

Anthony Pompili

CSC 415

Dr. Pulimood

Assignment 5 - Open Source Software: Analysis and Design

November 10, 2017

Link: [https://github.com/pomps8/CSC415\\_AnthonyPompili\\_ECar](https://github.com/pomps8/CSC415_AnthonyPompili_ECar)

Github Repository: pomps8/CSC415\_AnthonyPompili\_ECar

Project: E-Car

## Use Case Descriptions

|  |   |
|--|---|
| <b>Use case:</b> Check Singular Car in Carlist   |   |
| <p><b>Iteration:</b> 1, last modification: November 9, 2017</p> <p><b>Primary Actor:</b> E-car App user</p> <p><b>Goal in context:</b> Browse the catalog of cars and view specific car</p> <p><b>Preconditions:</b> Application must be downloaded on iOS Device, and the user must have an account created</p> <p><b>Trigger:</b> User clicks on the “View car info” button in the main menu</p> <p><b>Scenario:</b></p> <ol style="list-style-type: none"><li>1. User logs into the E-Car app</li><li>2. User selects the “View Car info” button</li><li>3. User selects the brand of car</li><li>4. User selects the model of car</li><li>5. User selects the year of the car</li><li>6. Car data is outputted to the users iOS Device</li></ol> | <p><b>Exceptions:</b></p> <ol style="list-style-type: none"><li>1. Data base is not up to date and does not contain a specific car</li><li>2. Car data was pulled incorrectly, and data is not correct</li><li>3. No image / data is available</li></ol> <p><b>Priority:</b> High priority, to be implemented after foundations to the application are complete</p> <p><b>When available:</b> second increment</p> <p><b>Frequency of use:</b> Semi-Frequent</p> <p><b>Chanel to actor:</b> Through E-Car Application</p> <p><b>Secondary Actors:</b> Webscraper to update information in list</p> <p><b>Channels to secondary actors:</b></p> <ol style="list-style-type: none"><li>1. Webscraper: Database access</li></ol> <p><b>Open Issues:</b></p> <ol style="list-style-type: none"><li>1. Should webscraper be implemented through swift file? Or external development + push to database to be pulled by app?</li><li>2. How / where to save data, database?</li></ol> |

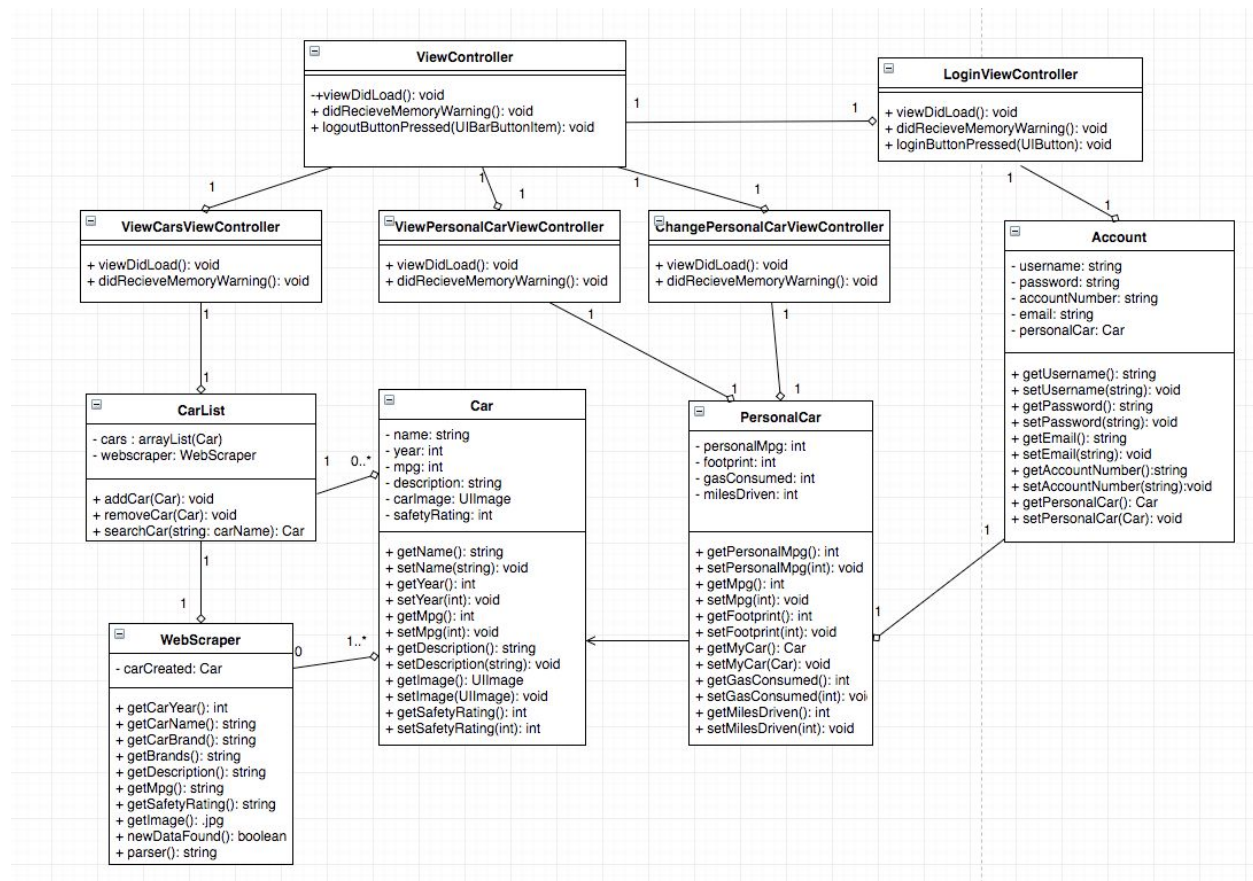
| <b>Use case:</b> Add Data (Gas consumption and Mileage)  |   |
|--|---|
| <p><b>Iteration:</b> 1, last modification: November 9, 2017</p> <p><b>Primary Actor:</b> E-car App user</p> <p><b>Goal in context:</b> Add gas consumption and mileage to users data</p> <p><b>Preconditions:</b> Application must be downloaded on iOS Device, the user must have an account created and a personal car set.</p> <p><b>Trigger:</b> User clicks on the “View Personal Car” button in the main menu</p> <p><b>Scenario:</b></p> <ol style="list-style-type: none"> <li>1. User logs into the E-Car app</li> <li>2. User selects the “View Personal Car” button</li> <li>3. User selects the “Add gas” button at the bottom</li> <li>4. User inputs valid info for gas consumed and current mileage</li> <li>5. User info is updated (Footprint, MPG, mileage, etc.)</li> </ol> | <p><b>Exceptions:</b></p> <ol style="list-style-type: none"> <li>1. User does not have personal car set</li> <li>2. Negative / zero gas is inputted into the system</li> <li>3. Mileage set is less than last inputted data set</li> </ol> <p><b>Priority:</b> High priority, to be implemented after foundations to the application are complete</p> <p><b>When available:</b> third increment</p> <p><b>Frequency of use:</b> Highly Frequent</p> <p><b>Chanel to actor:</b> Through E-Car Application</p> <p><b>Secondary Actors:</b> None</p> <p><b>Channels to secondary actors:</b></p> <ol style="list-style-type: none"> <li>1. None</li> </ol> <p><b>Open Issues:</b></p> <ol style="list-style-type: none"> <li>1. How to make sure user sets up personal car first?</li> <li>2. How to calculate mpg, footprint?</li> <li>3. How to set up list of inputs?</li> <li>4. How to edit / delete from this list?</li> </ol> |

|   |  |
|---|--|
| <b>Use case:</b> Change Personal Car  |  |
| <p><b>Iteration:</b> 1, last modification: November 9, 2017</p> <p><b>Primary Actor:</b> E-car App user</p> <p><b>Goal in context:</b> Browse the catalog of cars and find new car to set as personal car</p> <p><b>Preconditions:</b> Application must be downloaded on iOS Device, and the user must have an account created</p> <p><b>Trigger:</b> User clicks on the “Change Personal Car” button in the main menu</p> <p><b>Scenario:</b></p> <ol style="list-style-type: none"> <li>1. User logs into the E-Car app</li> <li>2. User selects the “Change Personal Car” button</li> <li>3. User is presented a pop-up saying “If you change your personal car, all data stored on car will be lost”</li> <li>4. User goes through picking car just like at set up. View similar to seeking a car through “View Car Info” sequence</li> </ol> | <p><b>Exceptions:</b></p> <ol style="list-style-type: none"> <li>1. Data base is not up to date and does not contain the specific car</li> </ol> <p><b>Priority:</b> Medium priority, to be implemented after basics to the application are completed</p> <p><b>When available:</b> second increment</p> <p><b>Frequency of use:</b> Semi-Frequent</p> <p><b>Chanel to actor:</b> Through E-Car Application</p> <p><b>Secondary Actors:</b> None</p> <p><b>Channels to secondary actors:</b></p> <ol style="list-style-type: none"> <li>1. None</li> </ol> <p><b>Open Issues:</b></p> <ol style="list-style-type: none"> <li>1. How to delete old car data?</li> <li>2. Is there a way to let the user have multiple cars instead of deleting old data?</li> </ol> |

| <b>Use case:</b> Set Personal Car   |   |
|---|---|
| <p><b>Iteration:</b> 1, last modification: November 9, 2017</p> <p><b>Primary Actor:</b> E-car App user</p> <p><b>Goal in context:</b> Browse the catalog of cars and select personal car for app usage</p> <p><b>Preconditions:</b> Application must be downloaded on iOS Device, and the user must have an account created</p> <p><b>Trigger:</b> User clicks on the “View Car Info” button in the main menu</p> <p><b>Scenario:</b></p> <ol style="list-style-type: none"> <li>1. User logs into the E-Car app</li> <li>2. User selects the “View Car Info” button</li> <li>3. User is presented a pop-up saying “You do not have a current personal car set up, would you like to make one?”</li> <li>4. User selects “Yes” to continue</li> <li>5. User browse list just like searching for a car in the car list, and finds their car</li> <li>6. Data about car is inputted once car is found (Mileage)</li> </ol> | <p><b>Exceptions:</b></p> <ol style="list-style-type: none"> <li>1. Data base is not up to date and does not contain the specific car</li> <li>2. User selects “No” and goes back to main menu</li> <li>3. Bad info about mileage is inputted</li> </ol> <p><b>Priority:</b> High priority, to be implemented after foundations to the application are complete</p> <p><b>When available:</b> second increment</p> <p><b>Frequency of use:</b> Highly Frequent</p> <p><b>Chanel to actor:</b> Through E-Car Application</p> <p><b>Secondary Actors:</b> None</p> <p><b>Channels to secondary actors:</b></p> <ol style="list-style-type: none"> <li>2. None</li> </ol> <p><b>Open Issues:</b></p> <ol style="list-style-type: none"> <li>1. Create way for user to input data about car to estimate footprint size thus far prior to app use? (Est. mileage per week + average MPG = est. gas usage)</li> </ol> |

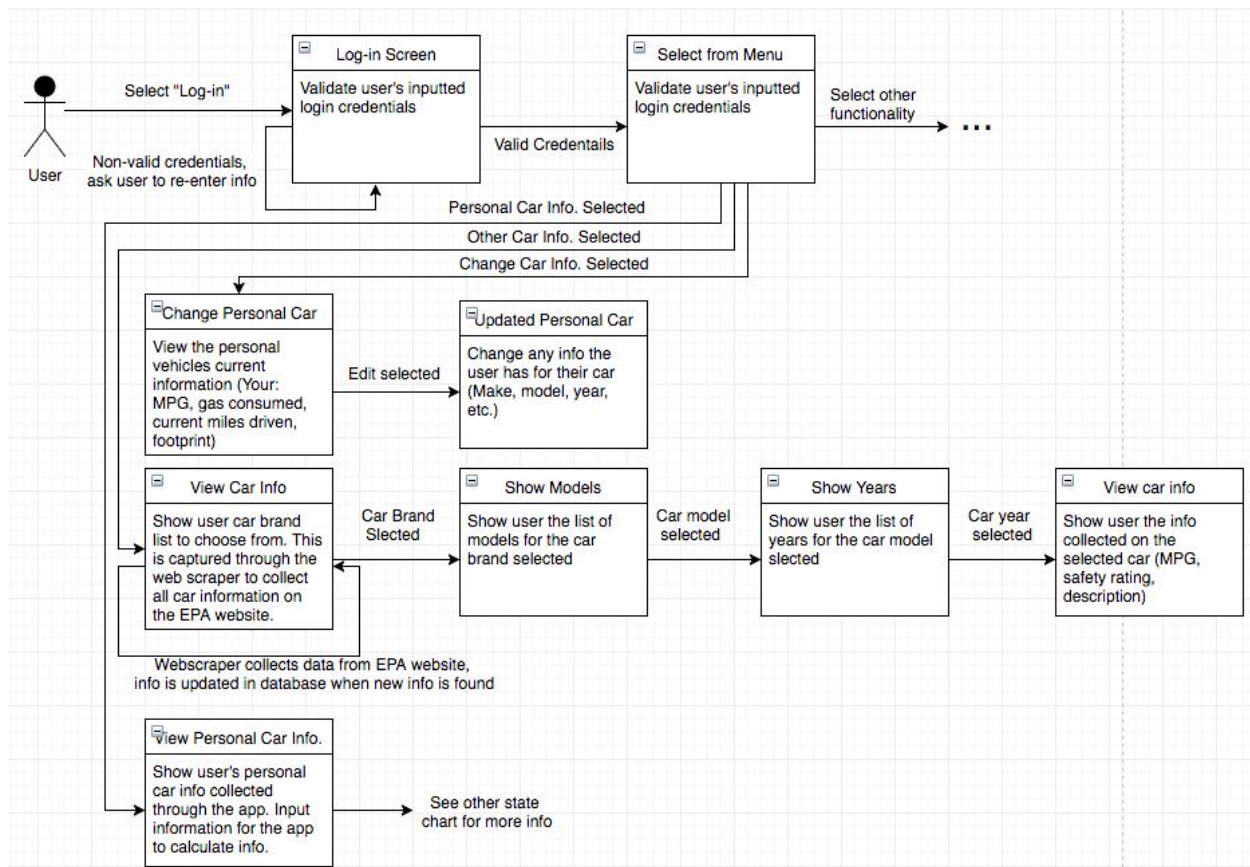
|   |  |
|---|--|
| <b>Use case:</b> Set up account   |  |
| <p><b>Iteration:</b> 1, last modification: November 9, 2017</p> <p><b>Primary Actor:</b> E-car App user</p> <p><b>Goal in context:</b> Browse the catalog of cars and view specific car</p> <p><b>Preconditions:</b> Application must be downloaded on iOS Device</p> <p><b>Trigger:</b> User clicks on the E-Car application on iOS Device</p> <p><b>Scenario:</b></p> <ol style="list-style-type: none"> <li>1. User opens application and selects "Log-in"</li> <li>2. User selects "create account"</li> <li>3. Basic account info is needed to create account (Email, username and password)</li> <li>4. Once email is validated through test email, the account is created and E-Car is ready to be used</li> </ol> | <p><b>Exceptions:</b></p> <ol style="list-style-type: none"> <li>1. Bad email is inputted</li> <li>2. Use does not have email to use, thus an account cannot be made</li> </ol> <p><b>Priority:</b> High priority, to be implemented after foundations to the application are complete</p> <p><b>When available:</b> second increment</p> <p><b>Frequency of use:</b> Highly Frequent</p> <p><b>Chanel to actor:</b> Through E-Car Application</p> <p><b>Secondary Actors:</b> None</p> <p><b>Channels to secondary actors:</b></p> <ol style="list-style-type: none"> <li>1. None</li> </ol> <p><b>Open Issues:</b></p> <ol style="list-style-type: none"> <li>1. How to have account validation through email?</li> <li>2. Where to store accounts?</li> <li>3. How to store account info (Passwords = main issue)?</li> </ol> |

## Detailed Design class diagram



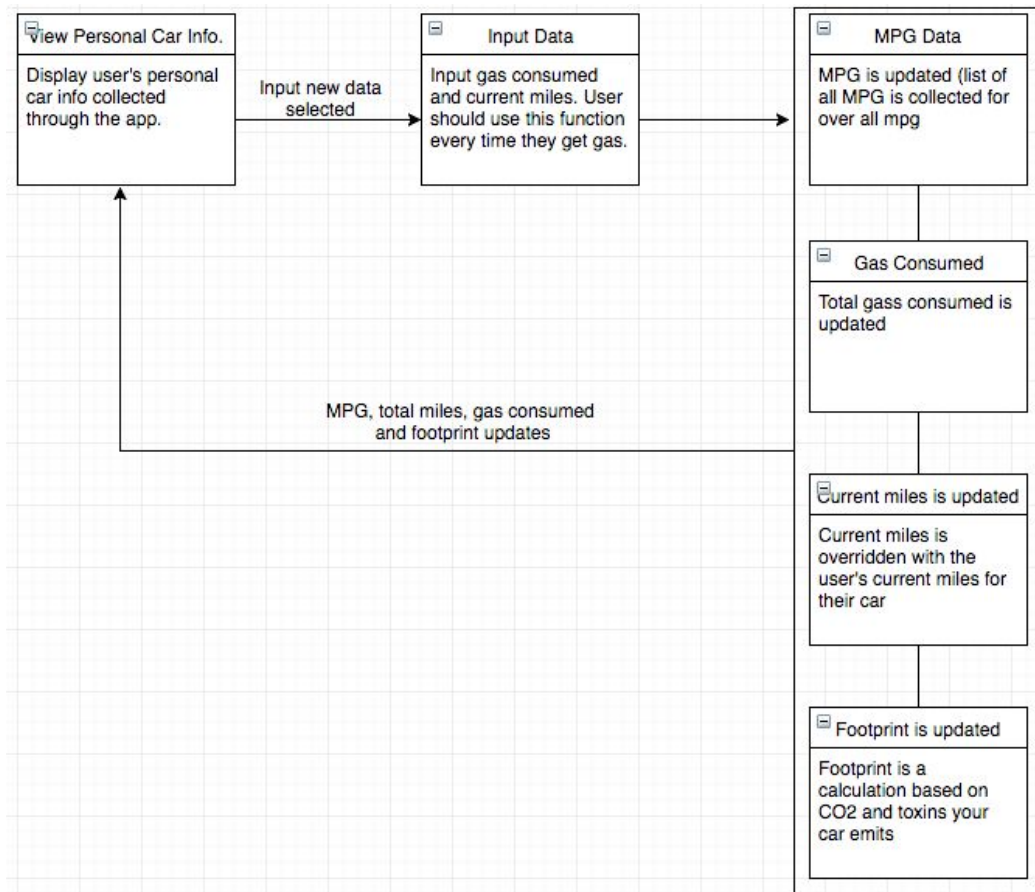
## Statecharts

### Overall System



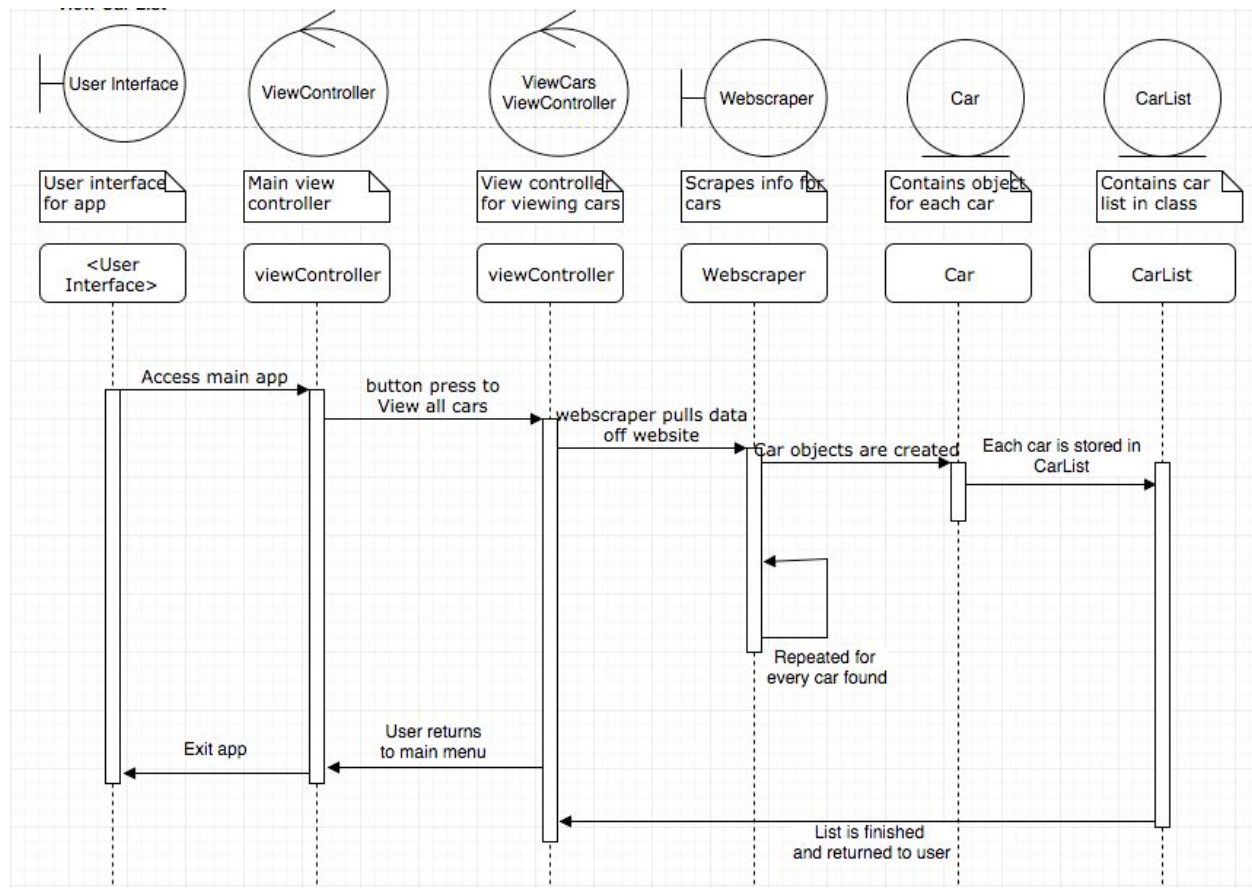


## Background calculations

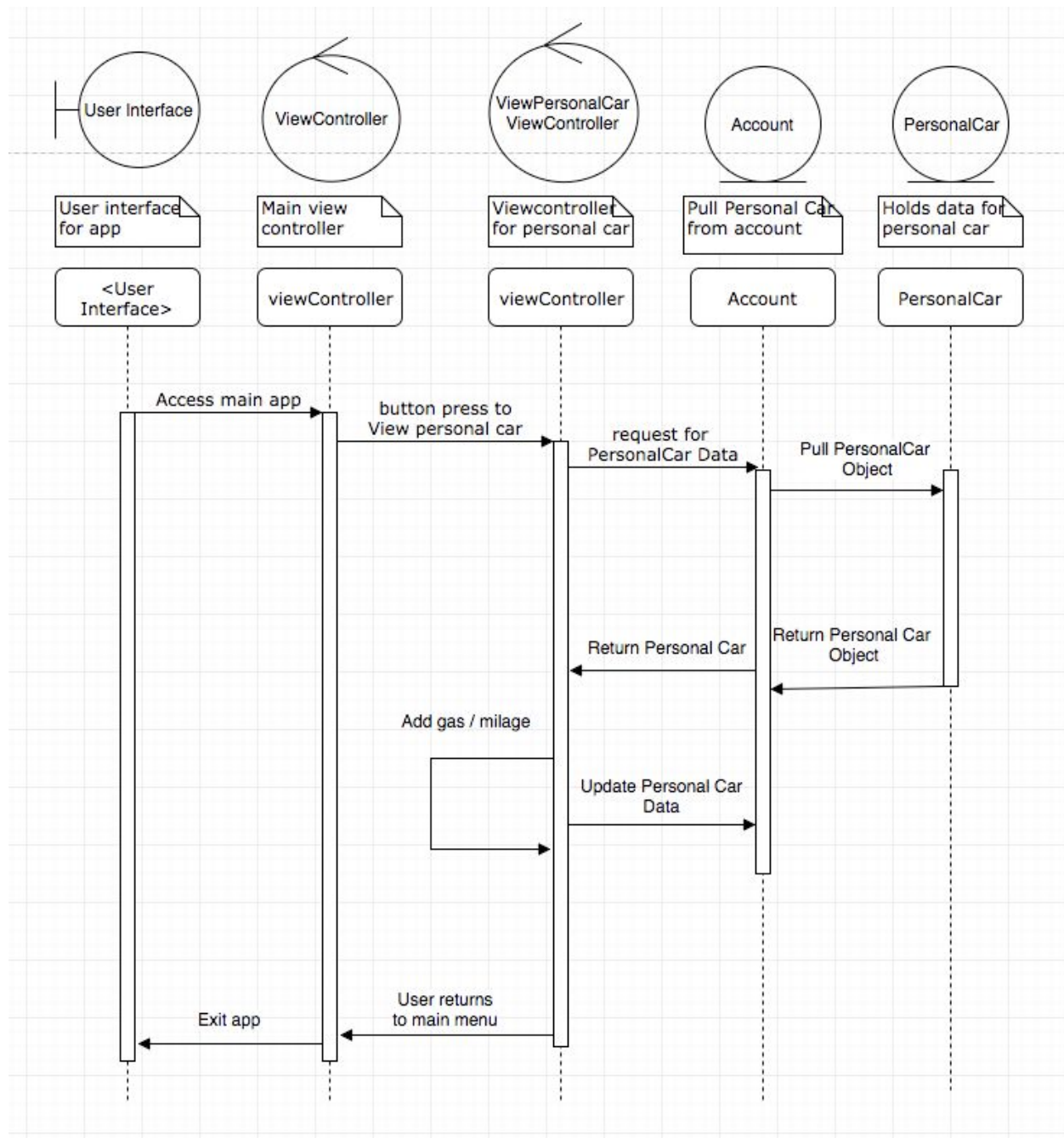


## Detailed SSD(s)

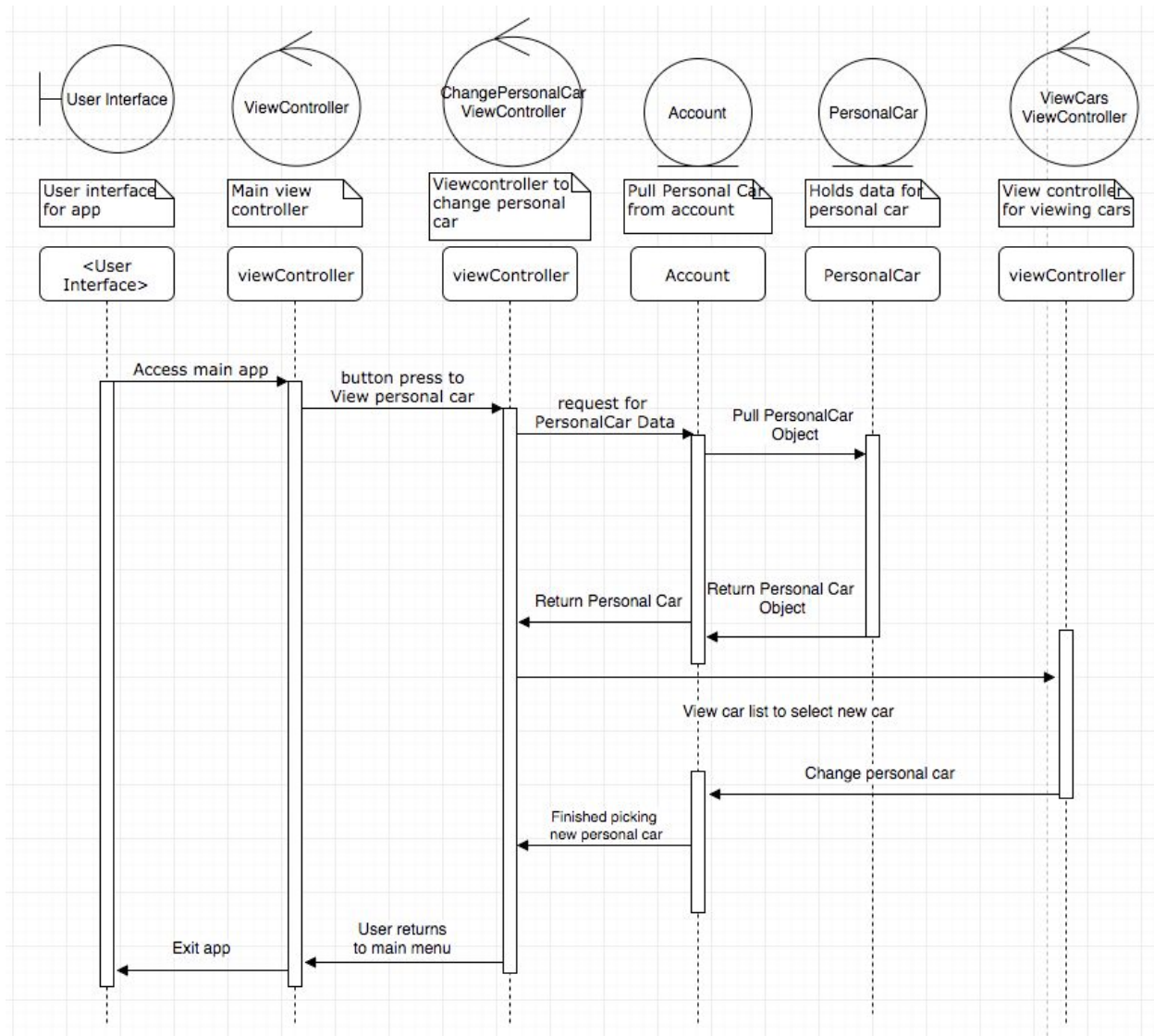
### Check Singular Car in Carlist



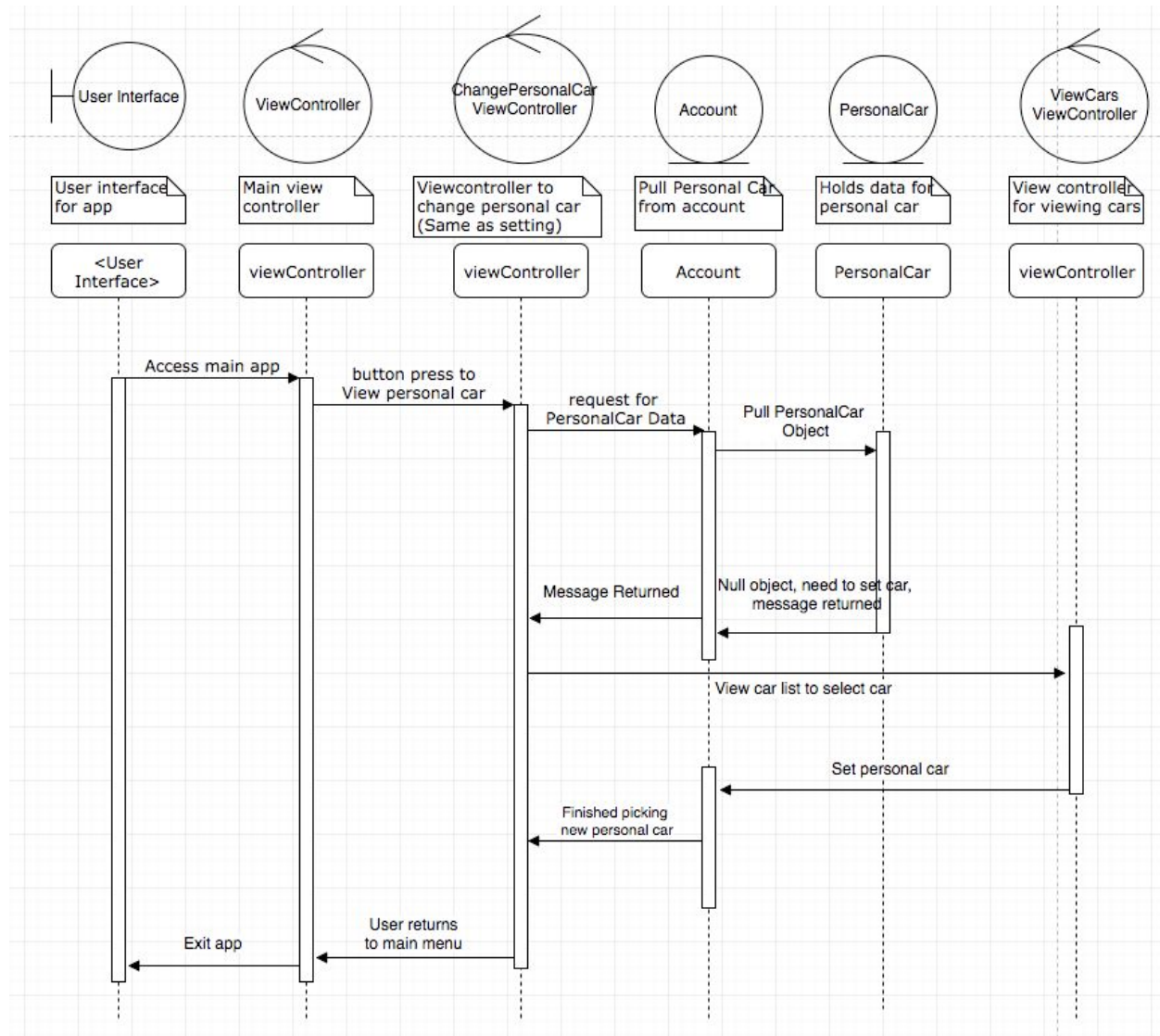
## Add Data (Gas consumption and Mileage)



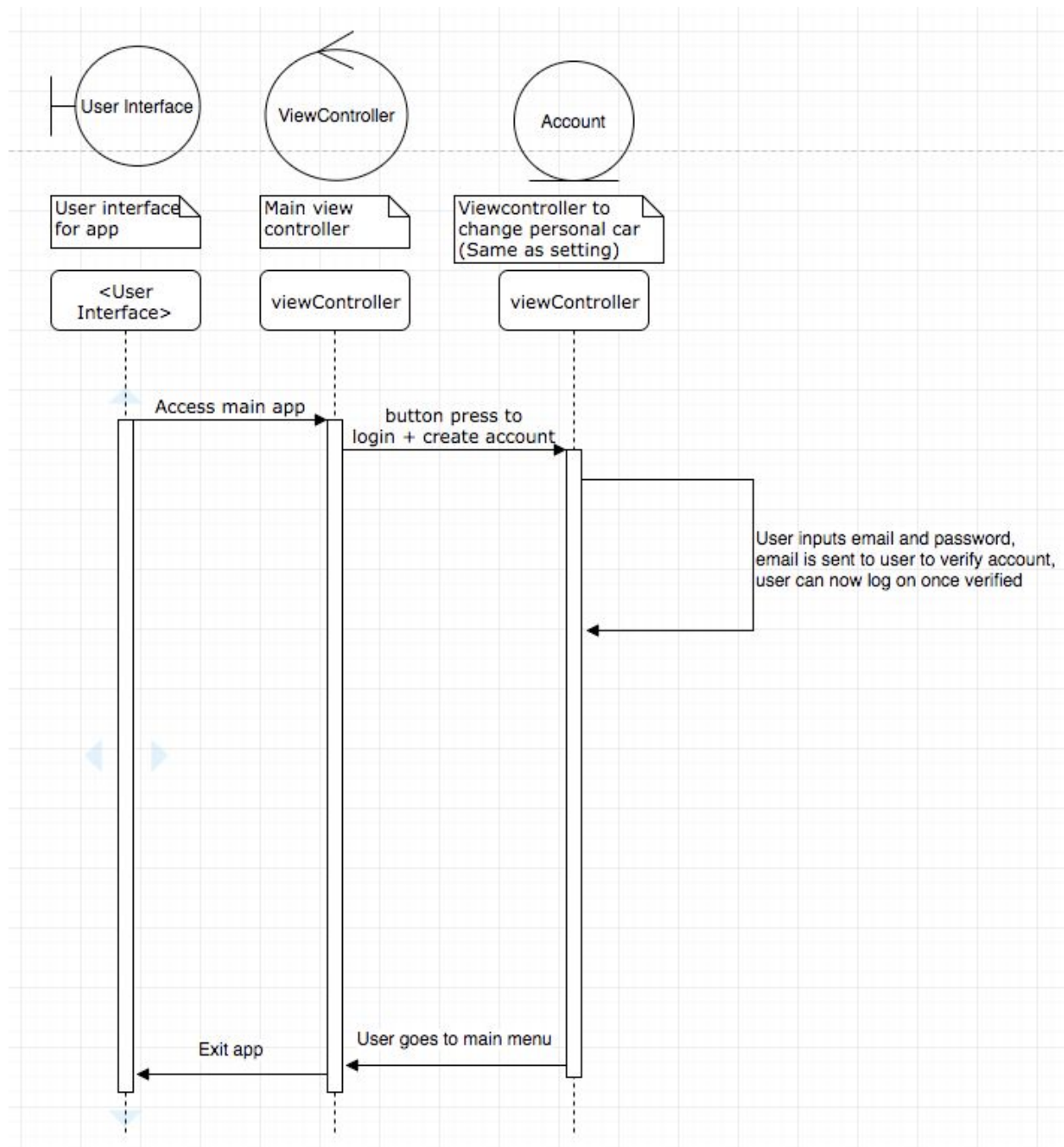
## Change Personal Car



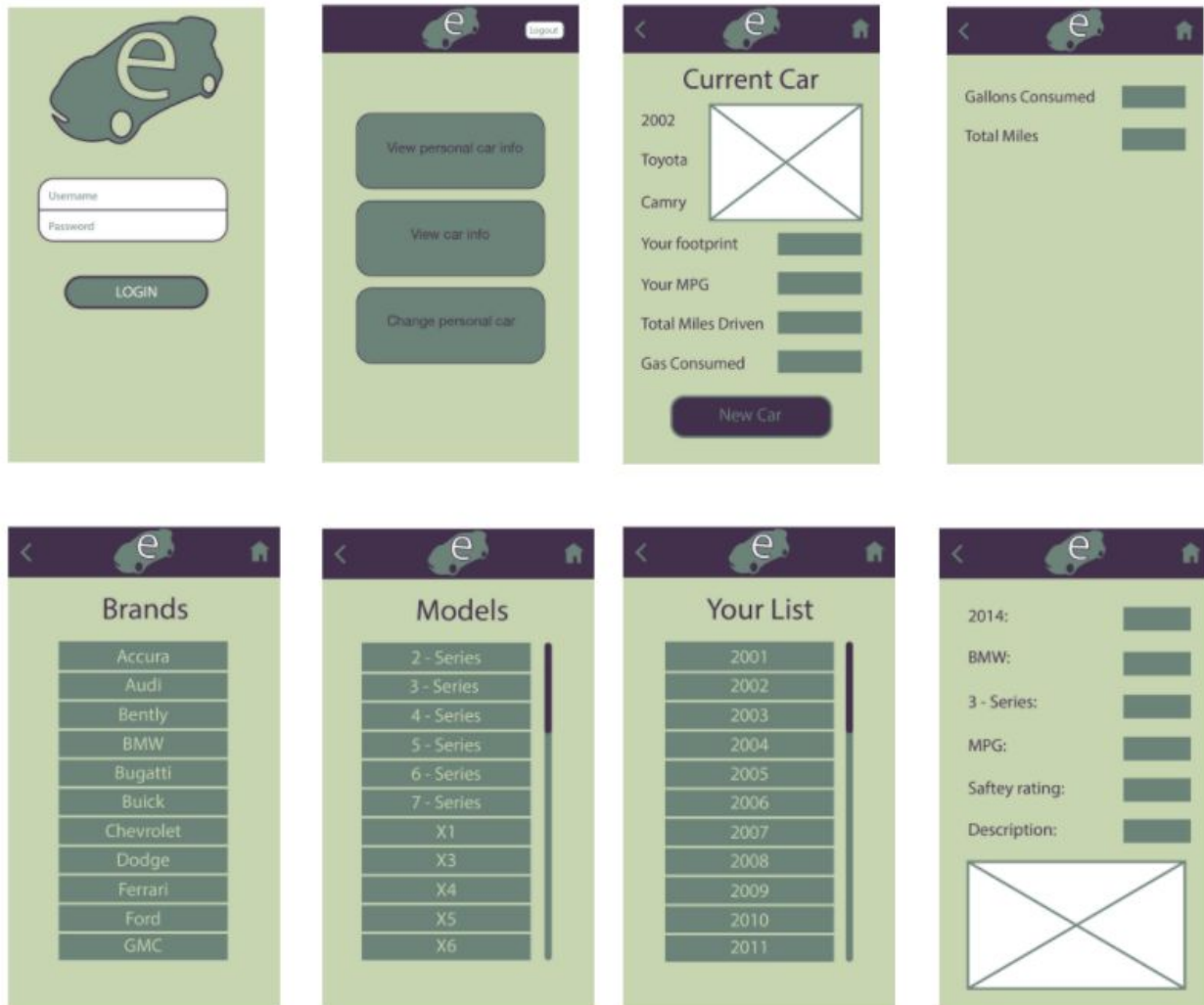
## Set Personal Car



## Setup Account



## User Interface Design



From top left to bottom right:

1. Log in screen
2. Main menu
3. Personal Car info
4. Gas + Mileage input screen
5. View car info Brand list
6. View car info Model list
7. View car info Year list
8. View car info for specific car



How my design reflects each of the 8 Golden Rules of design:

### **Rule 1: Strive for Consistency**

The user interface in each panel is completely consistent, from the color pallet, down to the layout of lists, and button layout. Green colors were chosen due to more “environmental” applications choose this color to reflect nature, and its cleanness. Additionally, the top portion of the application act as the navigation area, allowing the user to go back or to the home (main menu) whenever necessary.

### **Rule 2: Enable Frequent Users to Use Shortcuts**

The main shortcuts allowed in this application will be found on the top bar: the home and back buttons. Being that this interface allows the user to be nested in a very linear path, we want those users who know the app and do not want to go all the way through picking a car year for a specific car, to know they can hit the home button and go back to the beginning.

### **Rule 3: Offer Informative Feedback**

Any type of buttons pressed in the application will appear like they are being physically pressed, allowing the user to get the feedback they need to know that something was pressed. Any invalid information will not be accepted, resulting in a default pop-up box appearing on the screen.

### **Rule 4: Design Dialog to Yield Closure**

Any action that requires the user to input data, such as gas consumed and current miles, will have a “submit” button, which when pressed, and the data is validated, then a “success” message will be displayed that it was accepted.

### **Rule 5: Offer Simple Error Handling**

Although I attempt to make the system in such a way that errors will not happen, I am sure certain data input will result in errors. Having data checks on negative / extremely large numbers, making sure the user has a valid account, etc. will all have the correct error handling to ensure the data is checked properly / errors are dealt with in a proper manner to ensure the application will not fail.

### **Rule 6: Permit Easy Reversal of Actions**

To allow the user to reverse their actions, a back button was placed to go back to every pane in the path they are currently on. One main area they may wish to update / change is their current car, which will allow the user to change their car later on if they



need to do so. Additionally, seeing their full list of their gas inputs allows the user to edit / delete any input, in case of incorrect input.

#### **Rule 7: Support Internal Locus of Control**

This application allows the user to freely browse and input any data they need into the application. This data input covers their current car as well as gas and milage input. Anything the user wants to do, the user has the option to do, allowing them to believe that they have full control over anything inside the application.

#### **Rule 8: Reduce Short-Term Memory Load**

The user will have a delightful time using the application due to its ease of use. The flow of the application is easy to pick up for first time users, and these users will become very experienced shortly after use, and will become in tune with using any short cuts. This will allow them to just focus on what they're currently working on. All data is shown in a progressive manner, and the defaults for the application establish a mindful start up to the application.

## Modularity, Encapsulation, Algorithms, Data Structures

Modularity gives any application the ability to find any issues with testing, if an issue were to arise, and make the application more easily understood. My application will be broken down into very modular pieces to suit this, as well as encapsulate data by only having the data the particular objects needs within it. Additionally, data hiding will be used to make any class attribute private, and having respective getters and setters to mutate / access the data respectively. The main algorithm of calculating the footprint for a user based on gas, miles driven, personal vehicle, etc. will be created in an elegant manner. In speaking with the environmental department at TCNJ, I have the contact for a social scientist (Dr. Miriam Shakow) in the department to get an idea of the direction for the algorithm basis, and an environmental chemist (Dr. Mike Aucott) for the computational part of this application. Together, they will guide me in my search for an efficient algorithmic design for computing the carbon footprint for this application. Searching will be another taxing algorithm my application will have to face, going through all the different vehicles on the EPA's website through my webscraper, as well as storing and searching through them once scraped. A Binary Search Tree will aid me in storing all of the data for the cars, allowing the user to parse through this data as quick as possible. With all things considered, this application's code base will take encapsulation and modularity, efficiency of algorithms, and appropriate data structures into consideration.

## Test Case Design

Testing a system such as the one I am designing will require a lot of unit, integration and system testing. For unit testing, I see 3 main components with my project (View personal car and adding gas, view car list, and changing the user's personal car), that intend to test individually throughout development of each piece. As the project progresses, I will be integrating each piece together, and testing the integration of these parts to ensure that they work well together. Along each step of the way, I will be doing system testing, ensure integration of a certain component does not break, as well as at the end when I believe the project is complete.

For debugging, Xcode has a very robust system for debugging built in. This includes the debug area, debug navigator, adding breakpoints and a source editor. All of this is encapsulated in the IDE itself, so I can test while I am developing in a very easy manner. The following table will be followed throughout development for testing of the various components of the system.

| Functionality Test          | Input  | Expected Output  | Actual Output |
|-----------------------------|--|--|---------------|
| Account Creation            | Blank info   | Error, no account created  |               |
| Account Creation            | Correct info   | Account email confirmation sent                                  |               |
| Account Login               | Blank info   | Error, need to enter data  |               |
| Account Login               | Bad info   | Error, need to enter valid account info                          |               |
| Account Login               | Correct info   | Account info validated and login occurs                          |               |
| Set Personal Car            | Select correct info  | Correct car is selected  |               |
| Change Personal Car         | Select correct car to change to                            | Current car is changed to new car                                |               |
| Change Personal car         | Go to change personal car, but do not complete transaction | Current car info is retained, no data is changed                 |               |
| Add gas to personal car     | Add gas and update milage                                  | All corresponding data is updated accordingly                    |               |
| Add gas to personal car     | Go to add gas and update milage, but cancel transaction    | No data should change  |               |
| Remove gas from persona car | View list and remove a specific data input                 | All corresponding data is updated accordingly                    |               |
| View Personal Car info      | Go view personal car data                                  | See all personal data regarding car, all data should be correct. |               |