Requirements Analysis Document

Car Rental System
CSCI 4711 Software Engineering
Fall 2021
Augusta University
Augusta, GA
Date: 11/20/21
Version 5

Team Members

Sebastien Gonzalez
Alex Hall
Benito McClammy
Jonathan Parker

Table of Contents

1 INTRODUCTION

1.1 SCOPE OF SYSTEM

Our car rental application provides a system to make reservations from a selection of vehicles, and to monitor the availability of the fleet. It includes actors such as the *Customer*, who represent prospective vehicle renters, and *Employee*, an employee of the car rental service who can maintain awareness of the availability of their fleet.

The system keeps track of information about a number of preset vehicles and their availability. It stores this information in a database, that also securely stores user log in credentials, and an activity log. Both Customers and Employees interact with the car rental system through a Windows user interface.

The system includes functionality to verify and differentiate Customer and Employee log in credentials. It presents Customers with a selection of vehicles from which they can reserve over selected dates, pending availability. Employees are presented a query which allows them to check the availability of their fleet over a specified range of time. All data is stored internally as part of the system. The system is standalone, requires no network access, and external interaction is solely with Customer and Employee users.

2 REQUIREMENTS OF SYSTEM

2.1 FUNCTIONAL REQUIREMENTS

- Startup - This functionality performs the initial configuration of the system including its database and user interface.
- Login - Both Employee and Customer are required to login to access the system's other functionalities. Every login will be saved to the database.
- Logout - Any user that is logged in will be able to logout. Every logout is saved in the database.
- MakeReservation - This functionality is for Customer. They can select a car from the list of vehicles displayed and enter a date range to reserve the car. The system will validate the date range selected and allow the Customer to reserve the car if it is available for that date range.
- ViewAvailability - This functionality is for the Employee. They first enter a date range and are then presented two lists of the VIN numbers of cars in their inventory. One list shows the cars that are available to be rented for that date range. The other shows the cars that are not available or already rented as well as the specific date range that the car is unavailable.

2.2 NON-FUNCTIONAL REQUIREMENTS

- Platform
    - Target operating system is Microsoft Windows
    - System is not web-based and should not utilize a web browser
    - System should be implemented using the C# programming language
- Security
    - All user input is validated for appropriateness and prevention of SQL injection attacks
    - User passwords should not be stored as plaintext in the database
- Usability
    - System should implement a graphical user interface

2.3 USE CASES

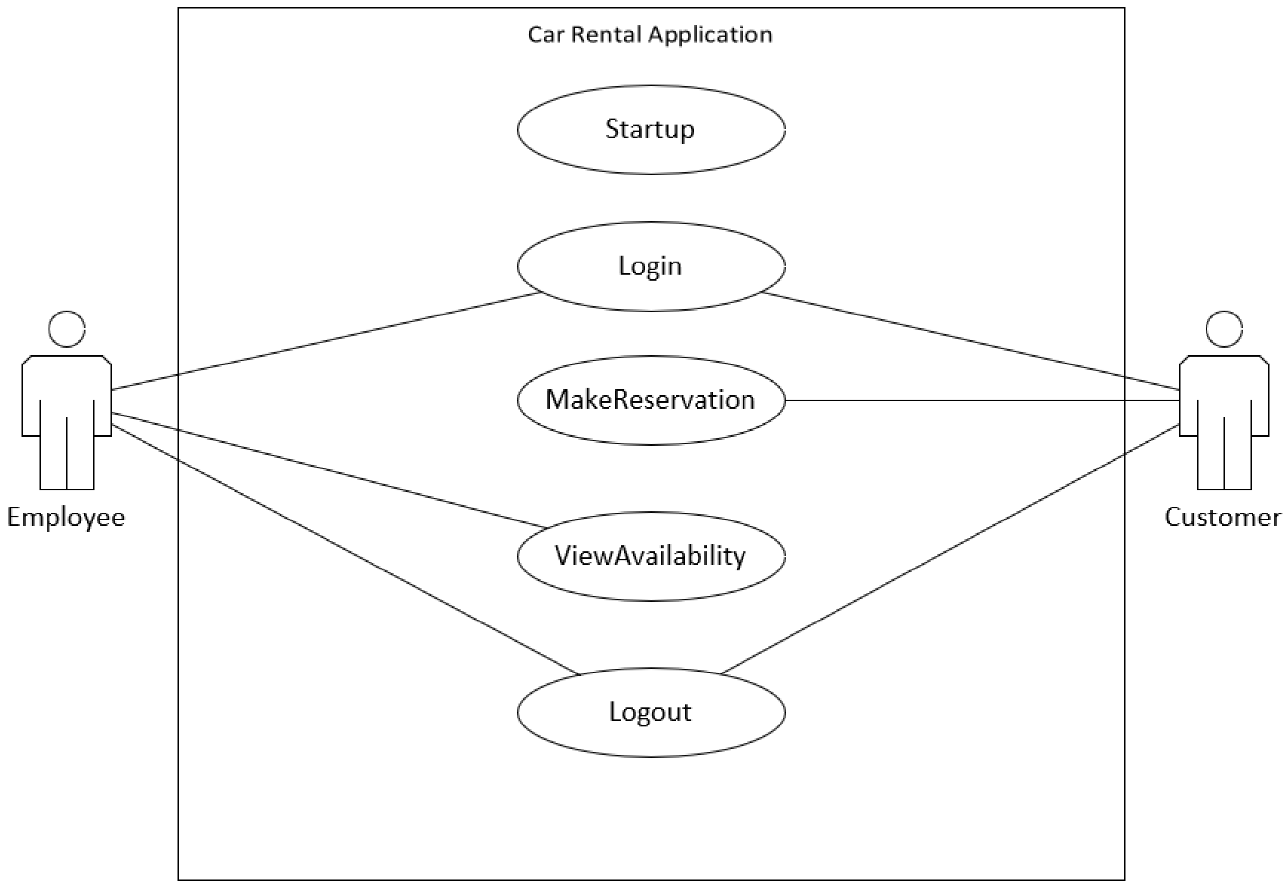


Figure 2.1: Use Case Diagram

2.4 USE CASE DESCRIPTIONS

Use case name	Startup
Participating actors	
Flow of events	1. This use case is initiated when the application is launched. **2. The system initializes the database and then the Login screen is displayed.**
Entry condition	–
Exit condition	– The Login screen is displayed.
Security requirements	– User credentials are not stored as plaintext in the database.

Figure 2.2: Startup

Use case name	Login
Participating actors	Initiated by Employee
Flow of events	1. The Employee enters the username and password information into the Login form and submits the form **2. The system verifies the credentials entered are stored in the database. The ViewAvailability screen is displayed and the login session is saved to the database.**
Entry condition	–
Exit condition	– The Employee is redirected to the initial ViewAvailability screen
Security requirements	– Login credentials are not visible or stored as plaintext in the database. – User input is validated for appropriateness and prevention of SQL injection attacks.

Figure 2.3: Login: Employee

Use case name	Login
Participating actors	Initiated by Customer
Flow of events	1. The actor enters the username and password information into the Login form and submits the form **2. The system verifies the credentials entered are stored in the database. The ReserveInitial screen is displayed and the login session is saved to the database. ReserveInitial screen contains a list of all vehicles.**
Entry condition	–
Exit condition	– The Employee is redirected to the ReserveInitial screen
Security requirements	– Login credentials are not visible or stored as plaintext in the database. – User input is validated for appropriateness and prevention of SQL injection attacks.

Figure 2.4: Login: Customer

Use case name	Login
Participating actors	Initiated by Employee and Customer
Flow of events	1. The actor enters the username and password information into the Login form and submits the form **2. The entered credentials are not verified in the database. Text on the Login screen will display that the username or password is incorrect.**
Entry condition	–
Exit condition	
Security requirements	– Login credentials are not visible or stored as plaintext in the database. – User input is validated for appropriateness and prevention of SQL injection attacks.

Figure 2.5: Login: Invalid Credentials

Use case name	MakeReservation
Participating actors	Initiated by Customer
Flow of events	**1.** The Customer selects a vehicle from the ReserveInitial screen, then clicks the View button. **2. The system presents the ReservationForm screen for the user to choose a date range. Make, model, and year are displayed on the screen. At this time, the ReserveVehicle button is unavailable.** **3.** The Customer clicks the ChooseDates button and inputs the desired date range for their rental. **4. The system checks the database and the vehicle is available for the selected dates. The ReserveVehicle button is now available.** **5.** The Customer clicks the ReserveVehicle button. **6. The system records the reservation information in the database, displays a message to the Customer confirming they have reserved the vehicle for the selected dates, and returns the Customer to the ReserveInitial screen.**
Entry condition	− User is logged in with a Customer account.
Exit condition	− The ReserveInitial screen is displayed to user, reservation information is stored in the database.
Security requirements	− User input for date entry must be verified to mitigate SQL injection vulnerability.

Figure 2.6: MakeReservation: Dates Available

Use case name	MakeReservation
Participating actors	Initiated by Customer
Flow of events	**1.** The Customer selects a vehicle from the ReserveInitial screen, then clicks the View button. **2. The system presents the ReservationForm screen for the user to choose a date range. Make, model, and year are displayed on the screen. At this time, the ReserveVehicle button is unavailable.** **3.** The Customer clicks the ChooseDates button and inputs the desired date range for their rental. **4. The system checks the database and displays a message to the Customer indicating the vehicle is not available for the selected dates. The ReserveVehicle button does not become available, the Customer must enter another date range.** **5.** The Customer clicks the ChooseDates button and inputs a new date range. **6. The system checks the database and displays a message to the Customer indicating the vehicle is not available for the selected dates.**
Entry condition	– User is logged in with a Customer account.
Exit condition	– A message is displayed on the ReservationForm screen
Security requirements	– User input for date entry must be verified to mitigate SQL injection vulnerability.

Figure 2.7: MakeReservation: Dates Unavailable

Use case name	ViewAvailability
Participating actors	Initiated by Employee
Flow of events	1. The employee selects the date range they wish to view and submits the form. **2. The system displays ViewAvailability screen showing vin numbers, make, and model of all vehicles available and unavailable for selected date range.**
Entry condition	− An Employee account is logged in.
Exit condition	− The ViewAvailability screen is displayed.
Security requirements	−

Figure 2.8: ViewAvailability

Use case name	Logout
Participating actors	Initiated by Employee and Customer
Flow of events	1. This function is activated when a Customer or Employee selects the “logout” button on the Reserve screens or ViewAvailability screen. **2. The system ends the user's session, logs the user logout in the database, then displays the Login screen.**
Entry condition	− User must be logged in to the system. − The Logout button is pressed by the user or the application is closed.
Exit condition	− The Login screen is displayed.
Security requirements	−

Figure 2.9: Logout

2.5 REQUIREMENT ANALYSIS

This page intentionally left blank

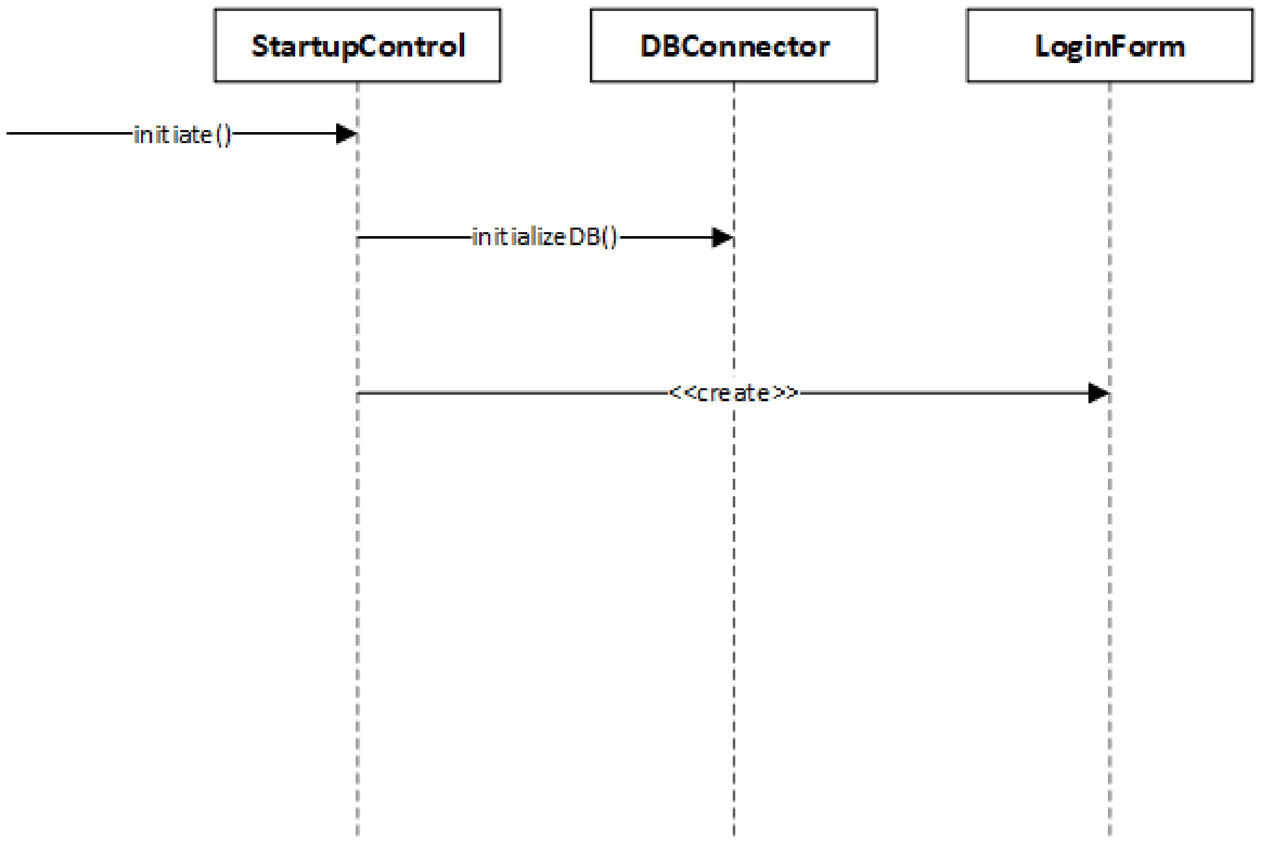


Figure 2.10: Startup

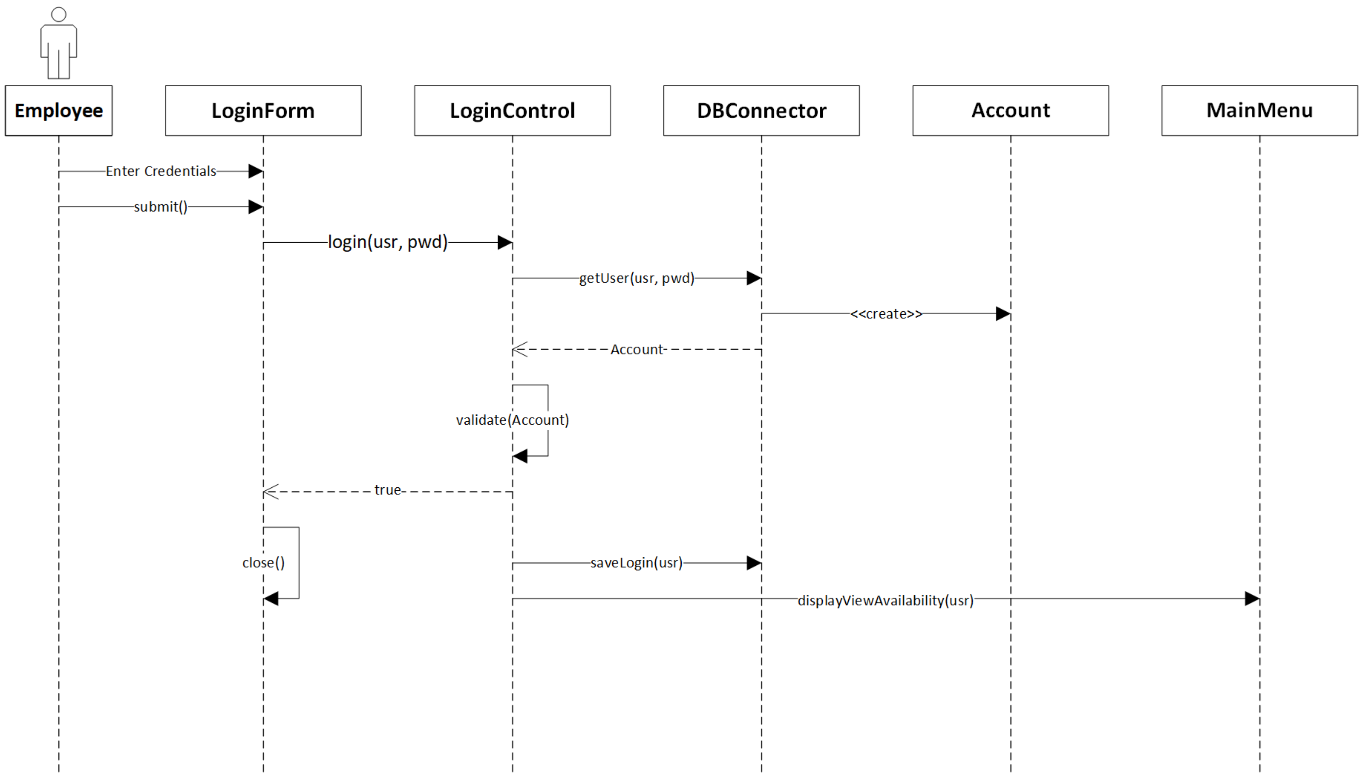


Figure 2.11: Login: Employee

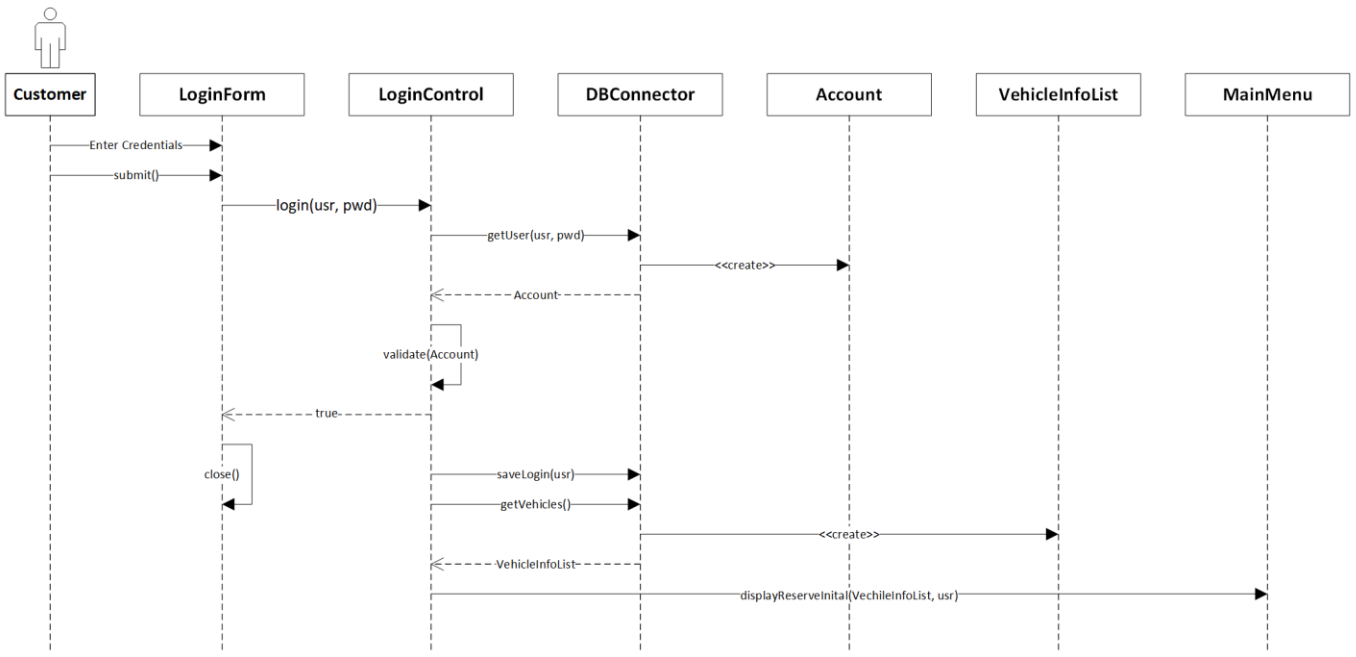


Figure 2.12: Login: Customer

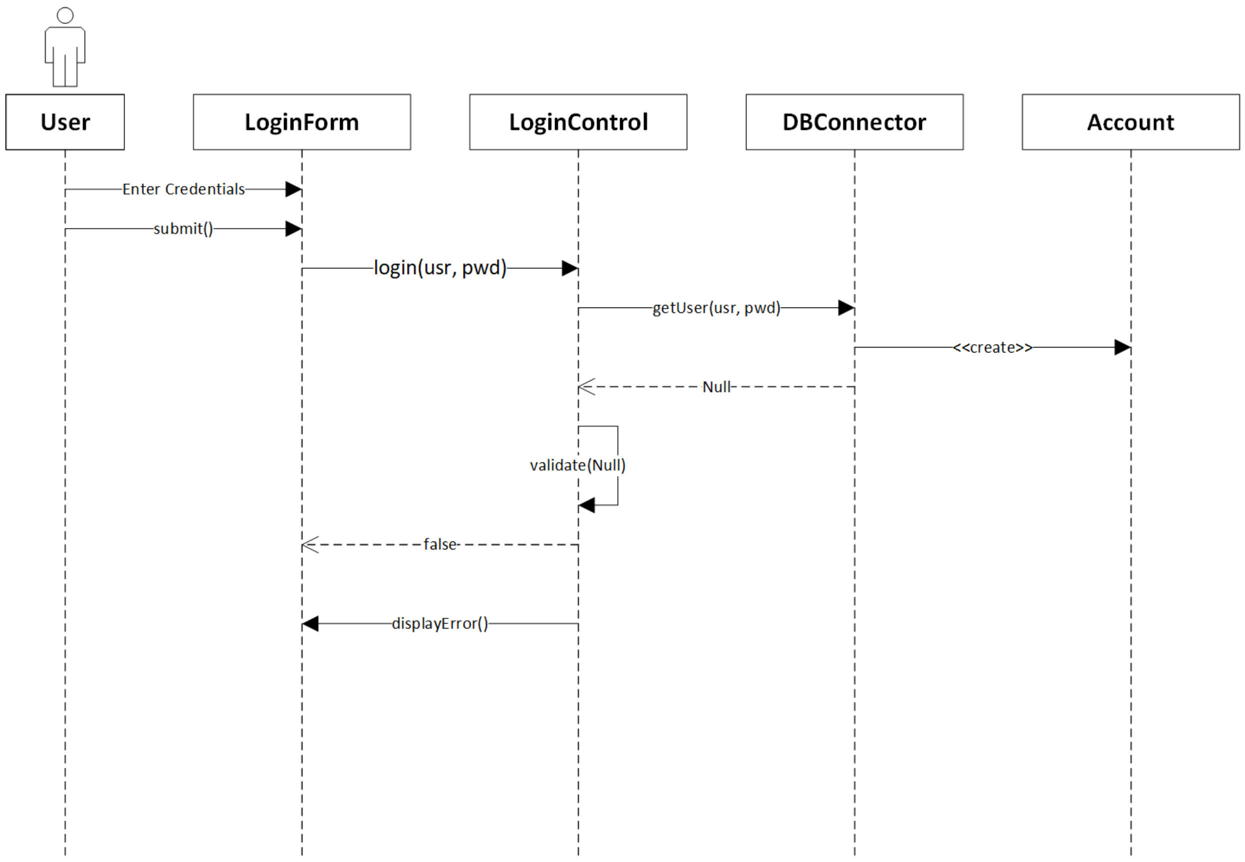


Figure 2.13: Login: Invalid

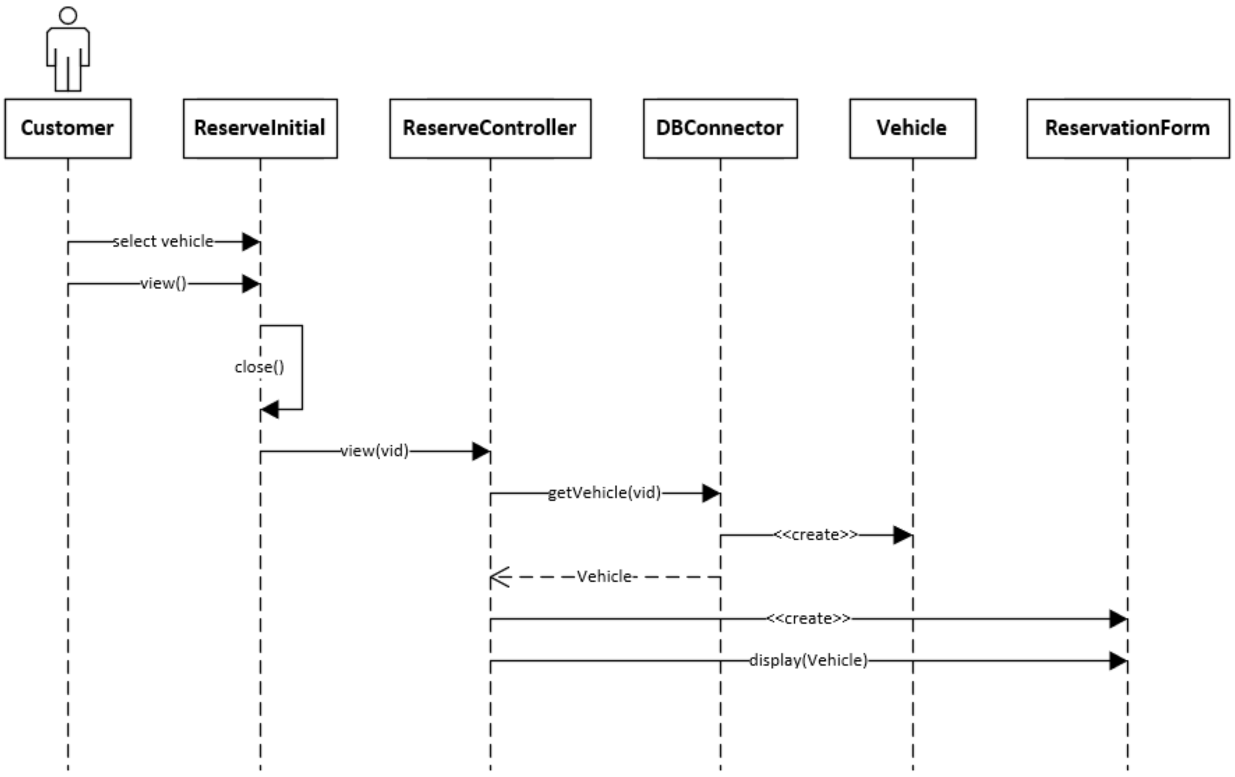


Figure 2.14: MakeReservation: Dates Available 1

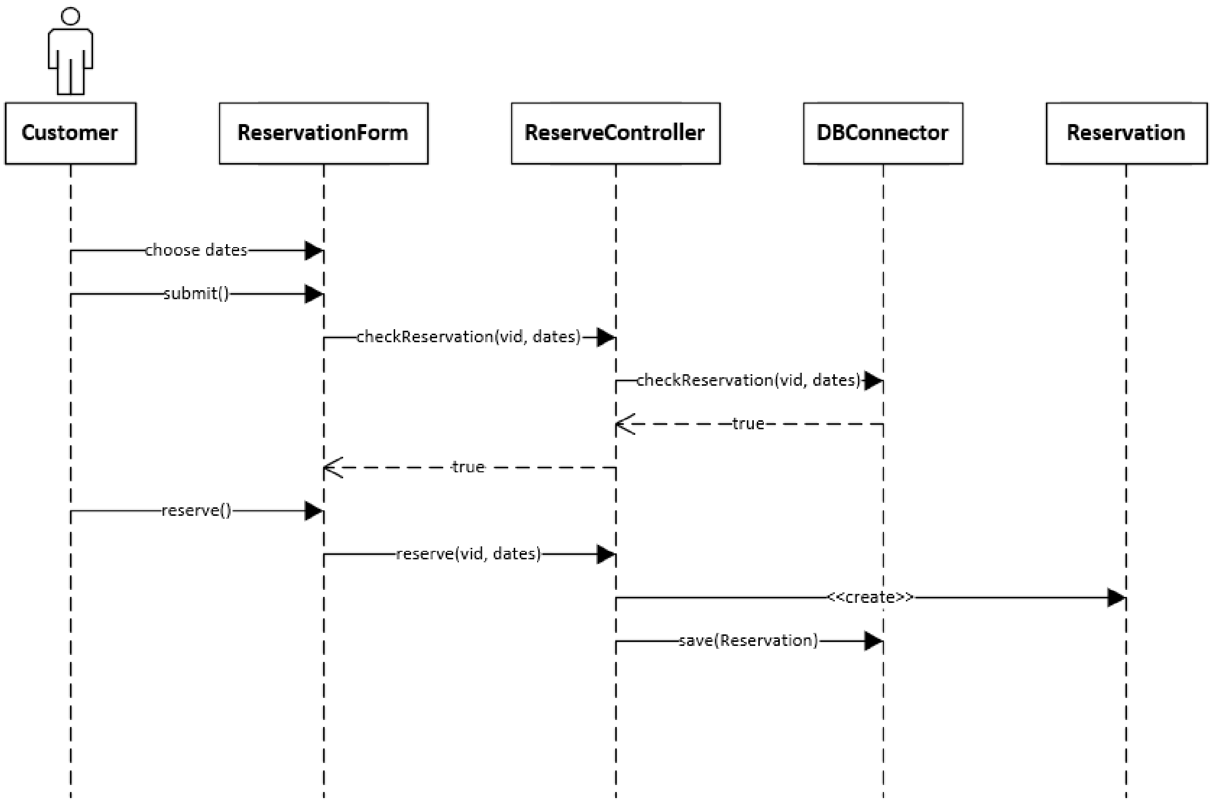


Figure 2.15: MakeReservation: Dates Available 2

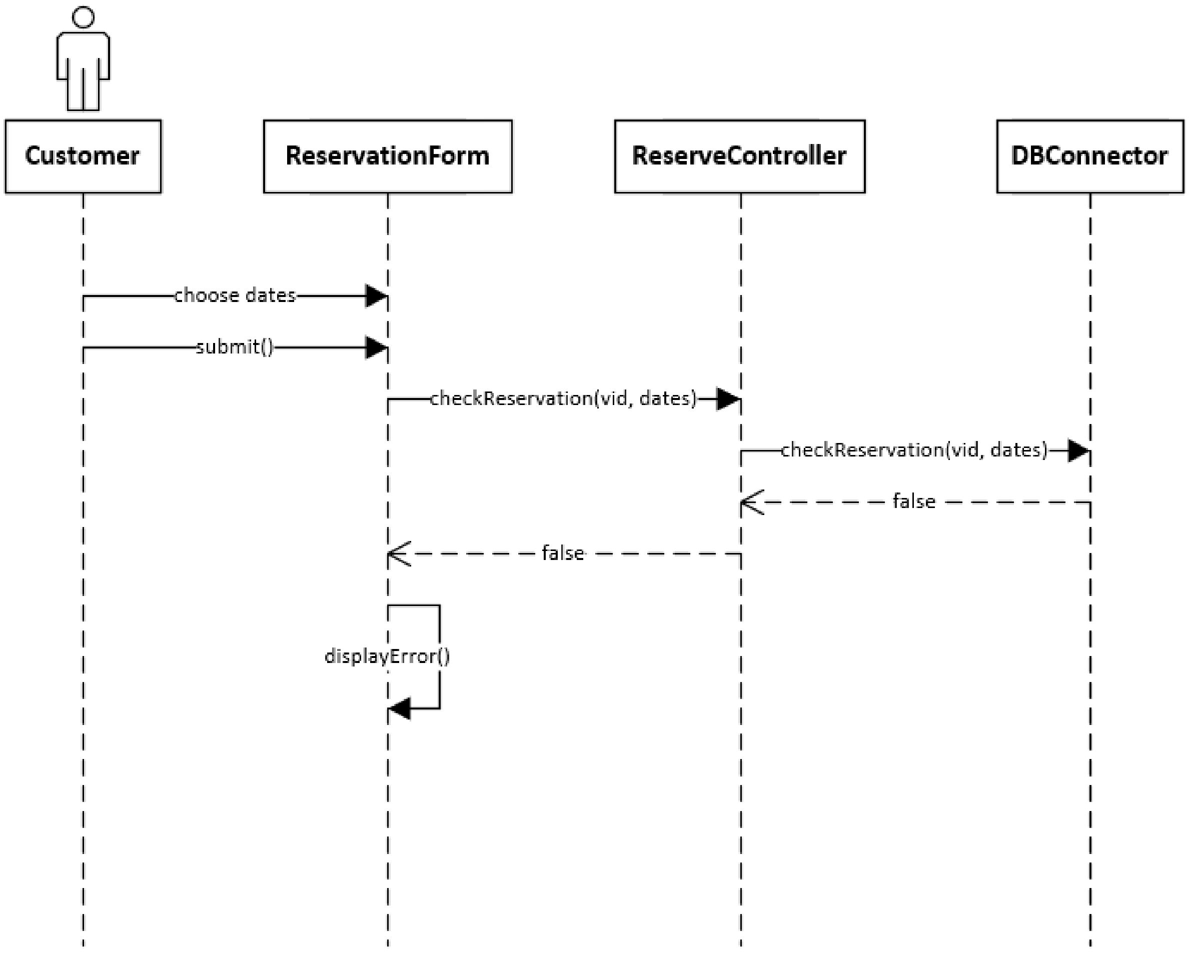


Figure 2.16: MakeReservation: Dates Unavailable

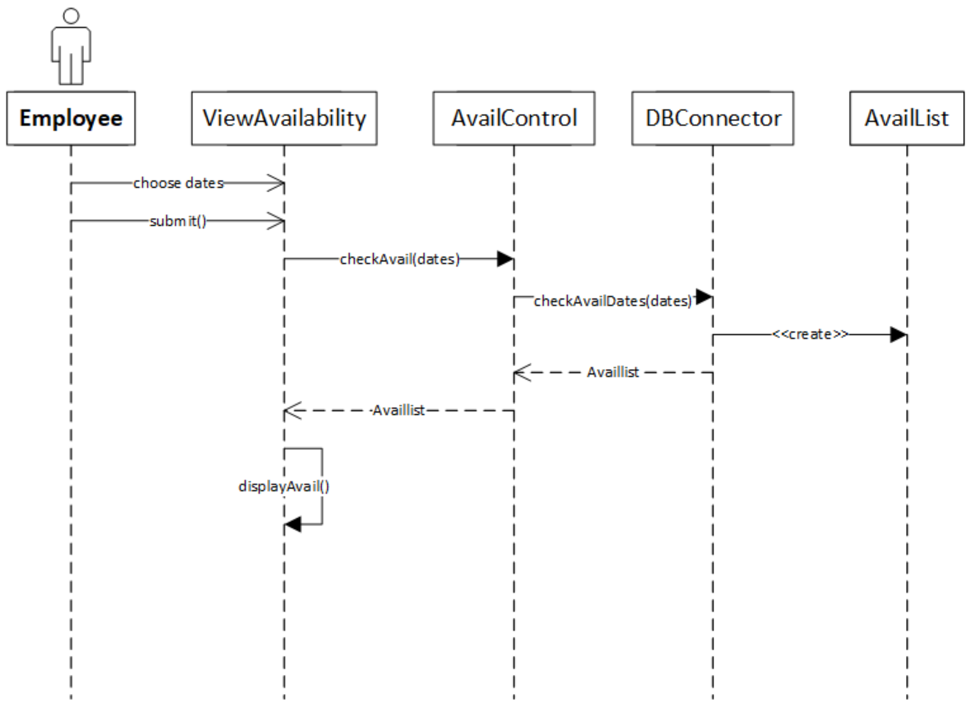


Figure 2.17: ViewAvailability

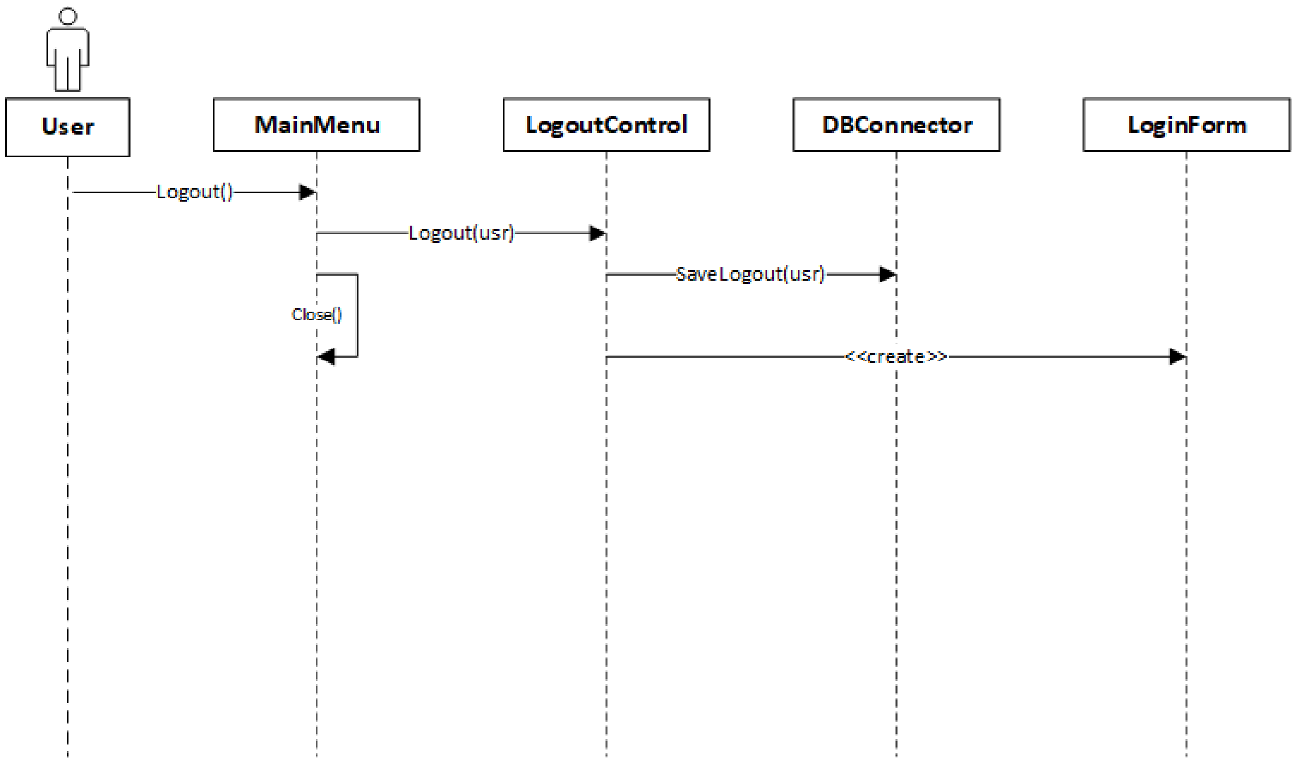


Figure 2.18: Logout

3 USER INTERFACE MOCKUPS

3.1 Login

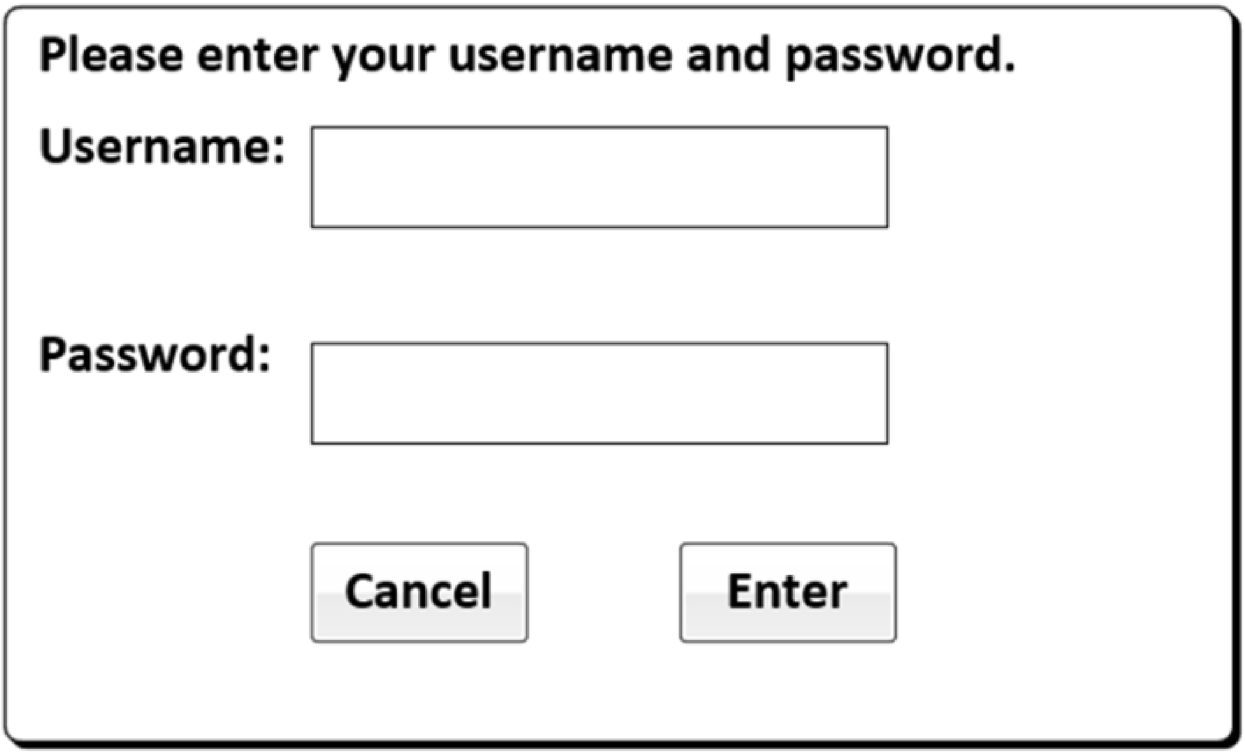


Figure 3.1: Login

3.2MAKERESERVATION

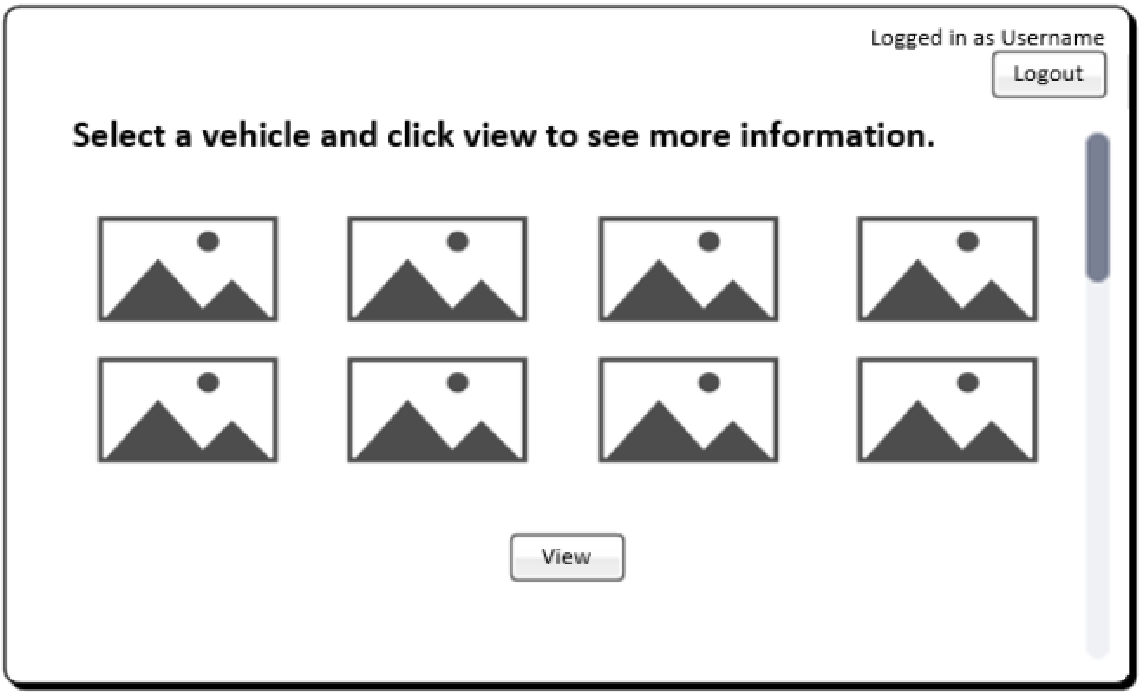


Figure 3.2: ReserveInitial

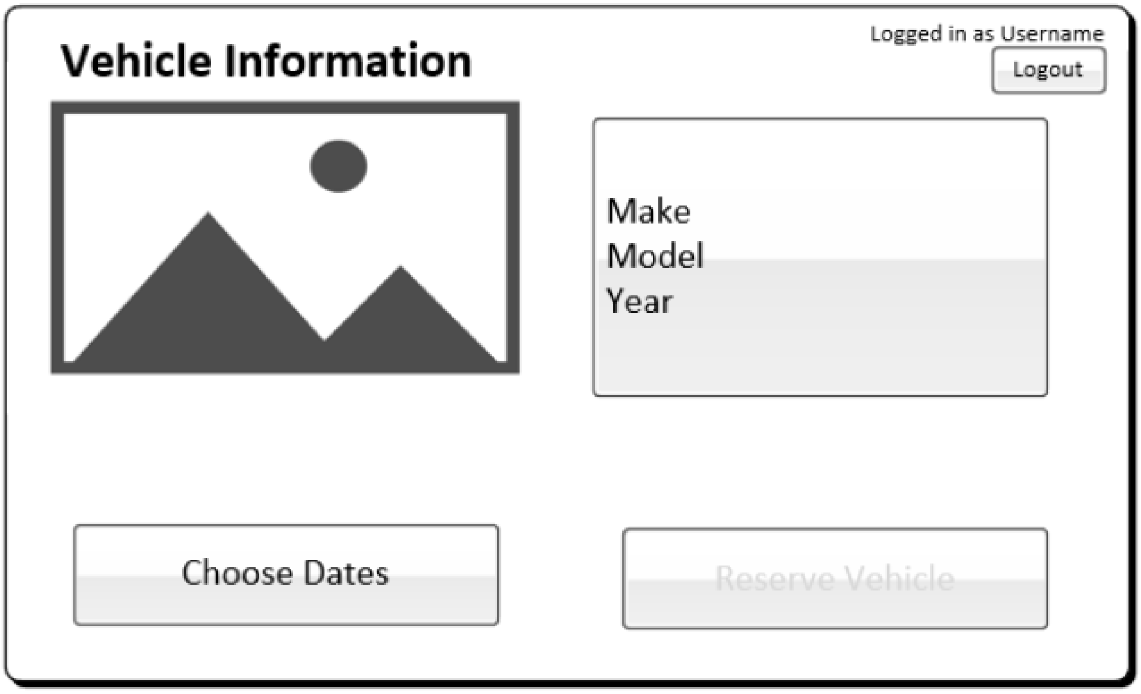


Figure 3.3: ReservationForm

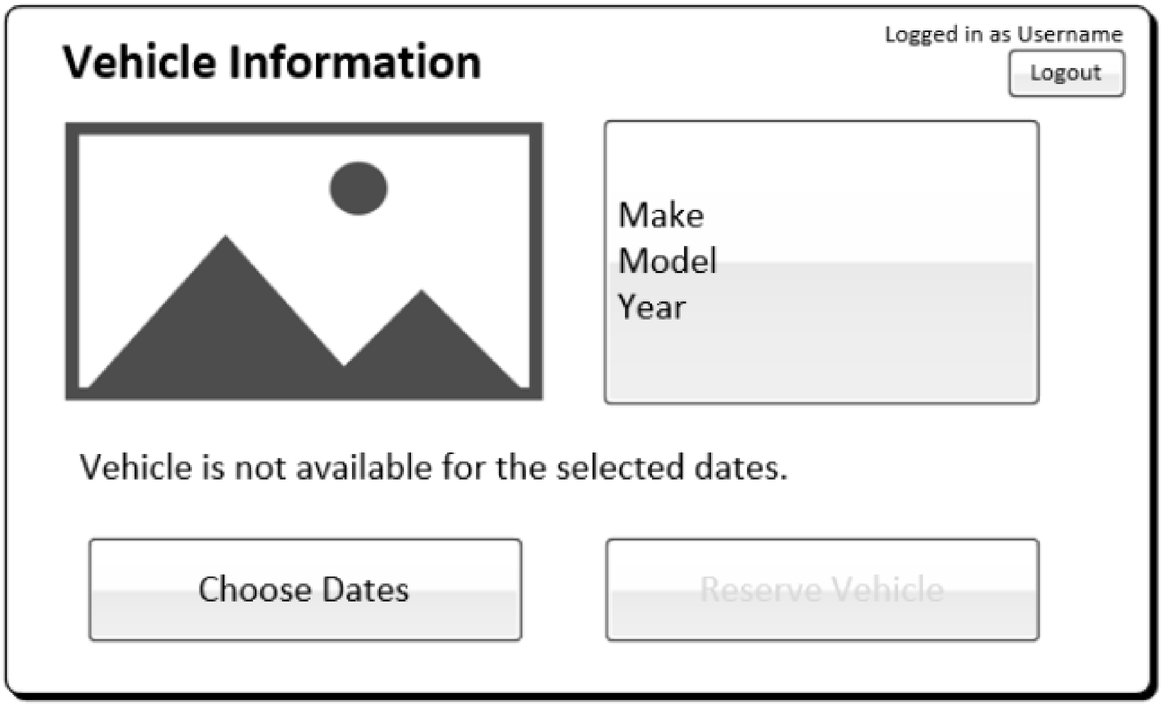


Figure 3.4: ReservationForm (not available)

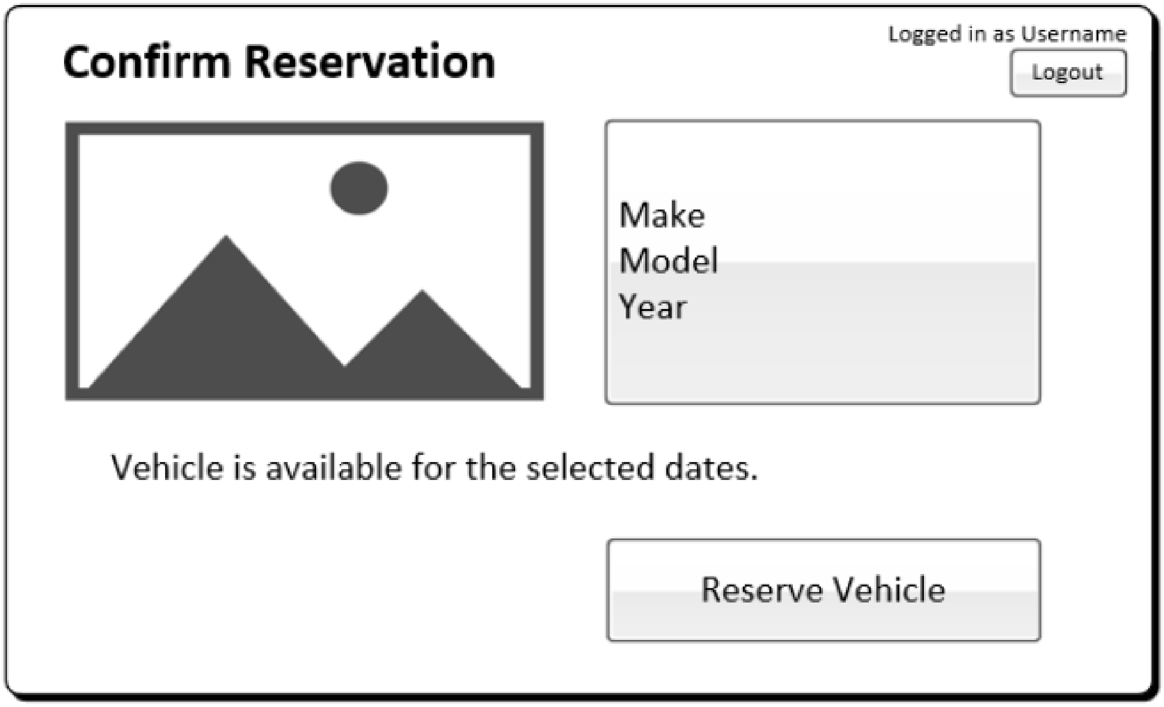


Figure 3.5: ReservationForm (confirmation)

3.3 VIEW AVAILABILITY

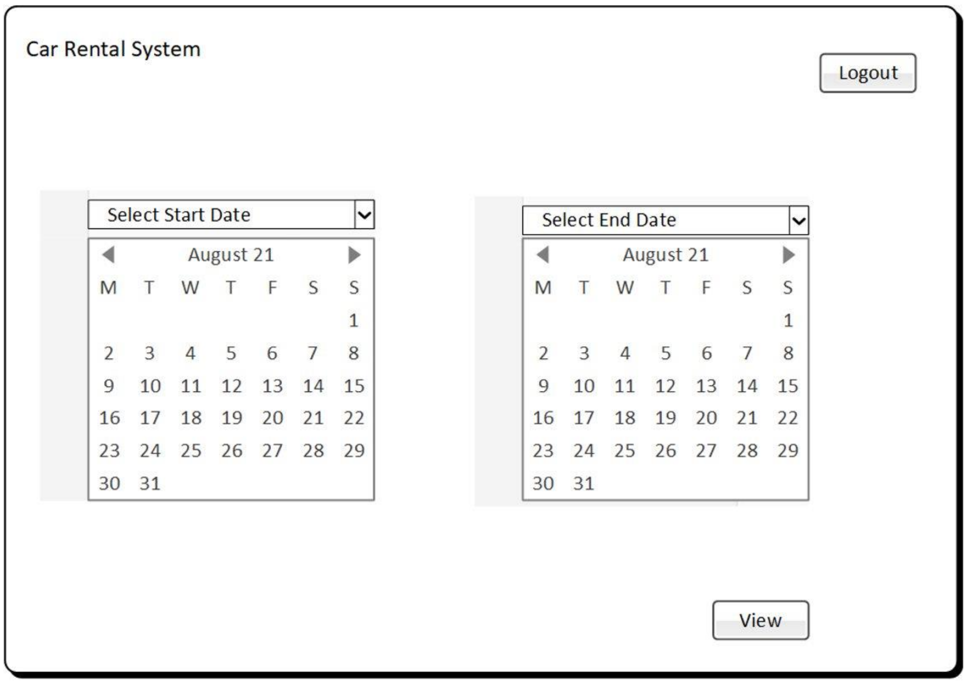


Figure 3.6: ViewAvailabilityDateSelection

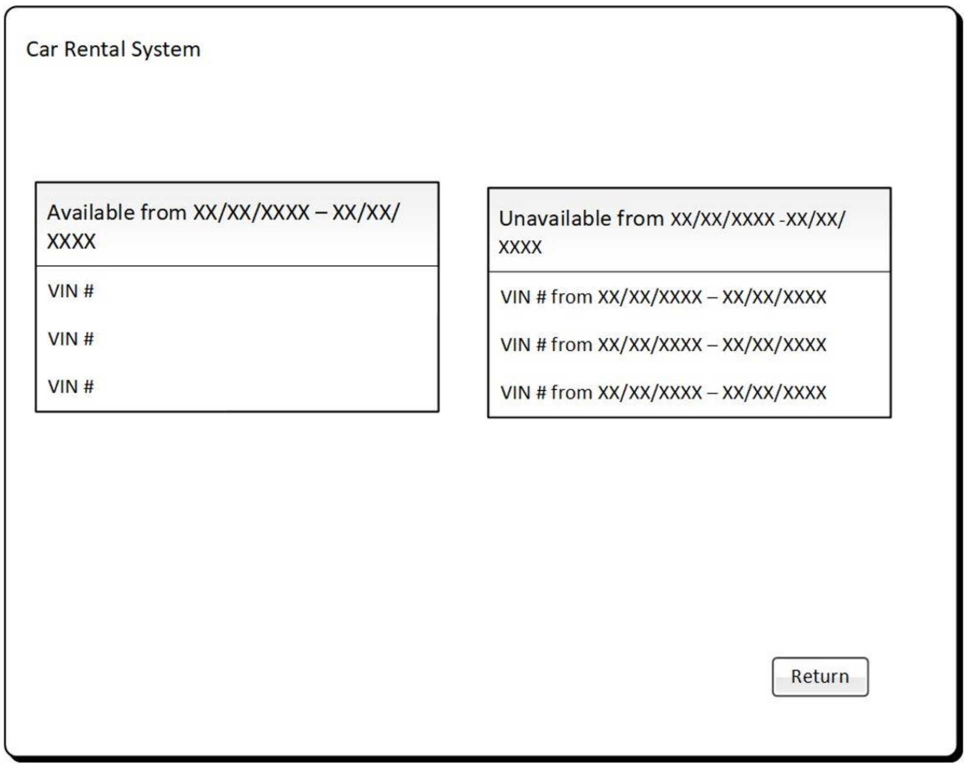

Car Rental System
Available from XX/XX/XXXX – XX/XX/XXXX
VIN #
VIN #
VIN #
Unavailable from XX/XX/XXXX -XX/XX/XXXX
VIN # from XX/XX/XXXX – XX/XX/XXXX
VIN # from XX/XX/XXXX – XX/XX/XXXX
VIN # from XX/XX/XXXX – XX/XX/XXXX
Return

Figure 3.7: ViewAvailability

4 OBJECT DESIGN

4.1 OBJECT RELATIONSHIP

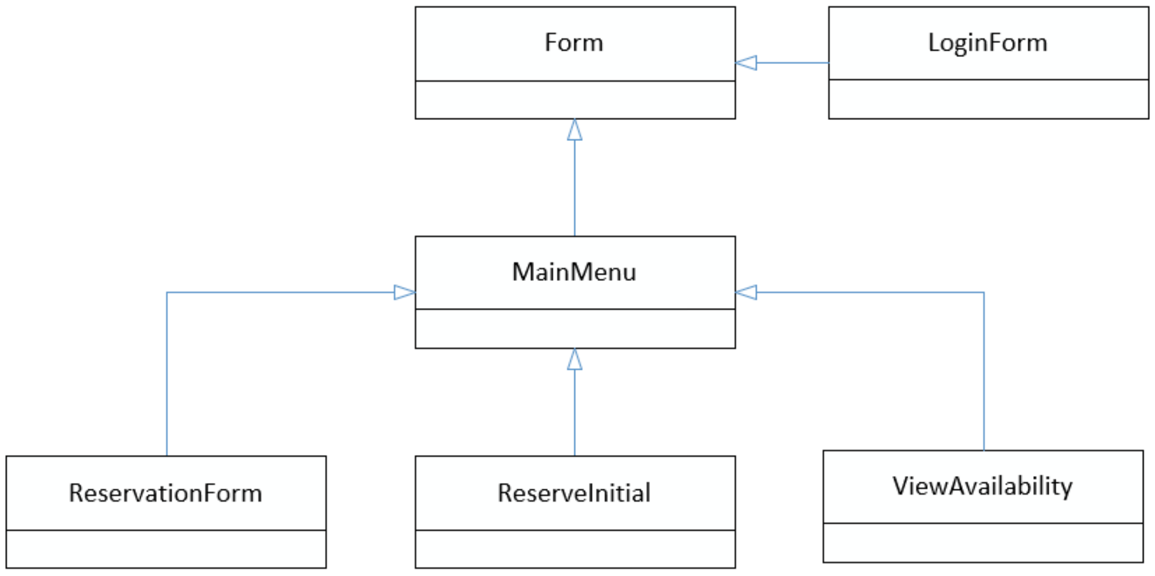


Figure 4.1: Class Diagram: Boundary

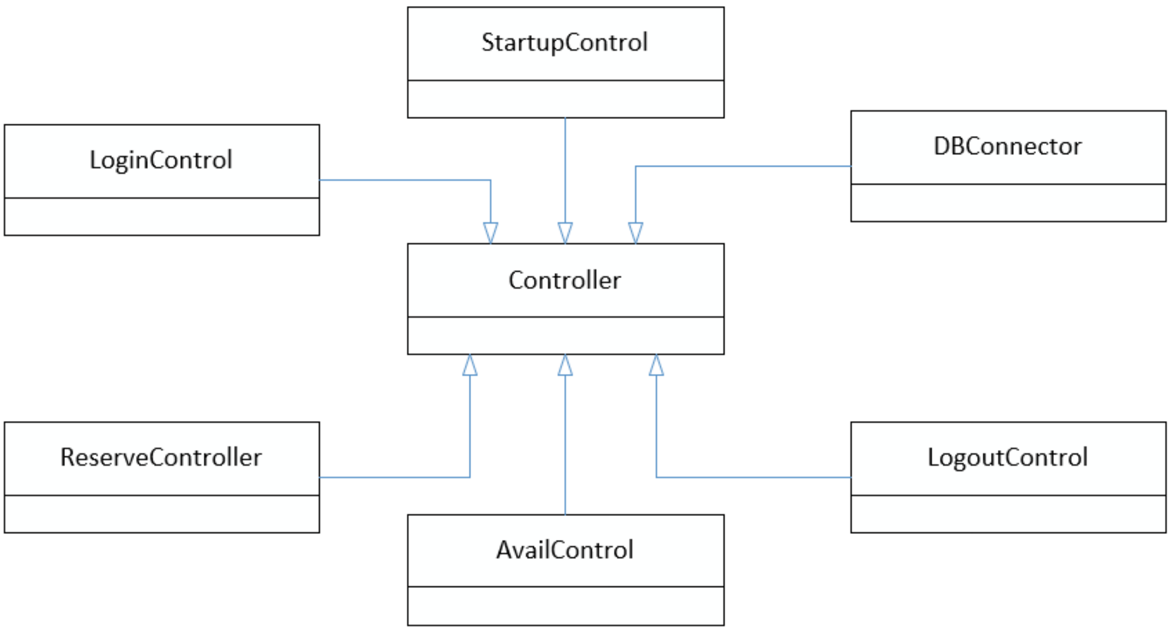


Figure 4.2: Class Diagram: Control

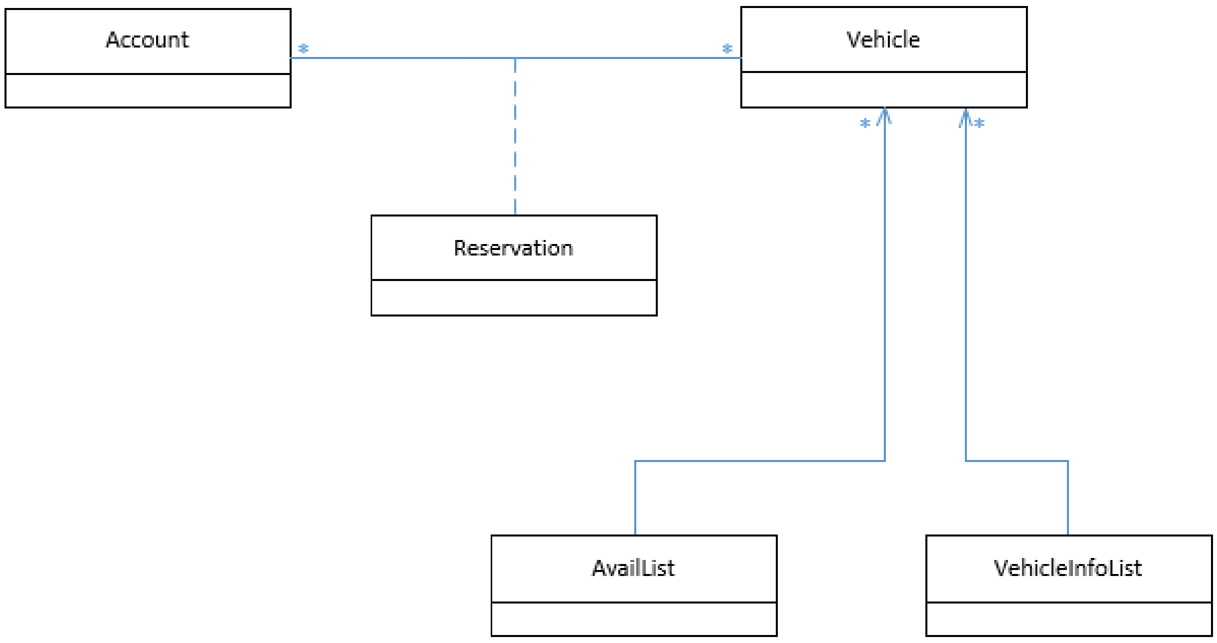


Figure 4.3: Class Diagram: Entity

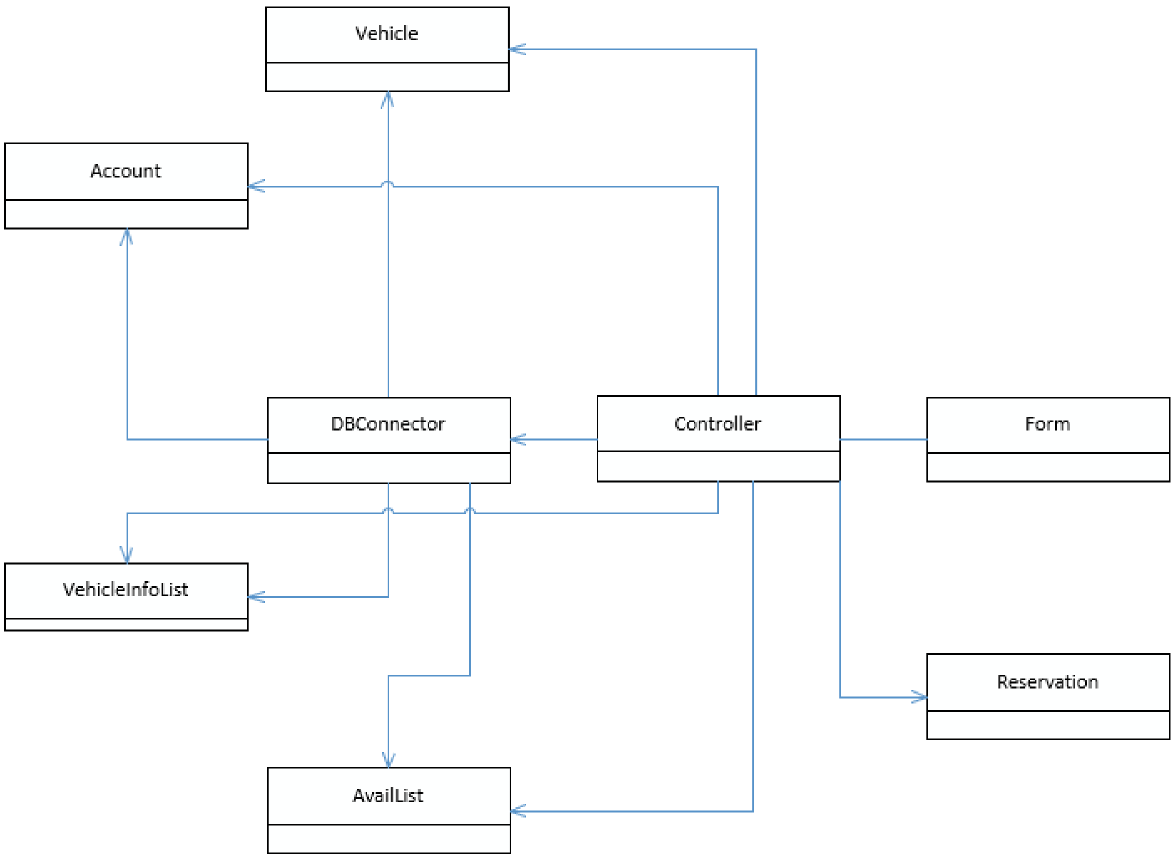


Figure 4.4: Class Diagram

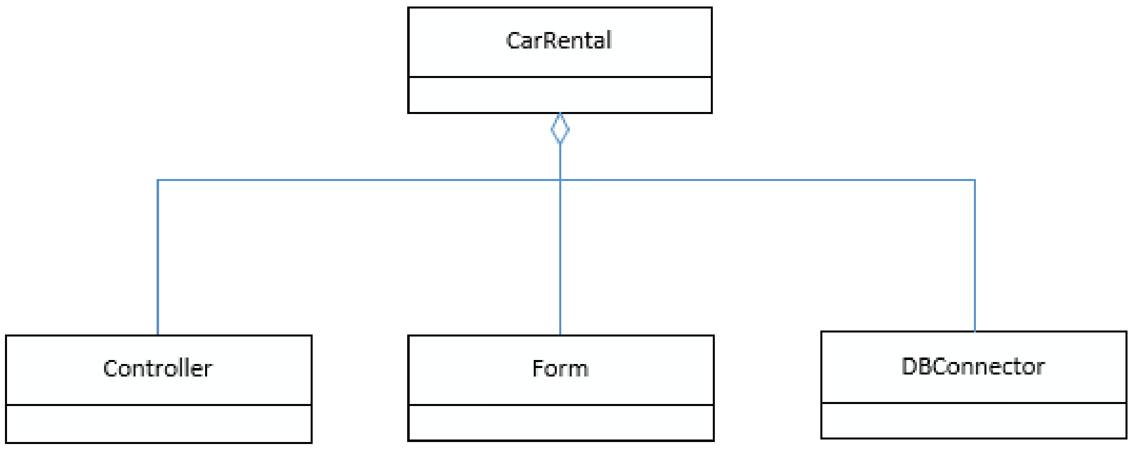
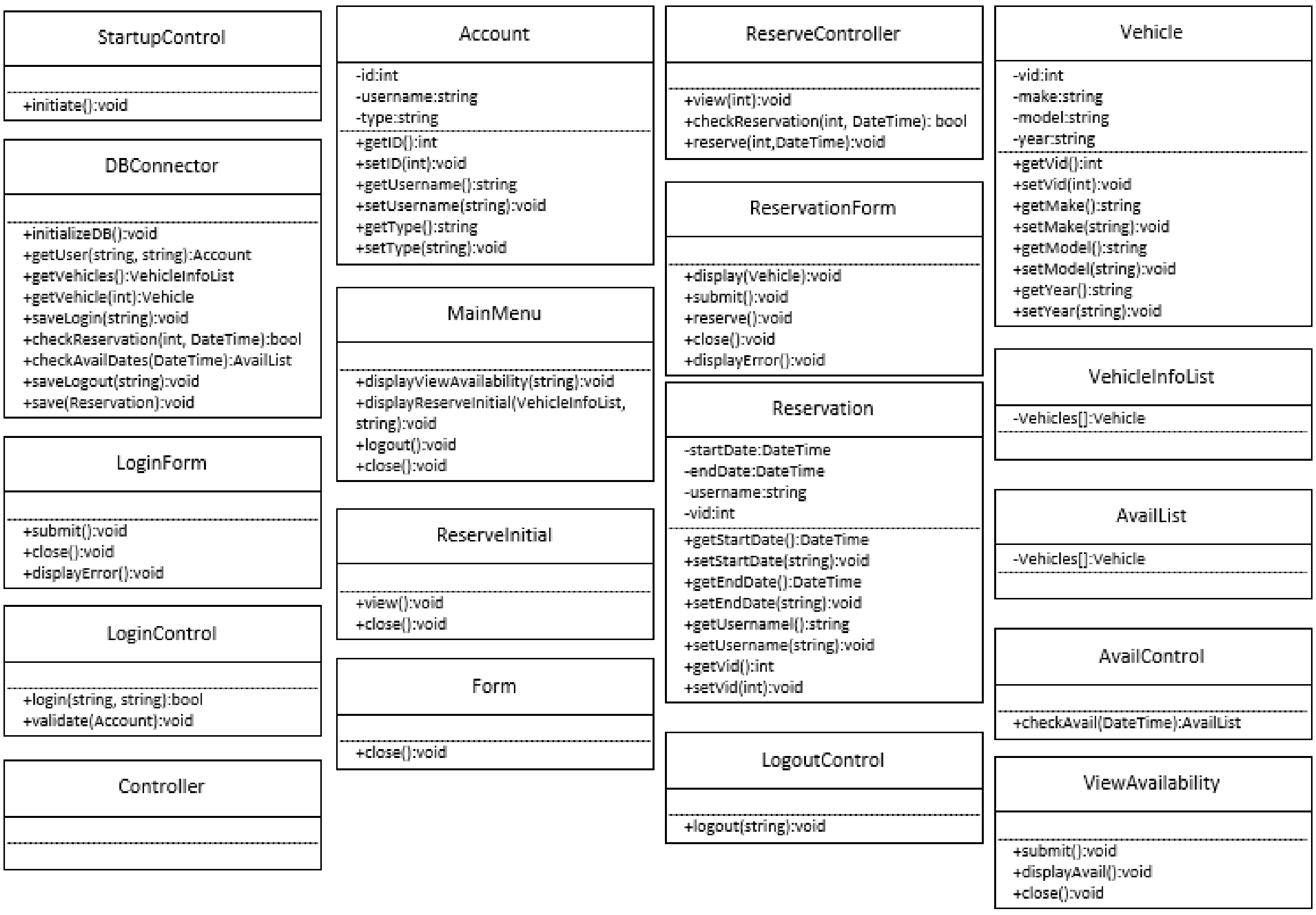


Figure 4.5: Class Diagram

4.2DETAILED CLASS DESIGN



4.3DATABASE DESIGN

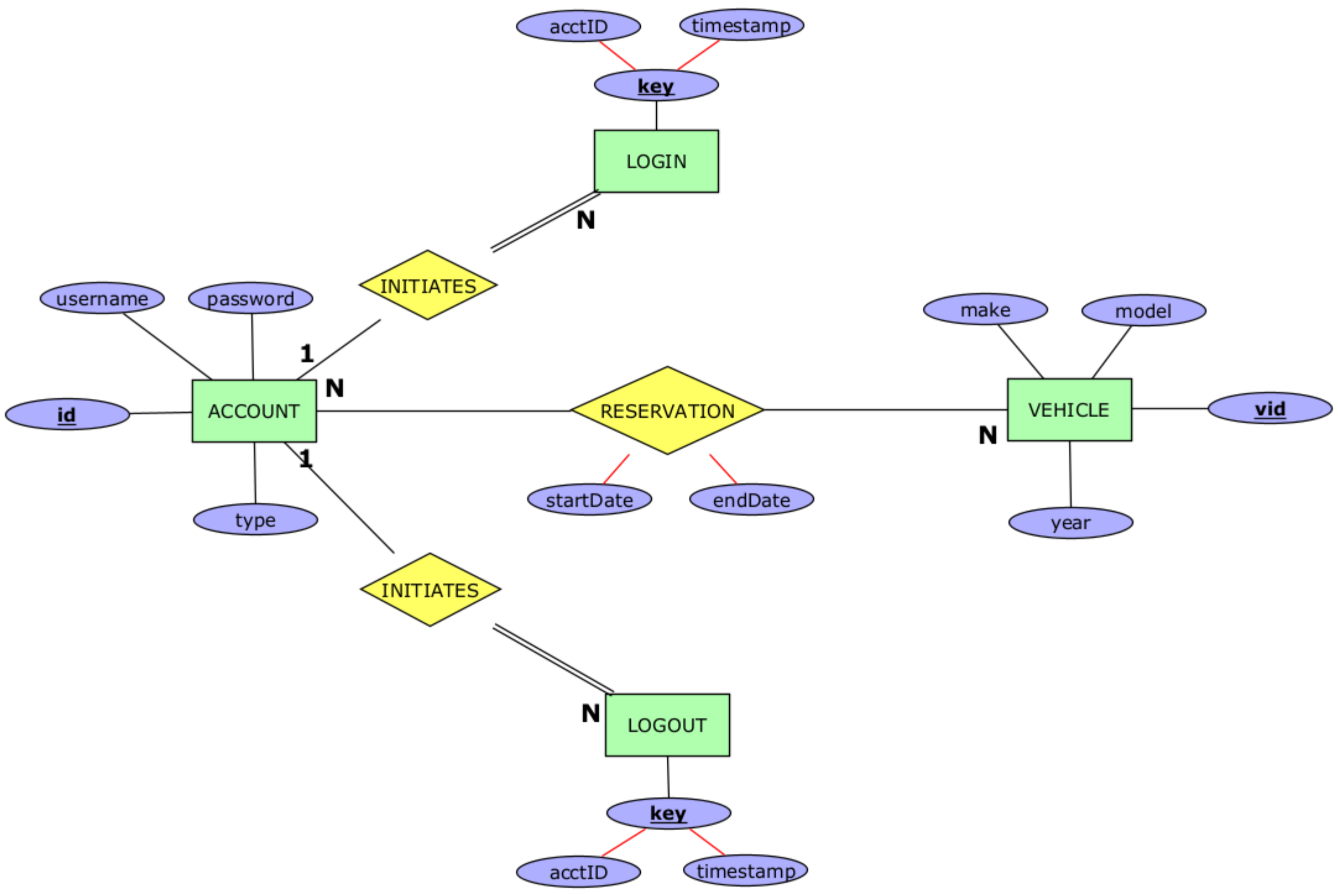


Figure 4.6: ER Diagram

RELATIONAL MODEL

ACCOUNT(<u>id</u>, username, password, type)

LOGIN(<u>acctID</u>^{FK}, <u>timestamp</u>)

LOGOUT(<u>acctID</u>^{FK}, <u>timestamp</u>)

VEHICLE(<u>vid</u>, make, model, year)

RESERVATION(<u>acctID</u>^{FK}, <u>vid</u>^{FK}, <u>startDate</u>, <u>endDate</u>)

STATECHART DIAGRAMS

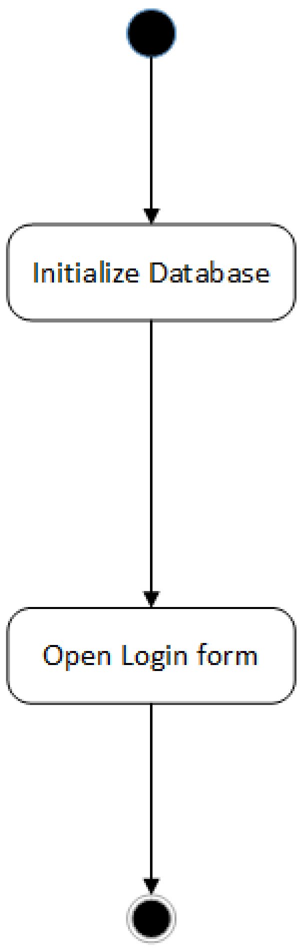


Figure 4.7: StartupControl

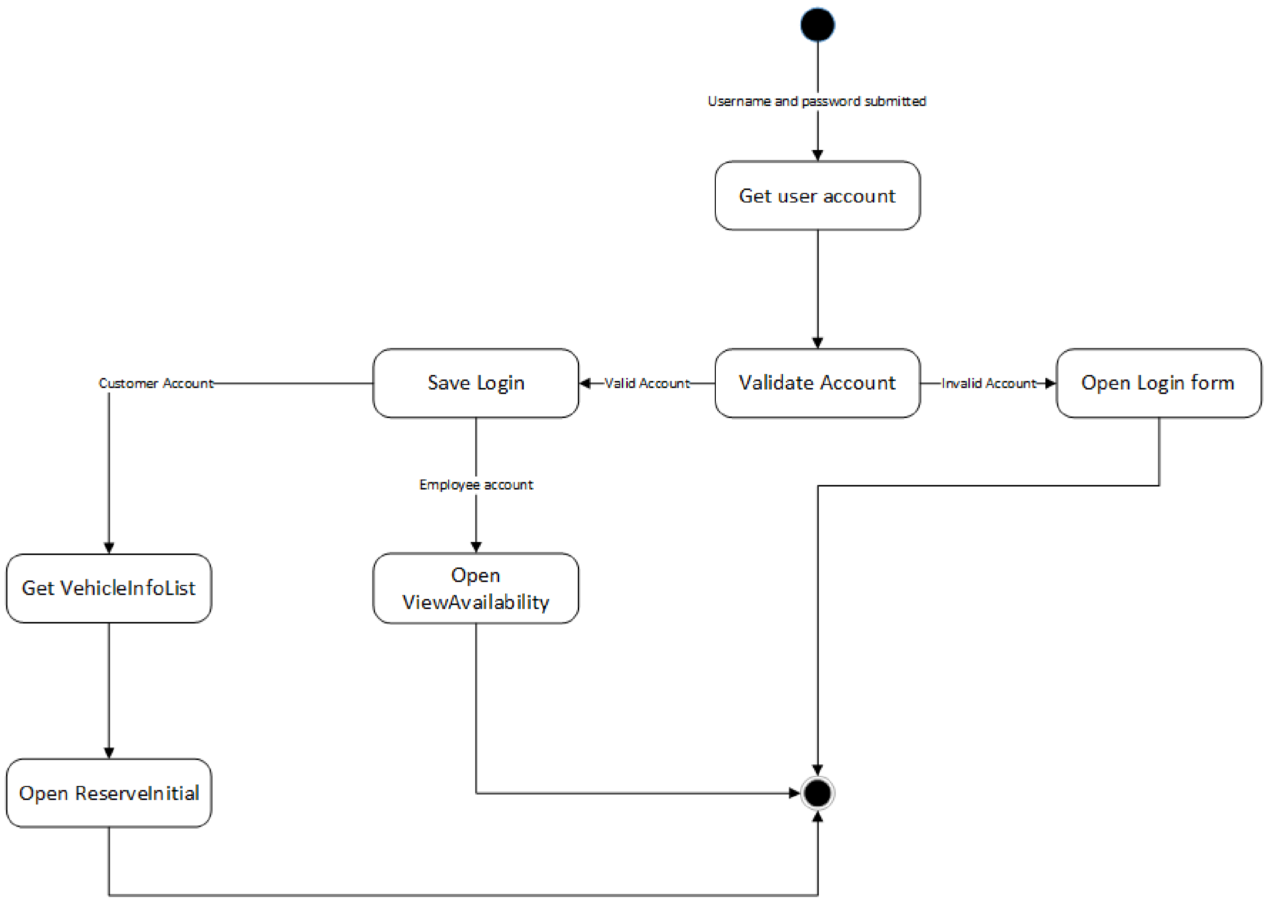


Figure 4.8: LoginControl

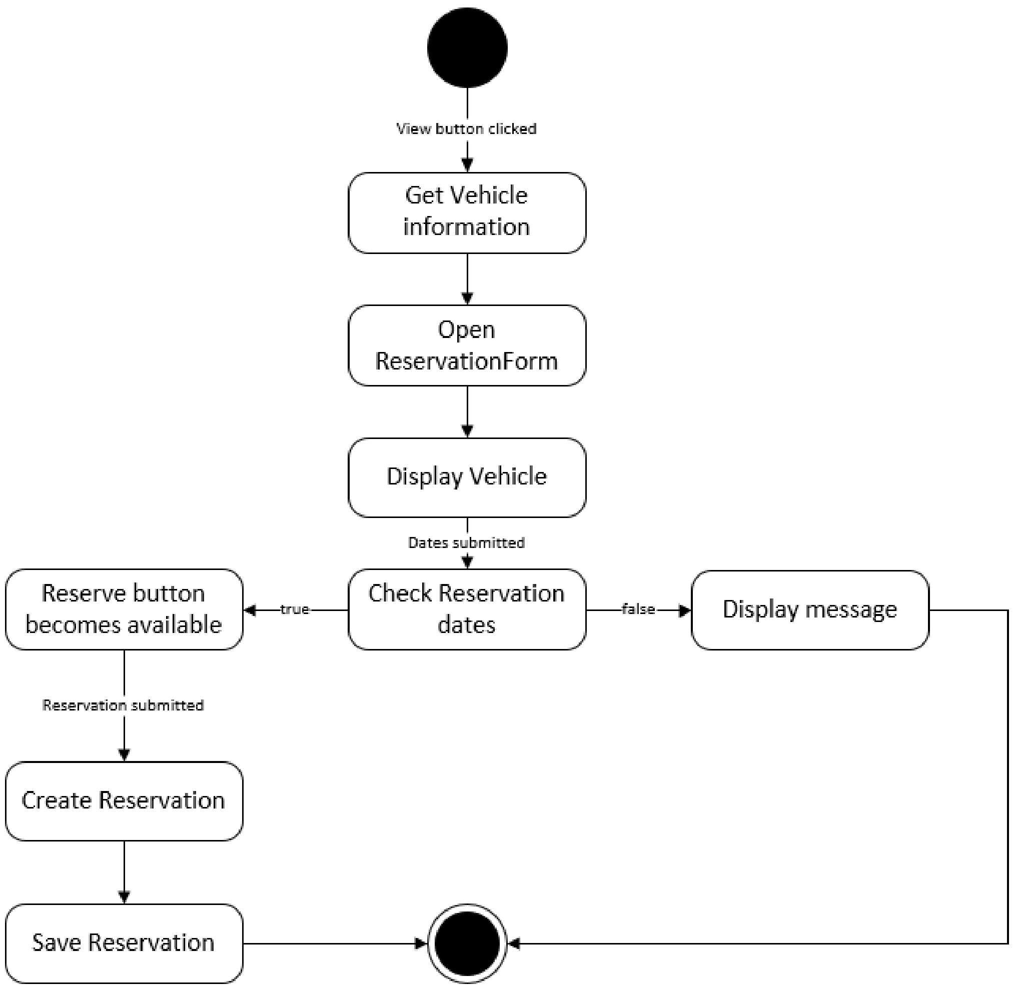


Figure 4.9: ReserveControl

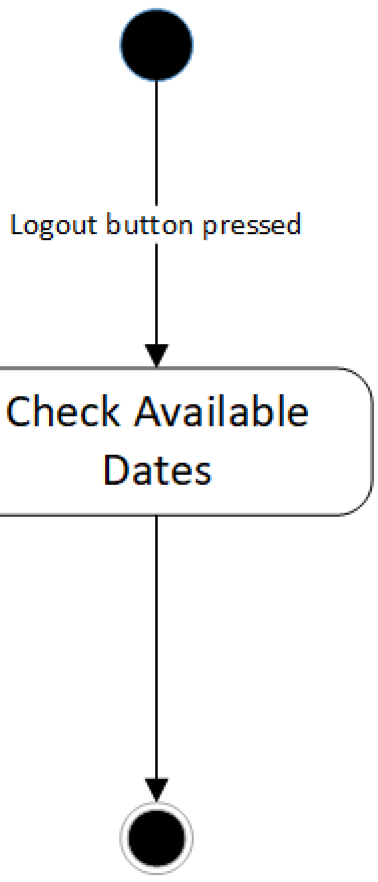


Figure 4.10: AvailControl

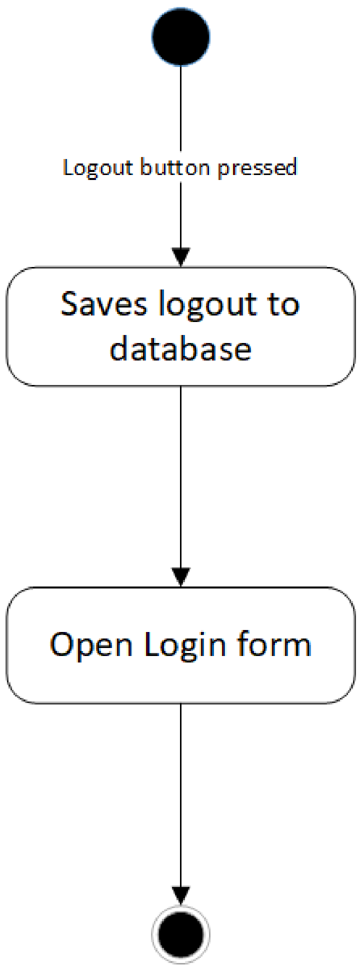


Figure 4.11: LogoutControl

5 SYSTEM DESIGN

SUBSYSTEM DECOMPOSITION

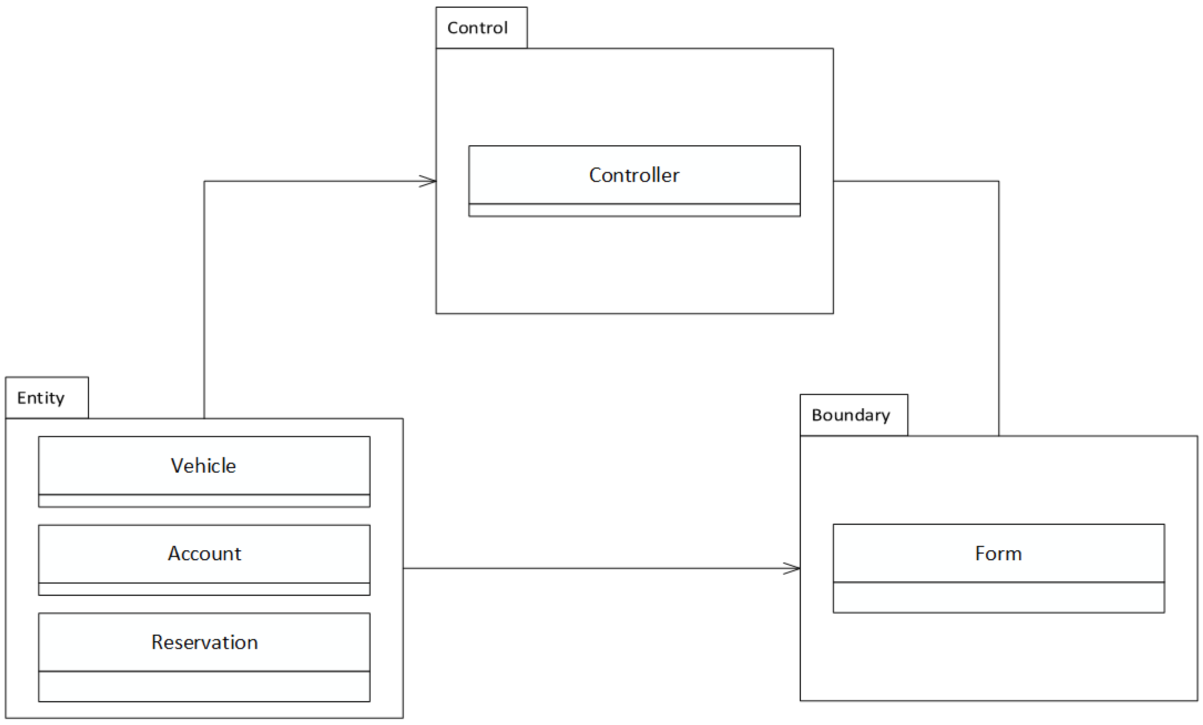


Figure 5.1: Subsystems

APPENDIX

APPENDIX A – SOURCE CODE

```
namespace Entity
{
    public class Reservation
    {
        private DateTime startDate;
        private DateTime endDate;
        private string username;
        private int vid;

        public Reservation(DateTime mstartDate, DateTime mendDate, string
musername, int mvid)
        {
            setStartDate(mstartDate);
            setEndDate(mendDate);
            setUsername(musername);
            setVid(mvid);
        }

        public DateTime getStartDate()
        {
            return startDate;
        }

        public void setStartDate(DateTime mstartDate)
        {
            startDate = mstartDate;
        }

        public DateTime getEndDate()
        {
            return endDate;
        }

        public void setEndDate(DateTime mendDate)
        {
            endDate = mendDate;
        }

        public string getUsername()
        {
            return username;
        }

        public void setUsername(string musername)
        {
            username = musername;
        }

        public int getVid()
        {
            return vid;
        }

        public void setVid(int mvid)
        {
            vid = mvid;
        }
    }
}
```

```
namespace Entity
{
    class Account
    {
        private int id;
        private string username;
        private string type;

        public Account(int mid, string musername, string mtype)
        {
            setId(mid);
            setUsername(musername);
            setType(mtype);
        }

        public int getId()
        {
            return id;
        }

        public void setId(int mid)
        {
            id = mid;
        }

        public string getUsername()
        {
            return username;
        }

        public void setUsername(string musername)
        {
            username = musername;
        }

        public string getType()
        {
            return type;
        }

        public void setType(string mtype)
        {
            type = mtype;
        }
    }
}
```

```
namespace CarRentalSystem.Controllers
{
    public static class ReserveController // controller for Reserve use
case
    {
        public static Vehicle selectedVehicle; // public access to
currently selected vehicle
        public static void View(int vid)
        {
            selectedVehicle = DBConnector.GetVehicle(vid); // stores
selected vehicle info from database
            ReservationForm.Display(selectedVehicle); // display
ReservationForm to user with selected vehicle info
            ReserveInitial.instance.Hide();
        }
        public static bool CheckReservation(int vid, int start, int end) //
"go-between" from check reservation button to database
        {
            if (DBConnector.CheckReservation(vid, start, end) == true)
            {
                return true;
            }
            else return false;
        }
        public static void Reserve(int vid, int start, int end) // "go-
between" from reserve button to database
        {
            DBConnector.Save(vid, start, end);
        }
    }
}
```

```
namespace CarRentalSystem.Boundary
{
    public partial class ReserveInitial : Form // landing page after user
logs in with customer account
    {
        public static ReserveInitial instance; // public access to current
instance of ReserveInitial
        public int vid; // public access to vehicle id of user's selection
from ReserveInitial
        public ReserveInitial()
        {
            InitializeComponent();
            instance = this;
        }
        private void btn_car1_Click(object sender, EventArgs e) // 2021
Honda Civic
        {
            vid = 1;
        }
        private void btn_car2_Click(object sender, EventArgs e) // 2021
Subaru Outback
        {
            vid = 2;
        }
        private void btn_view_Click(object sender, EventArgs e) // View
button takes user to see more info about selected vehicle
        {
            if (vid > 0) // validation to ensure a selection must be made
before clicking View button
            {
                ReserveController.View(vid);
            }
        }
        private void btn_logout_Click(object sender, EventArgs e)
        {
            ReserveInitial.instance.Close();
        }
        private void ReserveInitial_FormClosed(object sender,
FormClosedEventArgs e)
        {
            LogoutControl.Logout(LoginControl.thisUser);
        }
    }
}
```