
# 24-Hour Road Service Mobile Apps

## Project Plan

## Fall 2011

**Michigan State University Computer Science and Engineering Capstone Team Members:**

Paul Fritschen

Justin Hammack

Lingyong Wang

# Contents

## 1. Auto-Owners Insurance

Auto-Owners Insurance is a Fortune 500 company based in Lansing Michigan. It was founded in 1916 and currently provides life, home, car, and business insurance in 26 states.

## 2. Executive Summary:

Smartphones have become ubiquitous and can be put to many uses. A place where they can be useful is in the case of a roadside emergency. Applications for both Android and iPhone have been developed in order to assist in such an event. These applications provide the capability to send positional and insurance data through email, instead of the current method of verbal communication over the phone. An administrative website was also developed to present usage information about the applications and find trends in the requests for service.

## 3. Project Overview

The project consists of three main components and several supporting components. The main components are the two smartphone applications, for Android and iPhone, and a web site for administrative usage and logistical tracking.

The smartphone applications are to be used by people in need of roadside assistance. The primary functionality of the smartphone applications is to contact a company called Quest. This company arranges for tow-trucks or other assistance to be provided to the person in need of service.

The website is designed for internal use by Auto-Owners Insurance. Auto-Owners employees can use the website to view data about the number of requests for service based on parameters such as Road Trouble Service limit, the state the incident occurred in, or the time of day.

Supporting the smartphone applications and the website will be two external databases. One database is a mockup of an Auto-Owners Insurance database and stores insurance data. The other is a logging database that stores both insurance data and statistics about time and location of service requests. Additionally, a web service was created to communicate between the applications and databases.

### 3.1 Example Scenario

John, an insured member of Auto-Owners Insurance, is driving down the highway and gets two flat tires. He is incapable of dealing with this by himself. To remedy this situation the current method would be to call Quest and tell them your location and your insurance information. Quest would then deploy a tow truck to assist. With the Road Trouble Service Application, John can simply bring up the application on his phone, click a couple buttons to email his location, vehicle, and insurance information, and then be prompted to make a call to confirm that his information has been received. John does not need to remember his insurance policy number, nor know exactly where he is. Additionally, the application will allow John to view his own insurance information and determine if there are any nearby restaurants, hotels, or service centers, in case his vehicle takes time to receive service.

## 4. Project Features

A number of features were implemented for both the smartphone applications and the administrative website. These features are bulleted as follows.

### 4.1 Application Features

- The applications are able to communicate with Auto-Owners Insurance databases, for development purposes a database mimicking the setup and function of Auto-Owners Insurance databases was used.
- The applications have a login feature. Users are prompted to login when they open the application. The user will either input Auto-Owners Insurance login information or choose to proceed without logging in. After initial login the user does not need to login again and as such will not be prompted to.
- The applications pull insurance information from the mock Auto-Owners Insurance Database and store the data locally on the phone. The data updates when the user opens the application.
- The applications have a button to request Road Trouble Service. The user chooses to send their current position or a custom position, and which vehicle requires service. An email is sent to Quest and the user is prompted to call Quest to confirm reception of the data and the sending of service.
- The applications allow for the user to view their own insurance information.
- The applications were designed to have the look and feel of Auto-Owners Insurance. They were modeled off of a sample application and the Auto-Owners Insurance website.
- The applications allow users to locate nearby service centers, hotels, and restaurants.
- The applications are able to discover their longitude and latitude and translate this into a street address for easier use by towing companies. The applications can also send a localized map of their location.
- The applications push service request data to a logging database. This will happen anytime a request for service is made.
- Anyone can access the features of the applications that allow calling for help and finding nearby locations. However, only Auto-Owners Insurance members will be able to view their insurance information and make requests for service through email.

### 4.2 Website Features

- The Administrative Website pulls data from our logging database.
- The website has a statistics view that contains charts depicting usage frequency against variables such as the insured's Roadside Trouble Service limit as well as various filters for viewing data within a specified state, time frame or date range.
- The website has a table view for viewing all transactions at once.

## 5. Application Details

### 5.1 Screen Descriptions

The following section shows mockups of the different screens that comprise the applications. The screens were modeled off of the Auto-Owners Insurance website and a different application mockup provided by Auto-Owners Insurance.

In this section each figure is comprised of a screenshot from the iPhone on the left and a screenshot from the Android on the right. There are a few exceptions to this rule which will be explained when they arise.

### 5.1.1 Login Screen

The login screen allows the user to input their Auto-Owners Insurance information. This information is conveyed to a web service that will authenticate the information with the Auto-Owners Insurance database. On successful authentication, user insurance data will be downloaded to the phone and stored in a local database. The administrative website is explained in section 7. On this screen the user has the following two options

- **Login:** The user can login with their Auto-Owners Insurance username and password.
- **Skip:** If the user is not currently an Auto-Owners Insurance member, the user can skip login to search for nearby places, or use the call functionality of the application.

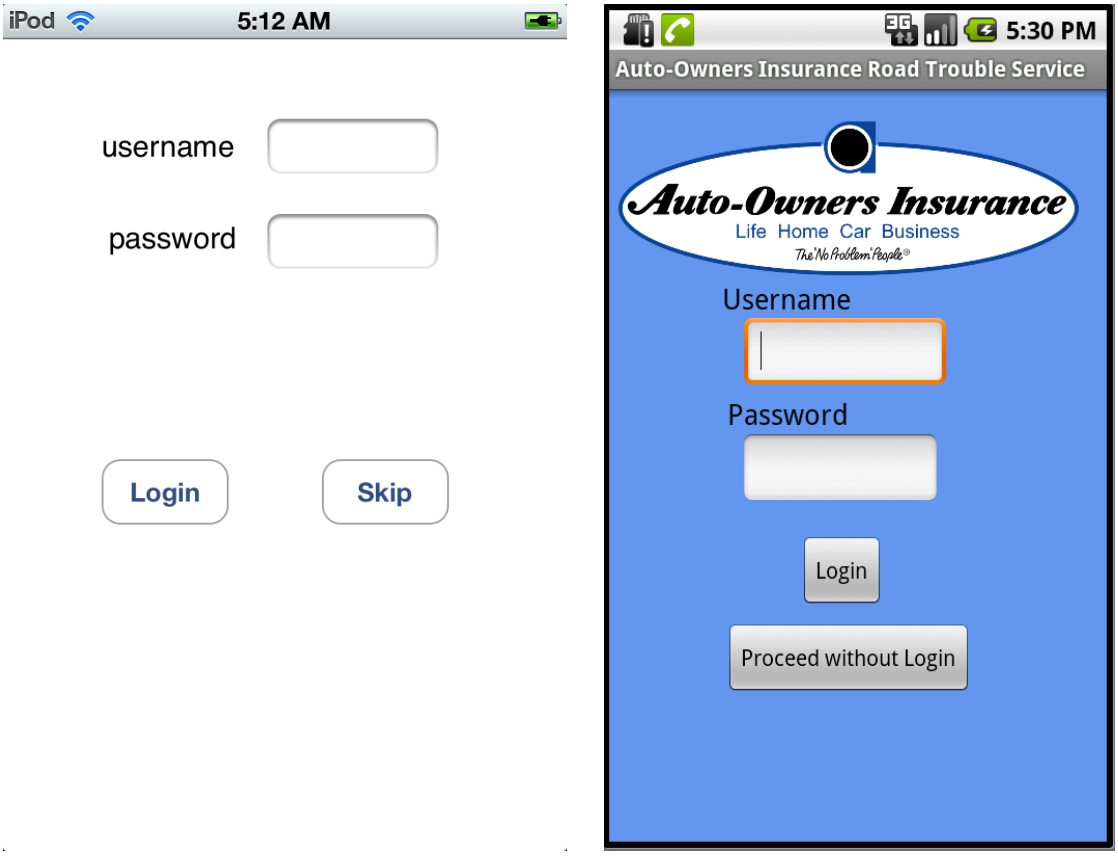



**Figure 5.1.1.1 Login Screen**

### 5.1.2 Main Screen

If a user has logged in before, the main screen will be the first screen to show up. From here the user can make a variety of selections. If the user does not have an internet connection the "Find" feature will be disabled, this is shown in figure 5.1.2.3.

- **Request Road Trouble Service:** The user can click this button to being the request for service. From here they will proceed to the Service Request Screens shown in section 5.1.3. If the user has not yet logged in they will be given the option to login, call Quest, or cancel. This is shown in figure 5.1.2.2.
- **View Insurance Data:** This button will take the user to a screen showing their login information. This is shown in section 5.1.4. If the user has not yet logged in, they will receive an alert allowing them to login or cancel. Figure 5.1.2.3 demonstrates this behavior.
- **Find Service Centers:** The user can click this button to find nearby service centers. This functionality is explained in section 5.1.5.
- **Find Hotels:** The user can click this button to find nearby hotels. This functionality is explained in section 5.1.5.
- **Find Restaurants:** The user can click this button to find nearby restaurants. This functionality is explained in section 5.1.5.

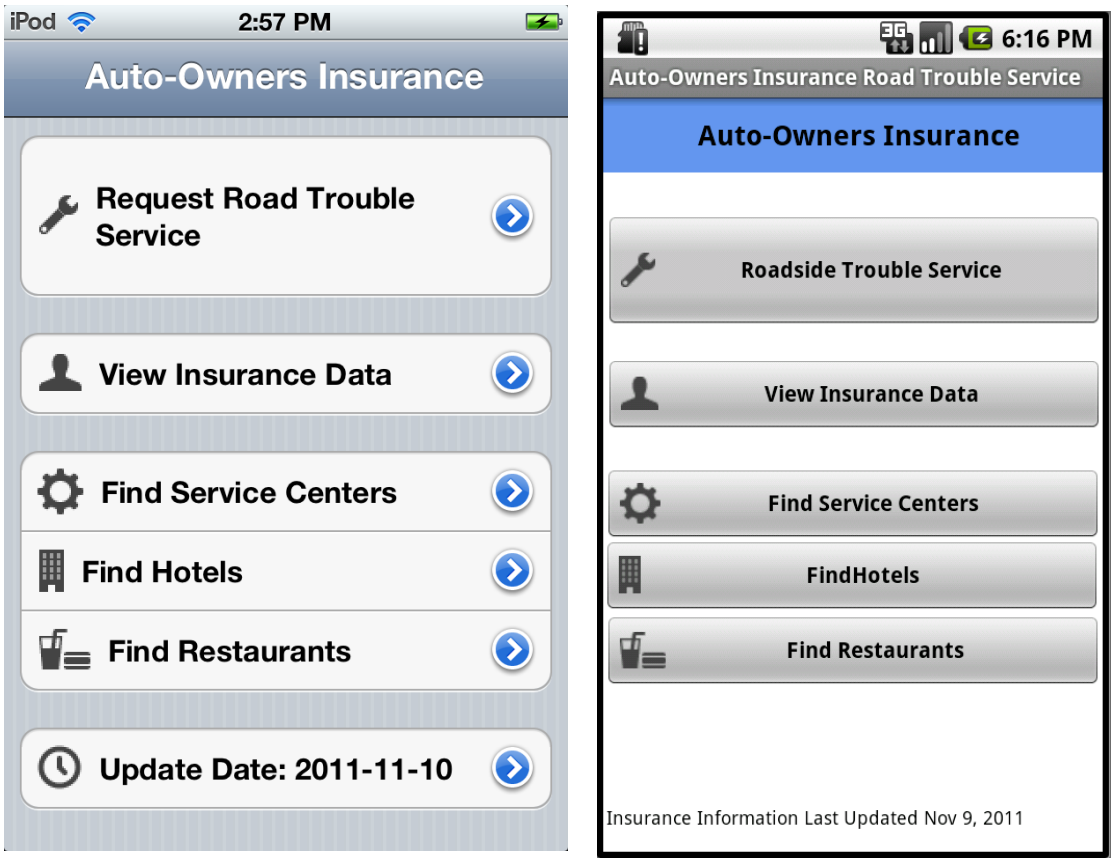



**Figure 5.1.2.1 Main Screen**

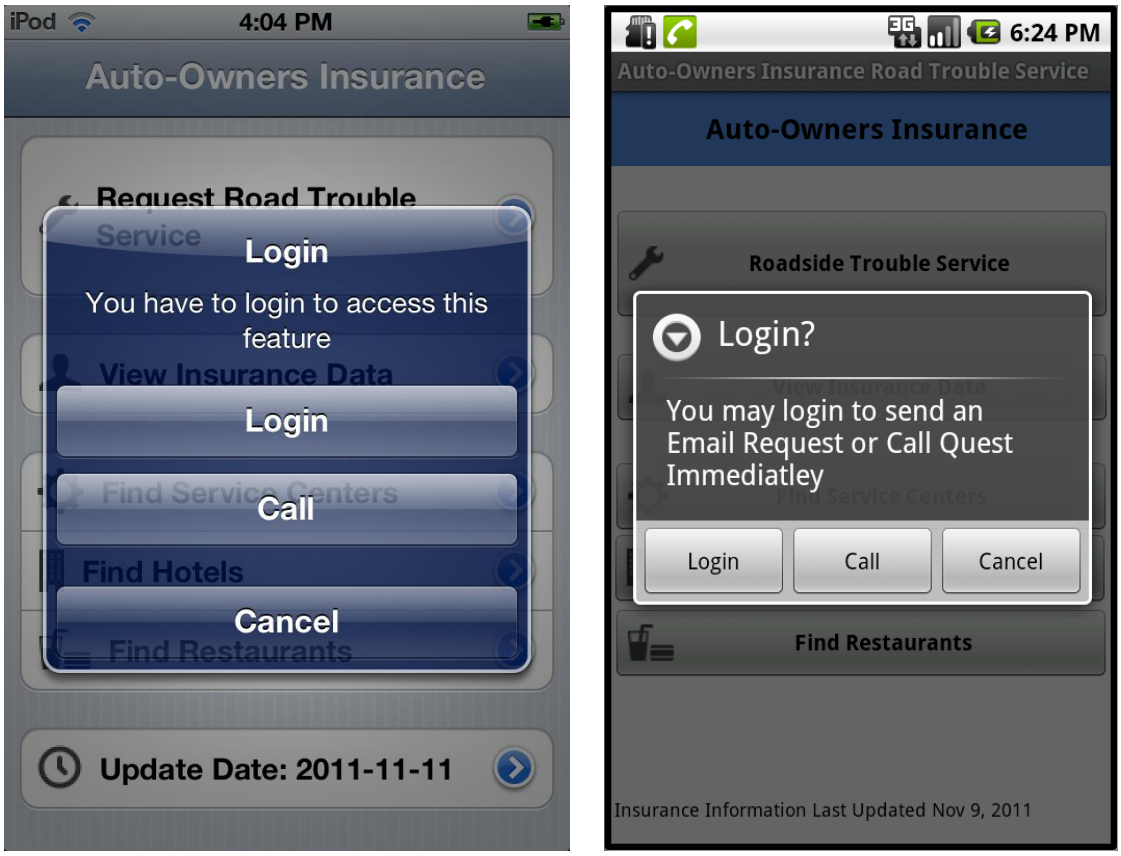



**Figure 5.1.2.2 Main Screen: Service Request Login Prompt**

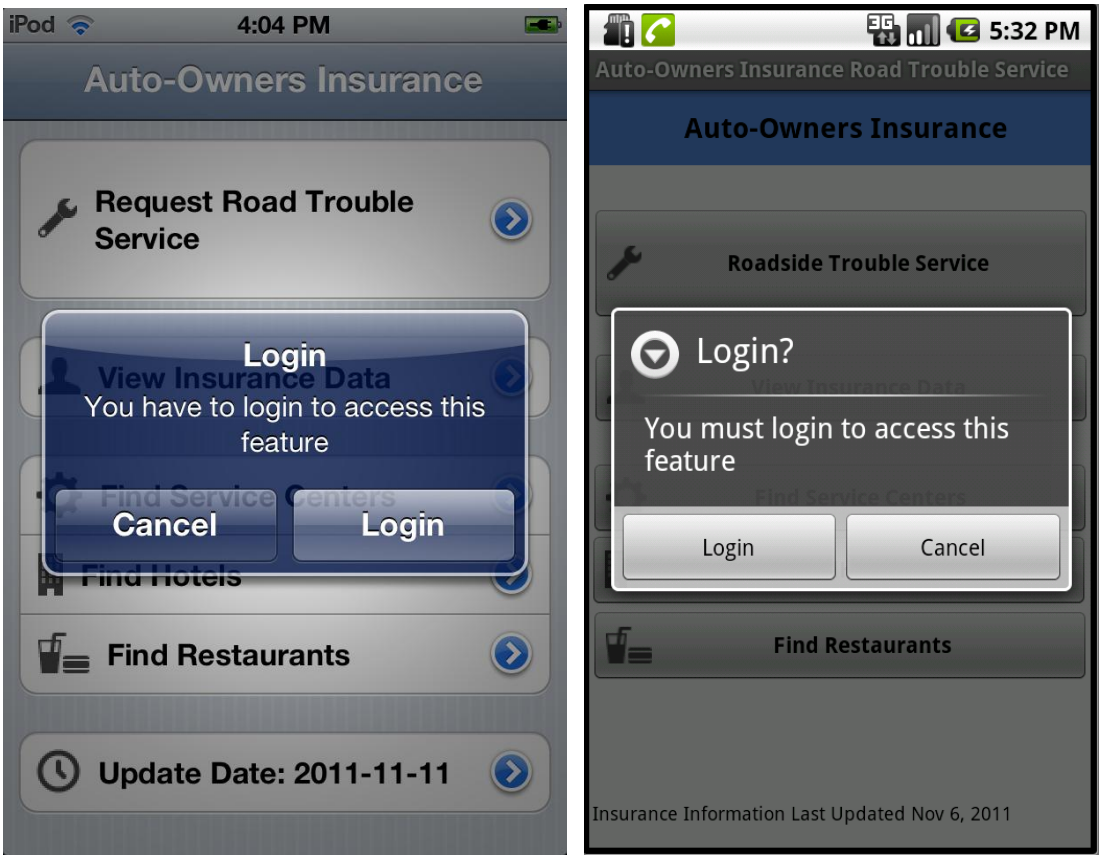



**Figure 5.1.2.3 Main Screen: Service Request Login Prompt**

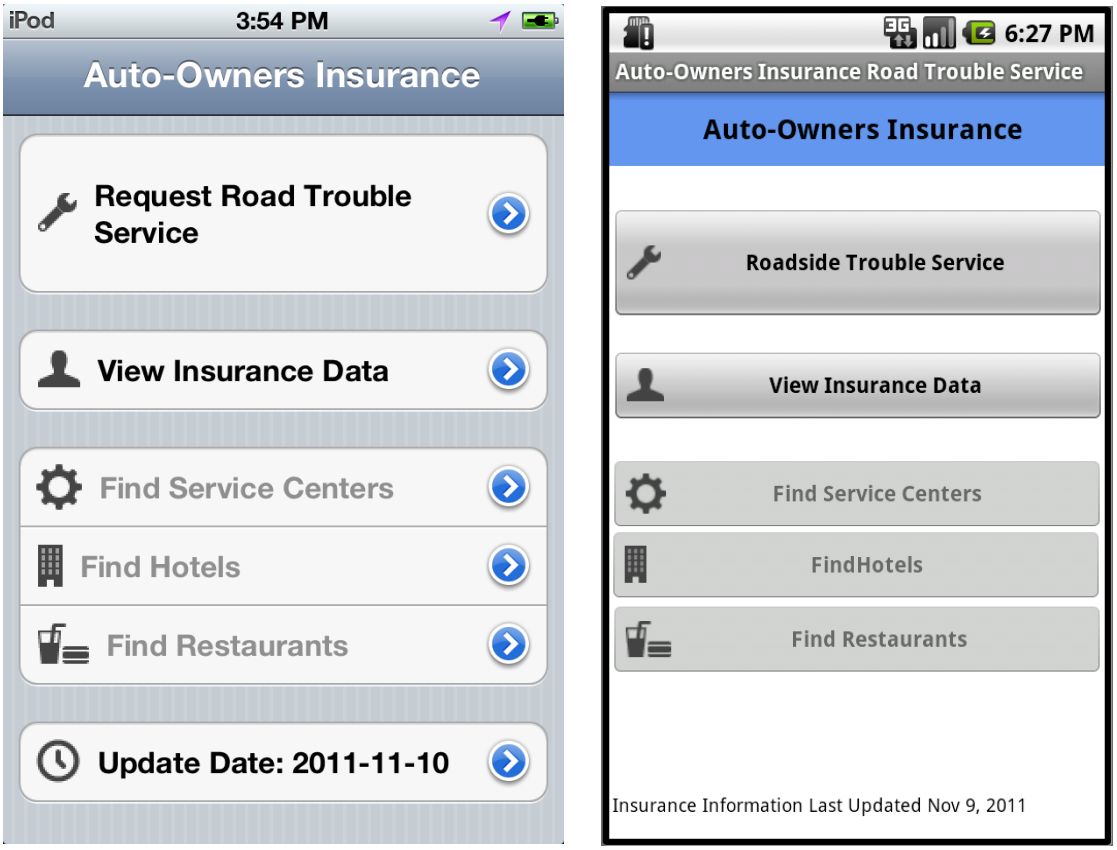



**Figure 5.1.2.4 Main Screen: No Internet Connection**

### 5.1.3 Service Request Screens

This series of screens guide the user through the steps to acquire service. The exact steps differ somewhat between Android and iPhone versions.

With the iPhone the user will see a screen that allows selection of a vehicle and then selection of which location to send. This is shown in figure 5.1.3.1. The text box populates with an address corresponding to the user's location and the map shown below will also visually display the user's current location.

- **Send With Current Location:** The user will select this button if they are at their vehicle.
- **Send With Modified Address:** The user should select this button if they are not at their vehicle. Before doing so, the user should type their modified location into the textbox.

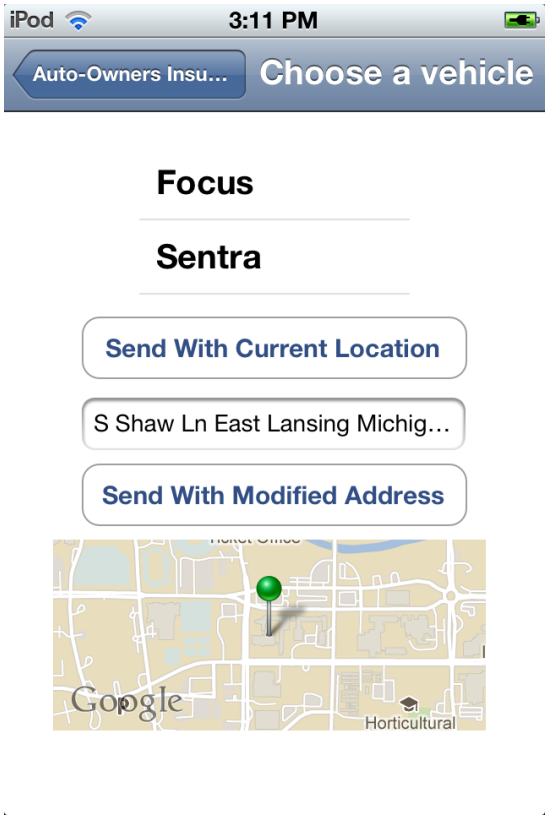



**Figure 5.1.3.1 Choose Location: iPhone**

When using an Android phone, a user will be prompted with the screen shown in figure 5.1.3.2. The map will display the user's current location and the text box will display a nearby address.

- **Send Current Location:** The user will select this button if they are at their vehicle.
- **Send Custom Location:** This user should select this button if they are not at their vehicle. Before doing so, the user should type their modified location into the textbox.

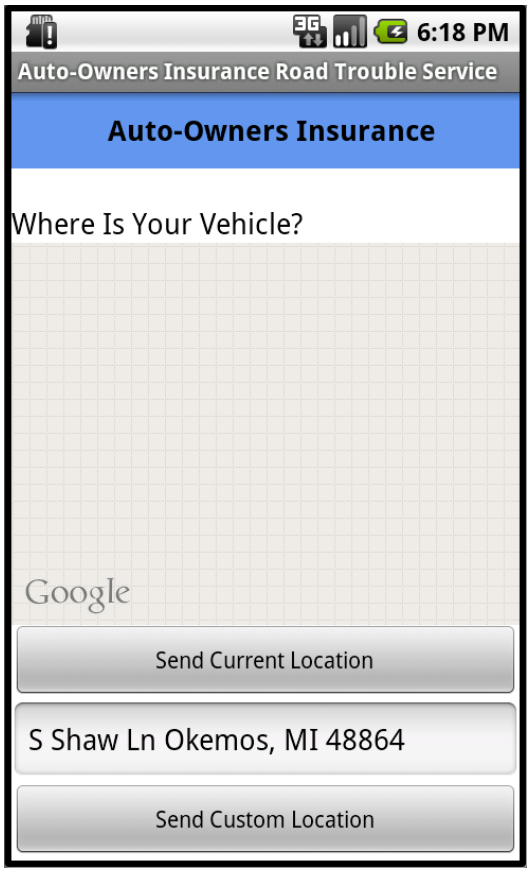



**Figure 5.1.3.2 Choose Location: Android**

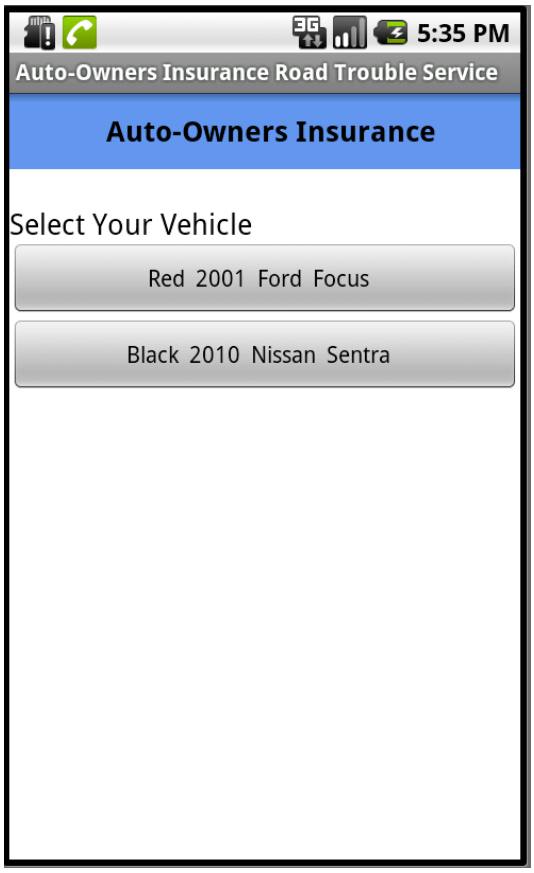



**Figure 5.1.3.3 Choose Vehicle: Android**

The user will then be sent to a vehicle selection screen shown in figure 5.1.3.3. This screen will be populated with all vehicles owned by the user that are insured by Auto-Owners Insurance. Upon selecting the vehicle in need of service the user will be sent to the smartphone's native email application. The native email client for iPhone is displayed in figure 5.1.3.4. Due to the inability to send emails from an Android emulator, only the iPhone version is displayed.

If the user had previously selected “Send Current Location” the email screen will be populated with the user’s latitude and longitude coordinates, their nearby address, and a map displaying their location. Otherwise the custom location will be displayed. Additionally, the user’s name, phone number and vehicle information will be shown.

- **Send:** The user will select this button if they are ready to email a service request to Quest. Additionally, this will send a transaction to the logging database. The user will be taken to the call screen shown in figure 5.1.3.5.
- **Cancel:** This user should select this button if they do not wish to request service.

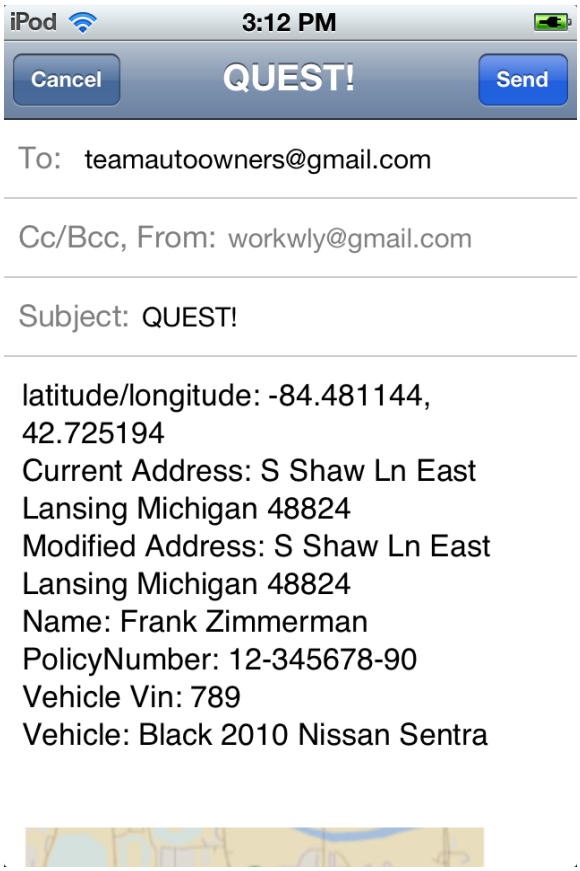



**Figure 5.1.3.4 Email Screen**

After the email is sent this user is prompted to call Quest.

- **Call Quest:** The user will select this button to make a confirmation call to Quest.

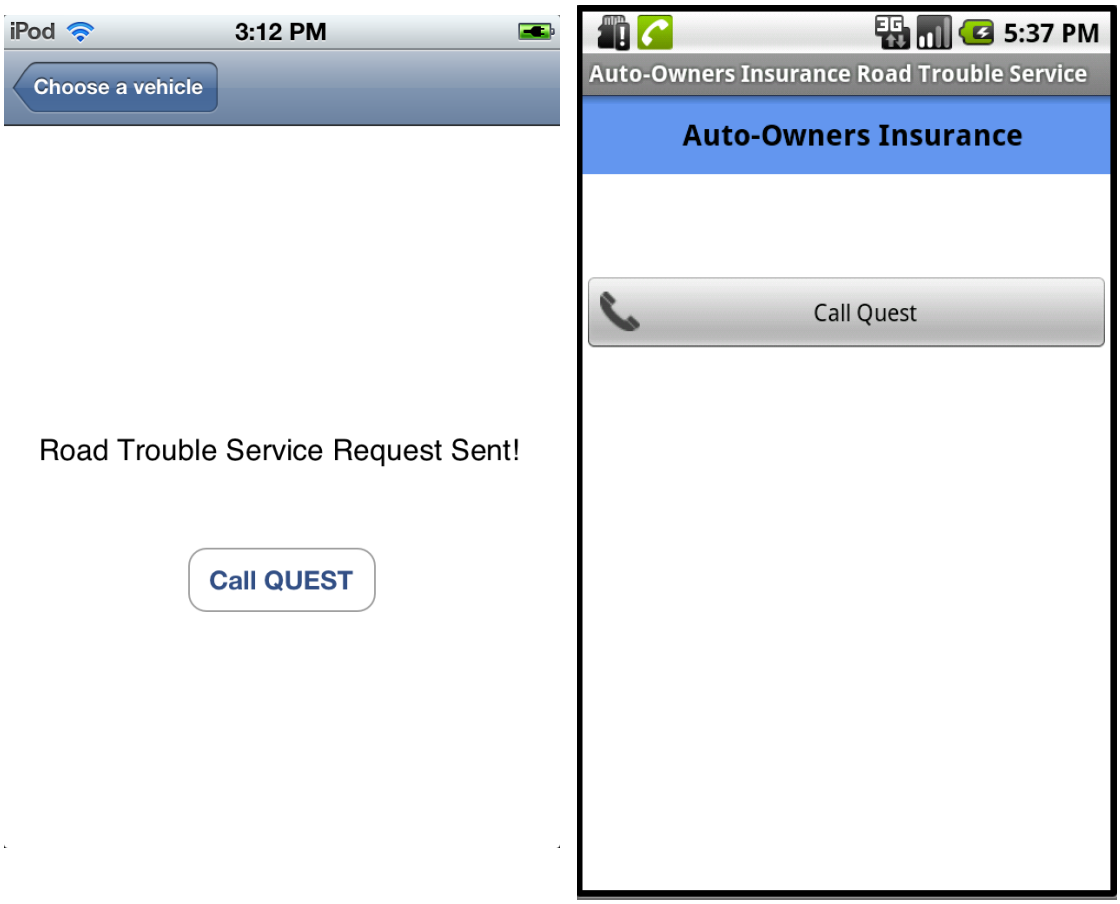



**Figure 5.1.3.5 Call Screen**

### 5.1.4 View Data Screens

By selecting "View Insurance Data" from the main screen, the user will be able to view their own insurance data. This data is stored locally on the phone after the initial login, and populated when this screen is opened. The user can click the policy button or any of the vehicles to obtain more detailed information.

- **Policy Button:** The user is taken to a screen showing their policy number as well as its start and end date. This screen is shown in figure 5.1.4.2.
- **Vehicle Button:** The user is taken to a screen showing their vehicle as well as its VIN and Roadside Trouble Service limit. This screen is shown in figure 5.1.4.3.

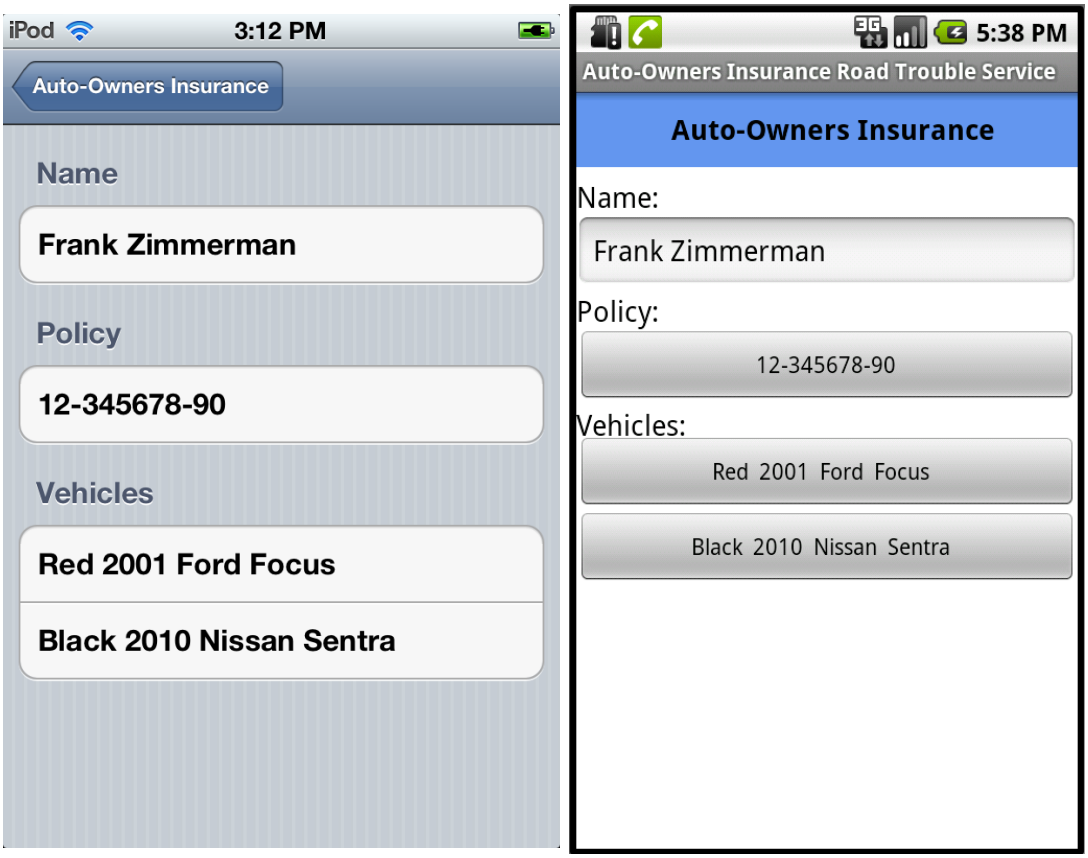



**Figure 5.1.4.1 View Insurance Data Screen**

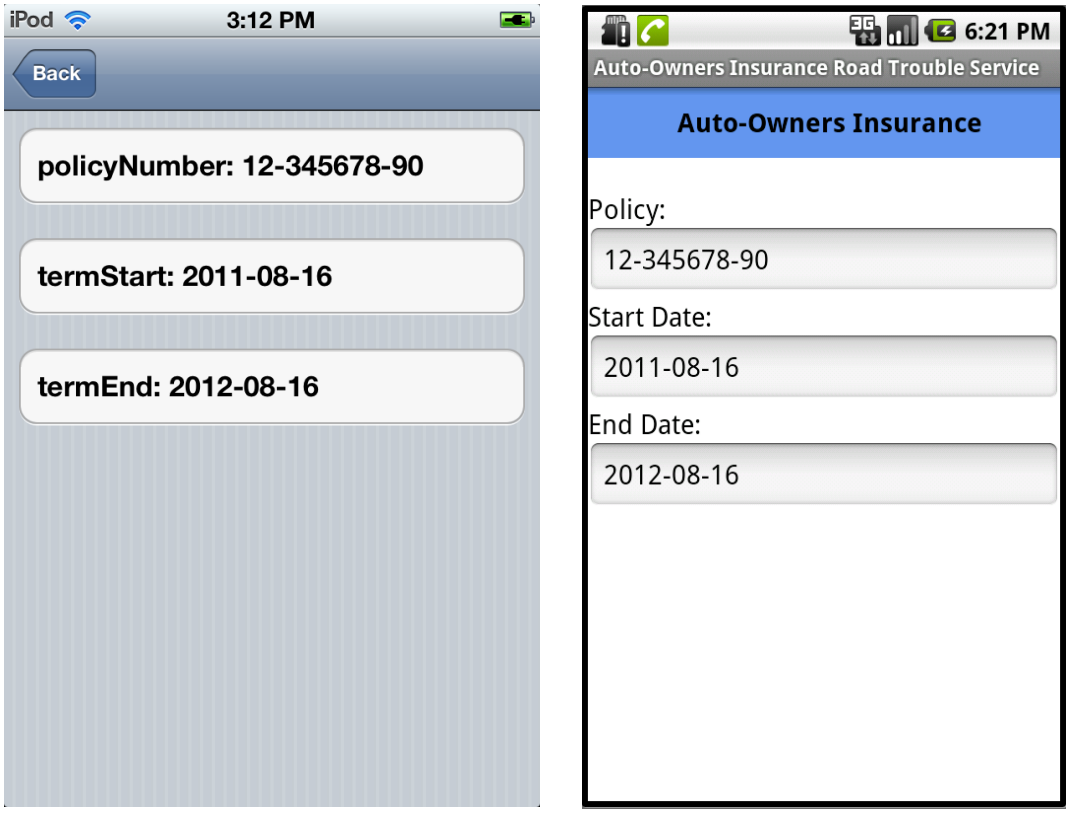



**Figure 5.1.4.2 View Policy Data Screen**

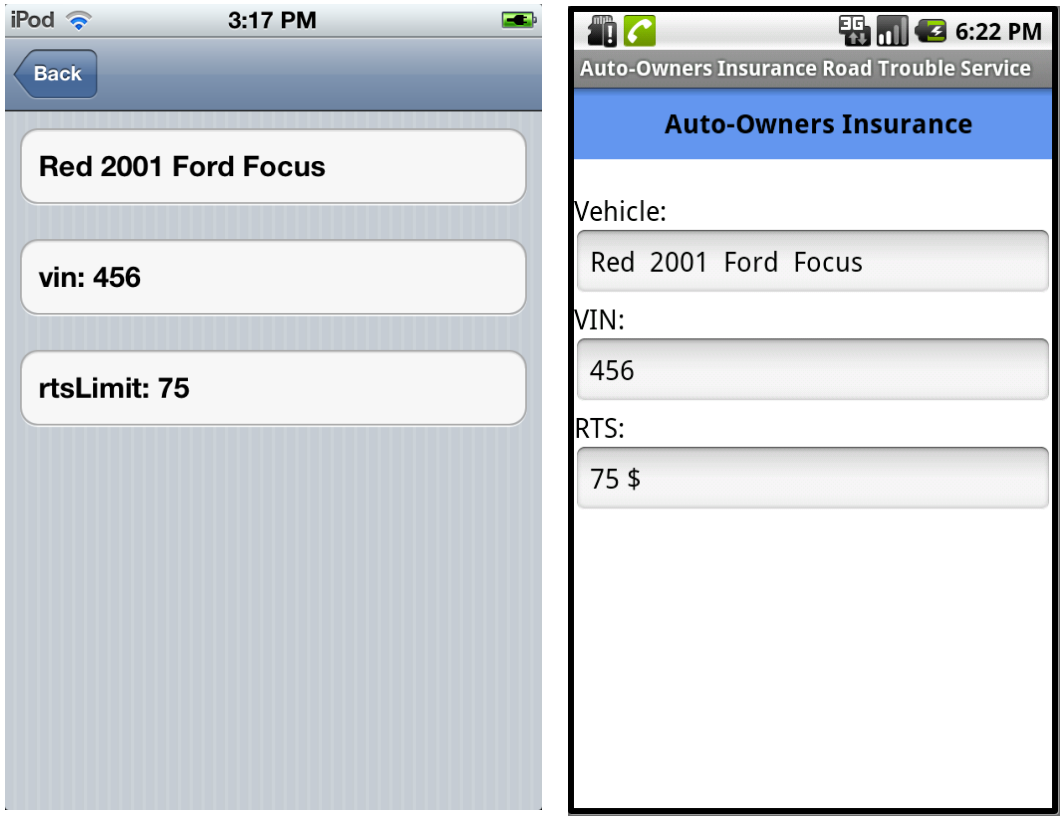



**Figure 5.1.4.3 View Vehicle Data Screen**

### 5.1.5 Nearby Locations Screens

In addition to requesting service, the application will be able to locate useful nearby places. These places consist of hotels, restaurants, and service centers. After selecting a place to search for off of the main screen, the application will send a query to Google Maps to find the requested nearby places. This will be returned in a list and displayed on the screen as shown in figure 5.1.5.1. The listing will show the place name as well as its distance away from the user's current location. By clicking on one of the listings, the native Google Maps application will open and display directions to the location

- **Location Button:** The phones Google Maps application is opened and directions to the selected location are displayed as shown in figure 5.1.5.2.

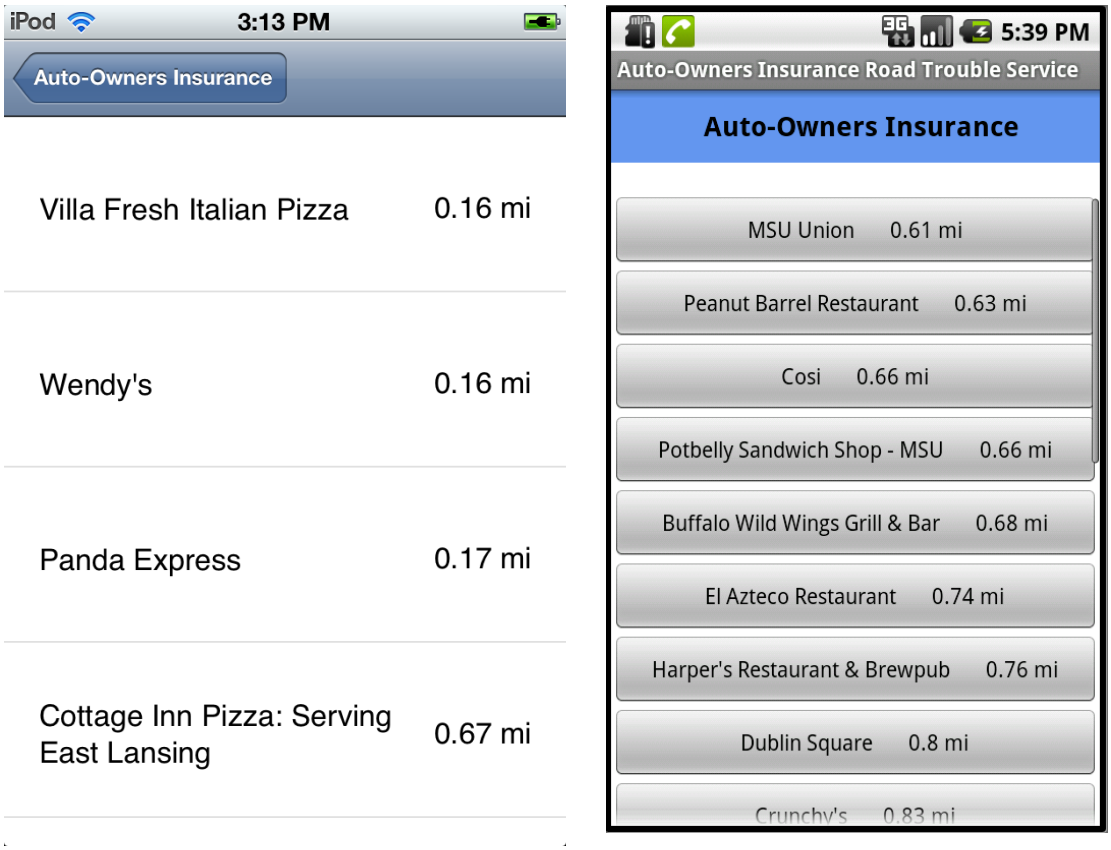



**Figure 5.1.5.1 Place Listing Screen**

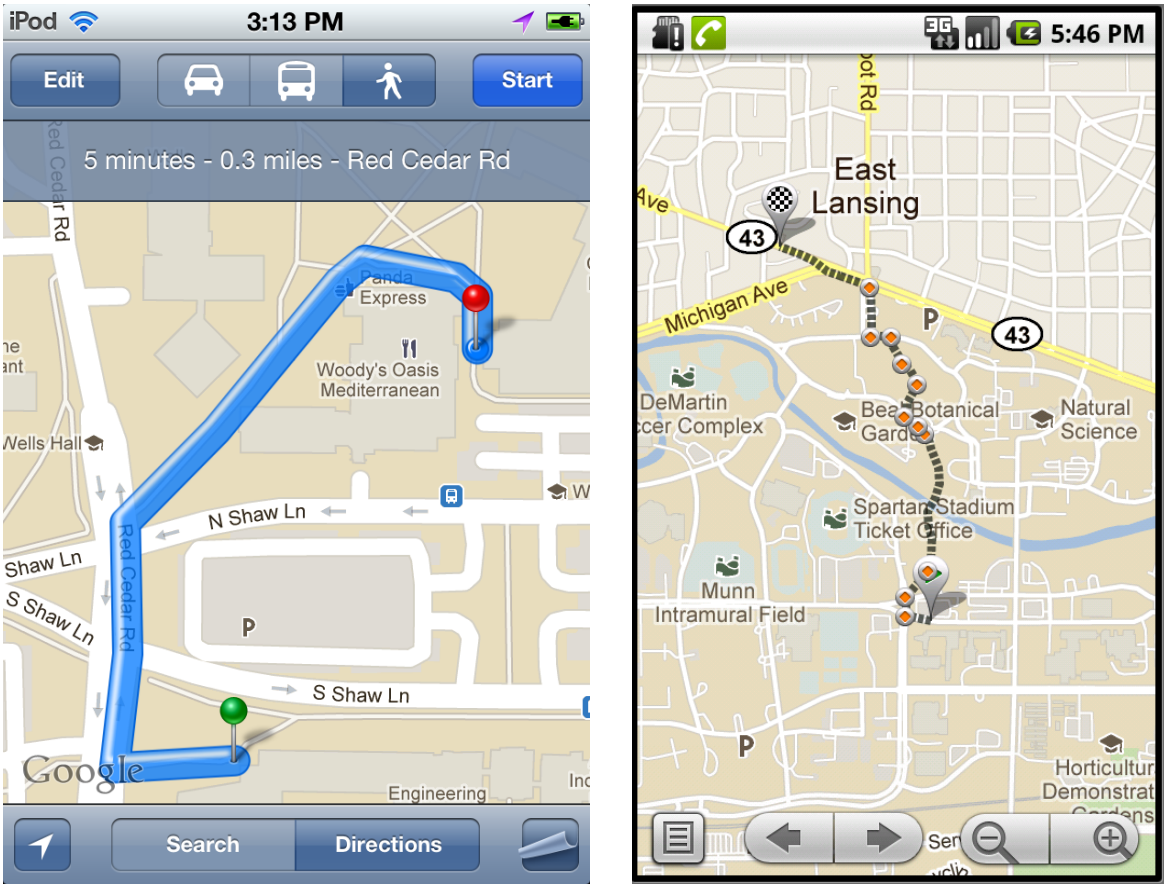



**Figure 5.1.5.2 Map Screens**

## 5.2 Class Diagrams

Although the functionality of the applications in both platforms is the same, details in implementation differ somewhat due to design decisions. Section 5.2.1 will describe the Android implementation while section 5.2.2 will describe the iPhone implementation. The classes diagramed are those involved in providing the back end functionality to the applications, classes for the user interface are not detailed.

### 5.2.1 Android Class Diagram

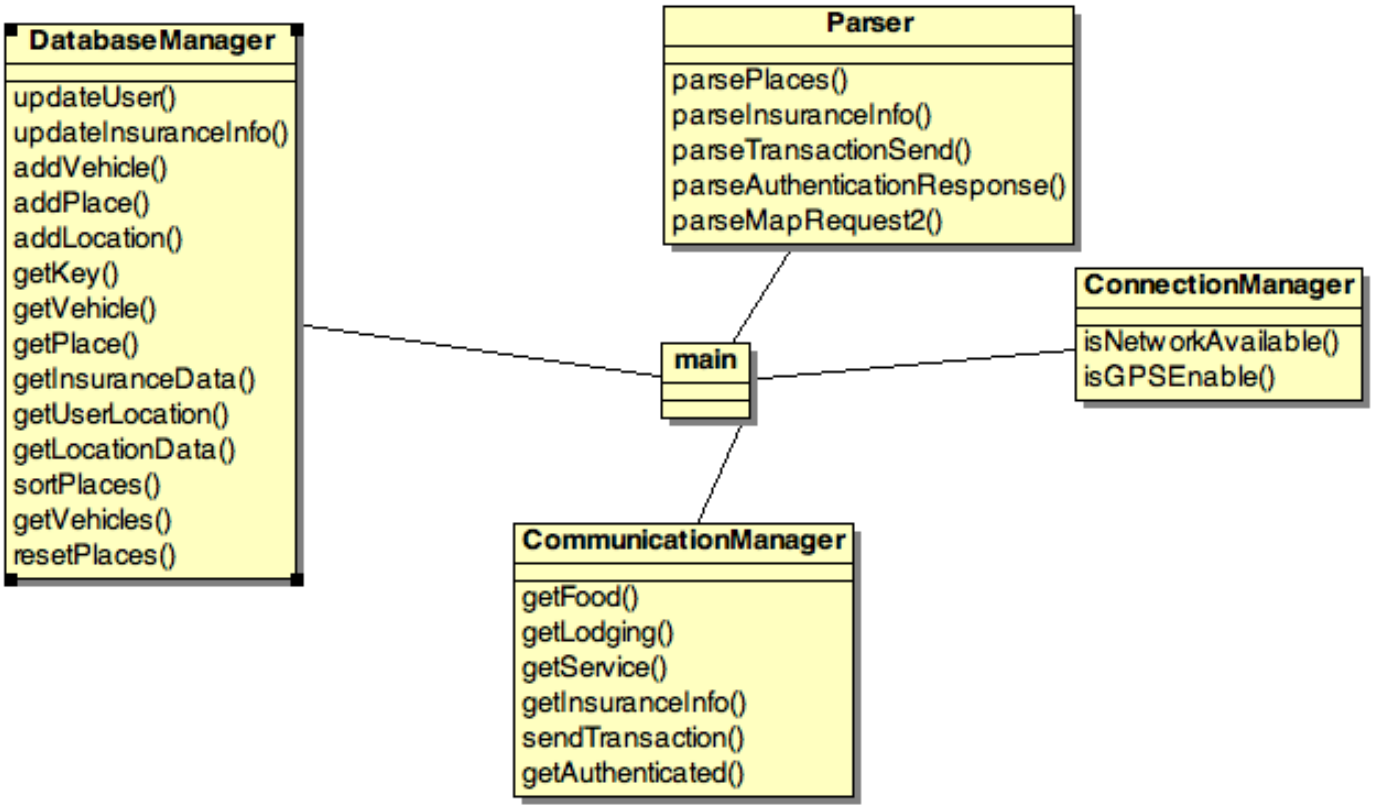



**Figure 5.2.1.1 Android Application Class Diagram**

#### 5.2.1.1 Main

The Main class is the root of the program. In this diagram it is representative of the activities and controllers that will use these classes. This will make calls to all the other various classes required while the program is in use.

#### 5.2.1.2 DatabaseManager

The DatabaseManager will store and retrieve data from the SQLite database on the phone. This data includes the user's insurance information, vehicle information, and nearby places locations.

#### 5.2.1.3 CommunicationManager

The CommunicationManager sends the requests to retrieve insurance information and nearby places. It also sends data to the logging database. Additionally it is able to tell if the provided login information is correct. The figure below shows the process for pushing and retrieving data to and from the databases.

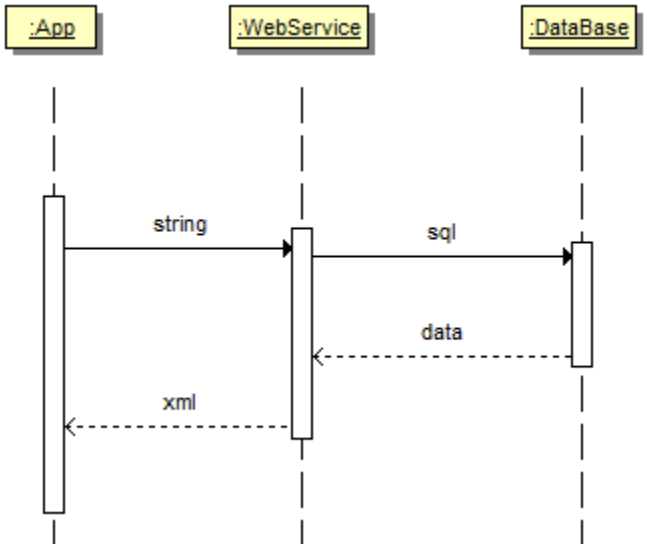



**Figure 5.2.1.3.1 Data Retrieval Diagram**

#### 5.2.1.4 ConnectionManager

The connection handler contains functions for determining network and GPS connectivity status. This is important to providing the user with the correct options depending on their access to the internet.

#### 5.2.1.5 Parser

The Parser class takes responses from the communications manager and processes them into formats the database will be able to hold. Additionally, it appropriately formats information to be sent to the communication manager to send to the logging database.

### 5.2.2 iPhone Class Diagram

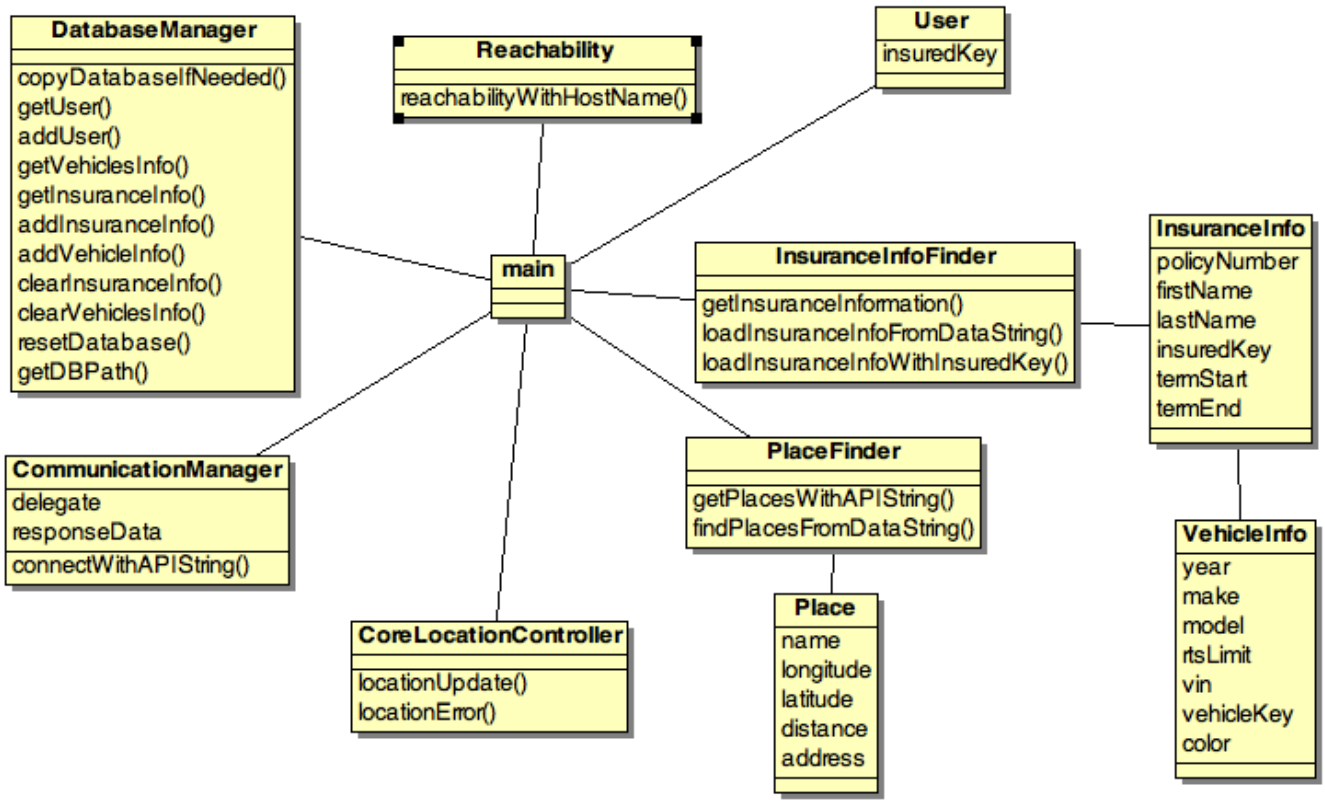



**Figure 5.2.2.1 iPhone Application Class Diagram**

#### 5.2.2.1 Main

The Main class is the root of the program. In this diagram it is representative of the activities and controllers that will use these classes. It will make calls to all the other various classes required while the program is in use.

#### 5.2.2.2 DatabaseManager

The DatabaseManager will store and retrieve data from the SQLite database on the phone. This data includes the user's insurance information, vehicle information, and nearby places locations.

#### 5.2.2.3 CommunicationManager

The CommunicationManager sends the requests to retrieve insurance information and nearby places. It also sends data to the logging database. Additionally it is able to tell if the provided login information is correct.

#### 5.2.2.4 InsuranceInfoFinder

The InsuranceInfoFinder parses the retrieved JSON string to get the insurance information.

#### 5.2.2.5 PlaceInfoFinder

The PlaceInfoFinder parses the retrieved JSON string to get nearby place information.

#### 5.2.2.6 CoreLocationController

The CoreLocationController is used to get the current longitude and latitude coordinates of the phone.

#### 5.2.2.7 Reachability:

The Reachability class detects if there is a network connection by using a host name like www.google.com.

#### 5.2.2.8 User/InsuranceInfo/VehicleInfo/Place

These classes provide a structure to hold their respective data.

## 6. Database Details

Two external databases were required to implement the project. Figure 6.1 represents the mock Auto-Owners Insurance database schema. The smartphone applications will pull insurance information from the database based on the provided username and password.

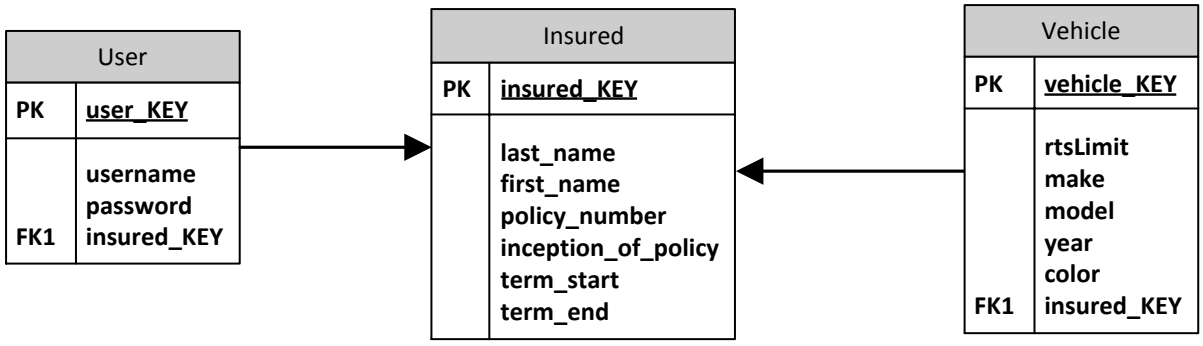



**Figure 6.1 Mock Auto-Owners Database Schema**

Figure 6.2 shows the database schema for the logging database. The smartphone applications will push transaction data to this database each time a service request is submitted.

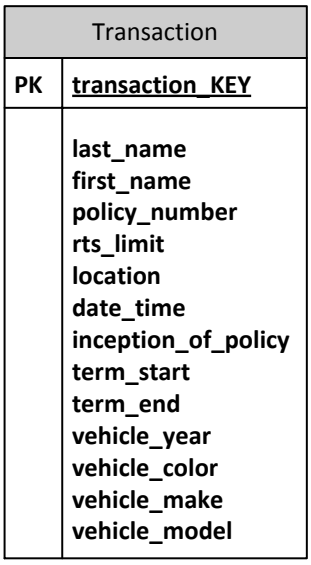



**Figure 6.2 Usage Logging Database Schema**

## 7. Website Details

A website will be provided for administrative purposes. The website will pull information about service requests from a logging database. This website will enable access to various graphs and tables based on the logging data.

### 7.1 Website Statistics View

The statistics view holds three charts (one is depicted below in figure 7.1.1). These three charts graph usage by Road Trouble Service Limit (RTS Limit), Vehicle Make, and the state that the request took place in. There are filtering options available for these charts. These filters can be seen in figure 7.1.2. They include state, start and end time, and start and end date.

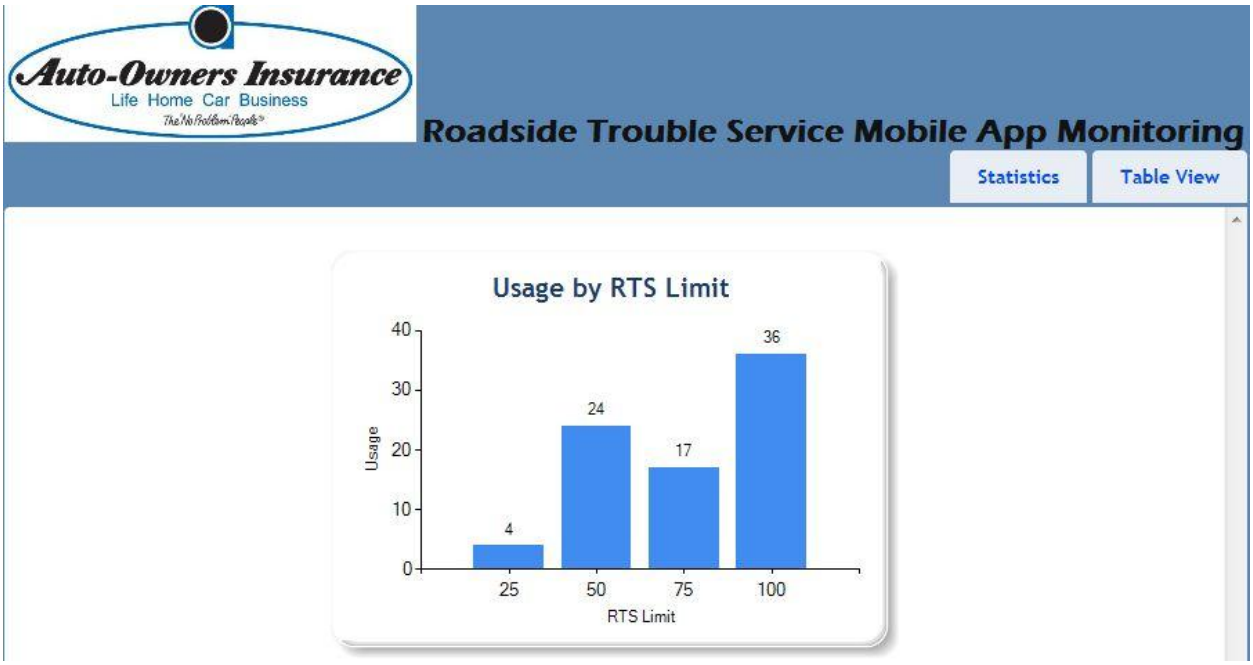
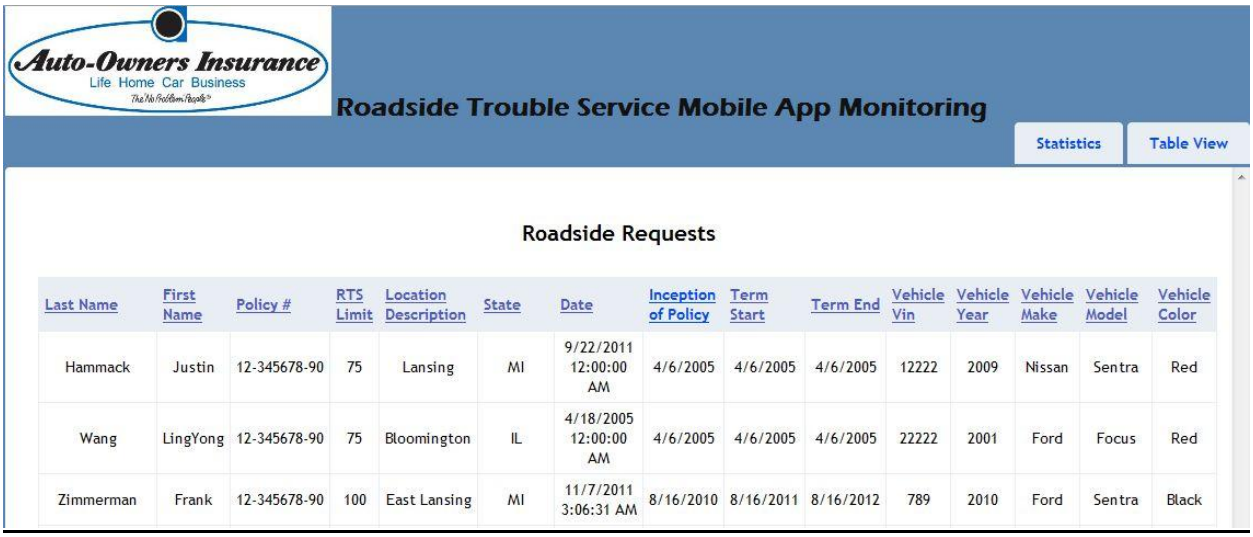



**Figure 7.1.1 Website Statistics View**


**Figure 7.1.2 Website Statistics Filters**

### 7.2 Website Table View

The website has a table view for viewing all transactions at once. This is depicted in figure 7.2.1. This table will sort by any column that the user clicks.




| Last Name | First Name | Policy # | RTS Limit | Location Description | State | Date | Inception of Policy | Term Start | Term End | Vehicle Vin | Vehicle Year | Vehicle Make | Vehicle Model | Vehicle Color |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Hammack | Justin | 12-345678-90 | 75 | Lansing | MI | 9/22/2011 12:00:00 AM | 4/6/2005 | 4/6/2005 | 4/6/2005 | 12222 | 2009 | Nissan | Sentra | Red |
| Wang | LingYong | 12-345678-90 | 75 | Bloomington | IL | 4/18/2005 12:00:00 AM | 4/6/2005 | 4/6/2005 | 4/6/2005 | 22222 | 2001 | Ford | Focus | Red |
| Zimmerman | Frank | 12-345678-90 | 100 | East Lansing | MI | 11/7/2011 3:06:31 AM | 8/16/2010 | 8/16/2011 | 8/16/2012 | 789 | 2010 | Ford | Sentra | Black |

**Figure 7.2.1 Website Table View**

## 8. Technologies

### 8.1 Development

The Android application was developed in Eclipse on Windows using the Java programming Language. The iPhone application was developed in Xcode using the ObjectiveC programming language. The website was developed in Visual Studio using ASP.NET and MVC 3. Version control was administered through TFS. The web service is a Restful web service programmed in Eclipse and running on Tomcat. The databases are run on our server computer using MSSQL Express. They are maintained through Visual Studio.

### 8.2 System Architecture

Our project consists of an application for both Android and iPhone platforms. They communicate with SQL databases through a Restful Web Service. Additionally, an administrative website for tracking usage also communicates with a database through the web service. This is illustrated in figure 8.2.1.

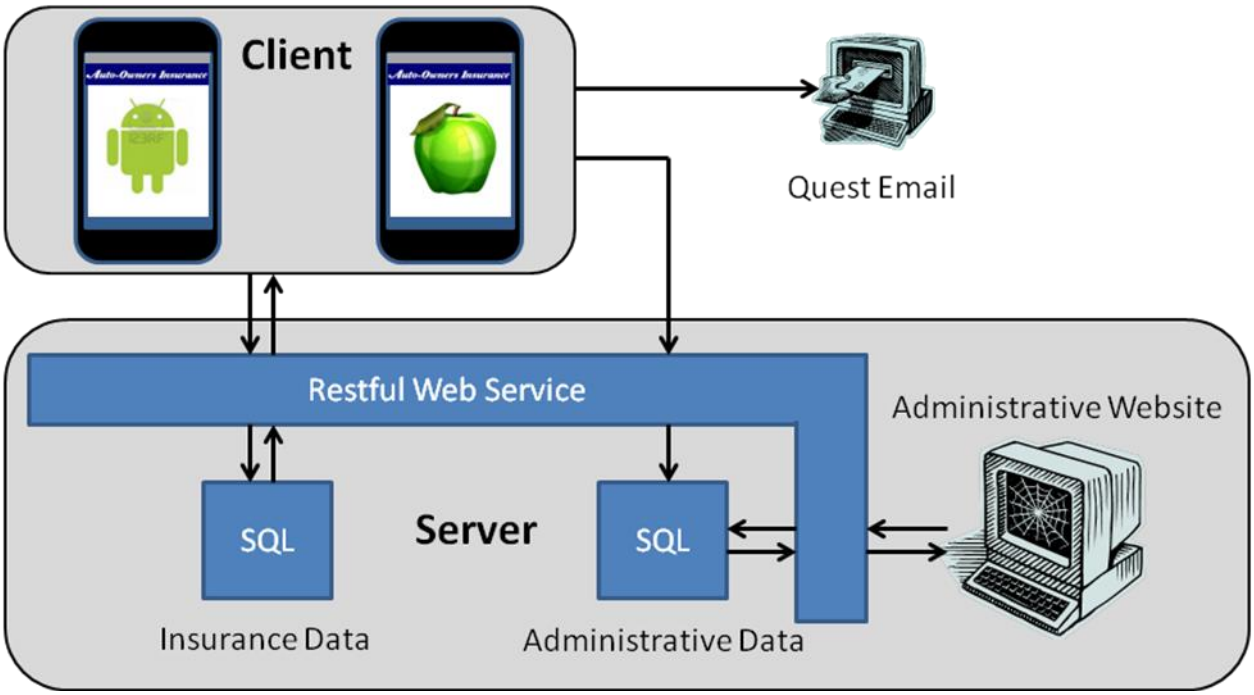



**Figure 8.2.1 Architecture Diagram**

### 8.3 Web service Architecture

To obtain information from the web service the application sends an HTTP request to our database. This HTTP request is parsed to determine what the specific database command is. This parsed URL is sent to the appropriate servlet, which sends a query to the MySQL database over a Java Database Connection. The servlet then formats and send the data back to the application over an HTTP response. This is illustrated in figure 8.3.1.

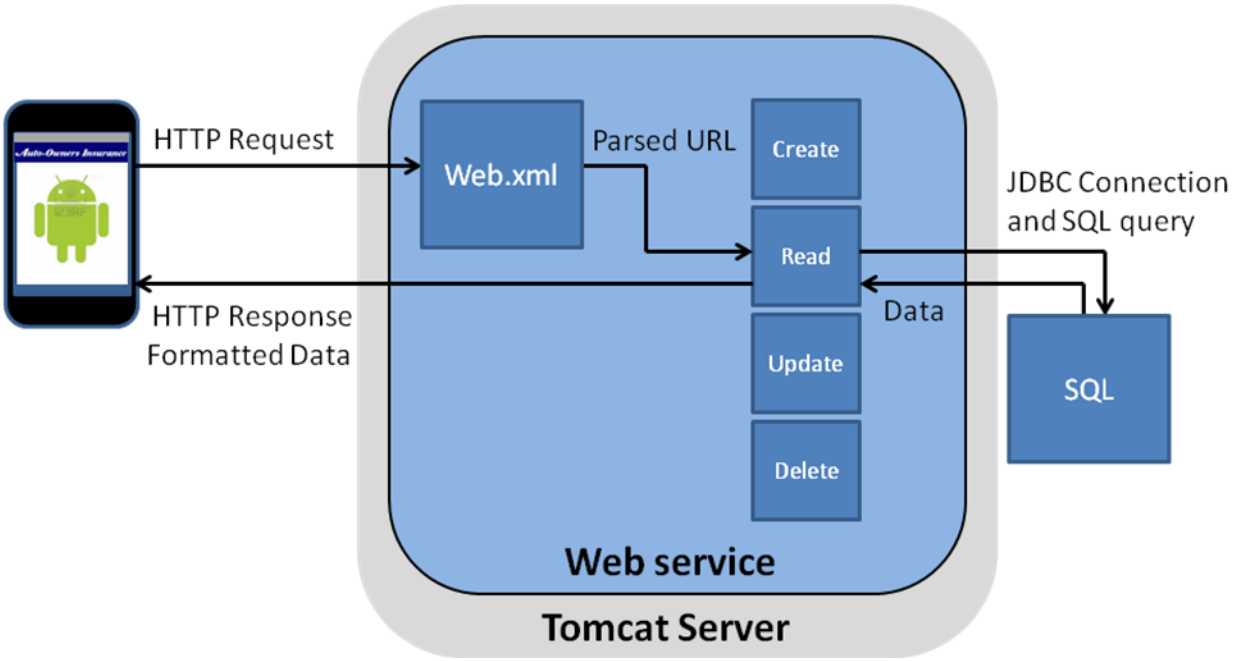



**Figure 8.3.1 Web Service Diagram**

## 9. Schedule

Week1: August 29 – September 4

- Received project description
- Met in person with Auto-Owners Insurance

Week2: September 5 – September 11

- Configured Lab Computers
- Created hello world apps in Android and iPhone
- Created a basic webpage.

Week3: September 12 – September 18

- Worked on project plan
- Set up SQL databases
- Work on web service

Week4: September 19 – September 25

- Complete Web Service
- Be able to push and pull data
- Finish project plan

Week5: September 26 - October 2

- Application and Web App can use webservice
- Begin GUI construction for web app and website

Week6: October 3 - October 9

- Application can get longitude and latitude
- Applications can email quest
- Applications can find hotels, restaurants, and service centers(with or without use of an external app)
- Website can make some sort of graph/table

Week7: October 10 –October 16

- Prepare for Alpha Presentation

Week8: October 17-October 23

- Finish and refine features
- Alpha Presentation: October 17

Week9: October 24 - October 30

- Finish and refine features

Week10: October 31 - November 6

- Finish and refine features
- Prepare for Beta Presentation

Week11: November 7- November 13

- UI refinements
- Bug Fixes
- Beta Presentation: November 9

Week12: November 14 - November 20

- UI refinements
- Bug Fixes
- Work on project video

Week13: November 21 - November 27

- UI refinements
- Bug Fixes
- Work on project video

Week14: November 28 - December 4

- Work on project video
- Prepare deliverables

Week15: December 5 - December 9

- Design Day: October 9
- Prepare for Design Day