

URBAN SCIENCE™

Dealership Inventory Solution

Project Plan
Spring 2016

Urban Science Contact:

Sam Bryfczynski
Mark Colosimo
David Goldschmidt
Linda Koeppe
Kathy Krauskopf
Michael Nelson
Rick Twydell
Chris Welch

Michigan State University Capstone Members:

Justin Girard
Tyler Huttenga
Joey Norwood
Anthony Santoro
Hannah White

Table of Contents

Executive Summary	3
Functional Specifications	4
Web Application.....	4
Generate an Inventory Recommendation.....	4
Inventory	4
Data Visualization	4
Mobile Application	5
Find Dealership.....	5
View and Filter a Dealership's Inventory	5
Locate A Vehicle.....	5
Manage Vehicles from Searchable Vehicles List.....	5
Design Specifications	6
User Interface.....	6
Web Application.....	6
Login	6
Inventory Overview.....	7
Purchase Plan Engine	8
Mobile Application.....	9
Home	9
Dealer Login.....	10
Vehicle Management.....	11
Browse Inventory and Filter.....	12
Vehicle Location Map	13
Use Cases.....	14
Web Application Use Case for Dealership	14
Mobile Application Use Case for Dealer.....	14
Mobile Application Use Case for Customer.....	15
Data Flow	16
Technical Specifications	18
System Architecture	18
Software Technologies	19
Database Schema.....	19
Development Environments.....	20
Testing Plan	21
Risk Analysis.....	23
Schedule.....	24

Executive Summary

Urban Science is a business-solutions company focused on supporting the sales and marketing functions of the automotive industry. Since 1977, Urban Science has been helping their client partners sell more vehicles, improve profitability, and increase customer loyalty. The Dealership Inventory Solution assists dealerships with inventory purchase recommendations to optimize their purchasing power. These recommendations help ensure that a dealer's inventory has a higher probability of a customer finding the vehicle they are looking for.

The process of purchasing vehicles from manufacturers is very complex, with many factors that must be considered before making a purchase. Some of these factors include consumer demand, current inventory, purchased vehicles that are in-transit to the dealership, and many more. Dealerships want to be better informed during their purchasing process to maximize the potential that the vehicles they purchase will be sold as soon as possible. Determining the correct quantity and mixture of vehicles for a dealership's inventory is a complicated problem that will directly impact a dealership's profitability and a customer's happiness. From a customer's point of view, they would like to visit a dealership once and find the vehicle they are looking for. The wide range of vehicles and options can make having the correct vehicle on the lot a difficult task for a dealership.

The Dealership Inventory Solution allows dealerships to determine which vehicles should be purchased based on the dealer's constraints, leading to higher customer satisfaction and profits. The system is comprised of two main components, a web application, for use by dealerships to optimize their inventory, and a mobile application, for use by customers to browse the inventory of a dealership and to find a vehicle on a dealer's lot. Within the web application, dealerships are able to view automatically generated recommendations based on the constraints the manufacturers and dealerships apply to the system. The recommendation conveys to the dealership which vehicles should be purchased to better meet customer expectations. The dealer can use these recommendations to build a list of vehicles they would like to purchase. Within the mobile application, customers can view the inventory of a dealership while also being able to filter the inventory to find their desired vehicle. Once a user has selected a vehicle the app will guide them to the vehicle's location on the lot via GPS. Finally, the mobile application allows the dealer to manage vehicles on their lot. The dealer can update the position of a vehicle, remove it from the system, and temporarily check it out if it's being taken for a test drive.

The goal of the Dealership Inventory Solution is to better assist dealerships in purchasing vehicles for their inventory, and to facilitate customers in their vehicle search. Through the combination of a web application and mobile application, the complex processes involved in purchasing vehicles for an inventory will now be simplified and streamlined. This will lead to higher customer satisfaction and increased profits for the dealership.

Functional Specifications

The primary goal of this project is to assist dealerships in determining which vehicles should be purchased for their inventory. A calculation is performed-- taking into account the dealer's inventory and other factors. The calculation returns a recommendation to the dealer with suggestions of which vehicles to purchase. The web application allows dealerships to adjust their recommendations based on their experience.

The secondary goal of this project is to allow potential customers to pinpoint the location of desired vehicles on the dealership lots. QR codes already present on the vehicles contain that vehicle specifics vin number. The dealer will use the mobile application to associate a location on the lot to the specific vehicle by scanning the vehicle with our QR code scanner.

From within the mobile application, customers can then browse the inventory of a dealership they are at. Once a vehicle has been selected, the customers are able to view a map that displays the path from the user to the vehicle location. This simplifies the process of physically scouring a dealership lot for a desired vehicle.

Web Application

Features

Generate an Inventory Recommendation

The calculator within the web application is powered by an algorithm used to determine the optimal distribution of vehicles to be purchased. The algorithm can be altered through constraints and inputs from the dealer. The recommendation calculation can then be used by the dealer to determine what assortment of vehicles will be able to yield the most profit, volume, or both.

Inventory

Dealers will be able to view their current inventory from within the web application dashboard. The dealers are also able to view in-transit shipments. This allows the dealers to constantly stay up to date with their inventory and gives the dealer a more overarching visual aspect to their inventory.

Data Visualization

Dealers will be able to view multiple graphs and data summaries related to their inventory data. The graphs and summaries allow the dealers to quickly generate a visualize overview of the current, incoming, and recommended vehicles. This feature also allows the dealers to condense their potentially large data set of vehicles into easily readable visualizations.

Mobile Application

Features

Find Dealership (Customers)

Customers are able to find the closest participating dealership from within the mobile application. When the user clicks the search button they are brought to the nearest dealership's inventory screen. This is done by using the GPS feature of the device.

View and Filter a Dealership's Inventory (Customers)

Once a dealership has been located the user is able to search the dealer's inventory. To make the browsing the inventory more conducive for the user, there is a filter feature within the inventory section. This filter feature allows the user to search by the criteria of make, model, year, class, and color. This filter feature streamlines the browsing of the inventory.

Locate a Vehicle (Customers)

Once a customer has located a desired vehicle in the inventory, the customer can click that specific vehicle. By clicking on the vehicle, the customer is brought to a map that pinpoints the vehicle's location on the dealership lot.

Manage Vehicles from Searchable Vehicles List (Dealers)

The dealer can sign into the dealership portal from within the mobile application. The dealer then can scan that specific QR code and associate that QR code with a longitude and latitude. This adds the vehicle to our searchable database and allows the vehicle to be clickable. Once the customer clicks on a vehicle, they are able to view the location of the tagged vehicle on a map.

If a vehicle is sold or no longer desired to be associated with a QR code, the dealer has the ability to scan the QR code and click the *remove* button. By clicking remove, the dealer will delete the vehicle information from our searchable database. This prevents the user from being able to view that specific vehicle on the inventory list.

In addition to the Add and Remove functionality the dealers will be able to check out vehicles. This is useful for situations such as, a vehicle leaving the lot for a test drive or off site maintenance. Checked out vehicles will have a notification banner associated with that specific vehicle to alert potential customers viewing that vehicle in the app. Once the vehicle has returned to the lot, the vehicle can be checked in. This will remove the "Checked Out" banner associated with that vehicle.

Design Specifications

The user experience of the Dealership Inventory Solution and accompanying mobile application is a high priority for the success of this project. The complex systems must be easily accessible to users. This section contains details and images on the design of the user interface. Additionally, this section also includes example use cases detailing how the web and mobile application could be used.

User Interface

Web Application

Login

The login screen is the first screen visited by a user. It requires that a user have an account with the system and be associated with a dealership. By entering a correct email and password, the user will be able to click the login button and enter the system. Optimally, our system will use the dealer's login information from their other dealership management systems, therefore, our system will not have open registration. Accordingly, this project will use test accounts instead of real dealer logins for development purposes.

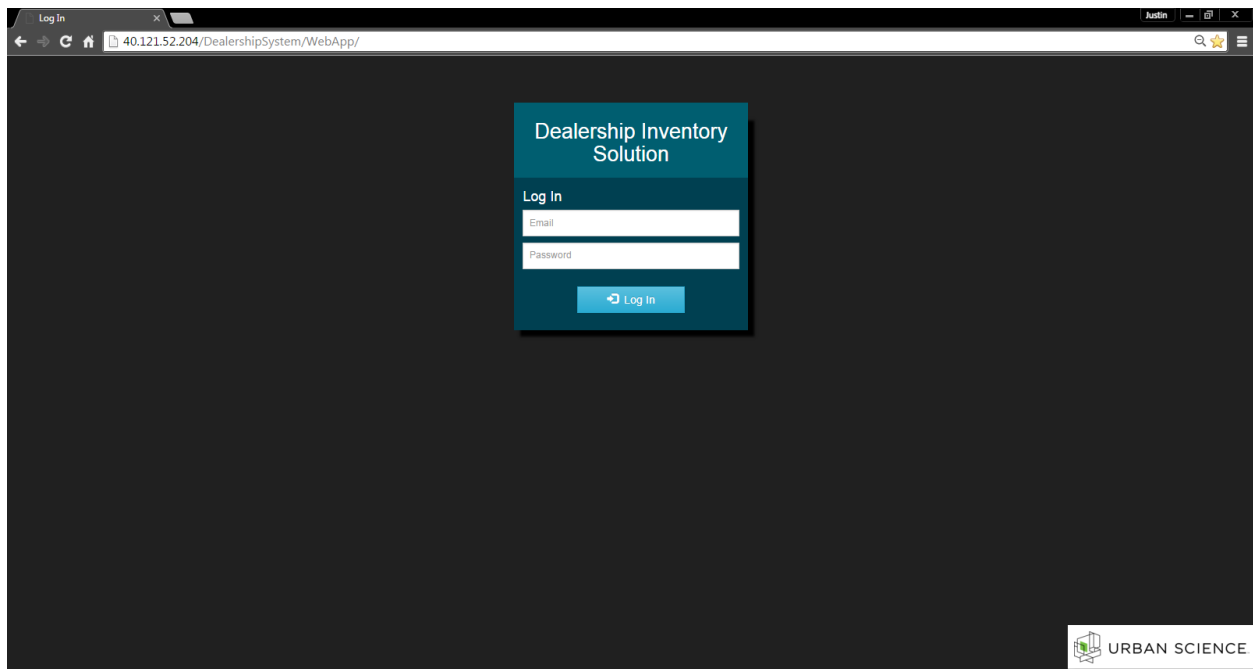


Figure 1: Login

Inventory Overview

The inventory screen allows the user to view their dealership's current selection of vehicles. This screen can be seen in Figure 2. The user can also see vehicles that have been purchased and are on the way. Two tabs in the upper-left corner allows the user to switch between vehicle lists, either current or incoming. These lists can be sorted by clicking one of the table headers. Clicking the same header multiple times will alternate the sorting between ascending and descending order. Graphs representing the distribution of different models can be seen in the top right. The user can find trim details about a certain model by selecting a model in the bottom right menu. There is a top navigation bar that allows the user to switch between the inventory overview, purchase plan engine, an about section, and a log out option. When one of the categories is selected it will be highlighted.

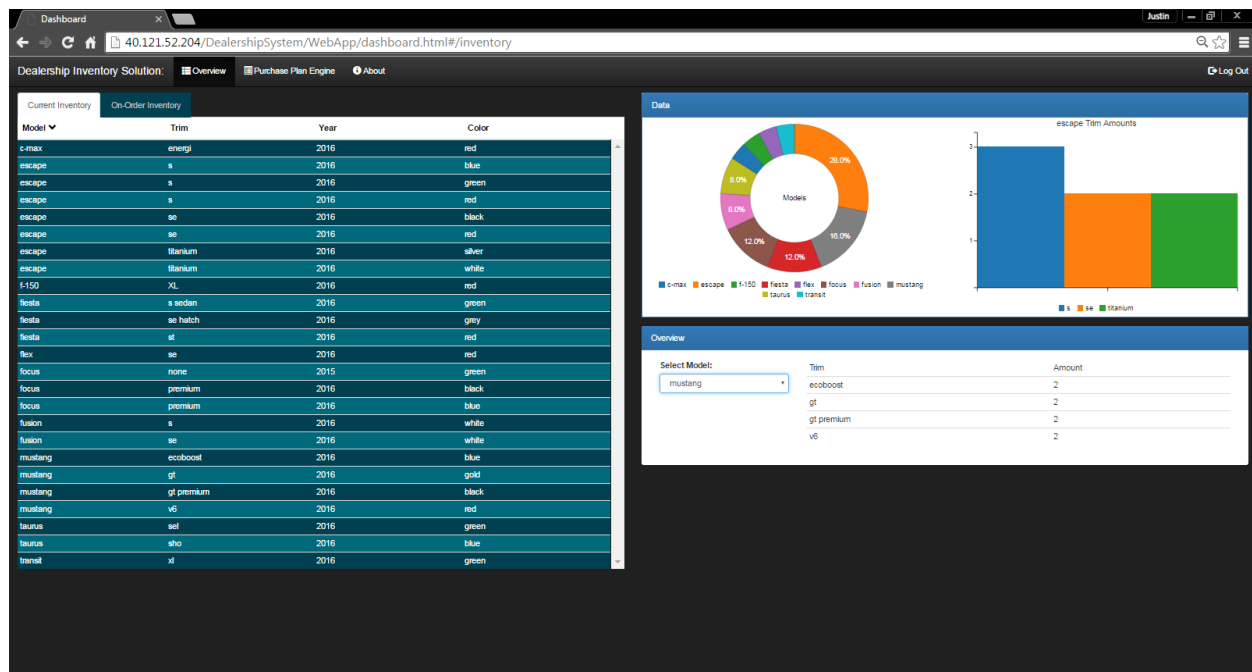


Figure 2: Inventory Overview

Purchase Plan Engine

The purchase plan engine is the core component of the whole application. This screen can be seen in Figure 3. When first visiting this page, the user is asked if they would like to start a new session or resume their previous one. Starting a new session will give the user an initial recommendation of vehicles that they should consider purchasing. Recommendations can be seen in the middle column in Figure 3. The user can add a vehicle to their purchase plan by clicking the add button. Once a vehicle is added it cannot be added again. The purchase plan is the right-most column. It contains a list of the vehicles the user would like to purchase. The user can customize the color of an item in the purchase plan by clicking the customize button. It can also be removed if the user no longer wants to consider buying that model/trim. The user is free to reorder the items in the purchase plan either by clicking the move buttons or by dragging and dropping.

A dealer may want to specify certain models or trims that they desire which may not appear in the initial recommendation. This can be done in the summary section which is the left-most column. In this section, the user can specify which trims they desire and the amount. After selecting their desired models and trims, a user can click the recalculate button which will repopulate the recommendation list based on their input.

Finally, the user can save their current session by clicking the save button. This allows them to resume their progress at a later time. When the user feels that they are finished they can click the print button to get a print friendly version of their purchase plan.

The screenshot shows a web application interface for the 'Purchase Plan Engine'. The browser address bar shows '40.121.52.204/DealershipSystem/WebApp/dashboard.html#/engine'. The interface is divided into three main columns: Summary, Recommendations, and Purchase Plan.

Summary Column: A table with columns: Model, Recommended, Desired, Planned, and Trims. It lists various car models like Mustang, C-max, Flex, Taunus, F-150, Focus, Escape, Fusion, Fiesta, Edge, and Transit, each with a recommended count, a desired input field, a planned count, and a trim selection button.

Recommendations Column: A table with columns: Priority, Model, Trim, Reason, and Add. It lists 16 recommended vehicles with reasons such as 'You currently have low inventory in this model' or 'This model has been selling very well in your area'. Each row has an 'Add' button.

Purchase Plan Column: A table with columns: Move, Priority, Model, Trim, Color, Customize, and Remove. It shows 5 vehicles currently in the purchase plan, each with a color selection button and a remove button.

At the bottom of the interface, there are three buttons: 'Re-Calculate Recommendations', 'Add All', and a group of 'Print', 'Reset', and 'Save' buttons.

Figure 3: Purchase Plan Engine

Mobile Application

Home

The home screen is the first page that the user is brought to after opening the application. The top-most button takes the user's position via GPS and displays the inventory of the nearest dealership. The button below that allows a dealer to login and manage vehicles on their lot.

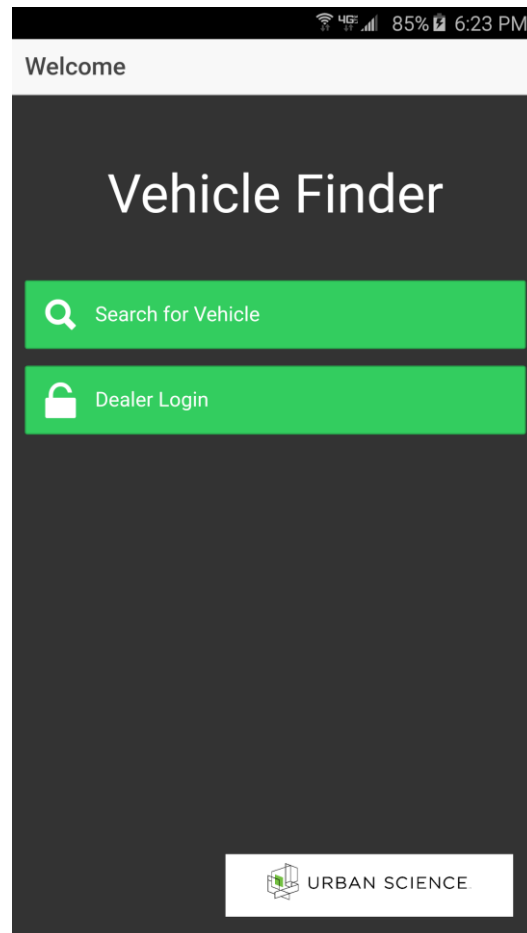


Figure 4: Home

Dealer Login

A dealer that wishes to update the position of a vehicle on their lot, or remove a vehicle from the system must first login. The user must provide a username and password that will be authenticated when the login button is clicked. A back button is located in the upper-left of the screen and allows the user to return to the home menu. After clicking the login button and successful authentication, the user will be taken to the vehicle management screen.

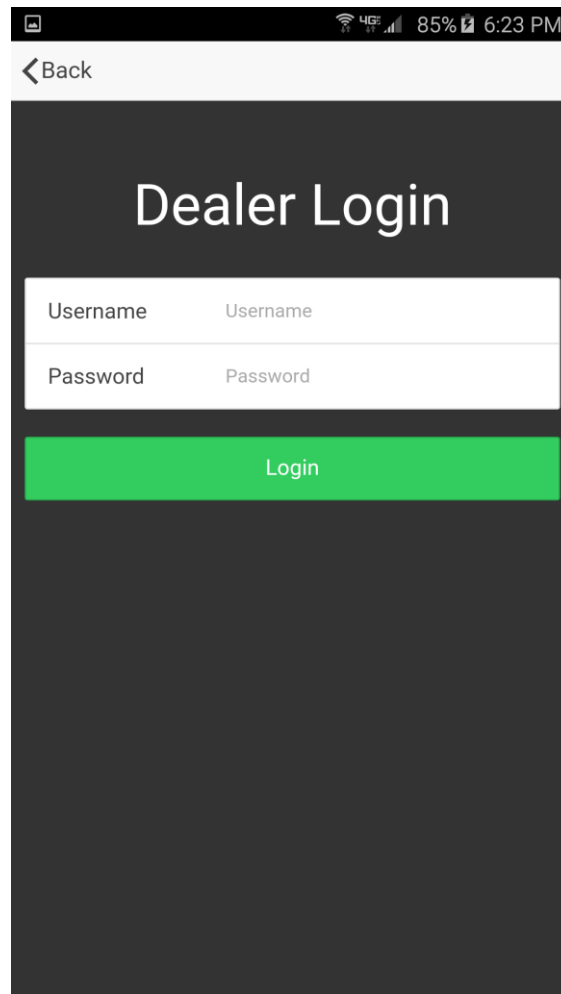
A screenshot of a mobile application interface for a 'Dealer Login' screen. At the top, a status bar shows a camera icon, 4G LTE signal, 85% battery, and the time 6:23 PM. Below the status bar is a white header bar with a left-pointing arrow and the text '< Back'. The main content area has a dark gray background. In the center, the text 'Dealer Login' is displayed in a large, white, sans-serif font. Below this, there are two white input fields. The first field is labeled 'Username' on the left and has 'Username' as placeholder text. The second field is labeled 'Password' on the left and has 'Password' as placeholder text. Below the input fields is a solid green rectangular button with the word 'Login' in white text. The bottom portion of the screen is a solid dark gray area.

Figure 5: Dealer Login

Vehicle Management

The application allows the user to update the position of a vehicle on the dealer's lot. It also allows them to remove a vehicle from the system if the vehicle has been sold. Sometimes a vehicle is taken on a test drive or to the car wash; a dealer can check out a vehicle so that it will not show up for a customer searching for a vehicle on the lot. When the vehicle is returned it can be checked in and customers will be able to find it again. Upon clicking the update/check in/check out buttons, the data from the scanned code and the GPS location of the phone are sent to the backend server where the location of the vehicle is updated. When the task is complete, a small notification will appear that tells the user whether the update was successful or failed. Clicking the back button while viewing the management screen returns the user to the home screen.

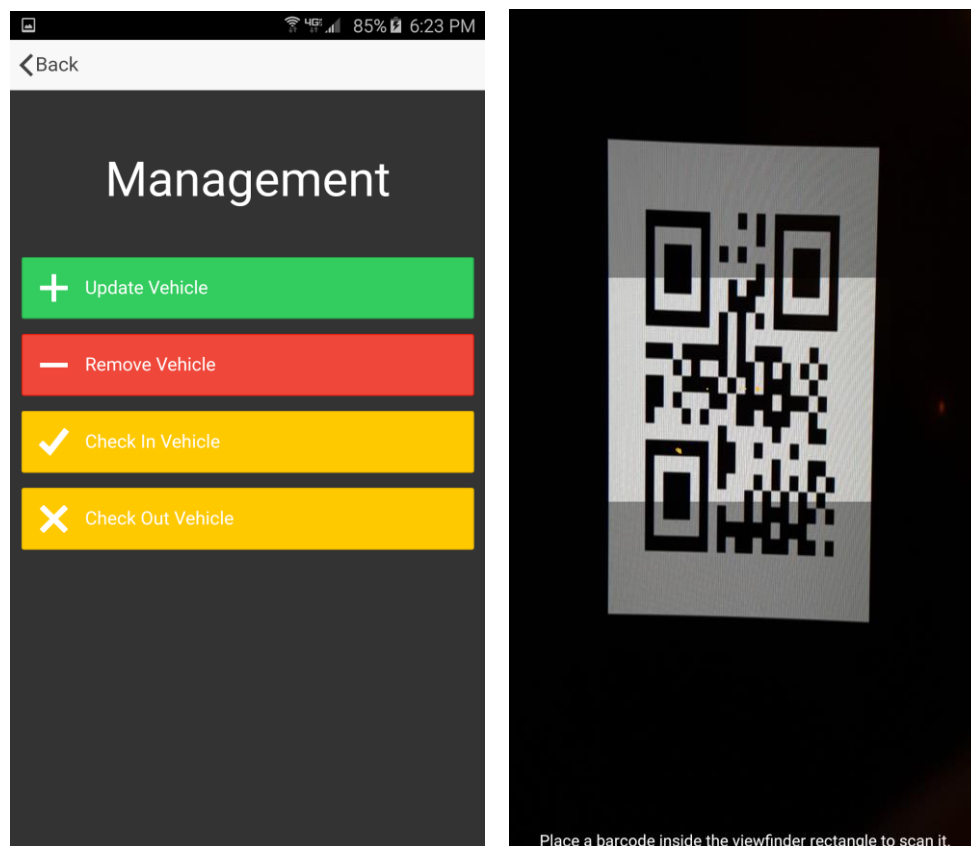


Figure 6: Vehicle Management (Left) and QR Scanner(Center)

Browse Inventory and Filter

From the home screen, when the user clicks the search button they are brought to the nearest dealership's inventory screen. In the left-most image in Figure 7, an unfiltered list of vehicles on a dealership's lot can be seen. The user is free to scroll through the inventory and select a vehicle they are interested in. To select a vehicle, the user just needs to tap on the row in the table. Selecting a vehicle takes the user to the vehicle's location map. More information about the vehicle location map can be found in the next section. A filter button, located in the top-right of the screen, takes the user to the filter screen. The filter screen allows the user to find a specific vehicle. After selecting a vehicle from the filter screen, a screen with dealership and vehicle information is displayed. The back button returns the user to the home screen.

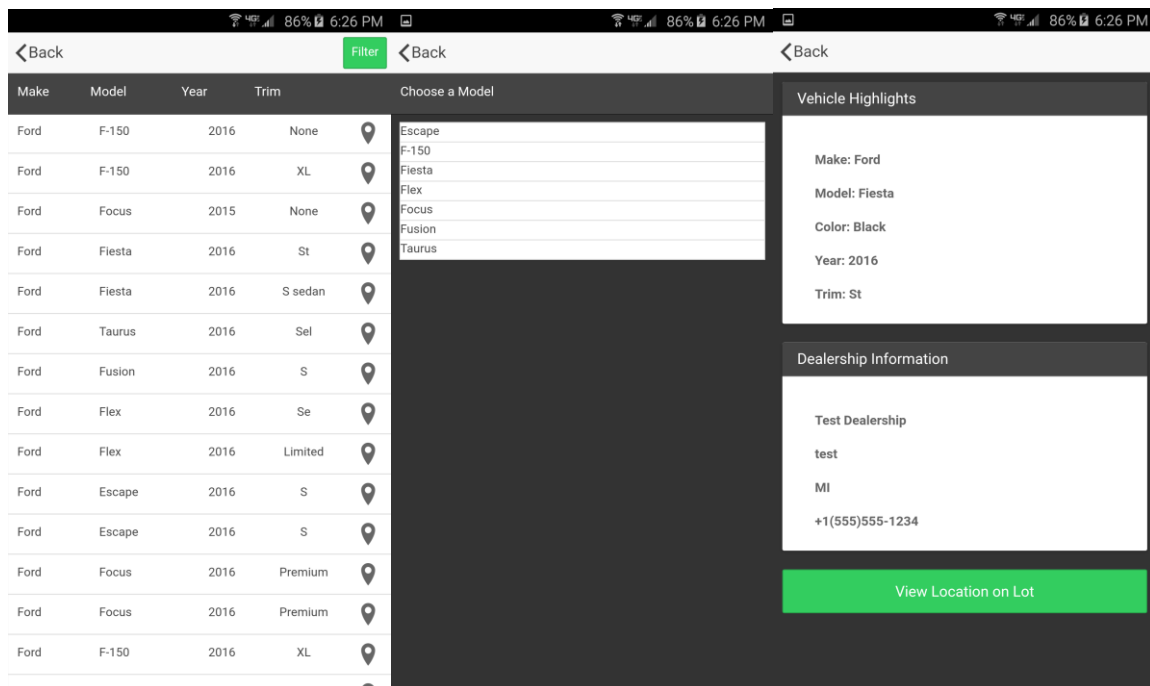


Figure 7: Inventory (Left), Filter (Center), and Vehicle Details(Right)

Vehicle Location Map

The vehicle location map assists a user in finding a vehicle on the dealership's lot. After selecting a vehicle from the inventory screen the user is brought to this map. A red marker indicates the position of the desired vehicle. A red line links the user's current position and the location of the vehicle. As the user navigates the lot their current position is updated on the map. Users can use this information to direct themselves to the vehicle's location. The back button returns the user to the dealership's inventory screen or to vehicle's information screen.

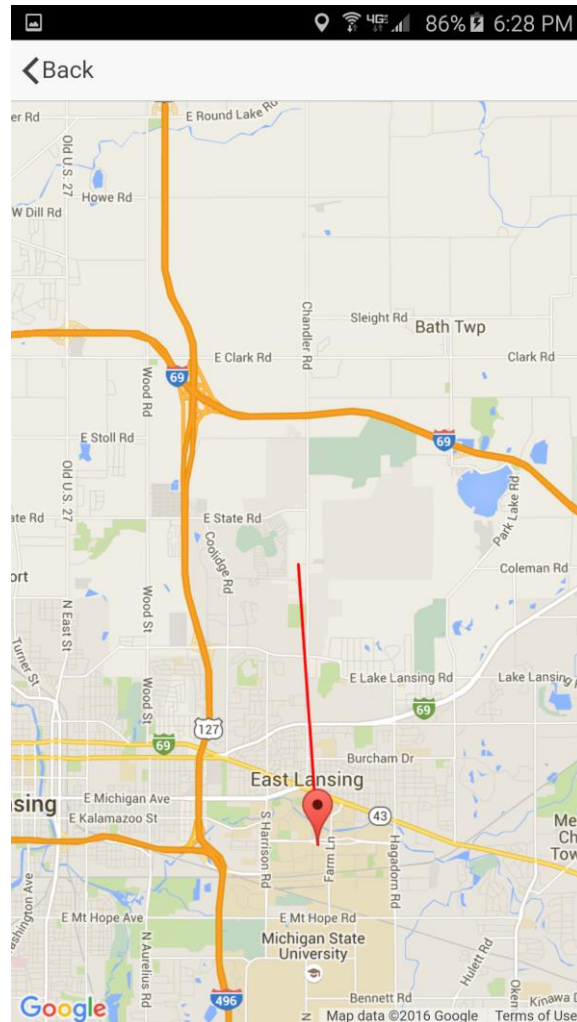


Figure 8: Vehicle Location

Use Cases

Web Application Use Case for Dealership

Bob is a dealership employee. Bob logs into the Urban Science Dealership Inventory Solution web application with his Original Equipment Manufacturer (OEM) credentials. Bob can now view the features that he is authorized to view for his dealership.

Bob decides that while he is logged into the website, he will check out what the system recommends he should purchase for his next restock. Bob navigates to the Purchase Plan Engine tab of the website, and is greeted by a menu asking him to start a new session or to continue a previous one. Bob decides that he would like to start a new session. Bob then sees an initial list of recommended models and trims that the system thinks he should purchase. Bob also sees all of the available models and trims in the summary table. Bob notices that the system only recommended two Ford F-150 Platinums but from Bob's experience he knows he really wants three. Bob finds F-150 in the summary list and clicks the trims button. Bob then enters three into the Platinum input box. Bob then clicks recalculate and a new list of recommendations are displayed, this time it includes three F-150 Platinums. Bob then clicks the add button next to the three F-150s and places them in the purchase plan.

Bob decides that he is satisfied with this purchase plan and he would like to save it for later viewing. Bob presses the save button which will add it to the database. He also decides he would like to print the purchase plan to show to a coworker. Bob clicks the print button and a new tab opens with a print friendly version of his purchase plan. Bob then uses the print option in his browser and prints the purchase plan.

Bob now wants to learn a little more about his current stock so he navigates to the Overview page. Bob is curious to see what trims he has for Mustangs. Bob selects Mustang from the dropdown menu in the Overview section and sees that he has two Ecoboost trims, two GT trims, two GT Premium trims, and two V6 trims. Bob is satisfied by easily finding the information he desired and logs out of the system.

Mobile Application Use Case for Dealer

Bob decides he would like to manage the Ford Focus the dealership just received. Bob opens his Urban Science Vehicle Finder app on his smartphone. He then clicks the "Dealer Login" button which brings him to a page requesting his OEM credentials. Bob enter his credentials and clicks the "Login" button.

Once verified, Bob is redirected to the vehicle management screen. Bob presses the update vehicle button. Bob's mobile device then opens a QR scanner. He uses this scanner to scan the code that is on the Ford Focus. Once scanned, the device's GPS marks it's current latitude and longitude. Bob is then returned to the vehicle management page. Now the vehicle will be included in the customer's list of vehicles they can find.

Bob then decides that while he has the app open, he might as well update the location of the Mustang that they just moved to a prominent place by the road. Bob walks over to the

Mustang, which already has a QR code sticker on it. He follows the same steps as before, scanning the code, and the location of the mustang is updated in the system, so customers will be directed to the new correct location of the Mustang.

Finally, Bob makes his way over to a customer who just purchased a Ford Edge and is waiting for his paperwork to be complete before driving away in the new vehicle. Bob decides that now would be a good time to remove that vehicle from the searchable vehicles because it will soon be taken away from the lot. Bob selects the Remove Vehicle option which opens the QR scanner. Bob scans the sticker as before and the vehicle is removed from the list of searchable vehicles, so no customers will mistakenly walk to the previous spot of the vehicle. Bob then removes the sticker and sends the customer on his way.

Mobile Application Use Case for Customer

Heather is a prospective vehicle buyer in the market for a new Chevy Malibu. Heather drives to the local Chevrolet dealership and upon arrival, notices a sign for the Urban Science Vehicle Finder app. She pulls out her smartphone and downloads the free app.

Upon opening the application, there is a prominent search button on the first page. She clicks this button which brings up a list of searchable vehicles at the dealership she is currently at. Since Heather knows she wants a Malibu, she clicks on the filter button, which brings up a screen with options to search for a specific type of vehicle.

Heather inputs that she is interested in a red Chevy Malibu LS. She is then brought to a vehicle information page that also has information about the dealership she is at. Heather decides that she would like help finding the vehicle so she clicks the vehicle location button. This action brings up a map displaying her current location and a marker showing where the vehicle is located on the lot.

Heather now knows which direction to walk to see this vehicle in particular. She makes her way to the vehicle and examines it. She decides after looking that she would prefer a different trim package than the one she is currently viewing so she clicks the back button. This brings her back to the previous page where she can select a different red Chevy Malibu with a higher end trim package.

She selects a different vehicle and the map directs her to a vehicle further down the row from where she was standing. After checking out this other Malibu, Heather decides that this is the vehicle for her. Heather clicks the provided phone number in the dealership information section and asks one of the dealers to come talk to her about the vehicle. She waves over a dealership employee and they proceed to fill out the necessary paperwork, and Heather drives away in her new Malibu.

Data Flow

The customer side of the mobile application in the Dealership Inventory Solution will receive data about dealerships when calculating which dealership the customer is currently at, and also receive data when viewing a dealership's current searchable inventory. The customer side of the mobile application will also be able to add data to the databases via analytical data about how customers use the app (when the application is opened on the dealer lot, when vehicle search is completed, when GPS matches vehicle location, etc.). The dealer side of the mobile application will store data when a vehicle's location on the lot is updated, provided a way to indicate if a vehicle is currently off the lot (for things like test drives) via check in/out, and will delete entries from the database when a vehicle is removed. The data flow for the user side of the mobile application is shown in its entirety in the green area in Figure 9, and the dealer side for the mobile is shown in its entirety in the upper portion of the white area of Figure 9.

The web application of the Dealership Inventory Solution, which has its data flow shown in the bottom portion of white section of Figure 9, pulls data in from the vehicle recommendation algorithm provided by the client. The web application will provide data when the recommendation algorithm is used, when the current working plan is saved or loaded, or when viewing the inventory.

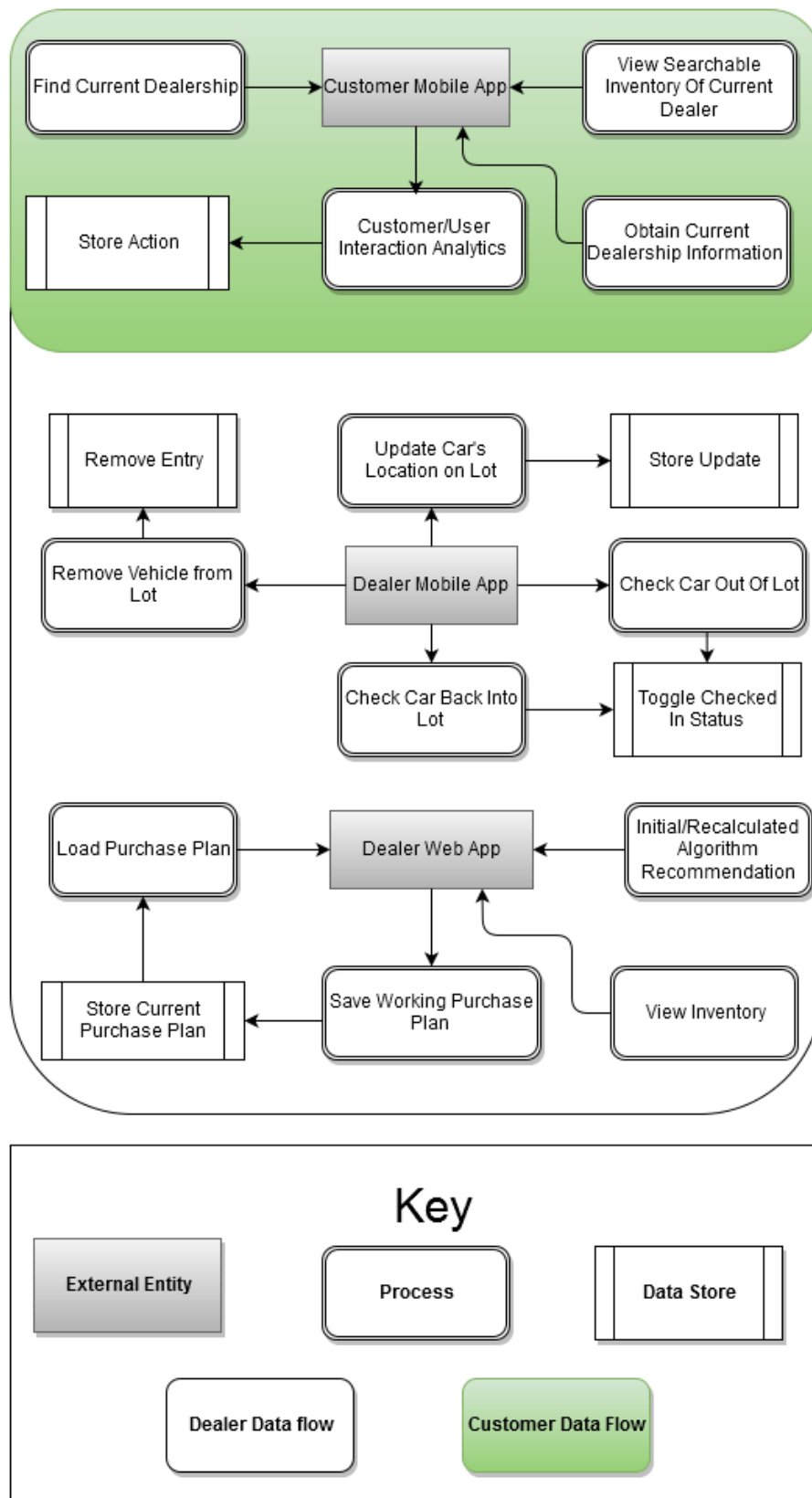


Figure 9: Data Flow Diagram

Technical Specifications

This section details the various technologies that will be used to create and maintain the system. Additionally, this section provides an overview of the anticipated database schema, as well as the development environments that will be used to build both the web and mobile applications.

System Architecture

The Dealership Inventory Solution involves the straightforward use of a SQL Server database for storage of the vehicle and dealership data. This database will be interfaced with using server-side PHP files and client-side RESTful calls. The SQL Server and PHP files will be hosted on the provided Windows Server 2008 R2.

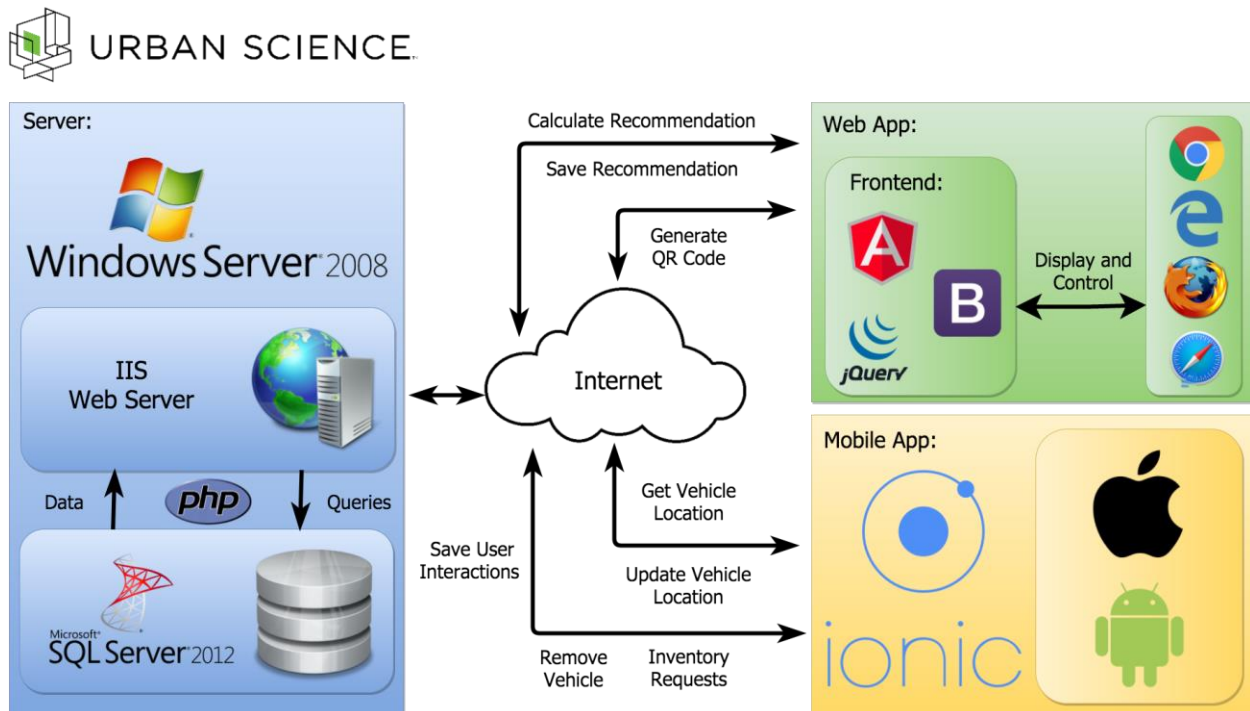


Figure 10: System Architecture Flow Diagram

Figure 10 provides a high-level overview of the system architecture and flow of data across components. Dealers will have access to the web application and mobile application, while customers will have access only to the mobile application. The dealership and vehicle data will be contained in the SQL Server 2012 database, which can be interfaced with through a REST layer. As the double-sided arrows indicate, both the web application and mobile application can send and receive data, meaning that they can both retrieve and alter the data stored on the SQL Server. Changes to the data would be in the way of saving a

recommendation (web app), updating vehicle location, removing a vehicle, and saving user interactions as users interact with the mobile application (mobile app).

Software Technologies

- HTML5/CSS
- Bootstrap
- AngularJS
- jQuery
- SQL Server 2012 + SQL
- PHP (implements jquery.qrcode.js)
- Ionic (requires HTML5, CSS, AngularJS, implements BarCodeScanner)

Database Schema

The system a single database with 7 primary tables with 4 translation tables (tables Make, Model, Color, and Trim, used to map dealer codes to real world meanings), which will be used to store information about dealerships (having some of the provided data being stored concretely and different data, such as the location of a vehicle on a lot, being derived at the mobile application level). There are foreign keys in the Inventory, Searchable_Vehicle, Employee, and Algorithm_Recommendation tables to the Dealership and Vehicle tables (with cascading deletions), as shown in Figure 11. The Dealership table stores general data about a given dealership, including a latitude and longitude field which is used to determine which dealership a customer is currently at when searching for a vehicle using the mobile application. The Employee table stores basic login credentials for an employee of a give dealership, which are used to access the dealer applications. The Inventory table stores a reference to a specific type of vehicle, along with the status of said vehicle (on the lot, in transit, ect.) and whether or not the vehicle has been sold. The Searchable_Vehicles table stores only the vehicles which are searchable on a lot using the mobile application, and is also the only table where any data will be deleted from the database (a deletion occurs when a dealer select the “remove” option on the mobile app and scans the desired vehicles’ QR code). The Vehicles table stores basic information about the make, model, and other main characteristics of a vehicle to be sold. The Usage_Analytics table will provide valuable data about consumer/dealer interaction which will be useful to improve future analytical modeling.

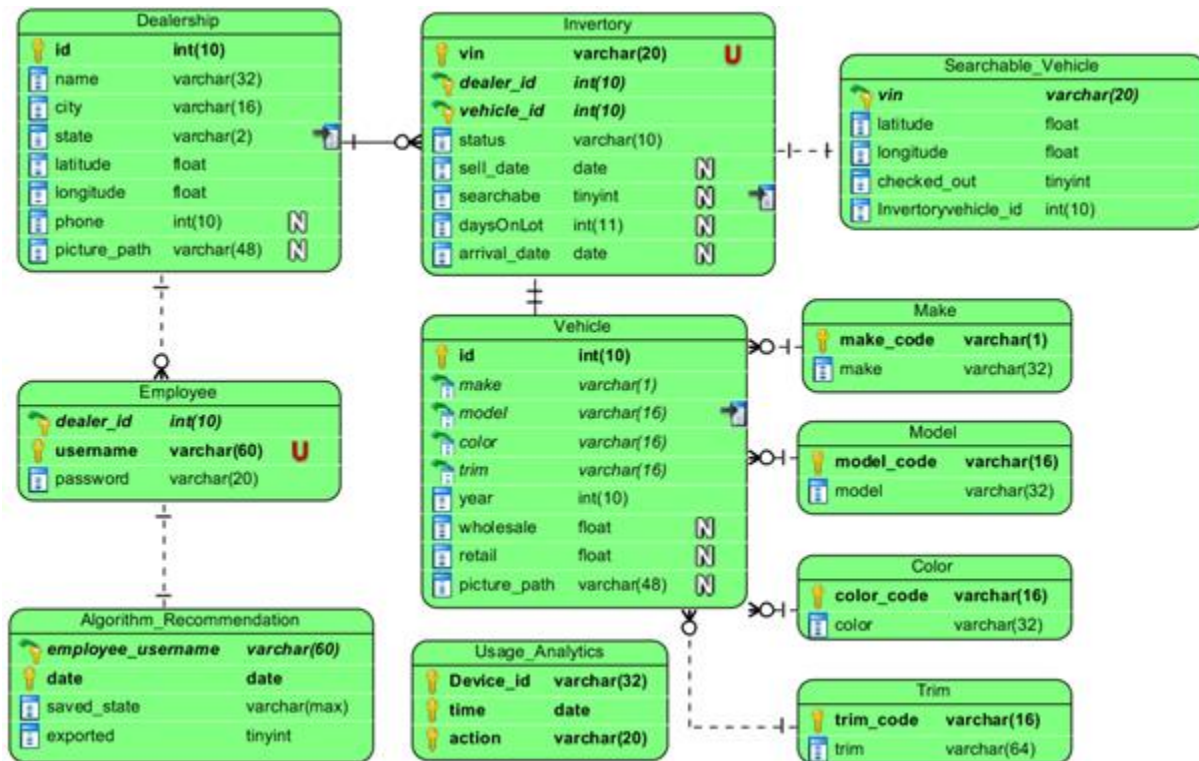


Figure 11: Database Schema

Development Environments

Both the web and mobile applications are written on OS X El Capitan machines. The web application is written using PhpStorm alongside Bootstrap, AngularJS, jQuery, as well as jquery.qrcode.js. The mobile applications are written using the Ionic framework, which utilizes HTML5, CSS, JavaScript, and BarCodeScanner. The server provided by the client is running Windows Server 2008 R2. Source control using TFS/Git is also provided by client.

Testing Plan

This section covers the testing methods involved with the creation and maintenance of the Dealership Inventory Solution. The testing methods incorporate unit testing on critical system parts, as well as manual, exploratory testing for bugs and other anomalies.

Web Application

AngularJS provides many different methods of unit testing, including:

- **Dependency injection** -- creating mock instantiations of objects and running code as if the objects were in a production environment
- **Karma** -- a command line tool that spawns a web server and runs the source code against multiple browsers to ensure compatibility/support
- **Jasmine** -- a JavaScript framework that provides functions allowing assertion statements and well-structured testing

```
describe('sorting the list of users', function() {  
  it('sorts in descending order by default', function() {  
    var users = ['jack', 'igor', 'jeff'];  
    var sorted = sortUsers(users);  
    expect(sorted).toEqual(['jeff', 'jack', 'igor']);  
  });  
});
```

Figure 12: A Jasmine sample testing method

Functions in Jasmine are relatively straightforward and use the *describe* function as a method of grouping tests together. Grouped within *describe* functions are *it* functions, which contain individual, specific tests. As seen in Figure 12, a Jasmine function is being used to test “sorting the list of users”. The “sorts in descending order by default” *it* function within the *describe* block contains code that verifies the array called “users” will be sorted in descending order. This is verified with the *expect...toEqual* line.

Describe and *it* blocks provide an easy and scalable way to structure testing, and will likely be used extensively to thoroughly test the web application for crucial functionality. It is also anticipated that Karma will be used to ensure browser compatibility when unit-testing has been successful.

Bootstrap unit-testing will require a third-party API called Qunit, which behaves almost identically to Jasmine. It is likely that the team will be able to leverage their experience and knowledge with Jasmine to use Qunit (or vice-versa).

Mobile Application

Similar to the web application, testing the mobile application involves the use of the Jasmine and Karma libraries. This similarity is caused from the reuse of AngularJS. The difference between the use of Karma between the two platforms is that Karma will be used from

```
describe('#doLogin', function() {  
    // TODO: Call doLogin on the Controller  
  
    it('should call login on dinnerService', function() {  
        expect(dinnerServiceMock.login).toHaveBeenCalledWith('test1', 'pas  
    });  
  
    describe('when the login is executed,', function() {  
        it('if successful, should change state to my-dinners', function()  
            // TODO: Mock the login response from DinnerService  
  
            expect(stateMock.go).toHaveBeenCalledWith('my-dinners');  
        });  
  
        it('if unsuccessful, should show a popup', function() {  
            // TODO: Mock the login response from DinnerService  
  
            expect(ionicPopupMock.alert).toHaveBeenCalled();  
        });  
    });  
});
```

the command line in this scenario to simply run the tests (it will not involve spawning a server).

Figure 13: A Jasmine sample testing method, used with mock controllers

Figure 13 shows the framework for testing that simulates or “mocks” the navigation that is handled by the would-be controllers. This is done using dependency injection (mentioned in the previous section: Web Application), wherein a mock instantiation is created instead of the real implementation. Just like in the testing for the web application, the Jasmine testing for Ionic allows for assertion statements and *define/it* functions.

Manual Testing

The team plans to utilize manual testing (particularly through the use of multiple different users/testers) as a way to filter out performance bugs, unfriendly and unintuitive UI, and other anomalies in both the mobile and web applications. Testers will attempt to “break” otherwise working functionality, in hopes of uncovering behavior that was previously unaccounted for.

Risk Analysis

Throughout development of the project, the team will handle the following risks (with their respective mitigations listed underneath):

- Uncertainty of how to use/interface with SQL Server 2012
 - Utilize online documentation and tutorials for self-learning (along with the ability to ask our client regarding data-specific questions)
- Little experience with Ionic framework
 - Utilize online documentation and tutorials, as well as utilize basic templates/demos for quick prototyping and understanding
- Incorporating unfamiliar libraries (AngularJS and Bootstrap)
 - Follow online tutorials, documentation, as well as prototyping often
- Interface design has to be intuitive and simple, despite loose guidelines and little design experience across the team
 - Frequent testing and reviews with our client
- Recommendation algorithm to be provided by the client not yet received
 - Simulate until complete and design for easy implementation

Schedule

Week 1 (1/11/16 - 1/15/16):

- Assigned team and client
- Conference call with Urban Science to answer initial questions
- Set up weekly meeting times with Urban Science
- Set up a trip to visit Urban Science in Detroit
- Met with MSU team to discuss possible software technologies to use
- Setup capstone server
- Started the project plan document

Week 2 (1/18/16 - 1/22/16):

- Visited Urban Science in Detroit
- Work on project plan document
- Explore web app libraries
- Configure server

Week 3 (1/25/16 - 1/29/16):

- 1/25 - Status report presentations
- Finalize server configuration
- Setup database and database tables
- Complete project plan document
- Web and Mobile App: Prototype technologies

Week 4 (2/1/16 - 2/5/16):

- 2/1 - Project plan presentation and project plan document due
- Web App: Inventory prototype
- Mobile App: Find dealership and view inventory prototype
- Mobile App: QR code reader

Week 5 (2/8/16 - 2/12/16):

- 2/10 - Present project plan
- Web App: Purchase plan engine prototype
- Mobile App: Filter inventory prototype
- Mobile App: Vehicle location map prototype

Week 6 (2/15/16 - 2/19/16):

- Web App: Dealer login prototype
- Mobile App: Dealer login prototype
- Mobile App: Update vehicle location and remove vehicle from system

Week 7 (2/22/16 - 2/26/16):

- 2/22 - Alpha presentations
- Polish web app features and ui
- Polish mobile app features and ui
- Start project video

Week 8 (2/29/16 - 3/4/16):

- Work on project video
- Web App: Finalize login and purchase plan engine
- Mobile App: Finalize dealer login
- Mobile App: Finalize find dealership and view inventory

Week 9 (3/7/16 - 3/11/16):

- Spring Break

Week 10 (3/14/16 - 3/18/16):

- 3/14 - Status report presentations
- Work on project video
- Web App: Finalize recommendations
- Mobile App: Finalize inventory filter and vehicle location map

Week 11 (3/21/16 - 3/25/16):

- Work on project video
- Web App: Implement small additional features
- Mobile App: Implement small addition features

Week 12 (3/28/16 - 4/1/16):

- Work on project video
- Web App and Mobile App: Tweak/polish user experience and UI
- Web App and Mobile App: User Tests and Bug Fixing

Week 13 (4/4/16 - 4/8/16):

- 4/4 - Beta presentations
- Work on project video
- Web App and Mobile App: Tweak/polish user experience and UI
- Web App and Mobile App: User Tests and Bug Fixing

Week 14 (4/11/16 - 4/15/16):

- Finish project video
- Web App and Mobile App: Tweak/polish user experience and UI
- Web App and Mobile App: User Tests and Bug Fixing

Week 15 (4/18/16 - 4/22/16):

- 4/18 - Status report presentations
- Web App: User Tests and Bug Fixing
- Mobile App: User Tests and Bug Fixing

Week 16 (4/25/16 - 4/29/16):

- 4/25 - Project video due
- 4/27 - All deliverables due
- 4/28 - Design Day setup
- 4/29 - Design Day