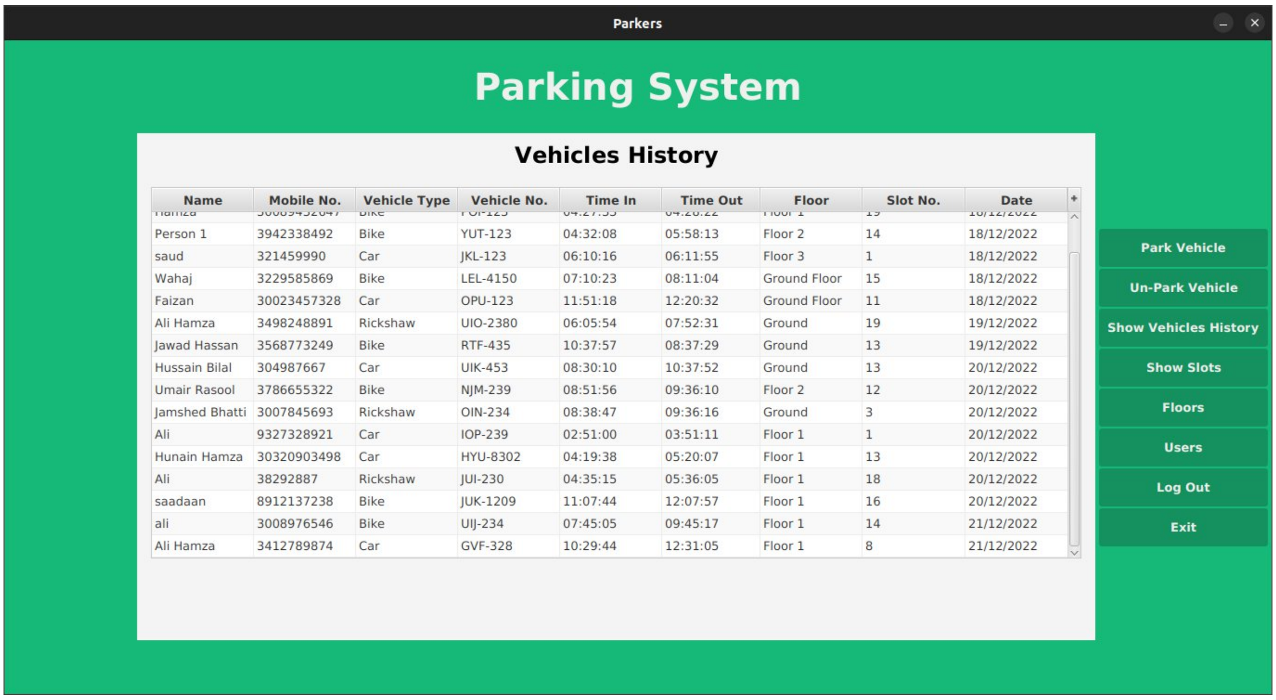
| ID | Customer Name | Vehicle Plate No. | Time In |
|---|---|---|---|
| 59189 | Jawad | POI-9023 | 08:23:23 |
| 6593 | Tahir Jamshed | MNK-2390 | 08:23:56 |
| 40597 | Nadeem Bashir | KUY-1290 | 08:24:23 |
| 80866 | Ali Hamza | GVF-328 | 10:29:44 |

Pay Bill

Does bill has been paid?

## History Area:

After the Vehicle has been unparked from the assigned slot, its history is added in the **"Vehicle's History Table"**. This table contains the history of all the vehicles which have been parked in our Parking Areas. This table can only be accessed by Admins. Controllers cannot access this area.



Vehicles History

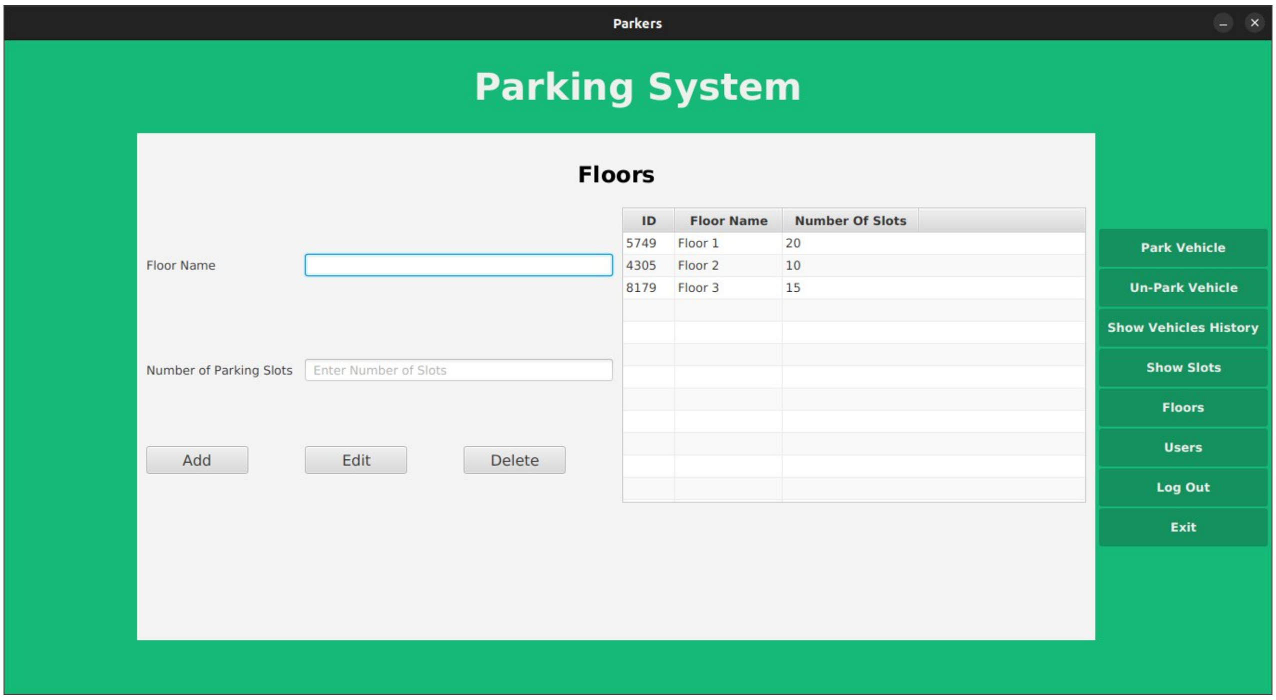
| Name | Mobile No. | Vehicle Type | Vehicle No. | Time In | Time Out | Floor | Slot No. | Date |
|---|---|---|---|---|---|---|---|---|
| Hamza | [illegible] | Bike | [illegible] | [illegible] | [illegible] | Floor 1 | [illegible] | [illegible] |
| Person 1 | 3942338492 | Bike | YUT-123 | 04:32:08 | 05:58:13 | Floor 2 | 14 | 18/12/2022 |
| saud | 321459990 | Car | JKL-123 | 06:10:16 | 06:11:55 | Floor 3 | 1 | 18/12/2022 |
| Wahaj | 3229585869 | Bike | LEL-4150 | 07:10:23 | 08:11:04 | Ground Floor | 15 | 18/12/2022 |
| Faizan | 30023457328 | Car | OPU-123 | 11:51:18 | 12:20:32 | Ground Floor | 11 | 18/12/2022 |
| Ali Hamza | 3498248891 | Rickshaw | UIO-2380 | 06:05:54 | 07:52:31 | Ground | 19 | 19/12/2022 |
| Jawad Hassan | 3568773249 | Bike | RTF-435 | 10:37:57 | 08:37:29 | Ground | 13 | 19/12/2022 |
| Hussain Bilal | 304987667 | Car | UIK-453 | 08:30:10 | 10:37:52 | Ground | 13 | 20/12/2022 |
| Umair Rasool | 3786655322 | Bike | NJM-239 | 08:51:56 | 09:36:10 | Floor 2 | 12 | 20/12/2022 |
| Jamshed Bhatti | 3007845693 | Rickshaw | OIN-234 | 08:38:47 | 09:36:16 | Ground | 3 | 20/12/2022 |
| Ali | 9327328921 | Car | IOP-239 | 02:51:00 | 03:51:11 | Floor 1 | 1 | 20/12/2022 |
| Hunain Hamza | 30320903498 | Car | HYU-8302 | 04:19:38 | 05:20:07 | Floor 1 | 13 | 20/12/2022 |
| Ali | 38292887 | Rickshaw | JUI-230 | 04:35:15 | 05:36:05 | Floor 1 | 18 | 20/12/2022 |
| saadaan | 8912137238 | Bike | JUK-1209 | 11:07:44 | 12:07:57 | Floor 1 | 16 | 20/12/2022 |
| ali | 3008976546 | Bike | UIJ-234 | 07:45:05 | 09:45:17 | Floor 1 | 14 | 21/12/2022 |
| Ali Hamza | 3412789874 | Car | GVF-328 | 10:29:44 | 12:31:05 | Floor 1 | 8 | 21/12/2022 |

## **Floors Area:**

In Floors area the total number of floors are shown to the Admin. Controller cannot access this area. Here the Admin can:

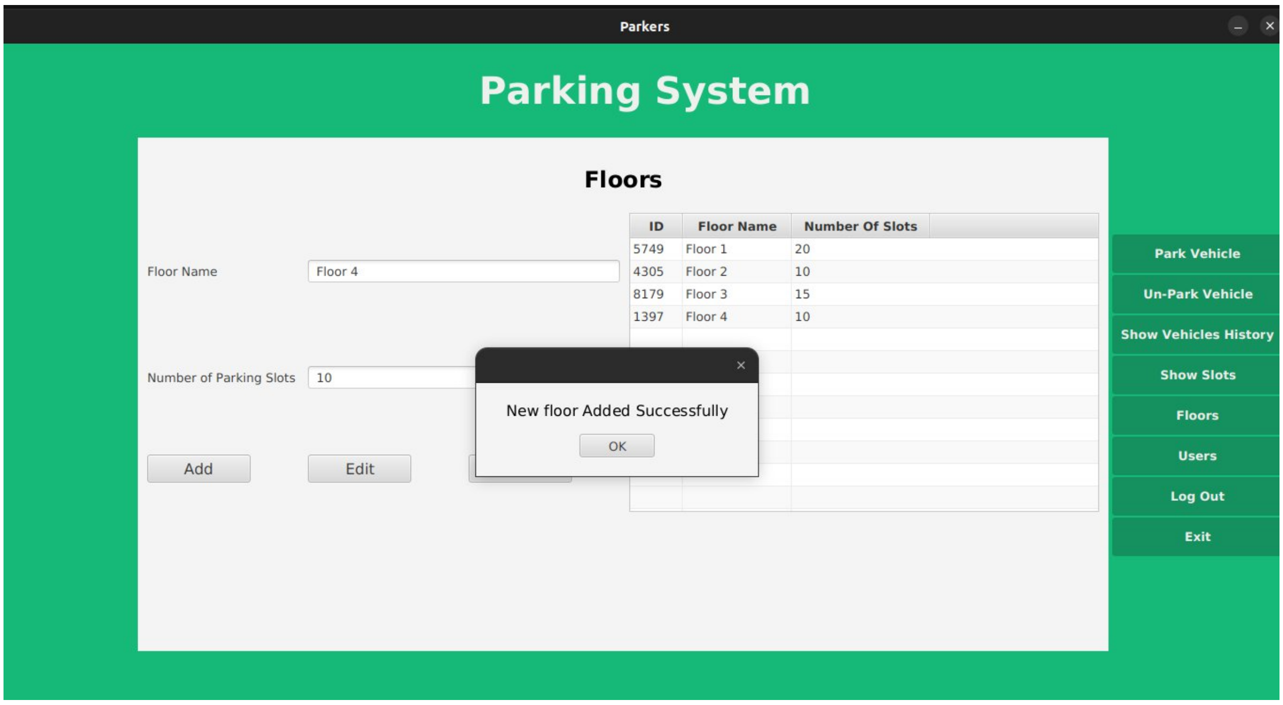
- Add a new Floor
- Edit an existing Floor Information
- Delete a Floor

A table on the side shows already existing floors info.



| ID | Floor Name | Number Of Slots |
|---|---|---|
| 5749 | Floor 1 | 20 |
| 4305 | Floor 2 | 10 |
| 8179 | Floor 3 | 15 |

To add a new floor the admin has to enter the **Floor name** and **number of slots available** on that floor.



Floors

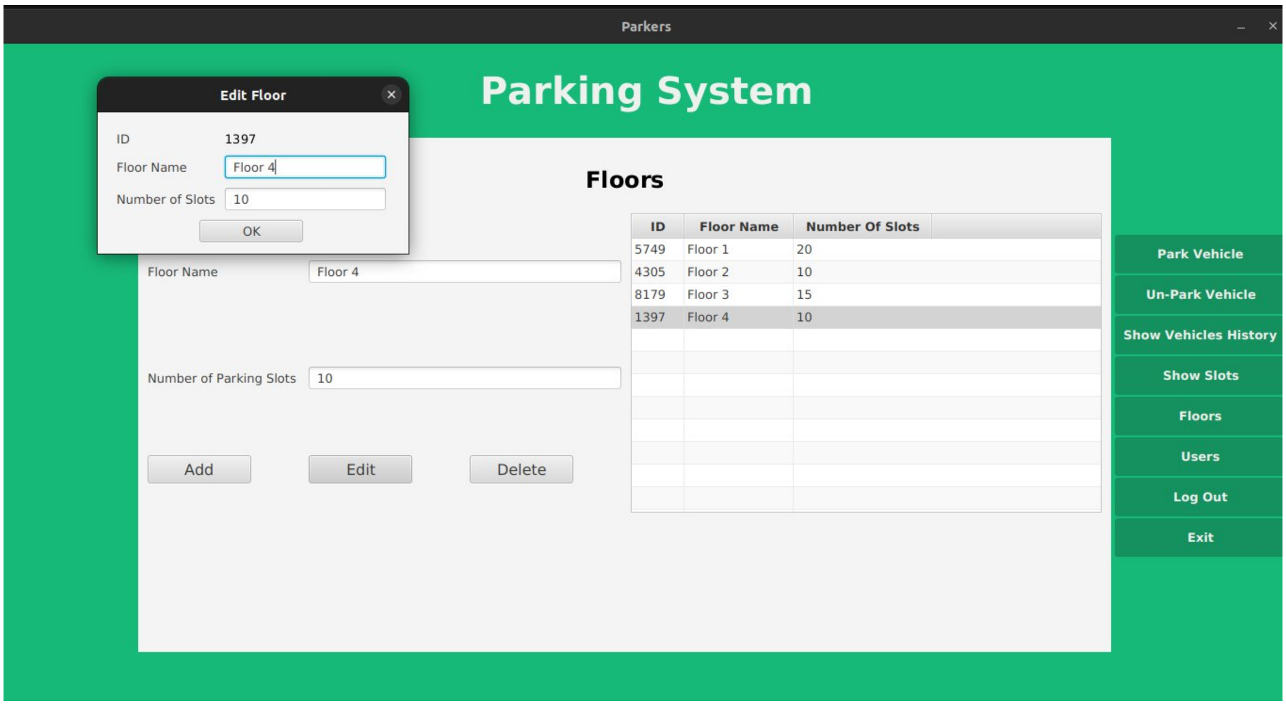
| ID | Floor Name | Number Of Slots |
|---|---|---|
| 5749 | Floor 1 | 20 |
| 4305 | Floor 2 | 10 |
| 8179 | Floor 3 | 15 |
| 1397 | Floor 4 | 10 |

Floor Name: Floor 4

Number of Parking Slots: 10

New floor Added Successfully

To edit a floor, the Admin has to select the Floor from the table and then click **Edit Button**. A new window appears where he can edit the floors info and then press OK. The floors info will be modified.



Edit Floor

ID: 1397

Floor Name: Floor 4

Number of Slots: 10

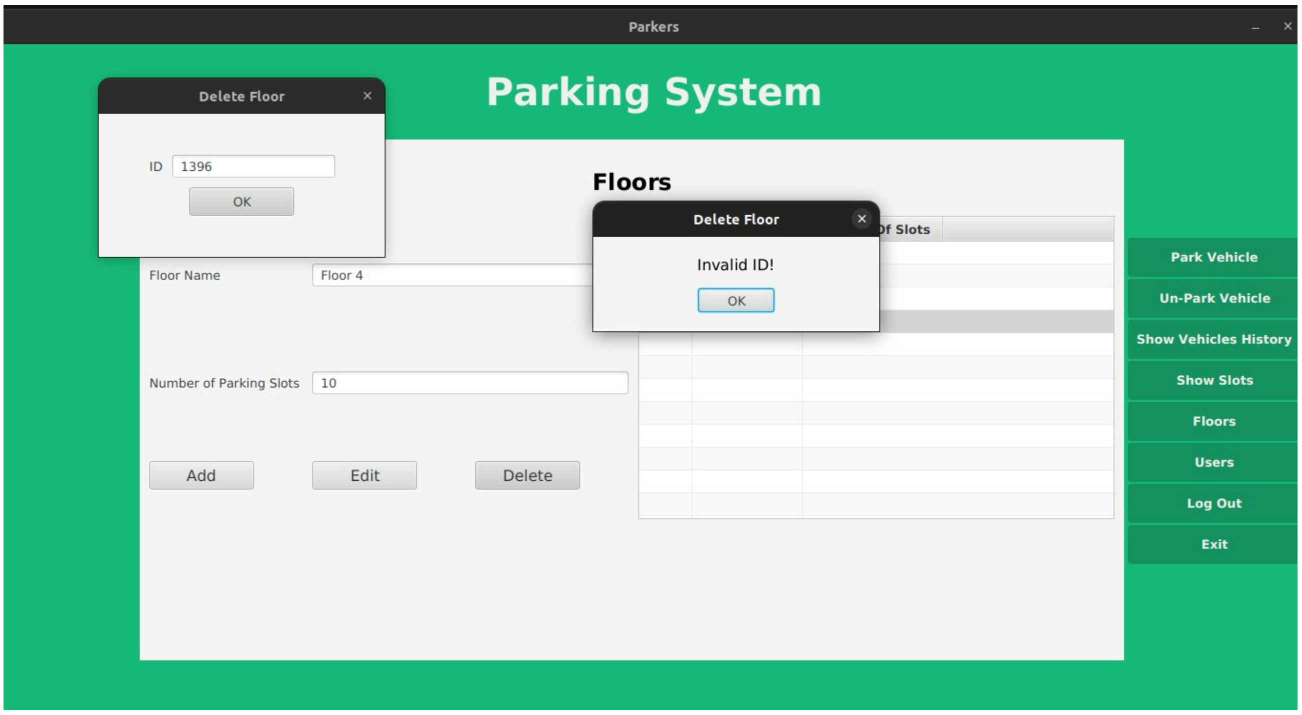
Floors

| ID | Floor Name | Number Of Slots |
|---|---|---|
| 5749 | Floor 1 | 20 |
| 4305 | Floor 2 | 10 |
| 8179 | Floor 3 | 15 |
| 1397 | Floor 4 | 10 |

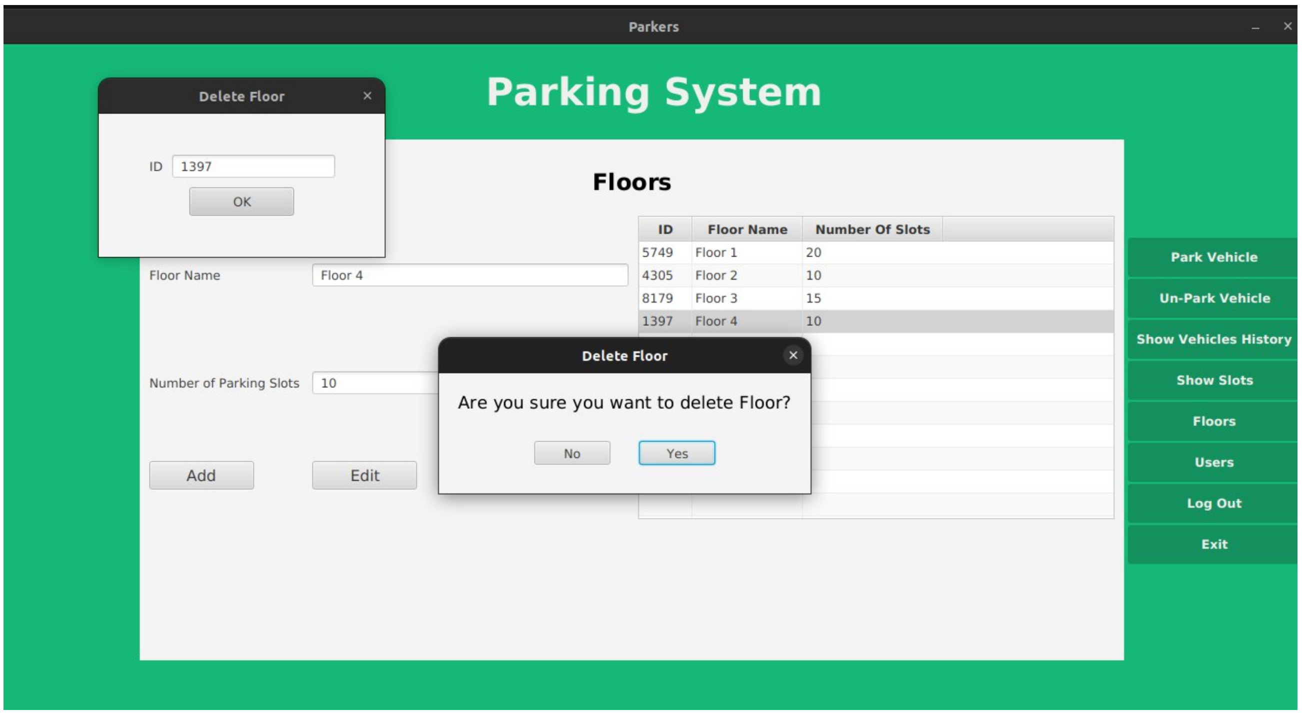
Floor Name: Floor 4

Number of Parking Slots: 10

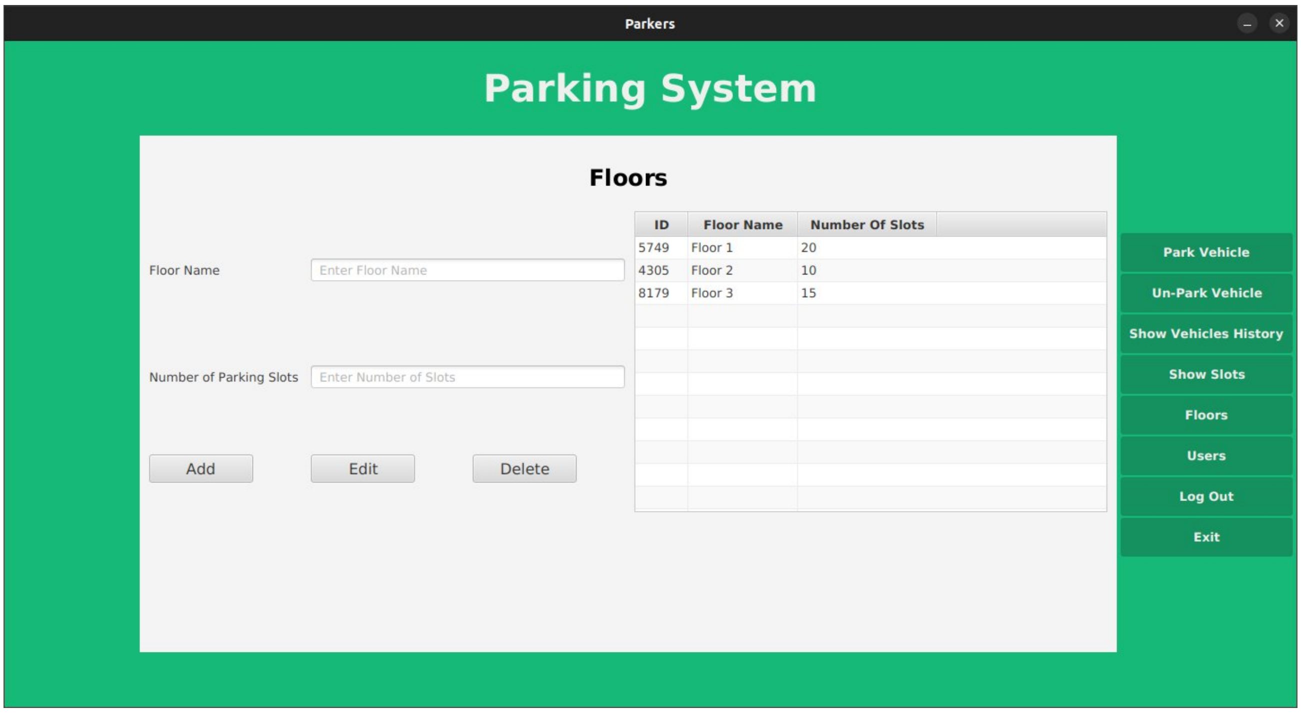
To delete a floor, the Admin has to click **Delete button**. A new window appears where he has to enter the floors ID which he wants to delete and press OK. If the ID is valid then floor will be deleted, otherwise a pop-up window appears saying invalid ID.



If the ID is valid then that floor will be deleted.



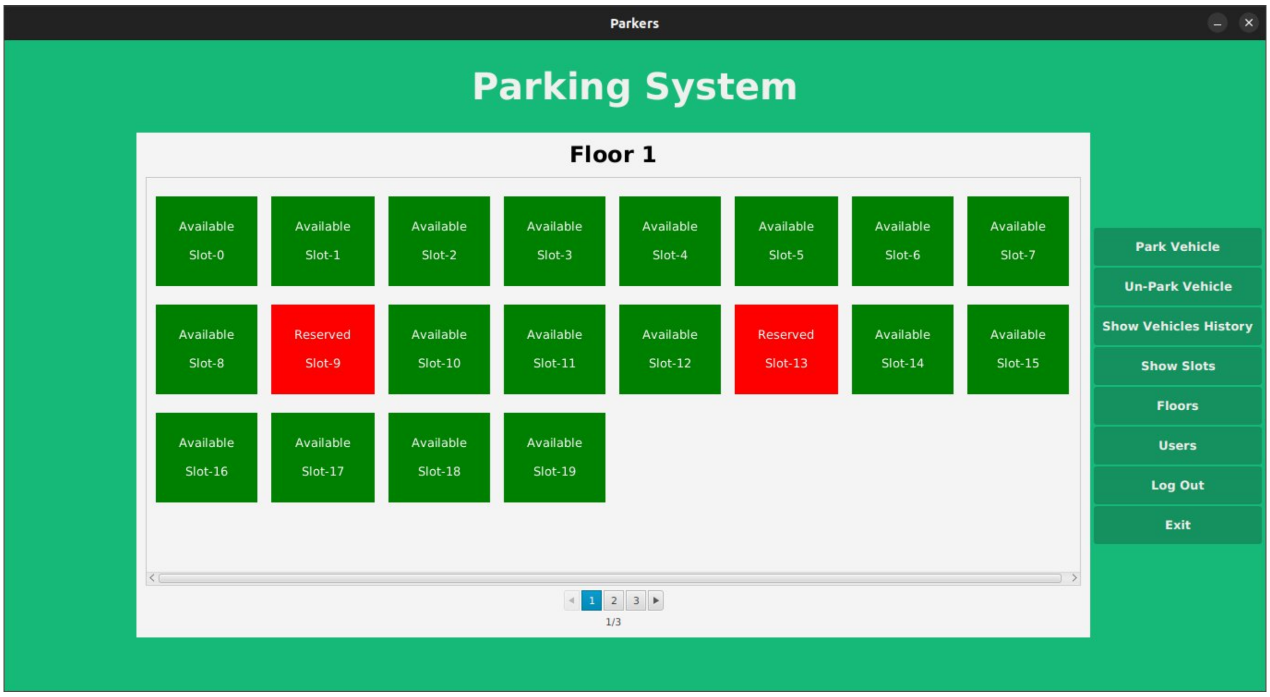
| ID | Floor Name | Number Of Slots |
|---|---|---|
| 5749 | Floor 1 | 20 |
| 4305 | Floor 2 | 10 |
| 8179 | Floor 3 | 15 |
| 1397 | Floor 4 | 10 |



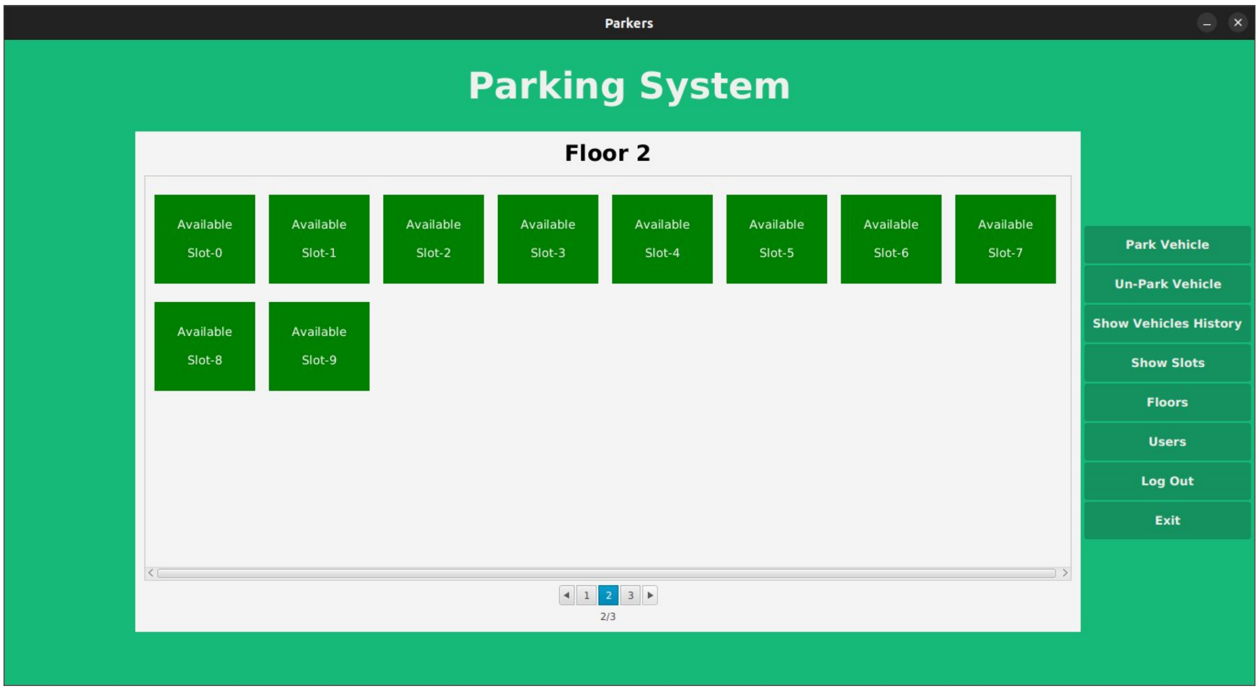
| ID | Floor Name | Number Of Slots |
|---|---|---|
| 5749 | Floor 1 | 20 |
| 4305 | Floor 2 | 10 |
| 8179 | Floor 3 | 15 |

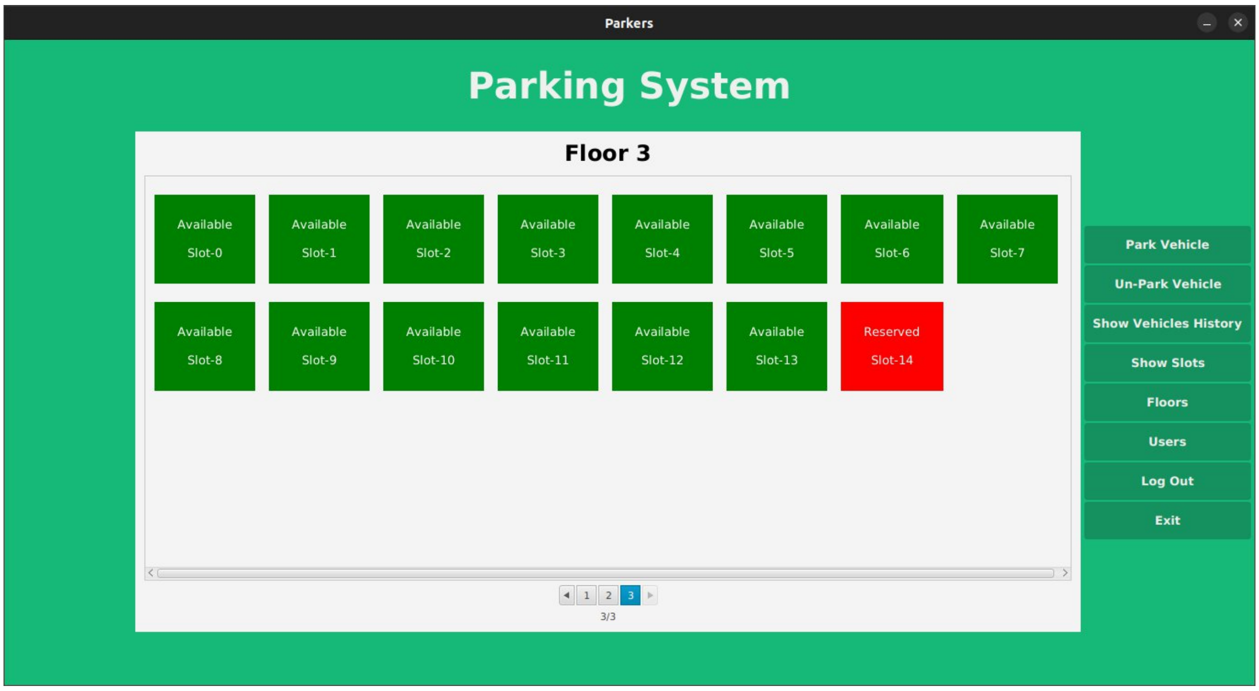
## **<u>Slots Area:</u>**

All the slots are showed in Slots Area in the Program. This Area can be accessed by both Controller and Admin. Here the slots which are reserved are shown in **Red Color** and available slots are shown in **Green Color**.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Available Slot-0 | Available Slot-1 | Available Slot-2 | Available Slot-3 | Available Slot-4 | Available Slot-5 | Available Slot-6 | Available Slot-7 |
| Available Slot-8 | Reserved Slot-9 | Available Slot-10 | Available Slot-11 | Available Slot-12 | Reserved Slot-13 | Available Slot-14 | Available Slot-15 |
| Available Slot-16 | Available Slot-17 | Available Slot-18 | Available Slot-19 | | | | |



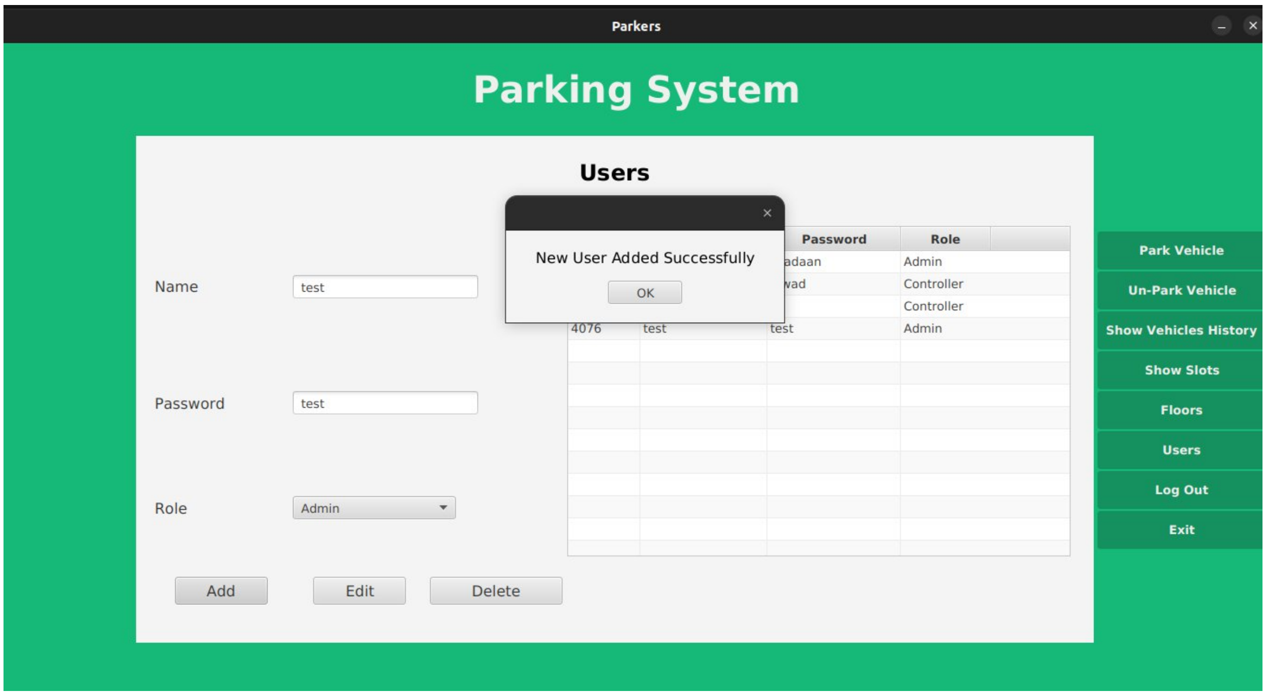


When a new vehicle is parked at a specific slot number, that slot number is marked as red. When that vehicle is unparked, the slot is again marked as green showing that it is available for parking.

## **Users Area:**

This Area can only be accessed by Admin. Here he can add new users as Admin or Controller. A table on the side shows already existing floors info.

To add a new user simply enter new user name, password and select the role of the new user to be as Admin or Controller. Then click the **Add button.**



New User Added Successfully

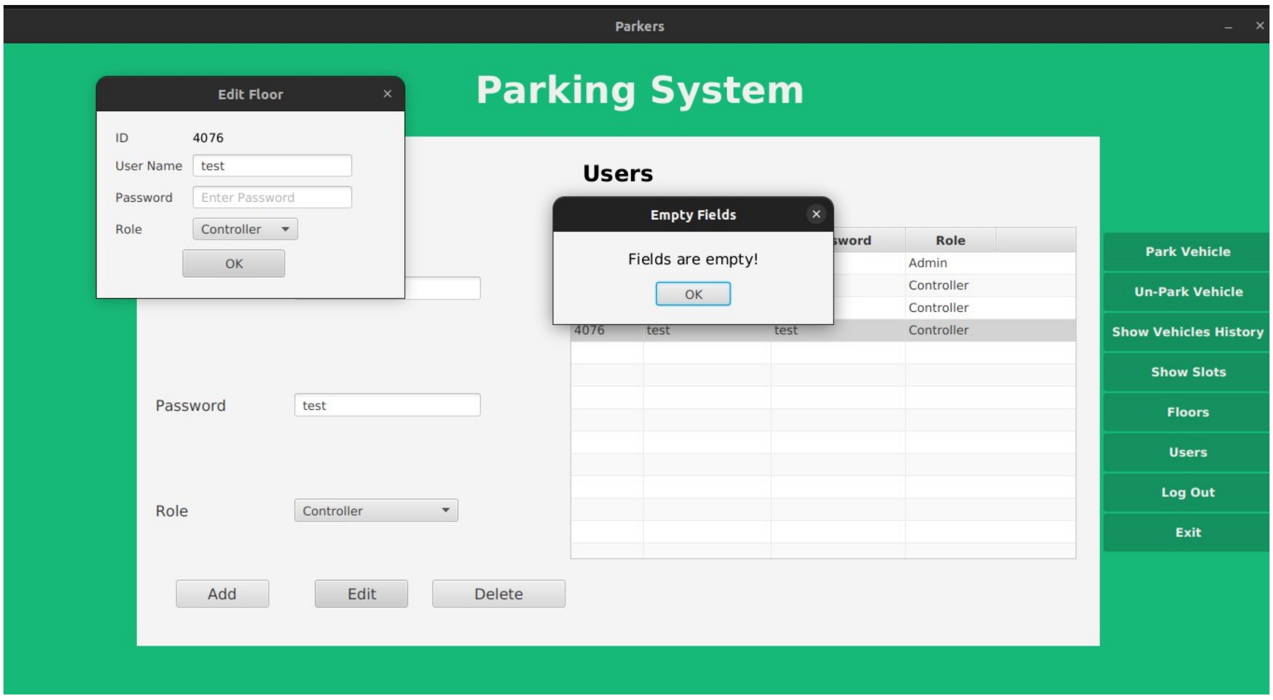
Name: test

Password: test

Role: Admin

|  |  | Password | Role |
|---|---|---|---|
|  |  | adaan | Admin |
|  |  | wad | Controller |
|  |  |  | Controller |
| 4076 | test | test | Admin |

To edit a user's info, the Admin has to select the user from the table and then click **Edit Button**. A new window appears where he can edit the user's info and then press OK. The user's info will be modified. If any field is set to empty then a pop-up appears saying "Fields are empty".



Edit Floor

ID: 4076

User Name: test

Password: Enter Password

Role: Controller

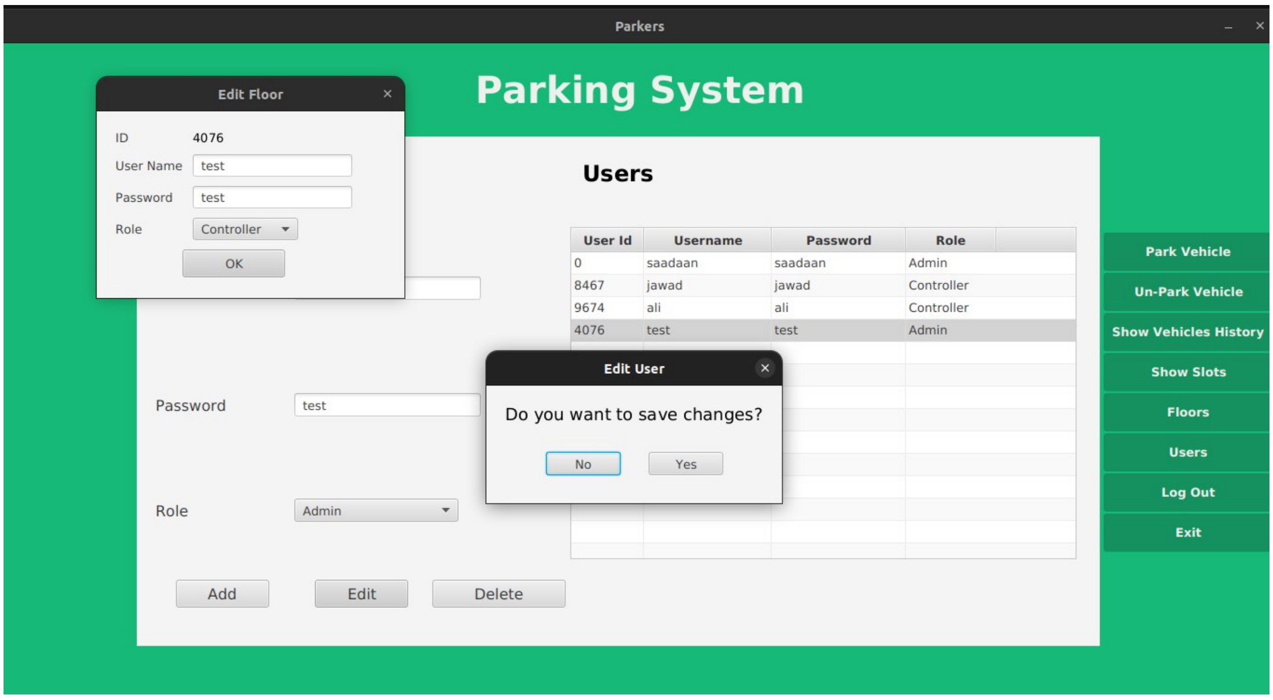
Empty Fields

Fields are empty!

Password: test

Role: Controller

|  |  | word | Role |
|---|---|---|---|
|  |  |  | Admin |
|  |  |  | Controller |
|  |  |  | Controller |
| 4076 | test | test | Controller |



Edit Floor

|  |  |
|---|---|
| ID | 4076 |
| User Name | test |
| Password | test |
| Role | Controller |

OK

Users

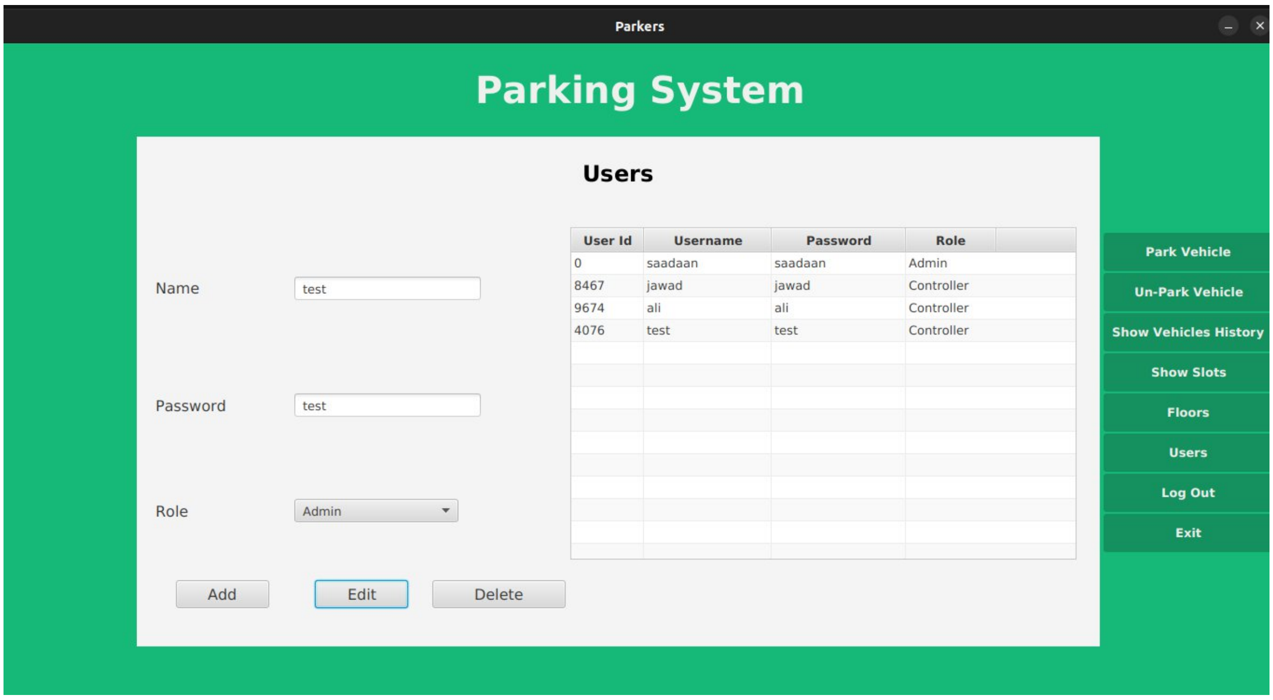
| User Id | Username | Password | Role |
|---|---|---|---|
| 0 | saadaan | saadaan | Admin |
| 8467 | jawad | jawad | Controller |
| 9674 | ali | ali | Controller |
| 4076 | test | test | Admin |

|  |  |
|---|---|
| Password | test |
| Role | Admin |

Edit User

Do you want to save changes?

No Yes

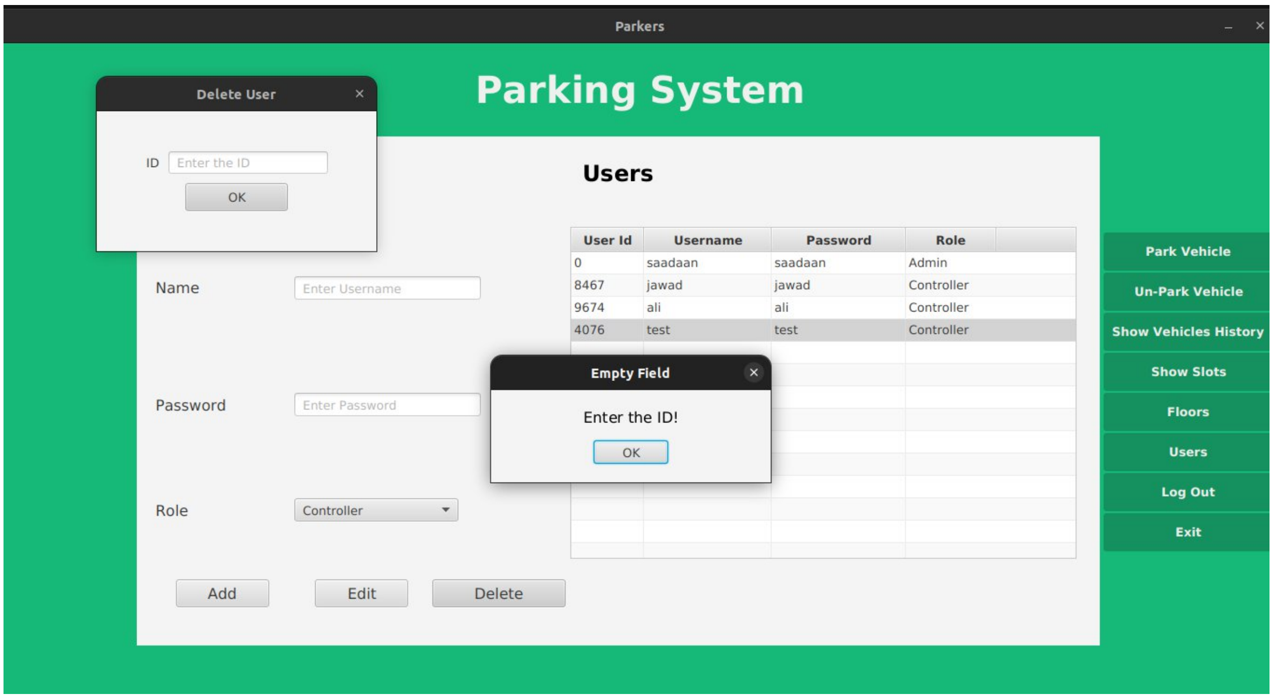


Users

|  |  |
|---|---|
| Name | test |
| Password | test |
| Role | Admin |

| User Id | Username | Password | Role |
|---|---|---|---|
| 0 | saadaan | saadaan | Admin |
| 8467 | jawad | jawad | Controller |
| 9674 | ali | ali | Controller |
| 4076 | test | test | Controller |

To delete a user, the Admin has to click **Delete button**. A new window appears where he has to enter the user ID which he wants to delete and press OK. If the ID is valid then floor will be deleted, otherwise a pop-up window appears saying invalid ID.



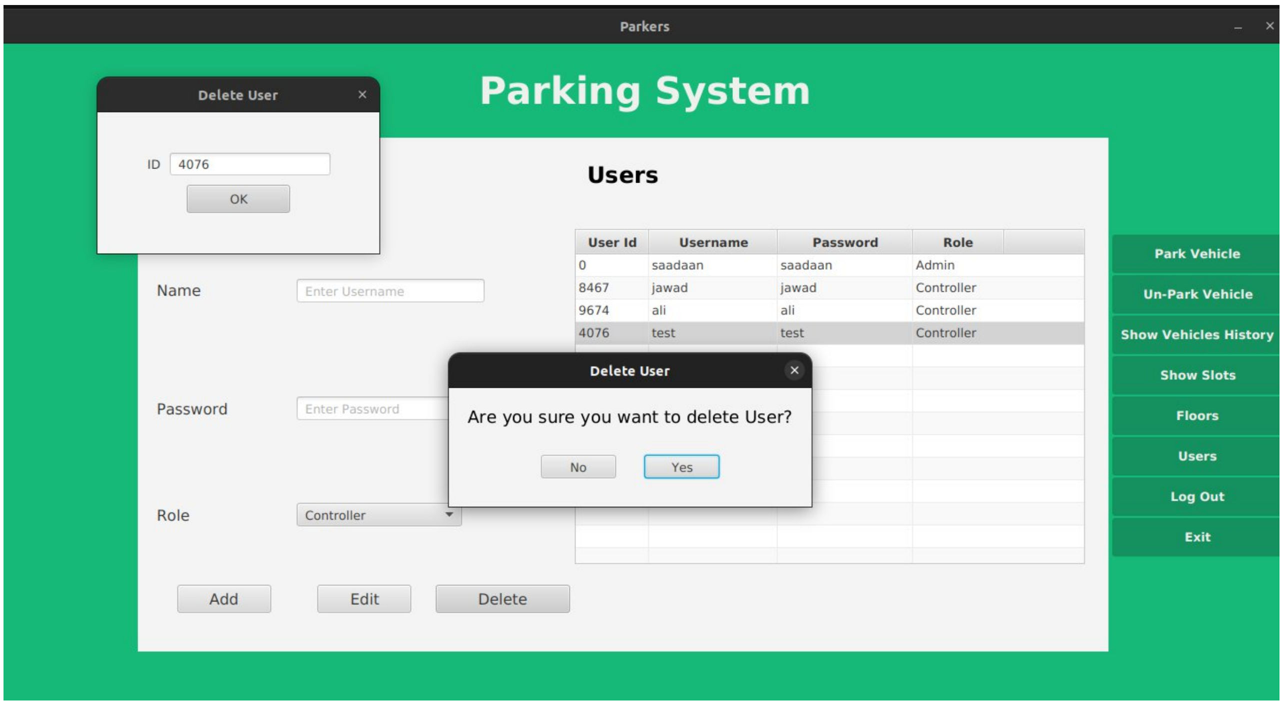
| User Id | Username | Password | Role |
|---|---|---|---|
| 0 | saadaan | saadaan | Admin |
| 8467 | jawad | jawad | Controller |
| 9674 | ali | ali | Controller |
| 4076 | test | test | Controller |

Empty Field

Enter the ID!

OK

If the ID is valid then that floor will be deleted.

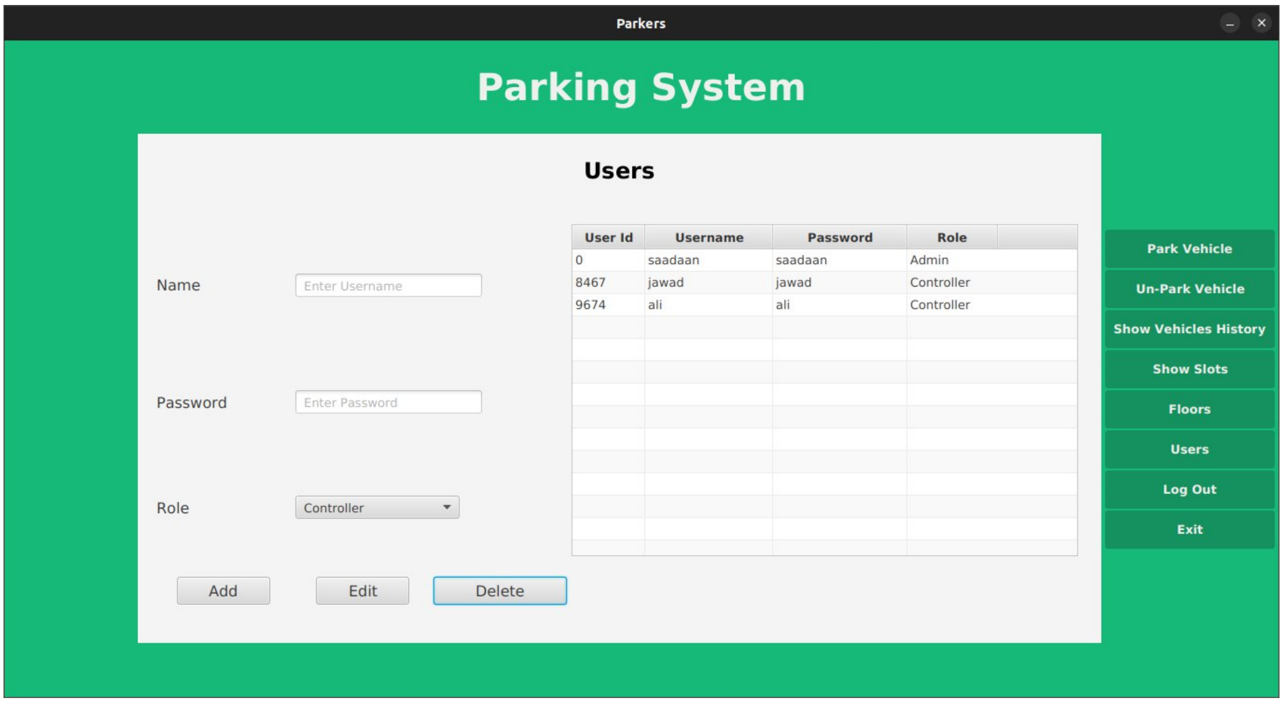


| User Id | Username | Password | Role |
|---|---|---|---|
| 0 | saadaan | saadaan | Admin |
| 8467 | jawad | jawad | Controller |
| 9674 | ali | ali | Controller |
| 4076 | test | test | Controller |

Delete User

Are you sure you want to delete User?

No Yes



Users

Name: Enter Username

Password: Enter Password

Role: Controller

| User Id | Username | Password | Role |
|---|---|---|---|
| 0 | saadaan | saadaan | Admin |
| 8467 | jawad | jawad | Controller |
| 9674 | ali | ali | Controller |

## Code:

Here is a part of our code which is has been written to develop this program.

```java
package com.Project;

import javafx.application.Platform;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.*;
import javafx.scene.text.Text;

import java.io.*;
import java.net.URL;
import java.util.ArrayList;
import java.util.ResourceBundle;

public class SystemController implements Initializable, Serializable {
    @FXML
    private Text floorName;
    @FXML
    private Button showVehicleHistoryBtn;
    @FXML
    private Button floorsBtn;
    @FXML
    private VBox vehicleHistoryPane;
    @FXML
    private TableView<Vehicle> tbVehicleHistory;
    @FXML
    private TableColumn<Vehicle, String> customerNameCol;
    @FXML
    private TableColumn<Vehicle, Long> mobileNoCol;
    @FXML
    private TableColumn<Vehicle, String> vehicleTypeCol;
    @FXML
    private TableColumn<Vehicle, String> vehicleNoCol;
    @FXML
    private TableColumn<Vehicle, String> timeInCol;
    @FXML
    private TableColumn<Vehicle, String> timeOutCol;
    @FXML
    private TableColumn<Vehicle, String> floorCol;
    @FXML
    private TableColumn<Vehicle, Integer> slotNoCol;
    @FXML
    private TableColumn<Vehicle, String> dateCol;
    @FXML
    private Button userBtn;
    @FXML
    private Text tCustomerName;
    @FXML
    private Text tMobileNo;
    @FXML
    private Text tVehicleType;
    @FXML
    private Text tVehiclePlateNo;
    @FXML
    private Text tTimeIn;
    @FXML
    private Text tFloor;
    @FXML
    private Text tTotalBill;
    @FXML
    private Text tPricePerHour;
    @FXML
    private TableView<Vehicle> tbUnparkVehicle;
    @FXML
    private TableColumn<Vehicle, Integer> vehicleIdCol;
    @FXML
    private TableColumn<Vehicle, String> vehiclePlateNoCol;
    @FXML
    private TableColumn<Vehicle, String> vehicleCustomerCol;
    @FXML
    private TableColumn<Vehicle, String> vehicleTimeInCol;
    @FXML
    private TextField tfTimeOut;
    @FXML
    private GridPane unparkVehiclePane;
    @FXML
    private Text tUnparkSlotNo;
    @FXML
    private ComboBox<String> cbFloor;
    @FXML
    private Text tvehiclePrice;
    @FXML
    private TextField tfTimeIn;
    @FXML
    private ComboBox<String> cbVehicleType;
    @FXML
    private TextField tfVehiclePlateNo;
    @FXML
    private TextField tfPersonName;
    @FXML
    private TextField tfMobileNo;
    @FXML
    private GridPane vehicleEntryPane;
    @FXML
    private Text tSlotNo;
    @FXML
    private AnchorPane SlotsPane;
    @FXML
    private GridPane UsersPane;
    @FXML
    private TableView<Floor> tbFloors;
    @FXML
    private TableColumn<Floor, Integer> floorIdCol;
    @FXML
    private TableColumn<Floor, String> floorNameCol;
    @FXML
    private TableColumn<Floor, Integer> NoOfSlotsCol;
    @FXML
    private Pagination pagination;
    @FXML
    private TextField tfFloorName;
    @FXML
    private TextField tfNumberOfSlots;
    @FXML
    private GridPane floorPane;
    @FXML
    private TableView<Users> usersTable;
    @FXML
    private TableColumn<Users, Integer> idColumn;
    @FXML
    private TableColumn<Users, String> usernameColumn;
    @FXML
    private TableColumn<Users, String> passwordColumn;
    @FXML
    private TableColumn<Users, String> roleColumn;
    @FXML
    private ComboBox<String> cbRole;
    @FXML
    private TextField tfName;
    @FXML
    private TextField tfPassword;

    /*=================================== Controlling User Access
===================================*/
    private static String status = null;

    //If user is entered as "Admin"
    public static Scene getAdminControl() throws IOException {
        status = "Admin";
        FXMLLoader fxmlLoader = new 
FXMLLoader(Driver.class.getResource("/com/Project/System.fxml"));
        return new Scene(fxmlLoader.load());
    }

    //If user is entered as "Controller"
    public static Scene getControllerControl() throws IOException {
        status = "Controller";
        FXMLLoader fxmlLoader = new 
FXMLLoader(Driver.class.getResource("/com/Project/System.fxml"));
        return new Scene(fxmlLoader.load());
    }

    //Disable options for "Controller"
    public void disableOptions(){
        showVehicleHistoryBtn.setDisable(true);
        floorsBtn.setDisable(true);
        userBtn.setDisable(true);
    }

    //Enable options for "Admin"
    public void enableOptions(){
        showVehicleHistoryBtn.setDisable(false);
        floorsBtn.setDisable(false);
        userBtn.setDisable(false);
    }


//==========================================================================
=======================//

    /*======================================== Panes Toggling 
========================================*/

    //This method toggles the visibility of Panes
    private<T extends Pane> void showPane(T t){
        for (Pane p :
                allPanes) {
            p.setVisible(t == p);
        }
    }

    //Toggling Vehicle Entry Pane
    public void showVehicleEntryPane(){
     showPane(vehicleEntryPane);
    }

    //Toggling Vehicle Un-park Pane
    public void showUnparkVehiclePane(){
        showPane(unparkVehiclePane);
    }

    //Toggling Vehicle History Pane
    public void showVehicleHistoyPane(){
        showPane(vehicleHistoryPane);
    }
    //Toggling Slots Pane
    public void showSlotsPane(){
        showPane(SlotsPane);
    }

    //Toggling Floor Pane
    public void showFloorPane(){
        showPane(floorPane);
    }

    //Toggling User Pane
    public void showUsersPane(){
        showPane(UsersPane);
    }

    //Logout Btn Action
    public void logoutBtnAction() throws IOException {

        if (Boxes.confirmBox("Logout", "Are you sure you want to logout?"))
{
            FXMLLoader fxmlLoader = new
FXMLLoader(Driver.class.getResource("Login.fxml"));
            Scene scene = new Scene(fxmlLoader.load());

            Driver.windowSetting();
            Driver.getWindow().setResizable(false);
            Driver.getWindow().setScene(scene);
        }
    }

    //Exit Button Action
    public void handleExit(){
        if (Boxes.confirmBox("Exit", "Are you sure you want to exit?"))
            Platform.exit();
    }


//============================================================================
===========================//

    /*======================================== Vehicle Entry Pane
========================================*/
    //Taking Entry of Vehicle
    public void addVehicleEntryBtnAction(){
        //If any field is empty it will an alert box is displayed
        if (tfPersonName.getText().equals("") ||
tfMobileNo.getText().equals("") || tfVehiclePlateNo.getText().equals("") ||
tfTimeIn.getText().equals("") || cbVehicleType.getValue() == null)
            Boxes.alertBox("Empty Fields", "Fields are empty!");
        else if (DateTime.timeValidity(tfTimeIn.getText())){
            Vehicle.addVehicle(new Vehicle(tfPersonName.getText(),
Long.parseLong(tfMobileNo.getText()), tfVehiclePlateNo.getText(),
cbVehicleType.getValue(), tfTimeIn.getText(), DateTime.setDate(),
cbFloor.getValue(), Integer.parseInt(tSlotNo.getText())), tbUnparkVehicle);

            //Clearing the fields after entering the Vehicle data
            tfPersonName.clear();
```

```java
            tfMobileNo.clear();
            tfVehiclePlateNo.clear();
            tfTimeIn.clear();
            tSlotNo.setText("");
            tvehiclePrice.setText("");

            //Updating the Slots Pane
            Slots.showSlots(pagination, floorName);
        }
    }

    //Setting Current Time in TimeIn Field of Vehicle Entry Pane
    public void setTimeInBtnAction(){
        DateTime.setTime(tfTimeIn);
    }

    //Setting Current Time in TimeOut Field in Unpark Pane
    public void setTimeOutBtnAction(){
        DateTime.setTime(tfTimeOut);
    }

    //Sets the Bill for parking vehicle in tTotalBill
    public void calculateBillBtnAction(){
        tTotalBill.setText(String.format("Rs. %.2f",
Vehicle.calculateTotalBill(Double.parseDouble(tPricePerHour.getText().split
(" ")[1]), tTimeIn.getText(), tfTimeOut.getText())));
    }


//===========================================================================
=============================//

    /*======================================== Un-park Vehicle Pane
========================================*/

    //Un-Park Vehicle Btn Action
    public void unparkVehicleBtnAction(){
        //Check if the un-park table row is selected or not
        if (selectUnparkTableRow() != null) {
            //Check if the timeOut field is empty or not
            if (tfTimeOut.getText().isEmpty())
                Boxes.alertBox("Empty Fields", "Enter Time Out of
Vehicle!");
            else {
                if (Boxes.confirmBox("Pay Bill", "Does bill has been
paid?"))
                    Vehicle.unparkVehicle(tbUnparkVehicle,
tbVehicleHistory, selectUnparkTableRow(), tfTimeOut.getText());   //Calls
the unparkVehicle Function from the Vehicle Class

                //Clearing the data from the field after un-parking the car
                tCustomerName.setText("");
                tMobileNo.setText("");
                tVehiclePlateNo.setText("");
                tVehicleType.setText("");
                tFloor.setText("");
                tUnparkSlotNo.setText("");
                tTimeIn.setText("");
                tfTimeOut.clear();
                tPricePerHour.setText("");
                tTotalBill.setText("");
            }
        }

    }

    //Selects the Row in Un-Park Table and returns the selected object as
Vehicle
    public Vehicle selectUnparkTableRow(){
        Vehicle selectedVehicle =
tbUnparkVehicle.getSelectionModel().getSelectedItem();

        //Sets the values of selected vehicle in the Un-Park Vehicle Pane
text fields and texts
        if (selectedVehicle != null){
            tCustomerName.setText(selectedVehicle.getCustomerName());

tMobileNo.setText(Long.toString(selectedVehicle.getMobileNumber()));
            tVehiclePlateNo.setText(selectedVehicle.getVehicleNo());
            tVehicleType.setText(selectedVehicle.getVehicleType());
            tFloor.setText(selectedVehicle.getFloorName());

tUnparkSlotNo.setText(Integer.toString(selectedVehicle.getSlotNo()));
            tTimeIn.setText(selectedVehicle.getTimeIn());

            if (tVehicleType.getText().equals("Rickshaw"))
                tPricePerHour.setText("Rs. 75");
            else if (tVehicleType.getText().equals("Bike"))
                tPricePerHour.setText("Rs. 100");
            else if (tVehicleType.getText().equals("Car"))
                tPricePerHour.setText("Rs. 150");
            else if (tVehicleType.getText().equals("Commercial Vehicle"))
                tPricePerHour.setText("Rs. 200");

        }
        return selectedVehicle;
    }


//===========================================================================
==============================//

    /*======================================== Floors Pane
========================================*/

    //Add Floor Button Action
    public void addFloorBtnAction(){
        //Check if the floor name and number of slots fields are empty or
not
        if (tfFloorName.getText().equals("") ||
tfNumberOfSlots.getText().equals(""))
            Boxes.alertBox("Empty Fields", "Fields are empty!");
        else {
            //Calls the addFloor Function from the Floor Class
            Floor.addFloor(new Floor(tfFloorName.getText(),
Integer.parseInt(tfNumberOfSlots.getText())), tbFloors, cbFloor);

            //Clearing the data from the fields after adding the floor
            tfFloorName.clear();
            tfNumberOfSlots.clear();

            //Updating the Slots Pane
            Slots.showSlots(pagination, floorName);
        }
    }

    //Edit Floor Button Action
    public void editFloorBtnAction(){
        if (selectFloorTableRow() != null)
            //Calls the editFloor Function from the Floor Class
            Floor.editFloor(tbFloors, pagination, selectFloorTableRow(),
floorName);
    }

    //Selects the Row in Floors Table and returns the selected object as
Floor
    public Floor selectFloorTableRow(){
        Floor selectedFloor =
tbFloors.getSelectionModel().getSelectedItem();
        if (selectedFloor != null){
            //Sets the values of selected floor in the Floor Pane text
fields
            tfFloorName.setText(selectedFloor.getFloorName());

tfNumberOfSlots.setText(Integer.toString(selectedFloor.getNoOfSlots()));
        }

    return selectedFloor;
    }

    //Delete Floor Button Action
    public void deleteFloorBtnAction(){
        Floor.delFloor(tbFloors, pagination, floorName);    //Calls the
delFloor Function from the Floor Class
    }


//===========================================================================
===================//

    /*======================================== Users Pane
========================================*/

    //Add User Button Action
    public void addUserBtnAction(){
        //Check if the Username, Password, and Role fields are empty or not
        if (tfName.getText().equals("") || tfPassword.getText().equals("")
|| cbRole.getValue() == null)
            Boxes.alertBox("Empty Fields", "Fields are empty or role not
selected!");
        else {
            //Calls the addUser Function from the Users Class
            Users.addUser(new Users(tfName.getText(), tfPassword.getText(),
cbRole.getValue()), usersTable);

            // Clearing the Fields
            tfName.clear();
            tfPassword.clear();
        }
    }

    //Edit User Button Action
    public void editUserBtnAction(){
        Users selectedUser = selectUserTableRow();
        if (selectedUser != null)
            Users.editUser(usersTable, selectedUser);   //Calls the
editUser Function from the Users Class
    }

    //Delete User Button Action
    public void deleteUserBtnAction(){
        //Calls the delUser Function from the Users Class
        Users.delUsers(usersTable);
    }

    //Selects the User in User Table and returns the selected object as
Users
    public Users selectUserTableRow(){
        Users selectedUser =
usersTable.getSelectionModel().getSelectedItem();
        if (selectedUser != null){
            //Sets the values of selected user in the User Pane text fields
            tfName.setText(selectedUser.getName());
            tfPassword.setText(selectedUser.getPassword());
            cbRole.setValue(selectedUser.getRole());
        }

        return selectedUser;
    }


//===========================================================================
====================//

    //Array list to store Panes
    private ArrayList<Pane> allPanes;
    @Override
    public void initialize(URL location, ResourceBundle resources) {
        //Checking if the Logged-In user is Admin or Controller
        if (status.equals("Controller"))
            disableOptions();
        else if (status.equals("Admin")) {
            enableOptions();
        }


//===========================================================================
=====//
```

```
        //Adding all the panes to the arraylist of Pane to handle the
visibility of panes
        allPanes = new ArrayList<>();
        allPanes.add(vehicleEntryPane);
        allPanes.add(unparkVehiclePane);
        allPanes.add(vehicleHistoryPane);
        allPanes.add(SlotsPane);
        allPanes.add(floorPane);
        allPanes.add(UsersPane);


//===========================================================================
============//

        /*==================== Initializing fields on Vehicle Entry Pane
====================*/

        //Adding items to "Vehicle Type" Combo Box on Vehicle Entry Pane
        cbVehicleType.getItems().addAll("Car", "Bike", "Rickshaw",
"Commercial Vehicle");

        //Adding items to "Floor No." Combo box on Vehicle Entry Pane
        ArrayList<Floor> floors =
FileHandling.readFromFile(Files.getFloorFile());
        for (Floor f :
                floors) {
            cbFloor.getItems().add(f.getFloorName());

        }

        cbFloor.setOnAction(event -> Slots.allocateSlot(cbFloor.getValue(),
tSlotNo));


        //Setting prices of vehicle
        cbVehicleType.setOnAction(event -> {
            if (cbVehicleType.getValue().equals("Rickshaw"))
                tvehiclePrice.setText("Rs. 75");
            else if (cbVehicleType.getValue().equals("Bike"))
                tvehiclePrice.setText("Rs. 100");
            else if (cbVehicleType.getValue().equals("Car"))
                tvehiclePrice.setText("Rs. 150");
            else if (cbVehicleType.getValue().equals("Commercial Vehicle"))
                tvehiclePrice.setText("Rs. 200");
        });


//===========================================================================
============//

        /*==================== Initializing Table on Un-Park Vehicle Pane
====================*/

        //Showing All Vehicles on the Table
        vehicleIdCol.setCellValueFactory(new PropertyValueFactory<>("id"));
        vehicleCustomerCol.setCellValueFactory(new
PropertyValueFactory<>("customerName"));
        vehiclePlateNoCol.setCellValueFactory(new
PropertyValueFactory<>("vehicleNo"));
        vehicleTimeInCol.setCellValueFactory(new
PropertyValueFactory<>("timeIn"));
        tbUnparkVehicle.setItems(Vehicle.showVehicles());


//==========================================================================
============//

        /*==================== Initializing Table on Vehicles History Pane
====================*/

        customerNameCol.setCellValueFactory(new
PropertyValueFactory<>("customerName"));
        mobileNoCol.setCellValueFactory(new
PropertyValueFactory<>("mobileNumber"));
        vehicleTypeCol.setCellValueFactory(new
PropertyValueFactory<>("vehicleType"));
        vehicleNoCol.setCellValueFactory(new
PropertyValueFactory<>("vehicleNo"));
        timeInCol.setCellValueFactory(new
PropertyValueFactory<>("timeIn"));
        timeOutCol.setCellValueFactory(new
PropertyValueFactory<>("timeOut"));
        floorCol.setCellValueFactory(new
PropertyValueFactory<>("floorName"));
        slotNoCol.setCellValueFactory(new
PropertyValueFactory<>("slotNo"));
        dateCol.setCellValueFactory(new PropertyValueFactory<>("date"));

        tbVehicleHistory.setItems(Vehicle.showVehiclesHistory());


//==========================================================================
======//

        /*==================== Initializing Pagination on Slots Pane
====================*/
        //Showing All Parking Slots
        Slots.showSlots(pagination, floorName);


//==========================================================================
======//
        /*====================== Initializing Table on Floors Pane
======================*/

        //Showing All Floor on the table
        floorIdCol.setCellValueFactory(new PropertyValueFactory<>("id"));
        floorNameCol.setCellValueFactory(new
PropertyValueFactory<>("floorName"));
        NoOfSlotsCol.setCellValueFactory(new
PropertyValueFactory<>("noOfSlots"));
        tbFloors.setItems(Floor.showFloor());
```

```
//============================================================================
====================//
        /*=============== Initializing Table and Combo Box on Un-Park
Vehicle Pane ====================*/

        //Adding items to Combo Box on Users Pane
        cbRole.getItems().addAll("Admin", "Controller");

        //Showing All user on the table
        idColumn.setCellValueFactory(new PropertyValueFactory<>("id"));
        usernameColumn.setCellValueFactory(new
PropertyValueFactory<>("name"));
        passwordColumn.setCellValueFactory(new
PropertyValueFactory<>("password"));
        roleColumn.setCellValueFactory(new PropertyValueFactory<>("role"));

        usersTable.setItems(Users.showUsers());
    }
}
```