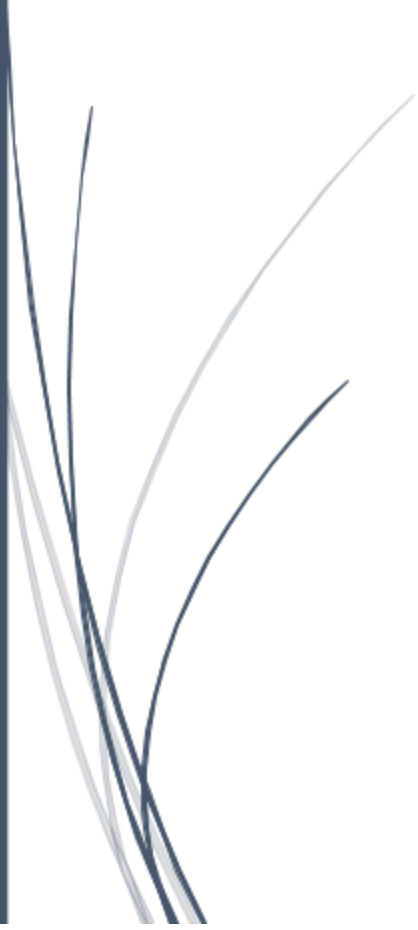
2015-2016

GuoSquad Senior Projects

Department of Software Engineering



David Piro, Matt Kress, Jay Shah, Wes Wickwire
FLORIDA GULF COAST UNIVERSITY

Florida Gulf Coast University
10501 FGCU Blvd, South
Ft. Myer, Fl. 33965

Contents

Author's Note: This project documentation is dedicated to the department in which it was derived.. Please note that this will be the final edit and maintenance for this project. Finally I'd like to also thank the team involved for their support and dedication. Without you guys the work involved would have never manifested into something of rigor, beauty, and professionalism. I hope your trials shall continue and whatever pointers you come across in life will refer you to new beginnings.

Hurtz don't it,

David Piro

iOS Mobile Application

Authored By,

Matt C. Kress
David J. Piro Jr.
Jay P. Shah
Wes Wickwire

Florida Gulf Coast University
10501 FGCU Blvd, South
Ft. Myers , Fl

Winner

Of The

2016 Research Day

Dean's Award

U.A Whitaker College of Engineering

Outstanding Undergraduate Poster

Hertz iOS Mobile Application (IMA)

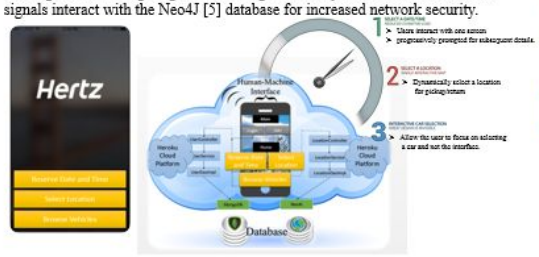
Matt Kress, David Piro, Jay Shah, Wes Wickwire
U.A. Whitaker College of Engineering, Florida Gulf Coast University
Instructor(s): Dr. Janusz Zalewski and Dr. Fernando Gonzalez
CEN 4935: Senior Software Engineering Project

Introduction

Today's market of mobile applications has emerged as primarily a consumer driven mode of production. That is the sustained exponential growth is driven by the rate mobile applications are purchased, the speed which they are developed and their efficiency in relation to the consumer. These market demands have prompted the motivation for an increased demand in programming employment as well as independent companies that offer programming services that accelerate software production. By drawing from these services, major Fortune 500 companies such as Hertz [1] utilize these opportunities to capitalize on today's software demands. For the most part, the marginal utility for mobile technologies and rate of mobile application production are unequivocal, allowing educational institutions to integrate mentor-client relationships with these companies. These relationships offer opportunities such as corporate experience, technical skills and supplemental education, which exist outside the restricted to a curriculum.

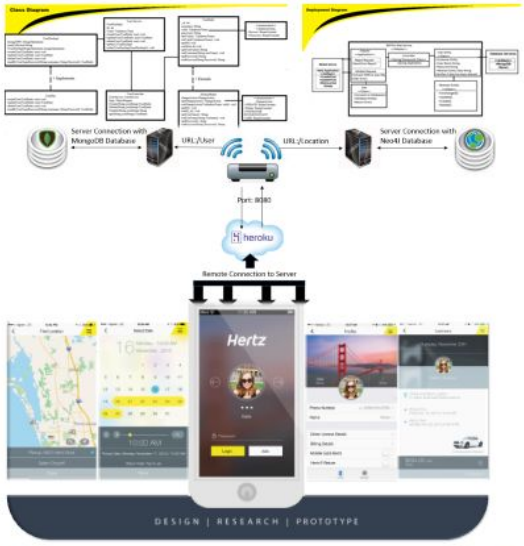
Solution Design

The IMA software architecture is divided into 4 modules including the iOS application (Human-Machine Interface), Heroku Cloud Platform, MongoDB and Neo4J. The iOS app [2] provides the user with the following functions; *Login, Join, Browse Vehicles, Select Location and Reserve Date and Time*. All five functions are programmed as buttons on the user screen, but exist on two different interfaces. The first interface, Main, offers the login and join buttons. Once one of these buttons is selected and carried out, the user will be brought to the Home interface. This interface, provides the user with three services, reserve a car, reservation location and Find Location. The Heroku Cloud Platform [3] manages the function requests based on the user sent HTTP requests. This platform offers two different micro services. The first service consists of the UserController, User Service, and UserDaoImpl. These services involve the sending and receiving of user related account information, billing information and account preferences, which are stored in MongoDB [4]. The second services are the LocationController, LocationServices and LocationDaoImpl. This set of services regard the sending and receiving of geolocation signals and system notifications. These signals interact with the Neo4J [5] database for increased network security.



Abstract

This project offers an object-oriented programming methodology, structuring this application to adapt and integrate into today's market. By developing this app., our approach corresponds to a 1-1 relationship with Hertz, who would provide the aspects that our project had to meet. Further entailing, a mobile application which wirelessly connects and interacts with customers. This project's research contributed on several aspects, the fulfillment of a machine interface integrated with web application services. The first Web application documents client information. While the other, a cloud computing service which mediates databases and client interface commands. Also client services including, weather tool, Forecast.io, Geolocation and vehicle reservation were incorporated into the software architecture. In this project, I designed the software architecture featuring the software requirements specification. My design features the integration of a minimalist interface, software services and functionality.



References

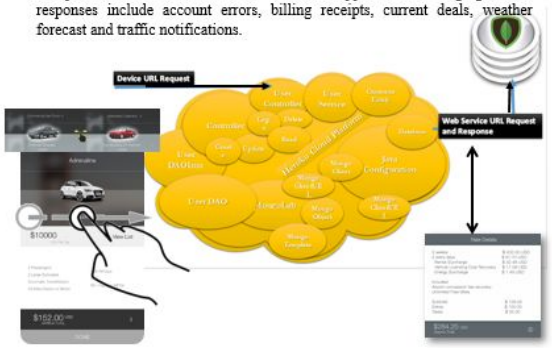
[1] Hertz. (2015). *The Hertz Corporation*. URL: https://www.hertz.com/rentacar/reservation/
[2] iOS. (2015). *Apple iOS 9*. URL: http://www.apple.com/ios/
[3] Heroku. (2015). *Cloud Application Platform*. URL: https://www.heroku.com/
[4] MongoDB. (2015). *Download MongoDB*. URL: https://www.mongodb.org/
[5] Neo4J. (2015). *Neo4j*. URL: http://neo4j.com/
[6] Spring MVC Framework. (2016). *Spring*. URL: http://spring.io/
[7] Walls, Craig. *Spring in Action Fourth Edition*. New York: Manning, 2015. Print

Conclusion

The Hertz Mobile Phone Application has completed the integration of a fully deployed server, application and databases. The interactions of these modules allow a customer to reserve a vehicle based on their preference and needs. The process in doing so is clean, sleek and quick to comfort the user, while rewarding the user with a unique experience. In addition, Hertz's customer satisfaction is a top priority and provides their guests with additional notifications. In the future, this system will feature both, a credit card and driver's license camera reader. This function would increase mobile phone security and create faster reservation times for the Vehicle Checkout Process.

Implementation and Testing

The IMA software implementation may be understood in 3 components; *Application software, Server Software and Database Functionality*. The iOS application software features a minimalist approach, where a total of 4 screens are interacted with to perform a single service. These services are carried out through the cloud platform, featuring RESTful Web Services [6] incorporated with a Spring Framework [7]. This design allows for the addition of future services, security measures and URI's (Uniform Resource Identifier) such as the *MongoClient URI* in the Server Context Diagram found below. This URI provides our application with access to MongoDB's database as well as the translation of User Requests from Java to JSON objects. Furthermore, the database functionality provides the appropriate responses to our platform, but also allows for additional application messaging. These responses include account errors, billing receipts, current deals, weather forecast and traffic notifications.



"Great Spirits have always encountered violent opposition from mediocre minds"

-Albert Einstein

1. Introduction

Mobile phone application development is a rapidly growing field, where corporations, businesses and developers are consistently meeting the demands of the consumer market. The reason is that mobile applications conveniently access services and are easily integrated with the limited operating systems available (e.g. Android [1] and iPhone [2]). A common example of this practice is mobile applications which incorporate ecommerce or location services (Figure 1.1).

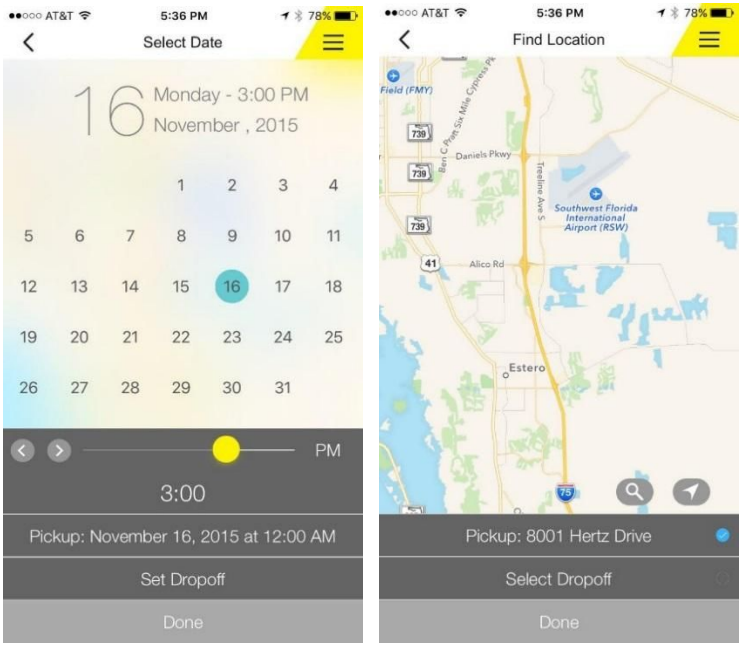


Figure 1.1 Hertz iOS Vehicle Reservation and Location Selection Interfaces

In line with this trend, the primary aim of this project is to develop an iOS user interface (Figure 1.2) that provides vehicle checkout and reservation services for Hertz [3], in particular, by integrating both, a front-end and back-end design.

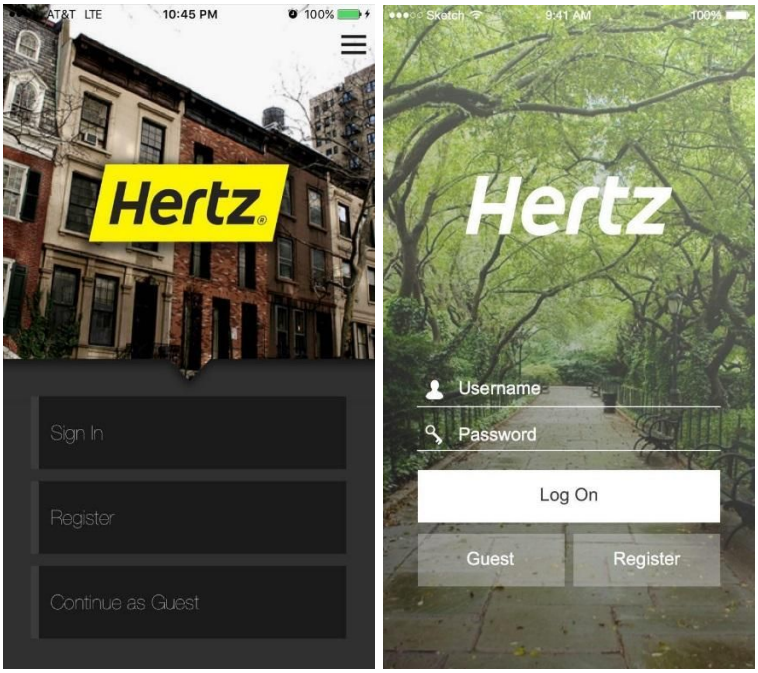


Figure 1.2 Hertz iOS User Interface Prototypes

2. Software Requirements Specification

2.1 Introduction to Requirements

2.1.1 Purpose

The two leading platforms for mobile phone development are the Android and iPhone operating systems. The primary aim of this project is to implement a functional mobile phone application that meets the design requirements from our client, Hertz. The purpose of this document describes the architecture and system design of the Hertz mobile phone application for iPhone platforms.

2.1.2. Scope

This Software Requirements Specification is for a high level system which works as a proof of concept for the use of building a mobile phone application. This document will define and describe; the software architecture design, functional requirements and the human interface in relation to the software.

2.1.3 Definitions, Acronyms, and Abbreviations

DAO - Data Access Object

Data Structure – A specialized format for organizing and storing files.

JSON – JavaScript Object Notation

MongoDB – NoSQL Document Database

Neo4j – Graphing database

RESTful Services – Representational State Transfer

Spring MVC Framework – A component from Spring that implements your web application according to the Model-View-Controller design pattern

URI – Uniform Resource Identifier

2.1.4 Physical Interactions

The following physical diagram includes all participating physical entities, especially networks, databases and web services (Figure 2.5). The framework provided by the web services is able to direct the flow of information exchange between all participating entities, as input or output.

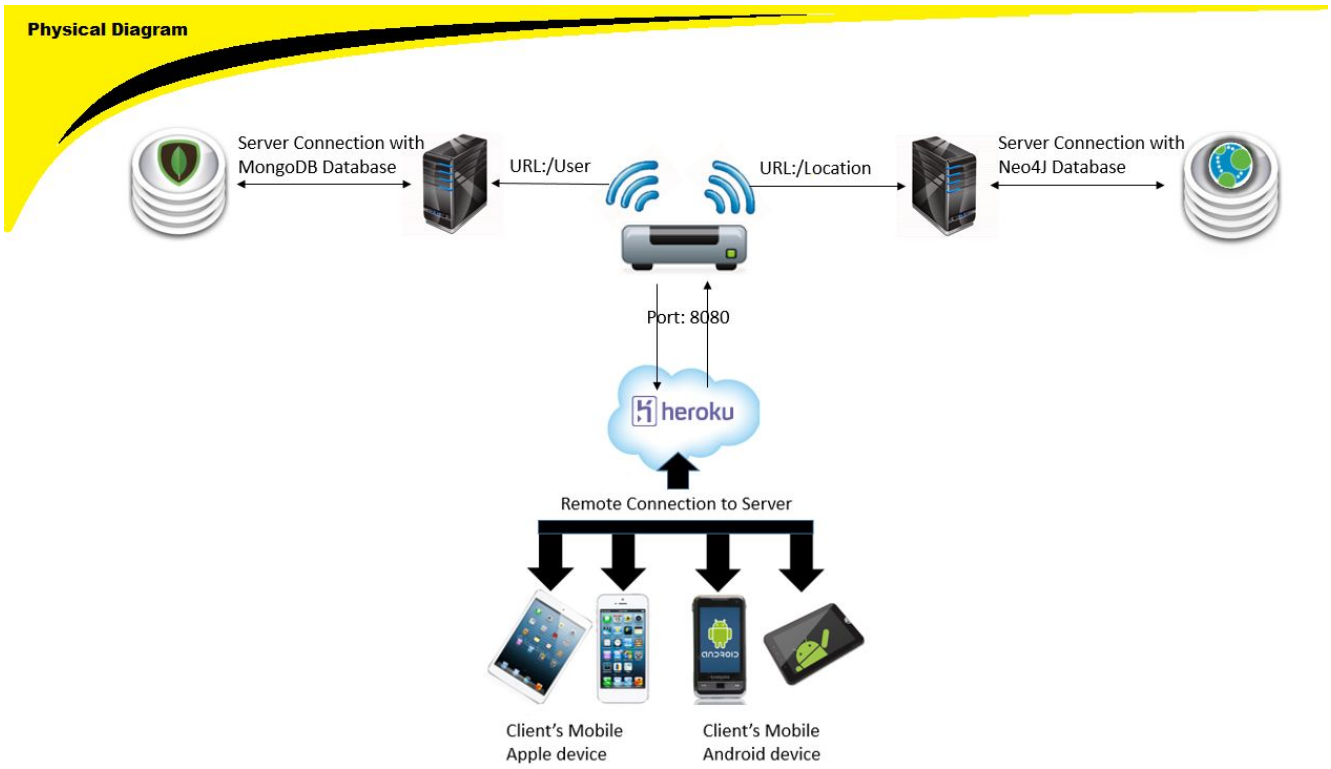


Figure 2.5 Physical Diagram

When the user begins the Hertz application, they will have to create an account with a username and login. Their inputted personal information will be converted into JSON format and then outputted to the remote server connection. The Heroku [6] server will receive this format as input, identifying it as an HTTP request with a specified application name. Within Heroku the RESTful services will determine how to handle HTTP requests based on the following functions;

Create (), Update (), Read (), and Delete (). Once the request is processed, the web services will use the provided database URI to connect to either, MongoDB [7] or Neo4J [8].

2.1.5 Software Interactions

The primary aim of the software is to provide the user with the ability to reserve a vehicle from any geographical location where Hertz offers rent-a-car services. Figure 2.6 is a Data Flow Diagram that describes this process and begins with the User's request for a reservation.

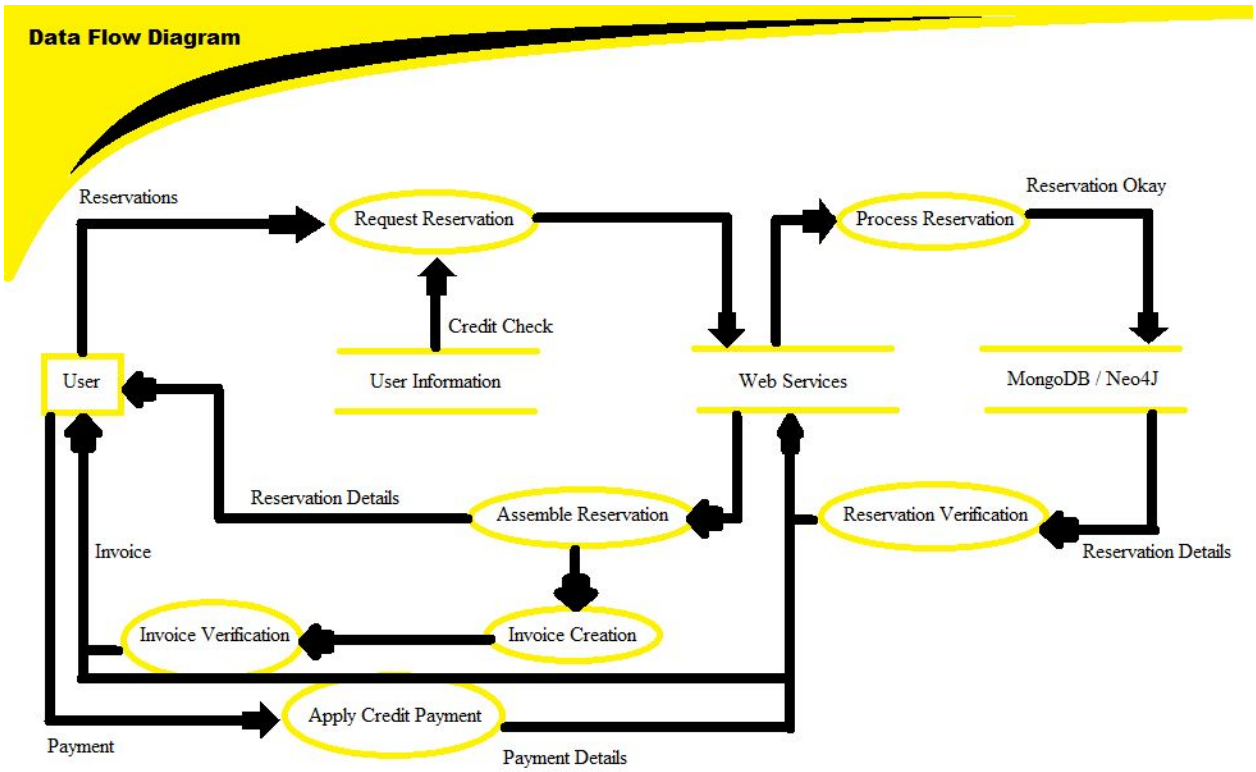


Figure 2.6 Data Flow Diagram

Upon doing so, the web services will process the reservation request, while exchanging information with the databases. Then, it will verify with Hertz the reservation order and customer details. The web services validates the returned information and generates a reservation response to the user. At the same time, an invoice is created and submitted to the user with the reservation response, followed by a payment method.

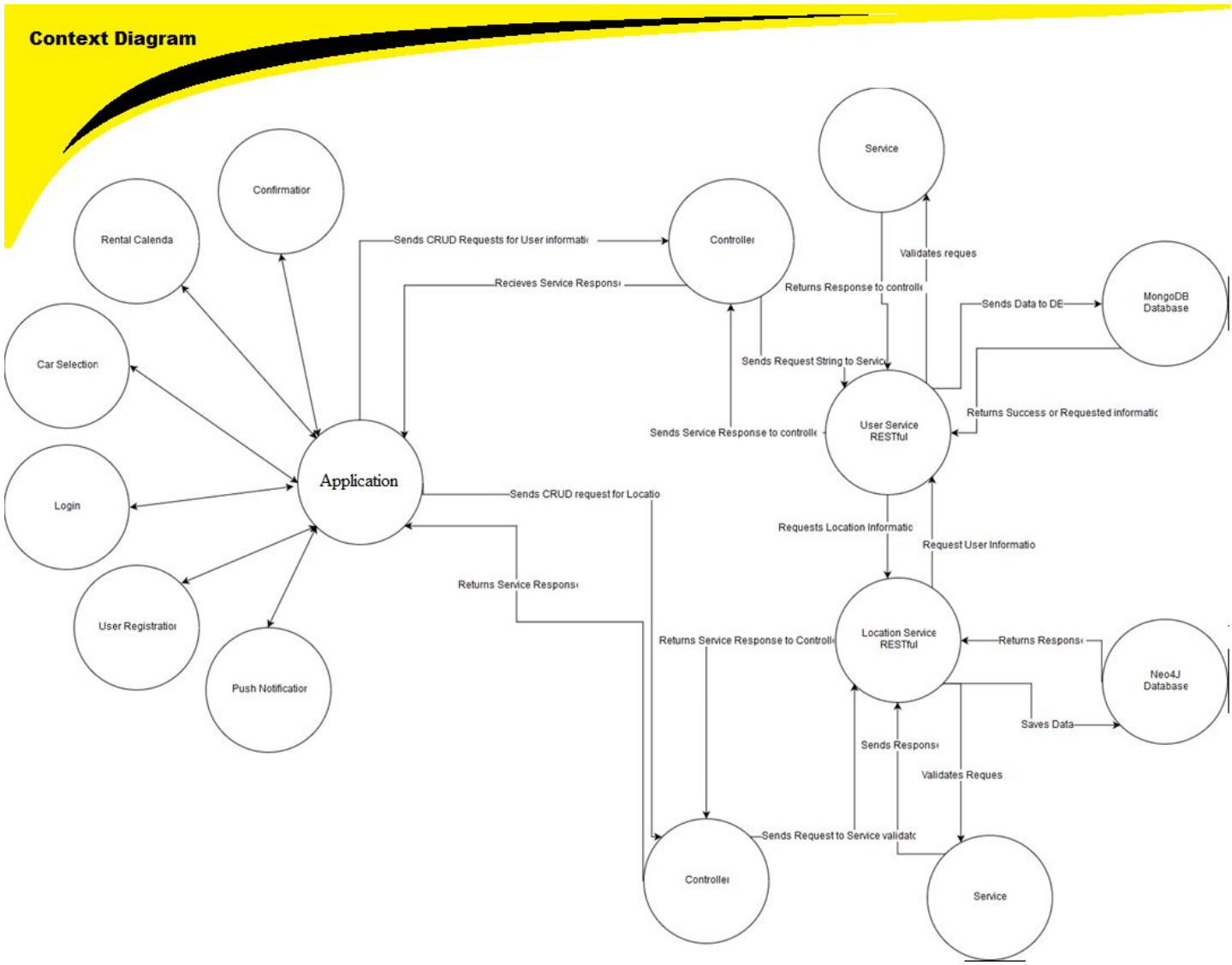


Figure 2.7 Context Diagram

Figure 2.7 represents the complexity of the system on a context diagram. This diagram describes the abstract system functionality, which can be partitioned into several parts, including; an application, user controller, location services, user services, mongoDB [6], and neo4j [7]. This

diagram illustrates how the user can register an account, login, select a car, select a rental date and receive both, confirmations and push notifications. These user interactions allow the communication with the mobile device, which sends the user commands, signals and responses to be processed within the RESTful services.

2.1.6 Front End vs. Back End Functions

The front-end design consists of the development of an aesthetic design, user-interface design and software architecture. The software architecture is concerned with the cohesive flow from page to page or from functions to function, by implementing high level software structures. This architecture is contingent on the user-interface design (Figure 2.1) because it is centered on aesthetic appeal to the user, as well as, user-friendly accessibility.

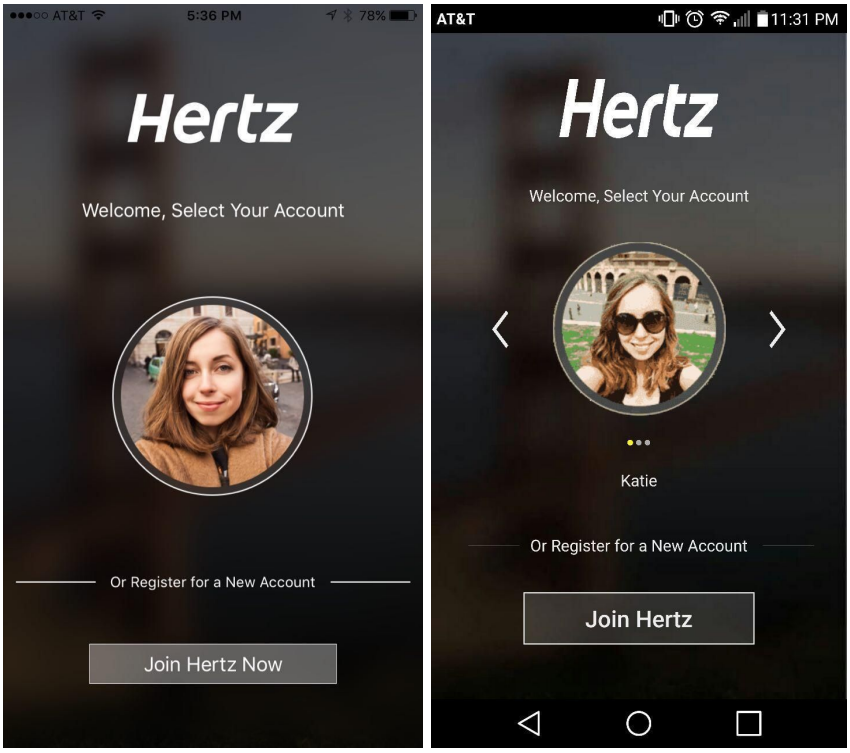


Figure 2.1 User Login

For example, reserving a vehicle should take a minimalist approach, by providing as much data as the software can for the user and yet, maintain a sleek user interface (Figure 2.2).

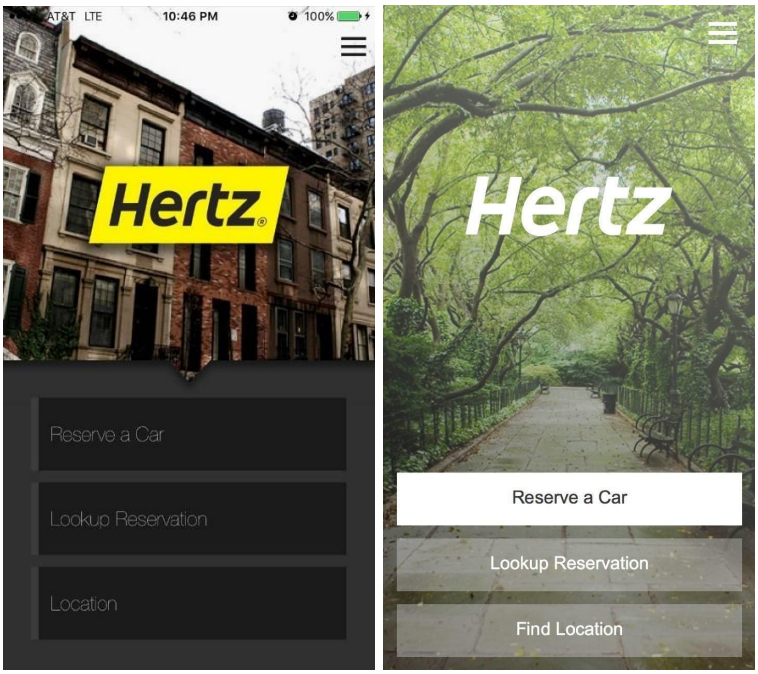


Figure 2.2 Hertz iOS Vehicle Checkout and Reservation Prototypes

The back-end development concerns software functionality, software architecture and data storage, based on non-functional requirements. The deployment diagram in Figure 2.3, illustrates this relationship.

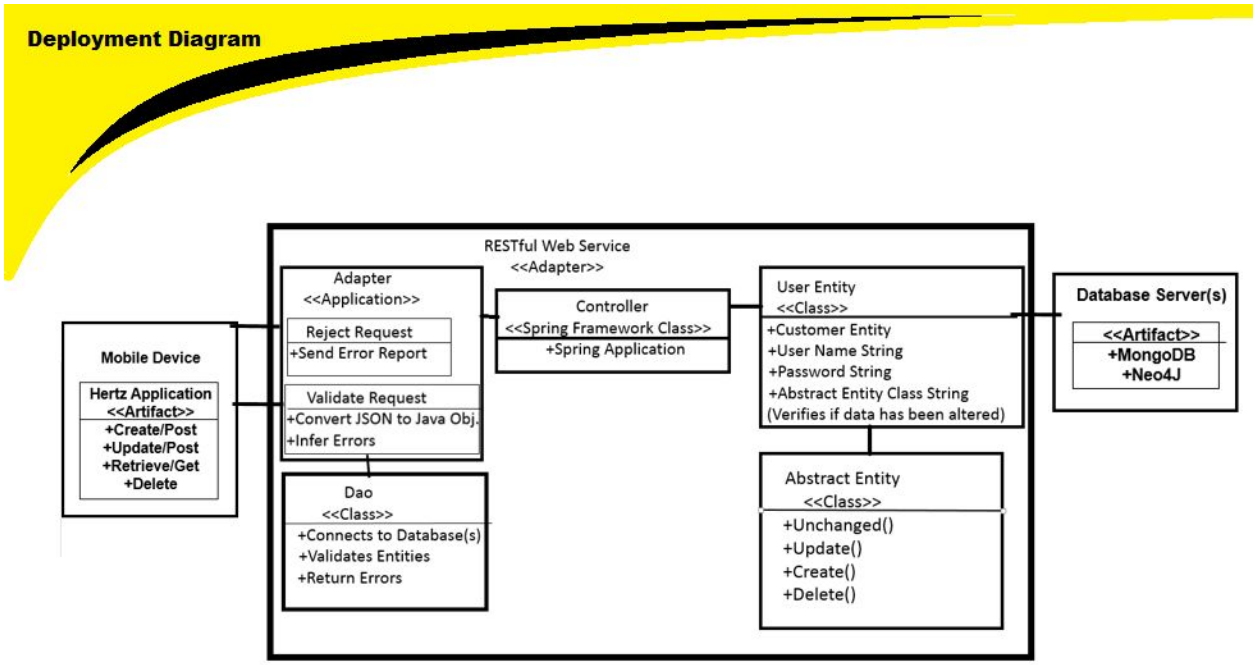


Figure 2.3 Back-End Deployment Diagram

These aspect require the client to provide, display and receive information from the user's mobile device. These interactions are implemented through the RESTful web services [4] and Spring Framework [5], allowing the URIs to access resources or data from MongoDB or Neo4J.

2.2 Requirements Specification

2.2.1 Input/output Requirements

2.2.1.1 The software shall save the user's name and password. (Hertz)

2.2.1.2 The software shall allow the user to select a reserve location for their vehicle reservation. (Hertz)

2.2.1.3 The software shall allow the user to select a return location for their vehicle reservation. (Hertz)

2.2.1.4 This software should output confirmation email verifications for

- 2.2.1.4.1 Vehicle reservation (Hertz)
- 2.2.1.4.2 Billing notifications (Hertz)

2.2.1.5 This software shall allow the client account to save

- 2.2.1.5.1 First Name (Hertz)
- 2.2.1.5.2 Last Name (Hertz)
- 2.2.1.5.3 Billing Information (Hertz)
- 2.2.1.5.4 Driver's License Information (Hertz)
- 2.2.1.5.5 Current Address (Hertz)

2.2.1.6 This software shall allow the client account to update

- 2.2.1.6.1 First Name (Hertz)
- 2.2.1.6.2 Last Name (Hertz)
- 2.2.1.6.3 Billing Information (Hertz)
- 2.2.1.6.4 Driver's License Information (Hertz)
- 2.2.1.6.5 Current Address (Hertz)

2.2.1.7 This software shall communicate with Neo4J database. (Hertz)

2.2.1.8 This software shall communicate with MongoDB database. (Hertz)

2.2.1.9 The software shall incorporate geolocation services into user interface. (Hertz)

2.2.2 Functional Requirements

2.2.2.1 The software shall have a location function to find Hertz Rent-A-Car Centers, from zip code input. (Hertz)

2.2.2.2 The software shall have a vehicle reservation function to

2.2.2.2.1 Select vehicle pick up time. (Hertz)

2.2.2.2.2 Designate vehicle return time. (Hertz)

2.2.2.2.3 Select vehicle pick up date. (Hertz)

2.2.2.2.4 Designate vehicle return date. (Hertz)

2.2.2.3 This software shall have a vehicle selection function to

2.2.2.3.1 Select vehicle based on type. (Hertz)

2.2.2.3.2 Designate vehicle based on model. (Hertz)

2.2.3 Non-Functional Requirements

2.2.3.1 This software shall have a response time of 3 seconds maximum when searching for a Hertz Location. (Hertz)

2.2.3.2 This software shall have no more than 10 screens or pages for all functionality. (Hertz)

2.2.3.3 This software shall notify the user if there is a loss of connection to the network, server or database. (Hertz)

2.2.3.4 The web services shall provide a layer of security between the network and database.

2.2.3.5 The web services shall have an object-oriented design to easily integrate

2.2.3.5.1 Software functionality. (Hertz)

2.2.3.5.2 Decrease software response time. (Hertz)

2.2.3.5.3 Decrease micro services run time. (Hertz)

2.2.4 Design Constraints

2.2.4.1 This software design requires an established network connection. (Hertz)

2.2.4.2 The iOS must enable global positioning for geolocation services. (Hertz)

2.2.4.2 The database shall receive Java Objects from RESTful Web. (Hertz)

2.2.4.3 The database shall send Java Objects to RESTful Web. (Hertz)

2.2.4.4 The RESTful Web service shall

2.2.4.4.1 Convert client GET requests via HTTP. (Hertz)

2.2.4.4.2 Convert JSON objects to Java objects. (Hertz)

2.2.4.4.3 Use the DAO (Data Access Object) to access MongoDB. (Hertz)

2.2.4.4.4 Send JSON-based response to the client application. (Hertz)

2.2.4.4.5 Validate customer's information before being sent to MongoDB. (Hertz)

2.2.4.4.6 Send push notifications to the client. (Hertz)

2.2.4.5 The user interfaces of the iOS should resize with equivalent proportions. (Hertz)

2.2.4.6 The Client shall receive push notifications from RESTful Web Services. (Hertz)

3. Software Design Description

3.1 Introduction

3.1.1 Scope

This section describes the software design and establishes the information content and organization of the Hertz Mobile Application. This detailed design is a representation of the Hertz Mobile Application software to be used for recording design information and software construction activities. These activities include, when design descriptions lead to code and when code leads to a design description.

3.1.2 Purpose

This design description specifies requirements on the module design, module organization and module interactions within the software.

3.1.3 Design Overview

The Hertz Mobile Application design description is divided into three sections; Software Architecture, Detailed Design and Human-Machine Interface Design. These sections emphasize the primary modules and the interactions between these modules.

3.1.4 Conformance

The Hertz Mobile Application Design Description conforms to the IEEE Standard 1016-2009 [] by satisfying the requirements specified in Section 2.2 ‘Requirements Specification’. Requirements are denoted by the verb shall.

3.2 Software Architecture

3.2.1 Software Architecture Introduction

The Hertz Mobile Application architecture establishes a conceptual model for the software design documentation. This conceptual model features a top-down design which includes, basic terms, modules and interactions.

3.2.2 Overview

The software architecture is divided into four modules including the, human-machine interface, Heroku cloud platform, MongoDB and Neo4J. The realization of the relationships between these modules is expressed in Figure 3.1

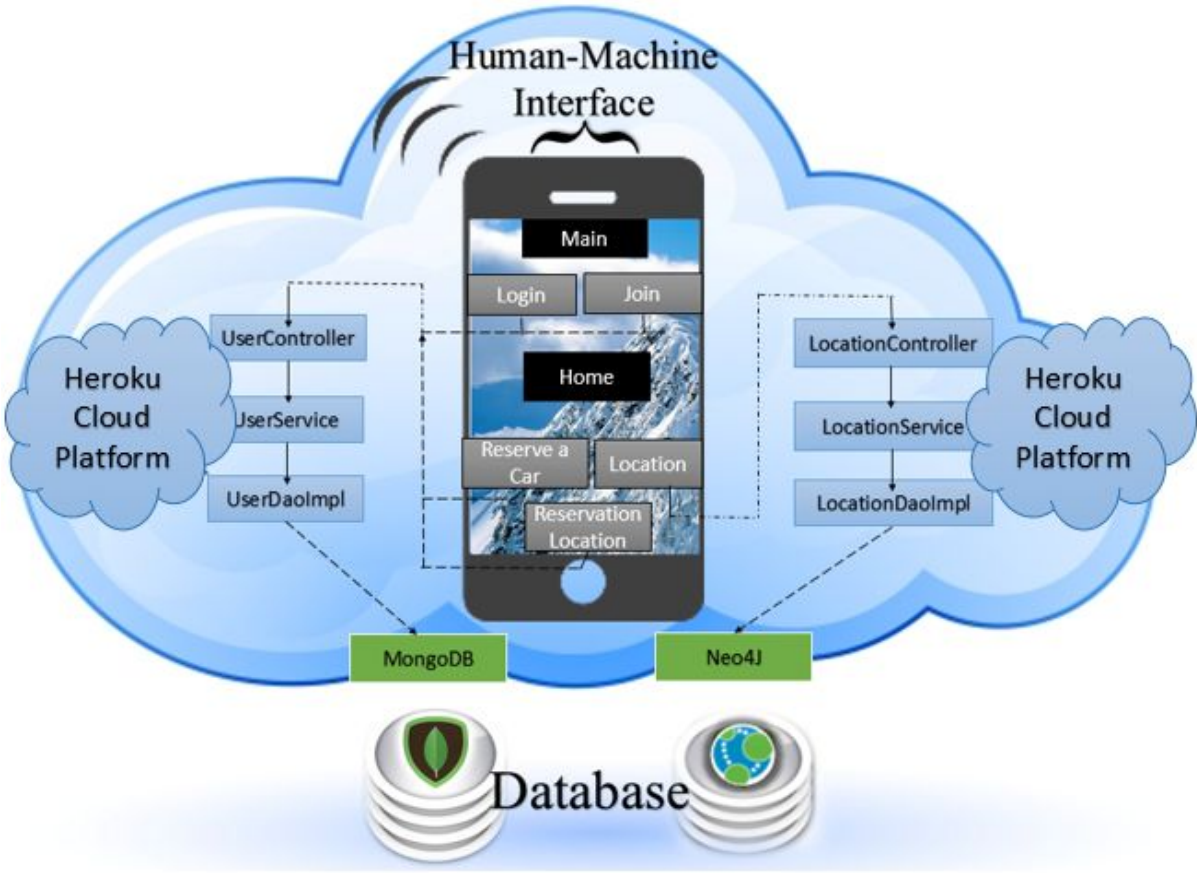


Figure 3.1 Structure Chart

The human-machine interface provides the user with the following functions; Login, Join, Reserve a Car, Reservation Location and Location. All five functions are programmed as

buttons on the user screen, but exist on two different interfaces. The first interface, Main, offers the login and join buttons. Once one of these buttons is selected and carried out, the user will be brought to the Home interface. This interface, provides the user with three services, reserve a car, reservation location and Find Location.

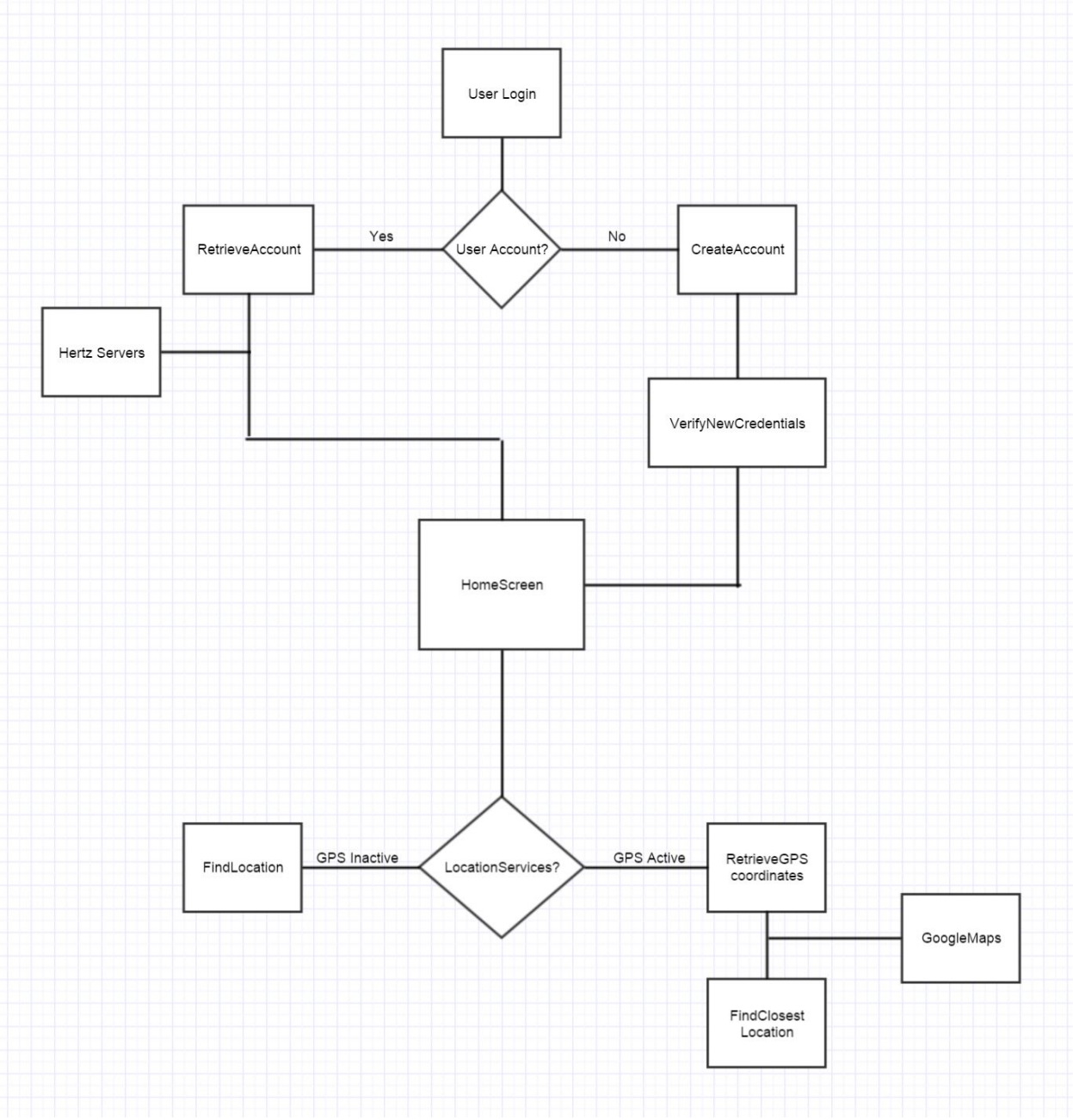
The Heroku cloud platform manages the function requests based on the user selection. This platform offers two different microservices. The first service consists of the UserController, User Service, and UserDaoImpl. These services involve the sending and receiving of user related account information, billing information and account preferences, which are stored in MongoDB.

The second services are the LocationController, LocationServices and LocationDaoImpl. This set of services regard the sending and receiving of geolocation signals. These signals interact with the Neo4J database for increased network security.

3.3 Detailed Design

3.3.1 Dynamic Perspective

Figure 3.2 illustrates the software behavior from when the Hertz Mobile Application launches to the time of completion.



3.2 Software Behavior Flowchart

Once the application has launched, the server software will validate the connection and the user interface will proceed to the Main screen based on Heroku's HTTP Post-Request in URL format. This interface offers the sign in and join commands that Heroku validates from the URL. If these commands return validated, the user is directed to the Home interface. This interface offers three services, Reserve a Car, Reservation Location and Find Location. The descriptions of these services are found in Section 3.4 Human-Interface Design.

3.3.2 Static Perspective

The client application sends REST requests via HTTP to the RESTful Web service. The Web service receives GET requests, which are JSON strings, and the REST Controller converts the strings into Java objects. The controller converts JSON into POJOs by implementing a method, which maps each JSON object to the specific Java object. Depending on the client's requests, the Web service can create a customer profile, update the customer's information, retrieve customer's data upon request, or delete a customer profile by accessing a NoSQL database, called MongoDB, from the DAO. The Web service is able to access the database from MongoDB methods and libraries. Figure 3.3 illustrates these class interactions as a Class Diagram.

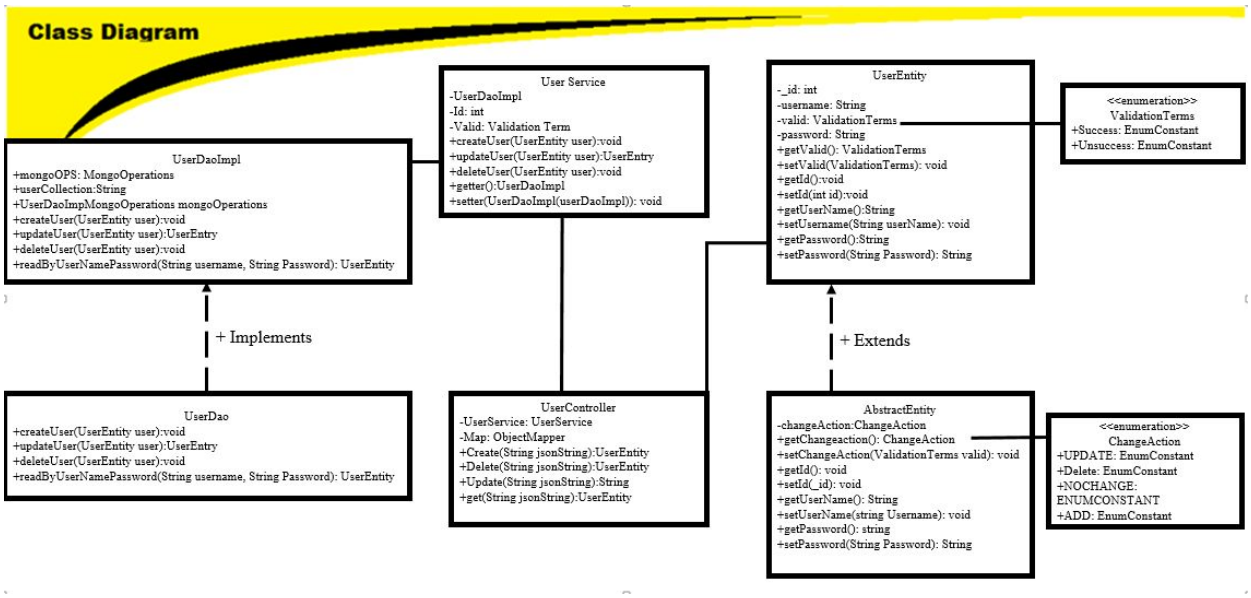


Figure 3.3 Class Diagram

3.4 Human-Machine Interface Design

3.4.1 Human-Machine Interface Design Introduction

The objectives of the Human Interface Design Document consists of being the primary document driving the implementation of the Hertz iOS mobile application human interface and provide the traceability of the design elements from the architecture documentation. Figure 3.4 offers a Flow Chart to describe the interfaces and their relationships.

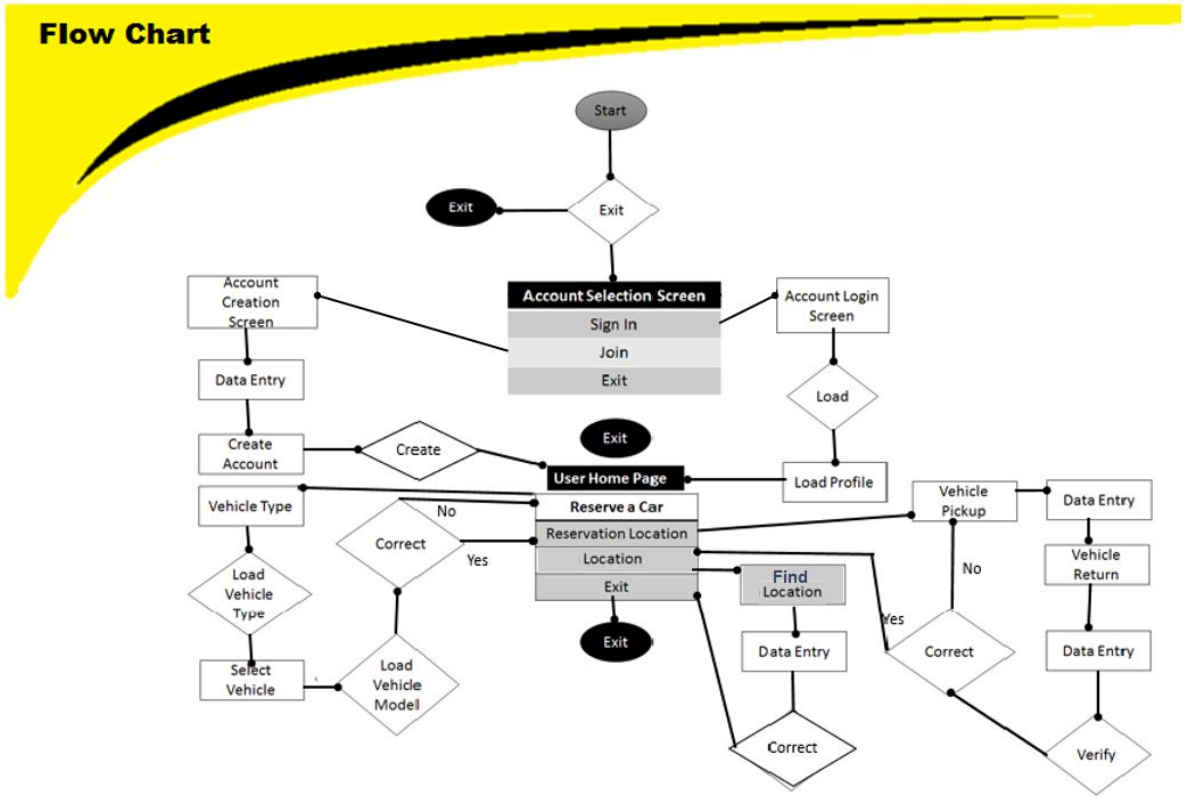


Figure 3.4 Interface Flow Chart

3.4.3 Application Screens

3.4.3.1 Account Selection

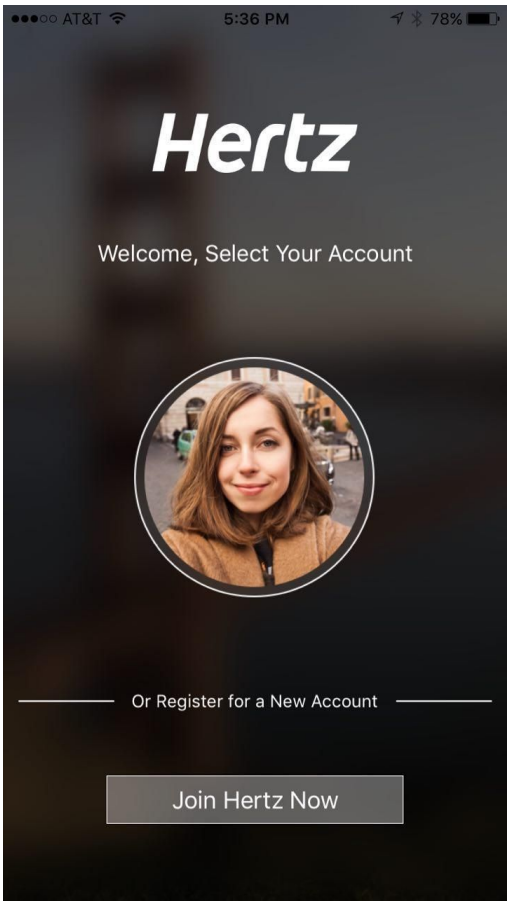


Figure 3.5 Mobile Application Screen – “Account Selection”

The join command proceeds to an account creation screen (Figure 3.5), where the user will input their personal information and billing information. These inputs will be converted to a JSON string object, relayed as an HTTP request to the web services and stored from the web services to MongoDB. If these requests are determined valid within the web services, then the user will be redirected to the Home screen.

3.4.3.2 Main

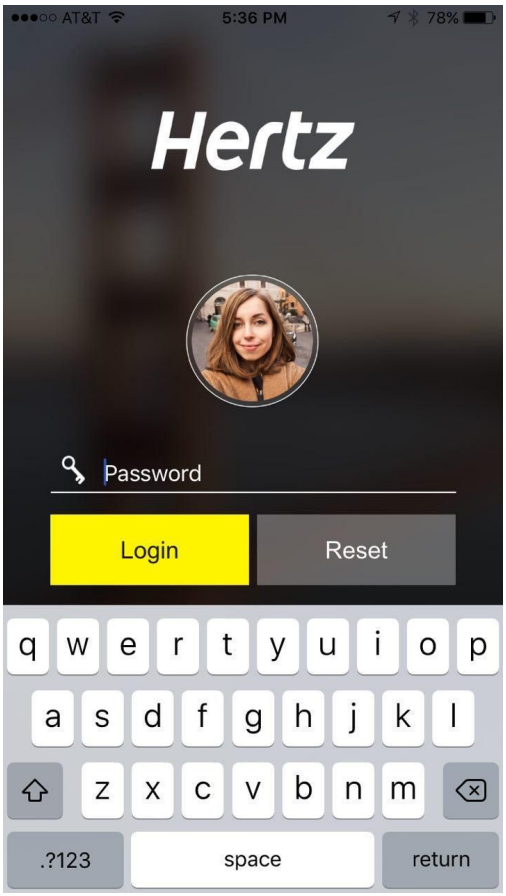


Figure 3.6 Home Screen

The sign in command reads the username and password as string variables within the mobile application. Both variables are converted to a JSON string object that is sent from an HTTP request to the web services. If the account string variables are validated by these services, then the interface will load the Home screen as shown in Figure 3.6

3.4.3.3 Home

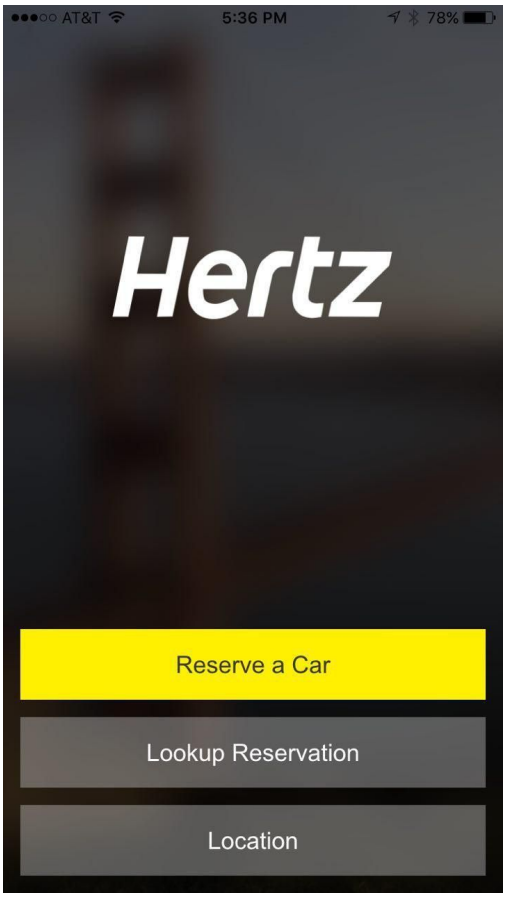


Figure 3.7 Home Screen Interface

The Home Screen Interface shown in Figure 3.7, offers users three options; Reserve a Car, Lookup Reservation, Location.

3.4.3.4 Car Reservation

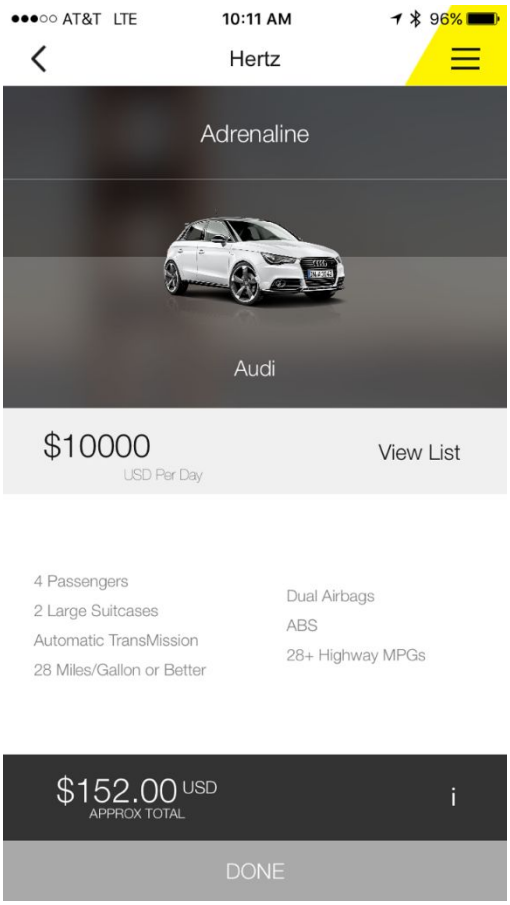


Figure 3.8 Reserve a Car Interface

The Reserve a Car (Figure 3.8) service redirects the user to several screens that provide the user with a choice of vehicle selection. First the vehicle type will be determined by the user as either luxury, sport, or convertible from a list menu. The user can then confirm the selected type to be directed to the following interface Vehicle Selection. This interface will offer several vehicles based on the vehicle types and are stored in MongoDB. In MongoDB the vehicle inventory is managed from the admin side. The user will can then confirm their vehicle selection and then, redirected to the Home screen.

3.4.3.4 Select Date

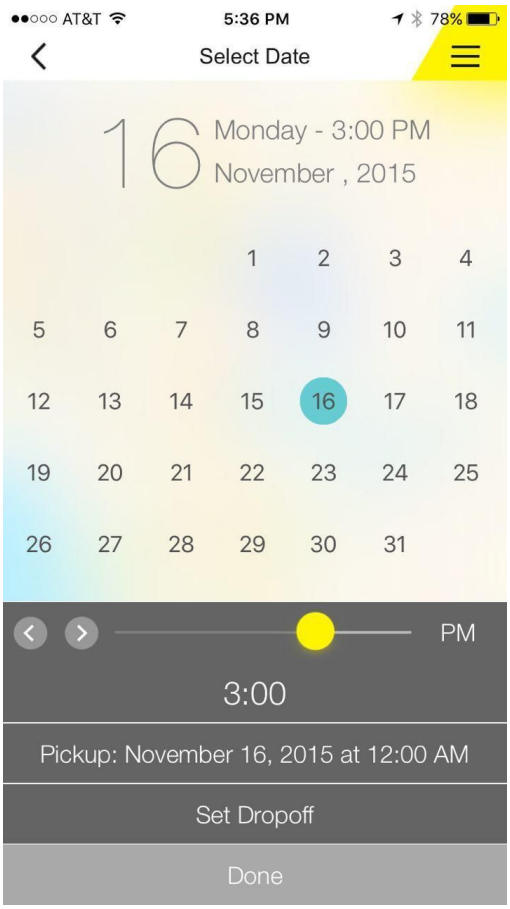


Figure 3.9 Reservation Location Interface

The Reservation Location (Figure 3.9) service is only accessible after the user has been through the Reserve a Car service. The Reservation Location service provides the user with a calendar view that is touch screen accessible to select the day and time of vehicle pickup. The user will then have to confirm these days to proceed to the vehicle return interface. The vehicle return interface operates the same as vehicle return, redirected the user to the Home interface after confirmation.

3.4.3.4 Find Location

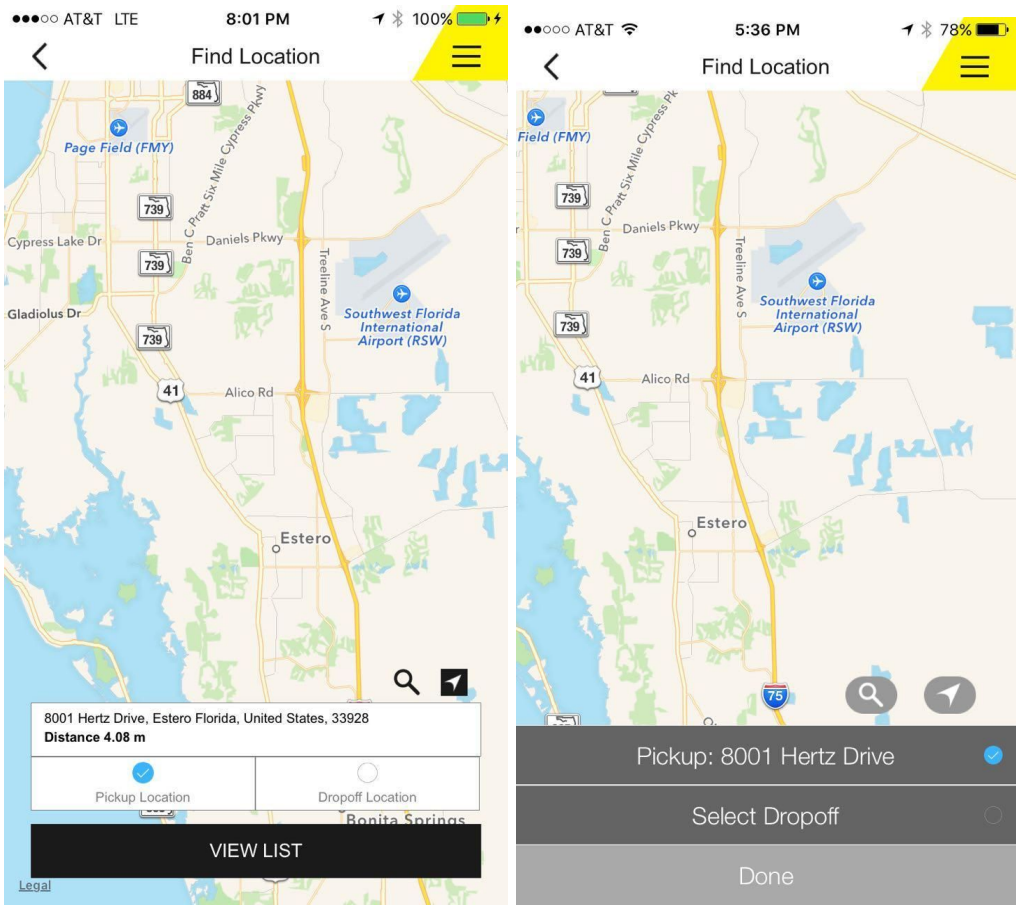


Figure 3.91 Find Location Interface

The Find Location service is only accessible after the Reserve a Car and Reservation Location service has been completed. This service will generate a list of Hertz locations for vehicle pickup based on the geographical location of the mobile phone. This service is explained further in detail in Section

4. References

[1] Android. (2015). *Android 6.0 Marshmallow*.
URL: http://www.android.com/ready-for-you/

[2] iOS. (2015). *Apple iOS 9*.
URL: http://www.apple.com/ios/

[3] Hertz. (2015). *The Hertz Corporation*.
URL: https://www.hertz.com/rentacar/reservation/

[4] Spring MVC Framework. (2016). *Spring*.
URL: http://spring.io/

[5] Walls, Craig. *Spring in Action Fourth Edition*. New York: Manning, 2015. Print

[6] Heroku. (2015). *Cloud Application Platform*.
URL: https://www.heroku.com/

[7] MongoDB. (2015). *Download MongoDB*.
URL: https://www.mongodb.org/

[8] Neo4J. (2015). *Neo4j*.
URL: http://neo4j.com/