Project Report

COSC 4360

Fall 2021

Callan Noak
Noah Abbott
Kaitlin Lunt
Jamie McAndrew

Introduction:
This report describes software development steps in designing a software for buying and selling a vehicle. Topics that will be discussed in this paper are as follows:

1. Discuss the limitations of the software
2. Discuss the different use cases of the software
3. Go over a preliminary domain model that will be later expanded
4. Discuss the final class diagrams for the software
5. Analyze the most complex use case with an interaction diagram
6. Go over the state chart for the main object of the software
7. Provide a generalized activity diagram for any use case
8. Go over the test cases used for automated testing.

Each topic represents a different segment in the software design process that we used. It is important to understand the different steps in software design in order to create a fully functional program.

Limitations:
There is one current major limitation to our program. When navigating to the admin page, it takes a while for the page to appear as the database is loading all the information that will need to be displayed. Currently the wait time is about one minute. During that time the software will freeze as the loading is done. This is a source for future improvement to the software. In the future, we can add an indicator that the database is loading information so that the user knows what is happening. We can also change the code so that the database does not take as long to load.

Use Cases:
In this section we will be going over the different use cases for the software and will end with an overall use case diagram. The use case diagram is shown in Figure 1.

Actors:
1. Buyers
2. Sellers
3. Admins
4. Users (Buyers, Sellers, and Admins are all Users)

Use Cases:
1. Log into software (‘Login’)
2. Purchase a vehicle (‘Buy Vehicle)
3. Sell a vehicle (‘Sell Vehicle’)
4. Check vehicle listings (‘View Listings’)

5. Get new password (‘Forgot Password’)
6. Register as new user (‘Register’)
7. Create a new admin (‘Create Admin’)
8. Check Previous Sales (‘View Previous Sales’)
9. Check list of users (‘View User List’)
10. Check lists of sale requests (‘View Requests’)
11. Accept or decline requests (‘Accept/Decline Requests’)

Use Case Descriptions:

1. <u>Login</u>
    a. Login: basic course of events
        i. User logs in to software

2. <u>Buy Vehicle</u>
    a. Buy Vehicle: basic course of events
        i. User logs in to software
        ii. System displays current listing of vehicles
        iii. User selects vehicle from listing
        iv. System displays information about vehicle
        v. User selects purchase vehicle
        vi. System asks for payment information
        vii. User provides payment
        viii. System remove vehicle from listing gives receipt of purchase

    b. Buy Vehicle – Vehicle Sold on Website: alternative course of events
        i. User logs in to software
        ii. System displays current listing of vehicles
        iii. User selects vehicle from listing
        iv. System displays information about vehicle
        v. User selects purchase vehicle
        vi. System detects that vehicle is be sold on a website and asks the user if they wish to be taken to their website.
        vii. If yes, open website in browser
        viii. If no, cancel purchase

    c. Buy Vehicle – Invalid Payment: exceptional course of events
        i. User logs in to software
        ii. System displays current listing of vehicles
        iii. User selects vehicle from listing
        iv. System displays information about vehicle
        v. User selects purchase vehicle
        vi. System asks for payment information

vii. User provides payment
viii. System detects invalid payment and asks user if they wish to try again with a new payment method
ix. If yes, user provides new payment method and continues with basic course of events
x. If no, cancel purchase

3. <u>Sell Vehicle</u>
    a. Sell Vehicle: basic course of events
        i. User logs in to software
        ii. System displays current listing of vehicles
        iii. User selects sell vehicle option
        iv. System provides an application that asks for the vehicles information
        v. User fills out application and selects submit
        vi. System displays an estimated selling price for the car and asks the user if they would like to use this price
        vii. If no, user enters their own selling price
        viii. If yes, the estimated price is used
        ix. System asks for final confirmation to sell vehicle
        x. If no, cancel selling
        xi. If yes, vehicle is added to listings and user is sent a confirmation

    b. Sell Vehicle – Link Vehicle to website: alternate course of events
        i. User logs in to software
        ii. System displays current listing of vehicles
        iii. User selects sell vehicle option
        iv. System provides an application that asks for the vehicles information
        v. User fills out application and fills out 'sold on website' section and provides the website link
        vi. System sets price to 'Shown on website'
        vii. System asks for final confirmation to sell vehicle
        viii. If no, cancel selling
        ix. If yes, vehicle is added to listings and user is sent a confirmation

    c. Sell Vehicle – Incorrect Information: exceptional course of events
        i. User logs in to software
        ii. System displays current listing of vehicles
        iii. User selects sell vehicle option
        iv. System provides an application that asks for the vehicles information
        v. User fills out application and selects submit
        vi. System detects that some information may be incorrect (enter 1821 for year model) and asks user to correct information
        vii. If no, cancel selling

viii. If yes, user enters corrected information and selects submit
ix. Continue with basic course of events starting from vi

4. <u>View Listings</u>
    a. View Listings: basic course of events
        i. User logs in to software
        ii. System displays current listing of vehicles
5. <u>Forgot Password</u>
    a. Forgot Password: basic course of events
        i. User selects forgot password option
        ii. User enters email information for recovery code
        iii. User enters recovery code
        iv. User enters new password

6. <u>Register</u>
    a. Register: basic course of events
        i. User selects registration option
        ii. User fills out registration form

7. <u>Create Admin</u>
    a. Create Admin: basic course of events
        i. User logs in to software (must be a current admin)
        ii. User selects create admin tab
        iii. User enters username of the user they wish to make an admin

8. <u>View Previous Sales</u>
    a. View Previous Sales: basic course of events
        i. User logs in to software (must be a current admin)
        ii. User selects previous sales tab

9. <u>View User List</u>
    a. View User List: basic course of events
        i. User logs in to software (must be a current admin)
        ii. User selects User List tab

10. <u>View Requests</u>
    a. View Requests: basic course of events
        i. User logs in to software (must be a current admin)
        ii. User selects Requests tab

11. View Requests
    a. View Requests: basic course of events
        i. User logs in to software (must be a current admin)
        ii. User selects Requests tab
        iii. User selects either the accepts or decline option for any given request

12. All Cases
    a. All Cases – Failed Login Attempt: exceptional course of events
        i. User logs in to software
        ii. System detects incorrect login information and asks user to enter information again
        iii. If no, cancel login
        iv. If yes, user attempts to log in again
        v. Continue with basic course of events for case

Use Case Diagram:

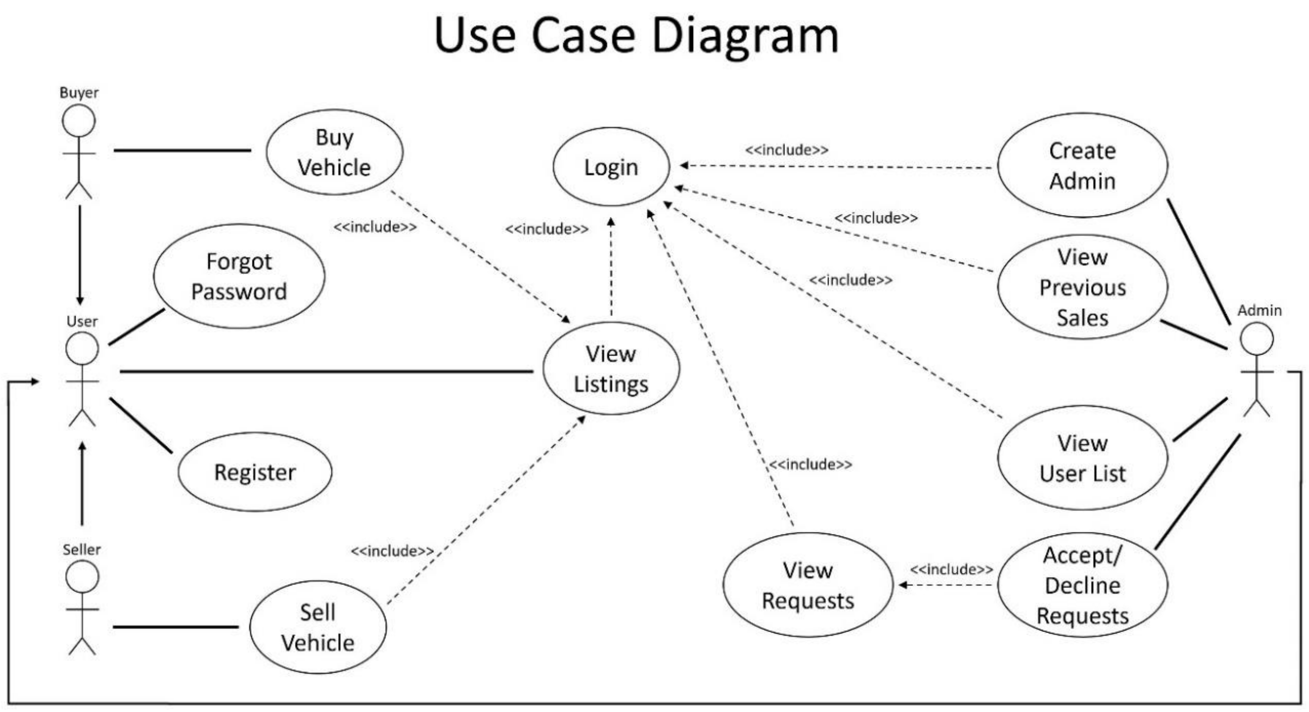

Figure 1: Use Case Diagram

Domain Model:
The domain model shows the early thought process of the classes that we would need to implement our design. Our software is centered around three objects: User, Vehicle, and Listing. Some of the early interactions between these classes that we came up with is:

1. A User can check a Listing to purchase a Vehicle
2. A User can sell a Vehicle to create a Listing
3. A User can view Listings
4. A User that is an admin can make another User an admin
5. A User that is an admin can accept or decline a pending Listing
6. A User that is an admin can view previous Listings that have been purchased
7. A User that is an admin can view a list of other Users

A graphic of our domain model is shown in Figure 2.

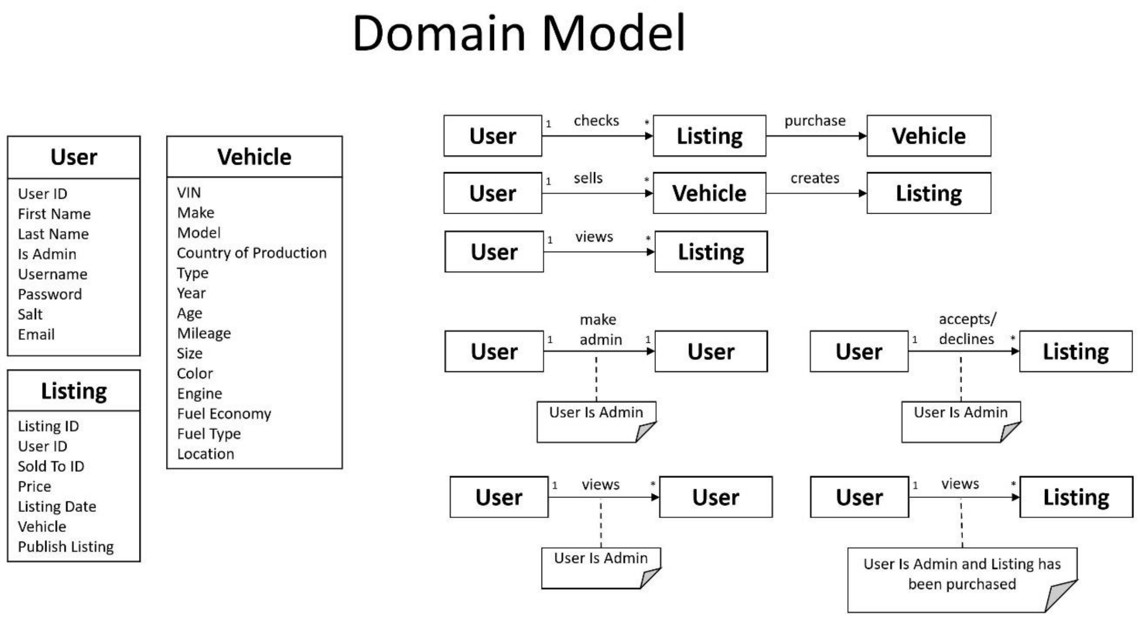

Figure 2: Domain Model

Class Diagram:
The class diagram further extends the domain model to show all of the main classes of the software. In addition to the User, Vehicle, and Listing classes from our domain model, we have added the Authentication, MemoryCache, UserManager, ListingManager, Vehicle Manager, UserInfo, ListingInfo, Request, and PreviousSale classes.

Each class has their own specific purpose: The MemoryCache class is for storing information about the current session on the database. The Authentication class is for checking a user's credentials to allow them to log in. All the manager classes are for handling their respective objects and saving information to the database. These classes are the main workhorses of the program. The UserInfo and ListingInfo classes are for formatting information to be ready to display in tables. The Request class is for handling and formatting requests. The PreviousSale class is for handling and formatting listings that have been sold. Our class diagram is shown in Figure 3. A pdf of the class diagram will be provided along with this report so that it is easier to read.

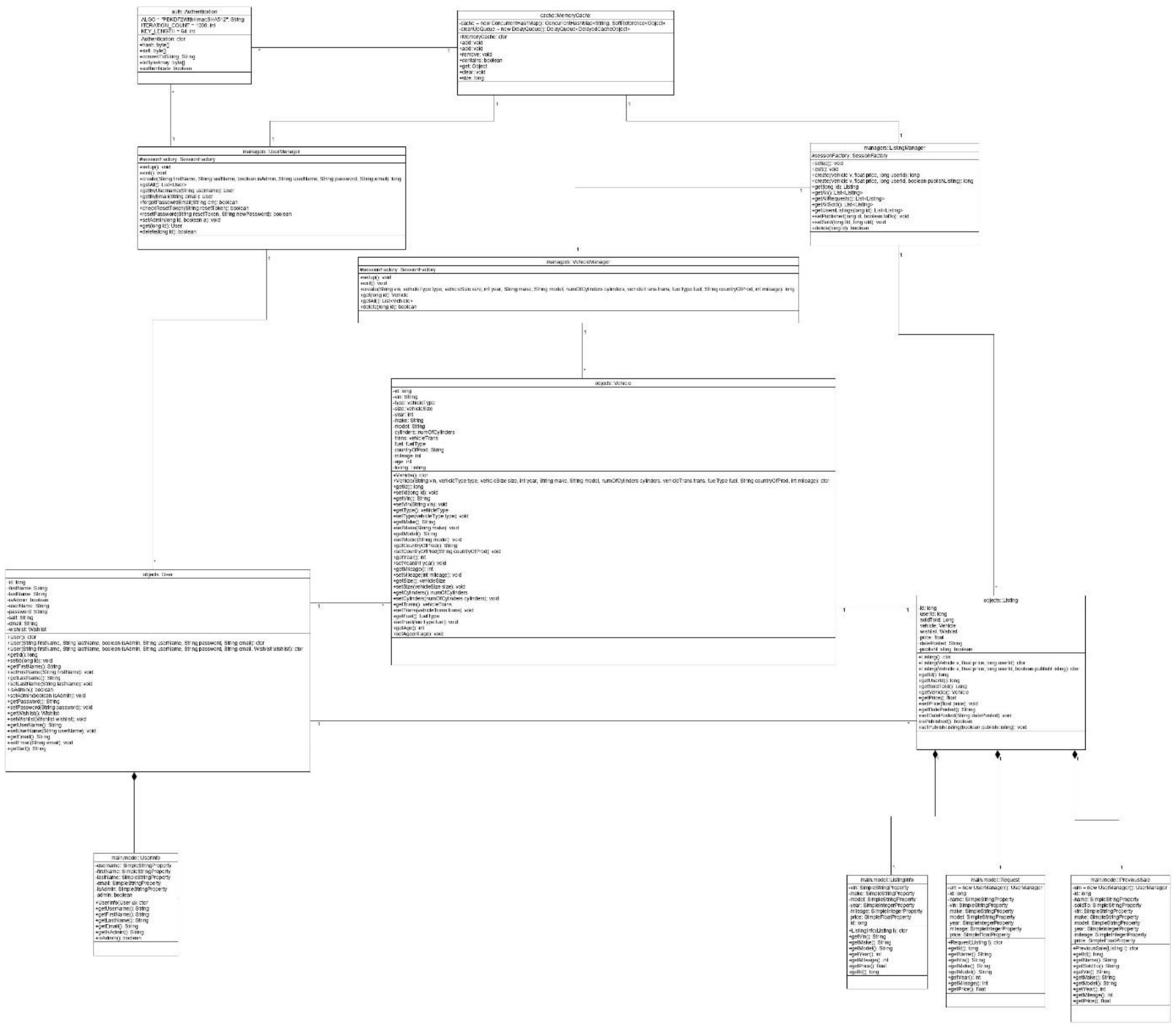

Figure 3: Class Diagram

Interaction Diagram:
Interaction diagrams are used to show the interactions between the different classes for a given use case. For this example, multiple use cases are combined to create a more complex use case. In this use case, we are combining the use cases Register, Login, Sell Vehicle, Buy Vehicle. The steps in this use case are:

1. The new user must register to gain access to the system
2. The user logs into the system.
3. The user sells their own vehicle by filling out a sale form
4. The user selects and purchases a different vehicle.

The interaction diagram showing this use case is shown in Figure 4.

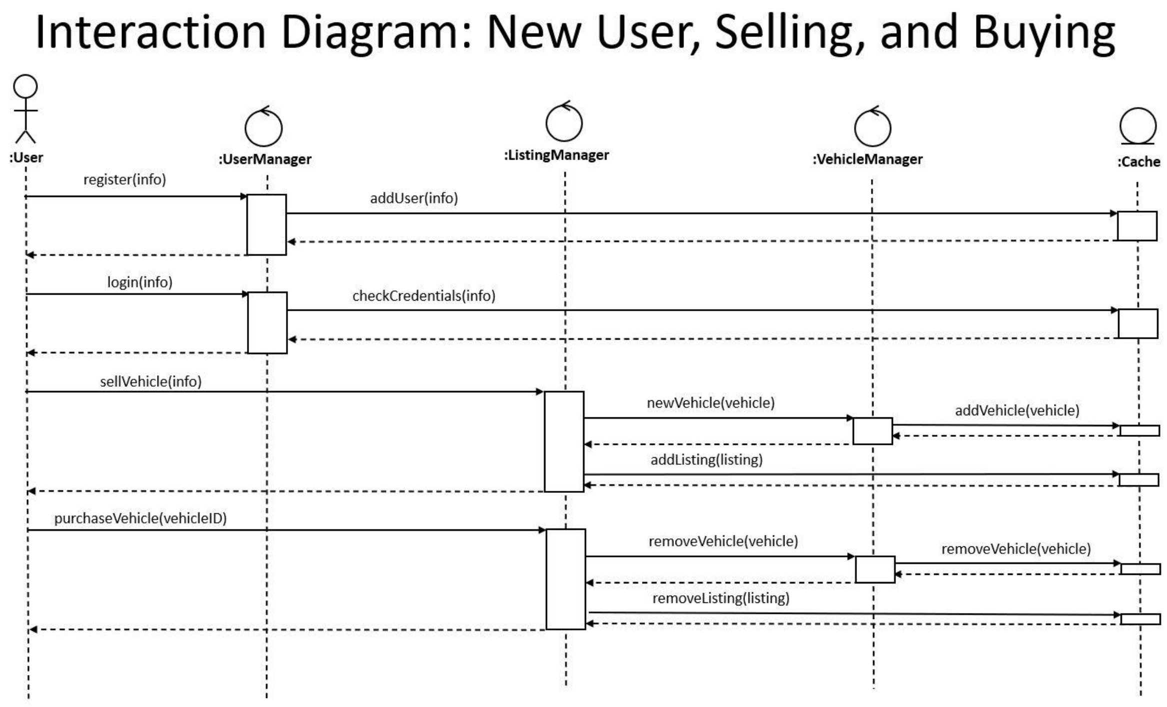

Figure 4: Interaction Diagram

State Chart:
State charts show different possible states that a particular object can be in at any given time. Our main object is the user that can be in a number of different states depending on what they are currently doing. The states for a user are:

1. Logging in
2. Registering as a new user
3. Changing their password
4. Viewing the vehicle listing
5. Viewing their personal listings
6. Purchasing a vehicle
7. Selling a vehicle
8. Viewing listing requests
9. Viewing a list of users
10. Viewing previous sales
11. Making another user an admin

The state chart for the User object is shown in Figure 5:

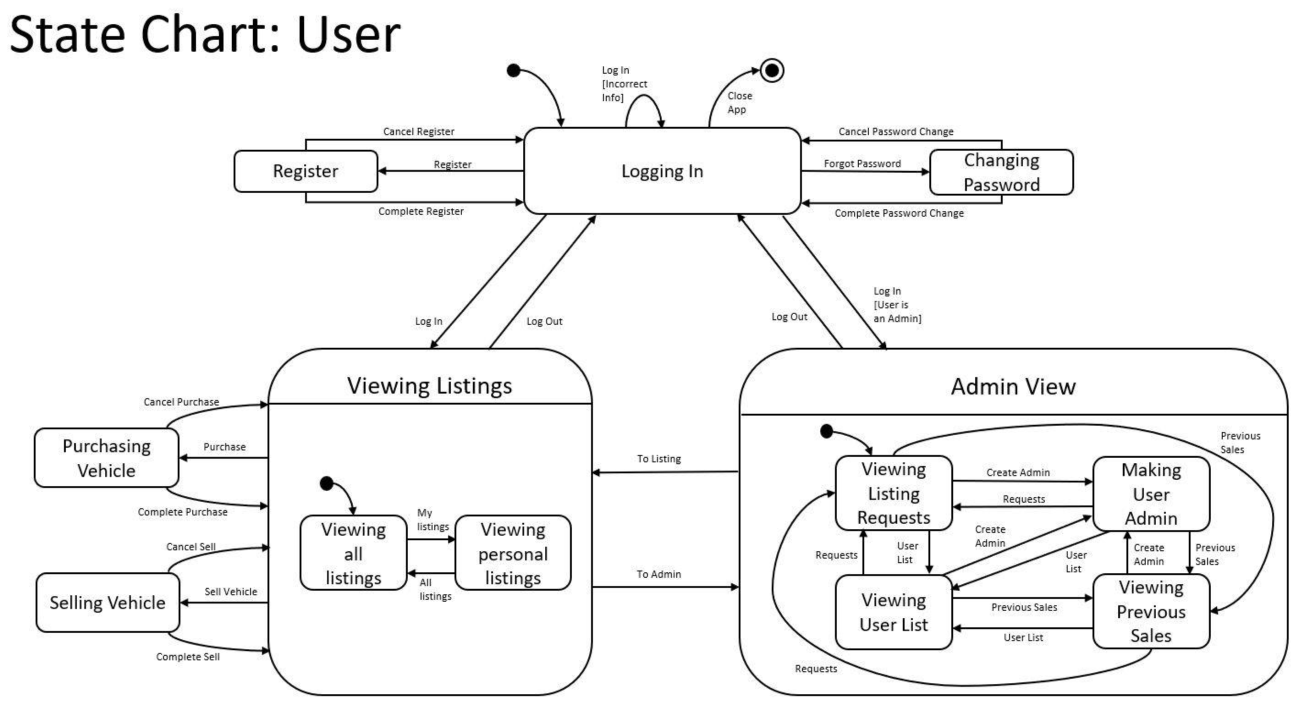

Figure 5: State Chart

Activity Diagram:
Activity Diagrams show all the different paths a user can take when using a piece of software. For our software, a user will always log in first to access the bulk of the system. The only other thing they may do first is either register if they are a new user or reset their password if the forgot theirs. Once logged in all the functions are available to the user if they are an admin. Users who are not admins are limited to buying and selling vehicles. The activity diagram for our software is shown in Figure 6.

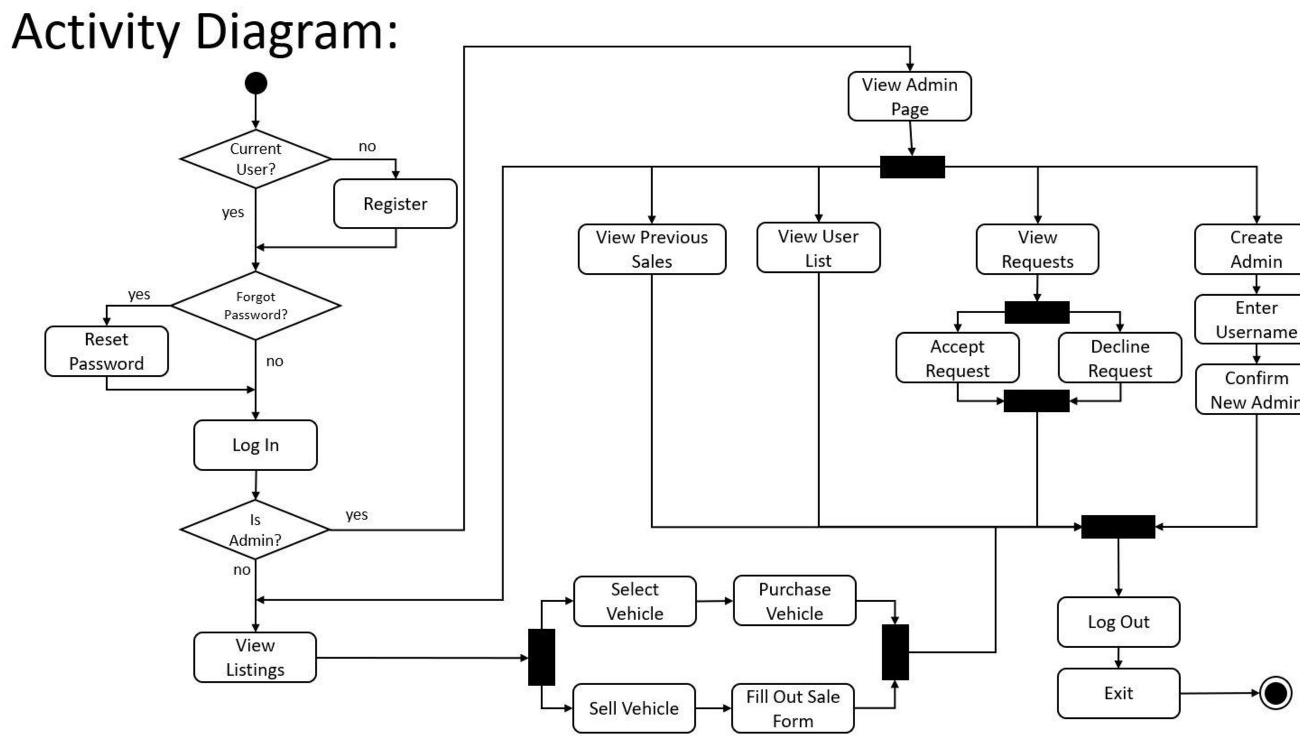

Figure 6: Activity Diagram

Test Cases:
To test our software, we tested the two main workhorse classes. These classes are the ListingManager and the UserManager. Both classes are used in almost every function of the program. We tested every method from each class to make sure they are working correctly. For the ListingManager class we created a default listing of:

Vehicle v = new Vehicle(genVIN(), vehicleType.**coupe**, vehicleSize.**mid**, 2016, "Toyota", "Camry", numOfCylinders.**six**, vehicleTrans.**automatic**, fuelType.**gasoline**, "Japan", 60000)
float price = 30000
long userId = 1
ListingManager.create(v, price, userID)

We needed to generate a new VIN number for each listing as to not create duplicates. For the UserManager class we created a default user of:

```
UserManager.create("Test", "Testerson", false, "ttesterson", "Test1234!", "test@test.com")
```

For both classes, the default cases were used to test every method in the class to make sure they were functioning correctly.