

RENT A HOME – A CROSS PLATFORM MOBILE APPLICATION TO LIST AND SEARCH
RENTAL HOMES

by

YUSUF ALI

B.E., Rajiv Gandhi Proudyogiki Vishwavidyalaya, India, 2012

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2016

Approved by:

Major Professor
Daniel Andresen

Abstract

Finding a desired off-campus home could be a difficult task for incoming students at K-State. This cross platform mobile application can help students in this task. This mobile application will serve as platform for students and home owners to find, list and review a property. Users can create their personal account and manage their profile and properties they listed. They can post multiple property listings with the key details and these listings will be visible to all the other users.

To avoid spam and repetitive postings there is an address verification system in place. While posting a property listing the address is verified with Google maps database. If Google maps fail to identify the address or if that address is already listed, the user will not be allowed to list that property.

Users can see all the property listings and their details including a Google map location image, a link to navigate to the address, contact details of the owner, reviews of other users. Users can rate a property from 1(low) to 5(high) along with an optional review. The review and rating can help other user to select a desired place to live.

The user interface is developed with web technologies and using Cordova the same code can be used to convert it into a mobile application for both iOS and Android which makes this application platform independent and easily available.

Table of Contents

List of Figures	v
List of Tables	vi
Acknowledgements	vii
Chapter 1 - Introduction	1
1.1 Motivation	1
1.2 Project Description	1
Chapter 2 - Related Work	2
2.1 Craigslist	2
2.2 Zillow	2
Chapter 3 - Requirement Analysis	3
3.1 Functional Requirements	3
3.2 Non-Functional Requirements	4
Chapter 4 - Tools and Technologies Used	5
4.1 Apache Cordova	5
4.2 ionic Creator	7
4.3 PhoneGap Developer app	7
4.4 S3 by Amazon Web Services	7
4.5 Spring Boot JPA	7
4.6 Google Maps API	8
Chapter 5 - System Architecture and Design	9
5.1 System Architecture	9
5.1.1 Mobile App	9
5.1.2 Application Server	10
5.2 System Design	11
5.2.1 ER Diagram	11
5.2.2 Use case diagram	12
5.2.3 Class Diagram	13
Chapter 6 - Implementation	14
6.1 Application GUI and working	15

6.1.1 Login	16
6.1.2 Create account.....	17
6.1.3 Search.....	18
6.1.4 Home	19
6.1.5 Property (housing) details	20
6.1.6 Add Review.....	21
6.1.7 Side Menu	22
6.1.8 Add a property	24
6.1.9 Address verification and duplicate address detection	26
6.1.10 Manage listings	29
6.1.11 View and edit personal details	32
Chapter 7 - Testing.....	33
4.1 Unit Testing.....	33
4.2 Integration Testing	35
4.3 Compatibility Testing	36
Chapter 8 - Conclusion and Future Scope	38
References	39

List of Figures

Figure 4.1 Cordova Application Architecture	6
Figure 5.1 System Architecture	9
Figure 5.2 ER Diagram.....	11
Figure 5.3 Use Case Diagram	12
Figure 5.4 Class Diagram	13
Figure 6.1 Login page	16
Figure 6.2 Create account page.....	17
Figure 6.3 Search screen.....	18
Figure 6.4 Home page	19
Figure 6.5 Property details page	20
Figure 6.6 Add review	21
Figure 6.7 Side menu (not logged in)	22
Figure 6.8 Side menu (logged in)	23
Figure 6.9 Add property form.....	24
Figure 6.10 Upload picture	25
Figure 6.11 Address verification.....	26
Figure 6.12 Error in address verification	27
Figure 6.13 Duplicate address in the system	28
Figure 6.14 Manage listings page	29
Figure 6.15 Edit property page	30
Figure 6.16 Edit and delete property.....	31
Figure 6.17 Edit personal details	32

List of Tables

Table 6.1 Lines of Code breakup	15
Table 7.1 Unit Test Cases	35
Table 7.2 Integration Test Cases.....	36
Table 7.3 Compatibility Test	37

Acknowledgements

First of all I would like to thank my major professor Dr. Daniel Andresen for his ideas, guidance and support throughout the project. I would also like to thank my committee members Dr. Torben Amtoft and Dr. Doina Caragea for serving in my committee and for the knowledge I gained while taking courses under them. I would like to thank the Department of Computing and Information Sciences for giving me the opportunity to pursue my graduate degree here.

Finally, I would like to thank my family, specially my brother Juzer Ali for constantly believing in me and supporting my decisions.

Chapter 1 - Introduction

1.1 Motivation

Searching for a desired home for rent could be a difficult task for a student, especially when they are new in town or planning to go to a new place. Two years ago I faced the same challenge when I was looking for a rental home in Manhattan KS. I was searching on Craigslist all the available housing but couldn't find some important information like experiences of the people who already lived there. To find that I had to contact many people already living in the city and my seniors. This took a lot of time and effort. A platform which allow landlords to list their housing and allow users to view, rate and review the housings would be a great help for the new students.

1.2 Project Description

Rent a home is a cross platform mobile application which can be used by students to search rental housing. Landlords who wish to rent their housing units can fill a form with all the required details of the house like availability, rent, address, number of rooms. Landlords can edit the details of their listings anytime they want. The listings will be visible to all the other users and will show up on their home pages. Users can also search housings by specifying search criteria which will only show the results matching the criteria. Users can view all the details of the housing and they can also post their review with a rating. Users can also view the reviews given by others and average rating of the housing.

Users can take advantage of the review feature to post their experiences about the housing. They can give a positive or negative feedback which will help new users to get more information about the housing which landlords might not provide. Apart from details provided by landlords, the experiences and feedbacks of previous tenants can prove to be the deciding factor for new students to select a suitable housing.

The mobile application using HTTP protocols interacts with a remote RESTful Web Service to fetch and store all the data. This server is an important part of this project because it is linked to a centralized MySQL database where all the data is stored and the server is responsible to complete all the requests made by the client. This will be discussed in detail in system architecture section.

Chapter 2 - Related Work

There are many applications that provide service to search for rental homes. Two of them are discussed in this section.

2.1 Craigslist

This app allows landlords to post their listings which can be seen by all the users. Users can search, view the housing details and get contact details of the owner. Craigslist is available on various platforms like web and mobile. It is popular and has large user base. However it does not allow users to rate and review housing.

2.2 Zillow

This application allows agents to post their listings which can be seen by all the users. This is a very promising application with a very good user interface and it is also available on web and mobile platforms. User can view all the available housings on a map as well as in a list form. This app also has features like sharing your listing on other platforms, rent comparison tool, neighborhood details, and recommendations. However this app also doesn't allow users to rate and review any housing.

There are more applications which provide similar services and on reviewing them I found out that none of them has the feature to rate and give feedback so that other users can benefit from them. The goal of this project is not to compete with these already existing applications, but to provide a free service having these missing important features to the users.

Chapter 3 - Requirement Analysis

The requirement for this project came up while discussing different project ideas with my major professor Dr. Daniel Andresen. While we had many ideas in mind Dr. Dan suggested working on this project since it may have a practical use in future for Off Campus Housing Support by K-State Student Legal Services. After agreeing upon the project and discussing about the higher level requirements the next part was to list down the basic features required in the application. These basic features made up the functional requirements of the application. To get these requirements I explored different applications in the same domain. This helped me eliciting what common features are required and how the structure of the application should look like.

The requirements were classified into two categories: functional and non-functional requirements.

3.1 Functional Requirements

Requirements, which are related to functional aspect of application fall into this category. They define functions and functionality within and from the software system. The major functional requirements of the application are:

- Users can view the list of all the rental houses.
- Searching: Users can filter the list by specifying search criteria.
- Users can select a result from the list and get more details about it like photograph, location on map, owner details, rating, reviews, etc.
- Users can provide their rating and feedback (they must be logged in first).
- Create account and login feature.
- Users can post a rental house listing by filling up a form and uploading a picture.
- Users can view the list of all the housings they have posted.
- Users can select and edit their listings.
- Before posting a listing address should be verified against Google Maps database so that no housing can be listed twice.

3.2 Non-Functional Requirements

Requirements, which are not related to functional aspect of software, fall into this category. They are implicit or expected characteristics of software, which users make assumption of. The major non-functional requirements of the application are:

- Flexibility: The server should be independent of the client technology such that it can serve different variety of clients on different platforms.
- Accessibility: the application should accessible anytime and anywhere (assuming there is proper internet connection)
- Cost: keep the development and operational cost minimum by using open source tools and software.
- Security: communication between clients and server should be secure.
- User friendly: the application should be easy to operate.
- Responsive user interface: the application view should be compatible with different devices and should adjust itself to different screen sizes and orientations.

Clear understanding of requirements helps in selecting the tools and technologies which should be used to get the desired product. Next chapter talks about the important technologies used to develop and run this application.

Chapter 4 - Tools and Technologies Used

This mobile application is developed using apache Cordova framework. To create clean project structure and the basic user interface template of the app, ionic creator was used. Code generated by ionic creator follows AngularJS framework, so rest of the development was done using JavaScript, CSS and HTML5 using AngularJS framework. To view changes quickly on the mobile phone Adobe PhoneGap desktop and develop apps were used.

The back or server of the application is a RESTful web service developed using Spring Boot framework which interacts with the MySQL database using Java Persistence API. All the data generated and consumed by the mobile application is stored in MySQL database, only the images are stored separately on AWS S3 cloud for faster storage and retrieval.

For address verification and checking duplicate address, Google Maps' Geocoding API is used. To display the location on housing details page, Google Maps' static API is used.

To run and debug the JavaScript and HTML5 code Google chrome browser was used with Google chrome developer tools. We shall now discuss these technologies to get a better understanding of how each component of the project works.

4.1 Apache Cordova

Apache Cordova is a set of device APIs that allow a mobile app developer to access native device function such as the camera or accelerometer from JavaScript. Using Apache Cordova one can build a complete mobile application using only JavaScript, HTML and CSS. Cordova uses Android SDK and JDK to generate the platform specific file (.apk for android and .ipa for ios). Technically the User Interface of a Cordova Application is effectively a Web View that occupies the complete screen and runs in the native Container. So, it is the same web view that is used by the Native Operating systems. This purely means that only the Native Containers changes according to the OS and internally the web pages remain the same. Below are some important terms of Cordova architecture and the high level view of Cordova application architecture.

Web View - The Cordova-enabled WebView may provide the application with its entire user interface. On some platforms, it can also be a component within a larger, hybrid application that mixes the WebView with native application components.

Web App - This is the part where the application code resides. The application itself is implemented as a web page, by default a local file named index.html, that references CSS, JavaScript, images, media files, or other resources are necessary for it to run. The app executes in a WebView within the native application wrapper, which we distribute to app stores.

Plugins - Plugins are an integral part of the Cordova ecosystem. They provide an interface for Cordova and native components to communicate with each other and bindings to standard device APIs. This enables us to invoke native code from JavaScript.

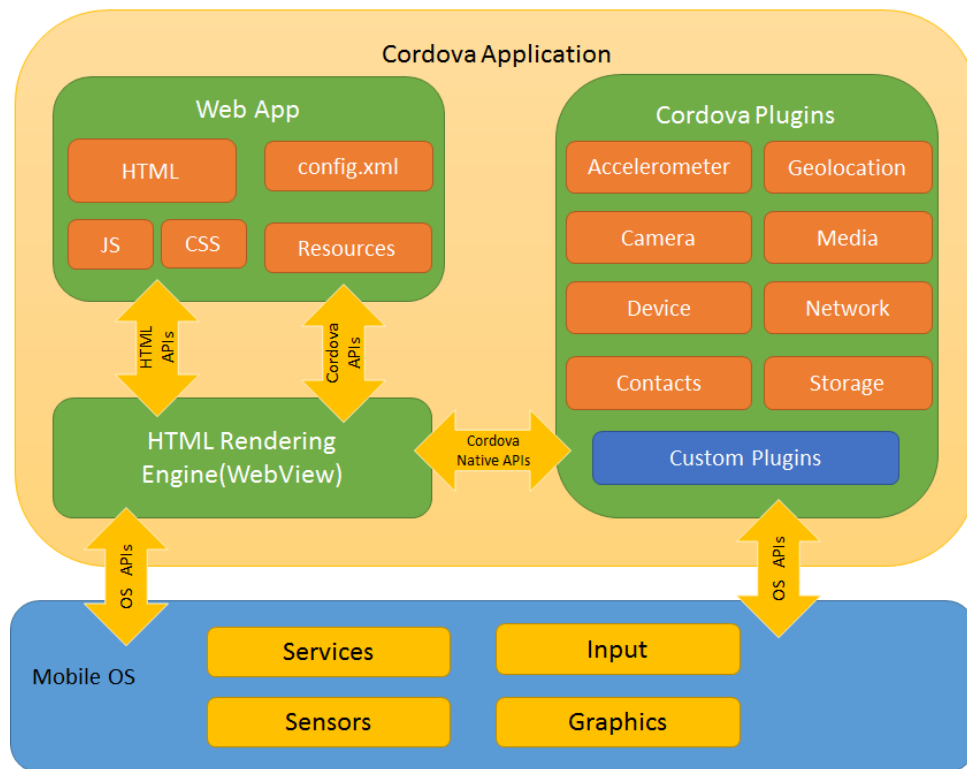


Figure 4.1 Cordova Application Architecture

For creating the native build file, SDK of that platform should be installed in the development environment. To create apk Android SDK version 6.0 and JDK 8 was used. To store app local user data, localStorage provided in Cordova's Storage API is used. This API is based on the W3C Web SQL Database Specification and W3C Web Storage API Specification. localStorage provides an interface to a W3C Storage interface. It allows one to save data as key-value pairs. Cordova version 6.0.0 is used to develop this application.

4.2 ionic Creator

ionic creator is an online tool which provides drag and drop feature to create user interface for mobile applications. ionic provides a rich user interface and many ready to use components which can be used to develop a rich, responsive UI. Using this tool only side menu and page layouts were created which gave a basic design to the application and a good project structure. Rest of the development was done in Web Storm.

Using ionic command line interface tools the web pages developed can be served on localhost and viewed in Google Chrome browser. This gives us access to Chrome developer tools for debugging JavaScript and HTML code.

4.3 PhoneGap Developer app

After making any changes in the application if we want to test it in the phone we have to create a built and push it in the device. It is a time consuming process and also running the app doesn't show any console messages for debugging purposes. PhoneGap developer app installed in the mobile device solves this issue. The changes are instantly seen in the PhoneGap web view and console messages can be seen in the PhoneGap desktop app. This makes development process easy and less time consuming.

4.4 S3 by Amazon Web Services

This service is used to store picture related to the housing details. The picture is directly sent from mobile app to S3 and also retrieved directly from S3 to mobile app. This is done for faster and efficient storage and retrieval of image files as compared to relational database. The database on server only stores link of the image. For security, S3 provides an access ID and a secret key which should be sent with each request, otherwise the request is rejected.

4.5 Spring Boot JPA

The server code is written in Java using Spring Boot framework. Spring Boot is designed to simplify the bootstrapping and development of a new Spring application. The framework takes an opinionated approach to configuration, freeing developers from the need to define boilerplate configuration.

JPA allows us to create persistence entities which are lightweight Java classes whose state is typically persisted to a table in a relational database. Instances of such an entity correspond to individual rows in the table. Entities typically have relationships with other entities, and these relationships are expressed through object/relational metadata. Object/relational metadata can be specified directly in the entity class file by using annotations, freeing us from the need to create tables in database.

4.6 Google Maps API

Google Maps provide wide range of APIs which a developer can use to integrate various services provided Google Maps. In this project two APIs are used:

Google Maps Geocoding API: Geocoding is the process of converting addresses into geographic coordinates, which we can use to place markers on a map, or position the map. Along with geographic coordinates the API also send the complete corrected address which can be used to verify the address provided by user and unique place_id which can used to check the duplicate entry of the address in the system. The API can be access over standard HTTP protocol.

Google Static Maps API: The Google Static Maps API lets us embed a Google Maps image on our web page without requiring JavaScript or any dynamic page loading. The Google Static Maps API service creates the map based on URL parameters sent through a standard HTTP request and returns the map as an image you can display on your web page. The main URL parameters are geographic coordinates.

Chapter 5 - System Architecture and Design

5.1 System Architecture

Below is the system architecture of this application.

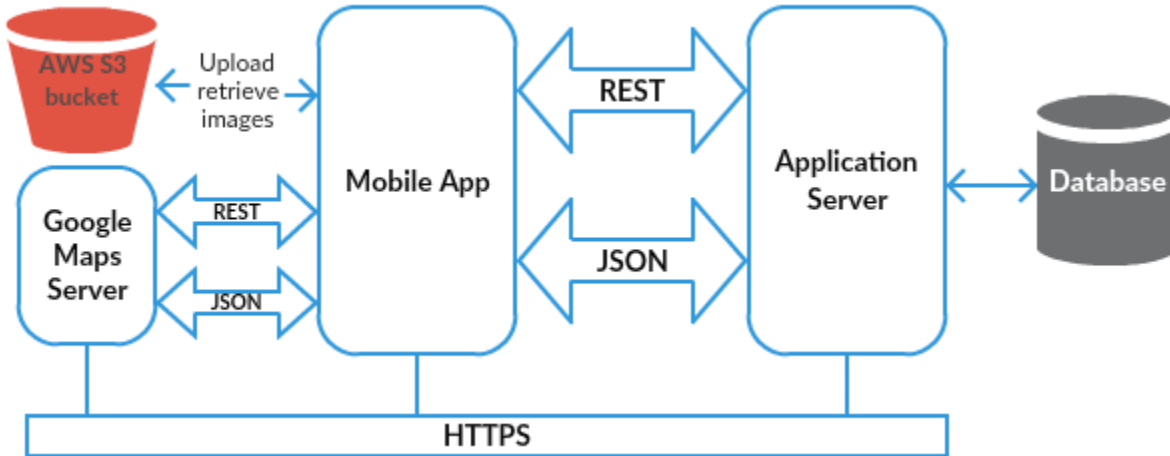


Figure 5.1 System Architecture

The above figure describes the system architecture of the application and also gives an idea how different components interact with each other. Let's discuss about the two main components Mobile App and Application Server.

5.1.1 Mobile App

The component with which user interacts is Mobile App (for detailed architecture see Section 4.1). In general a user can perform four actions: Create data, Delete data, Update data and Get data. To perform these actions Mobile App uses four HTTP methods: POST, DELETE, PUT and GET respectively. Using these methods the App sends HTTP requests to Application Server on the respective REST end points (or URIs) and server responds with appropriate data. All the data exchange between Mobile App and Server is in JSON format. The communication takes place over HTTPS so that the data transfer is secure. Also in each request sent by the Mobile App a security key is included in the request header which tells server that the request is coming from a correct source. If the key is missing in any request, the server will reject the request with HTTP 403 error (meaning that client is forbidden from sending requests to the server)

We can see that apart from Application Server, App also interacts with AWS S3 bucket and Google Maps Server for storing images and for address verification purpose respectively.

5.1.2 Application Server

Application Server is a RESTful Web Service. REST stands for Representational State Transfer. In REST architectural style, data and functionality are considered resources and are accessed using Uniform Resource Identifiers (URIs), typically links on the Web (or REST end points). The resources are acted upon by using a set of simple, well-defined operations that are POST, DELETE, PUT, and GET. The REST architectural style constrains architecture to client/server architecture and is designed to use a stateless communication protocol, typically HTTP. Below are the list of resources (Objects) managed by the server and associated URIs:

- **User:** This object stores the user details

URI: <https://<server-address>:<port>/users>

POST call on this URI with appropriate parameters will create a new user.

GET call on this URI will get details of all the users. GET call with particular user id will get that user.

DELETE call on this URI with user id will delete the user.

PUT call on this URI with user id and appropriate parameters will update that user.

- **Property:** This object stores the rental house details

URI: <https://<server-address>:<port>/property>

POST call on this URI with appropriate parameters will create a new property (housing).

GET call on this URI will get details of all the properties. GET call with particular property id will get that property.

DELETE call on this URI with property id will delete the property.

PUT call on this URI with property id and appropriate parameters will update that property.

- **Review:** This object stores review given by users on a property.

URI: <https://<server-address>:<port>/review>

POST call on this URI with appropriate parameters will create a new review.

GET call on this URI will get all the reviews. GET call with particular review id will get that review.

DELETE call on this URI with review id will delete the review.

PUT call on this URI with review id and appropriate parameters will update that review.

Apart from these URIs more custom URIs are added in the server to get data with the required query parameters.

5.2 System Design

Unified Modeling Language (UML) is a standard way to visualize design of a system. Use case diagram and Class diagram are the two UML diagrams used to describe a higher level requirements, design and structure of the system. Then to describe database design of the system ER diagram is used which shows the attributes and relationships between entities in the system.

5.2.1 ER Diagram

The main components of ER models are entities (things) and the relationships that can exist among them.

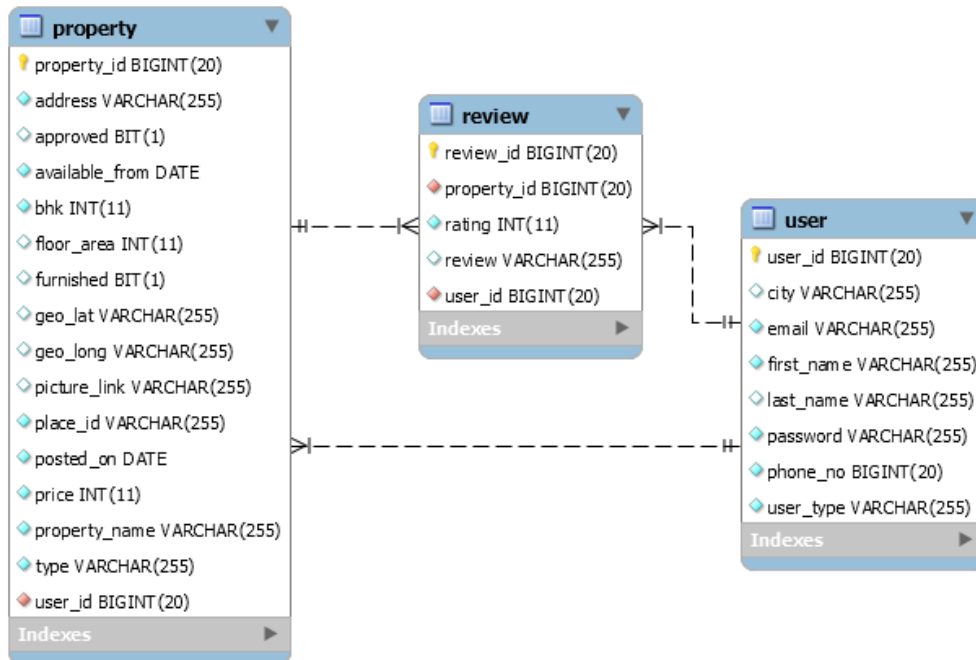


Figure 5.2 ER Diagram

5.2.2 Use case diagram

Use case diagram gives a behavioral view of the system. It depicts what actions a user can perform in the application. Below is the use case diagram for this system.

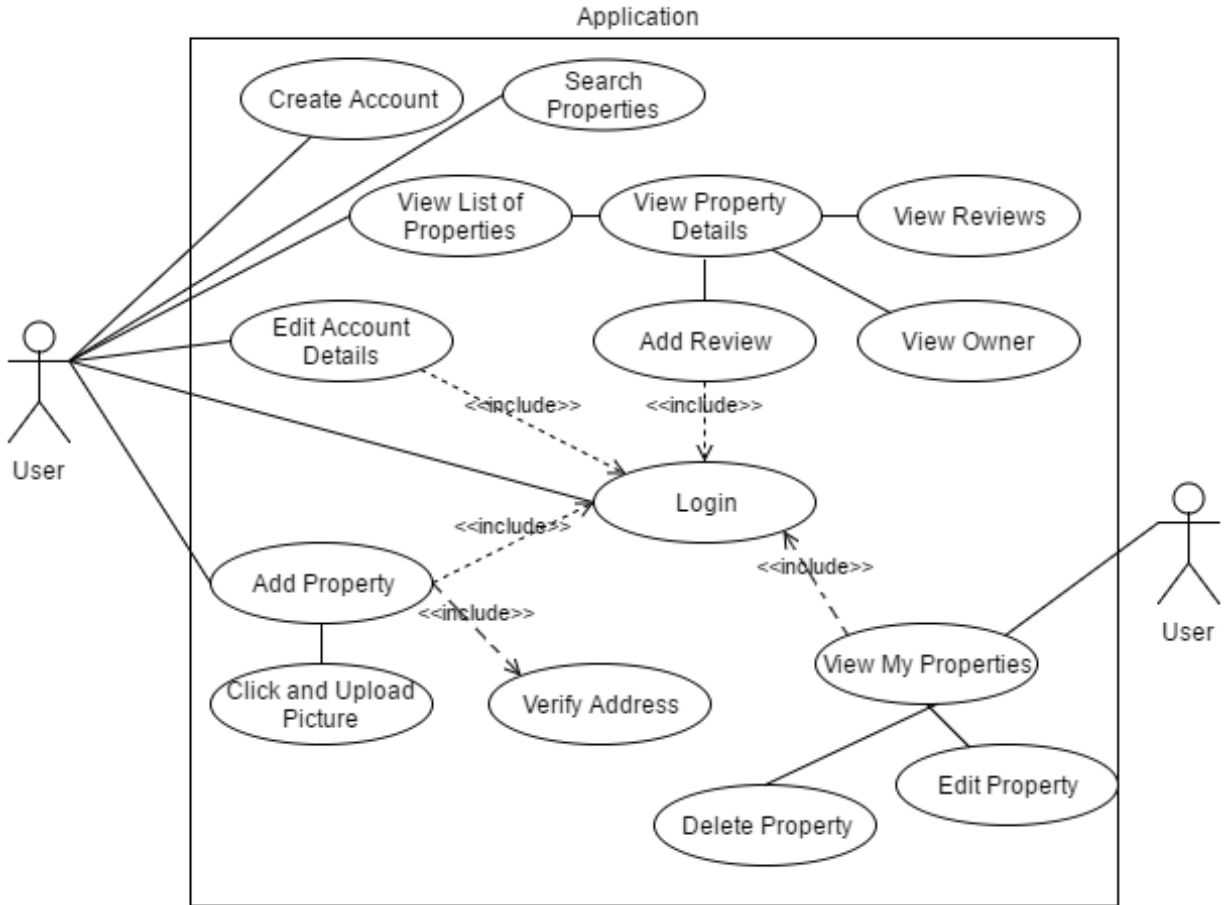


Figure 5.3 Use Case Diagram

5.2.3 Class Diagram

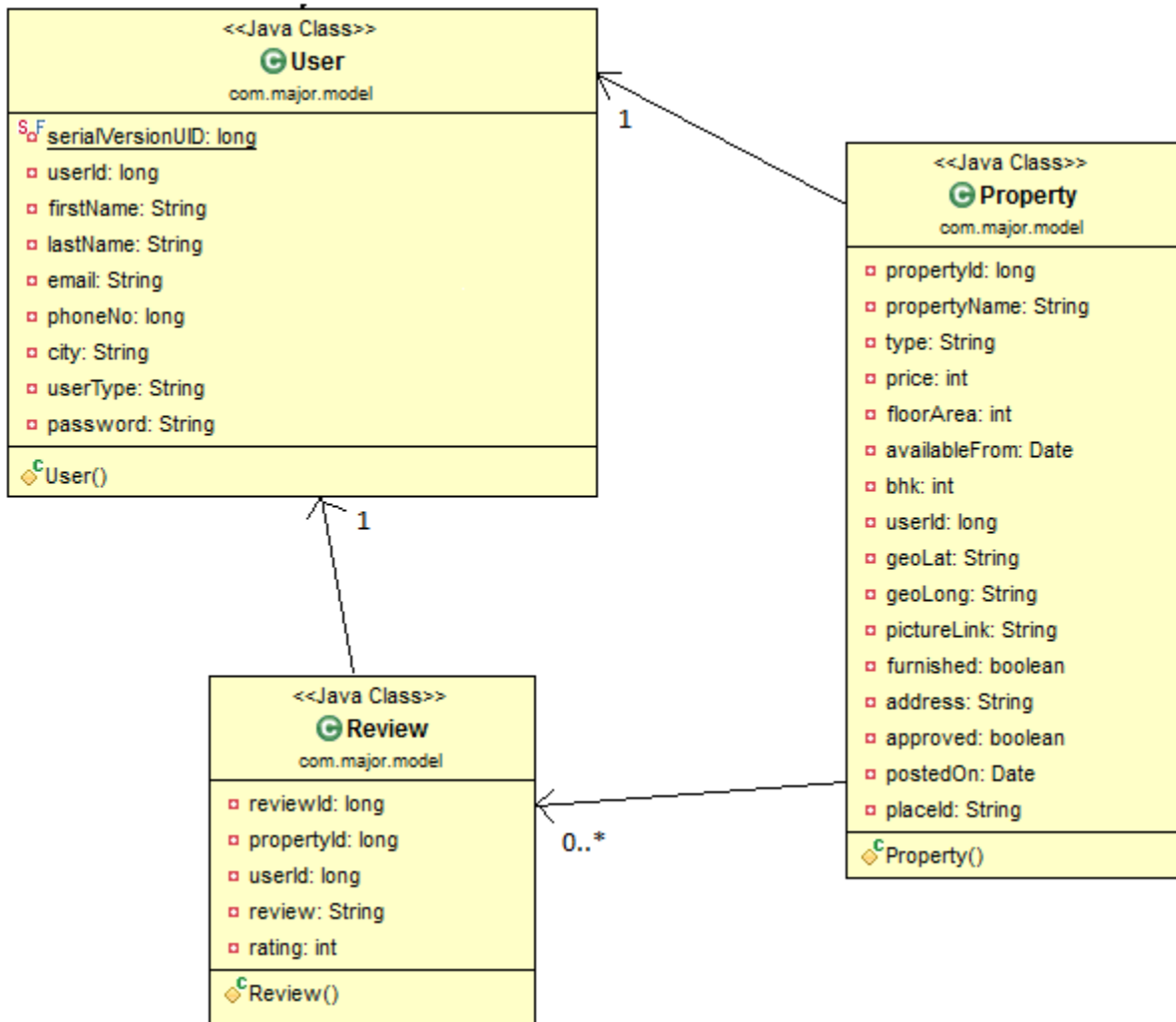


Figure 5.4 Class Diagram

The class diagram above suggests that a property can have one and only one user and 0 to n rating. A review can have one and only one user.

Chapter 6 - Implementation

The purpose of this application is to provide a service where users can search rental homes and see reviews provided by other users. This application provides a simple and easy to use user interface with the following features:

- **Create account** – User can view all the properties and their details without having an account. But to post a property listing user has to create account and login first. User can create account easily by just filling up a form.
- **Login** – Authorized users can login into the application to access more features like add and manage their property listings. Anytime user can go to the login link in the side menu and login to their account.
- **Search property** – User has an option to filter the available housings by starting a search with their desired criteria.
- **View all properties** – List of all the properties can be seen on the home page of the application. This list will have a picture and few important details.
- **View property details** – Selecting a property from the list will open a new page with all the details about it.
- **View owner details** – User can see owner details like email address and phone number.
- **See property location on map** – A map image with marker on the property address will be available to the user, on tapping it user can see that location in Google Maps application.
- **View reviews** – Property details page has all the reviews and average rating at the bottom of the page. User can scroll down and go through the feedback provided by other users.
- **Add a review** – If user is logged in to his account, he/she can add a review on the property currently in view along with a rating from 1 to 5.
- **Add your properties** – If user wishes to list his/her house on the application he can fill a form and submit it. This will make that property visible to all the users.
- **Manage your properties** – User can see the list of properties he/she has added and edit their details like rent, etc.
- **Manage account details** – User can edit his/her personal details which he/she entered during account creation.

Tools and technologies used to develop this application are already discussed in the previous chapters. Total lines of code written to develop this application are 2784 which includes coding in JavaScript, HTML and Java (breakup in Table 6.1). In addition to that efforts were spent in configuration, setting up the development environment, planning design and architecture, and learning tools required for development.

Language	Lines of code (LOC)
JavaScript	1563
HTML	553
Java	668
Total	2784

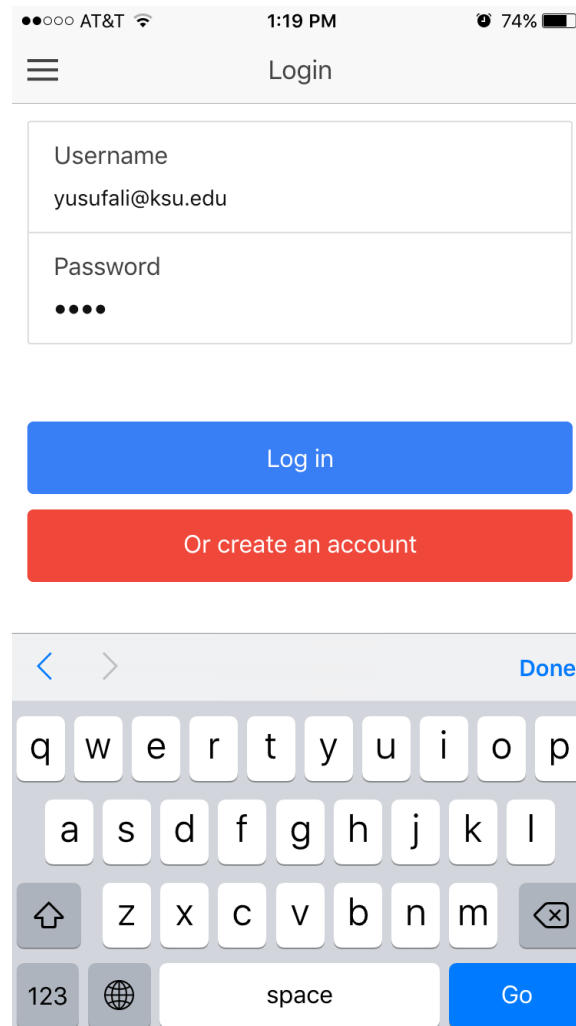
Table 6.1 Lines of Code breakup

Note that a lot of boilerplate code writing is saved by using frameworks like ionic, Cordova on client side and Spring Boot JPA on server side. Also because of these frameworks there was no need to dig into XML coding and configuration.

6.1 Application GUI and working

As mentioned before, the GUI is developed purely using web technologies. Next we see the different pages and their working.

6.1.1 Login



The screenshot displays a mobile application's login interface. At the top, the status bar shows 'AT&T', signal strength, time '1:19 PM', and battery level '74%'. The app's navigation bar includes a hamburger menu icon and the title 'Login'. The main content area contains two text input fields: 'Username' with the value 'yusufali@ksu.edu' and 'Password' with four black dots. Below these fields are two prominent buttons: a blue 'Log in' button and a red 'Or create an account' button. At the bottom of the screen, a standard iOS keyboard is shown, indicating the app is in a text entry mode.

Figure 6.1 Login page

Login page has two input fields username and password. Here username is the email id which user used to register. If user fills username field with a string which doesn't qualify as a valid email address type string (i.e. example@example.com) then the login button will remain disable. Login button will become enable once user fills a valid email address string and password.

6.1.2 Create account

The screenshot shows a mobile application interface for creating an account. At the top, the status bar displays 'AT&T', signal strength, Wi-Fi, time '1:19 PM', and battery level '74%'. Below the status bar is a navigation bar with a back arrow, 'Login', and 'Signup' options. The main form consists of several input fields: 'First Name' with the value 'Yusuf', 'Last Name' with 'Ali', 'Email' with 'yusufali@ksu.edu', 'Password' with four dots, 'Phone Number' with '4253010302', and 'City' with 'Manhattan'. At the bottom of the form is a prominent blue button labeled 'Sign up'.

Figure 6.2 Create account page

If a user doesn't have login credentials he/she can create an account by filling up the login form. The login form also has validations. If any of the name, email, password and phone number field is left empty then Sign up button will remain disabled and user cannot create account without providing those details.

6.1.3 Search

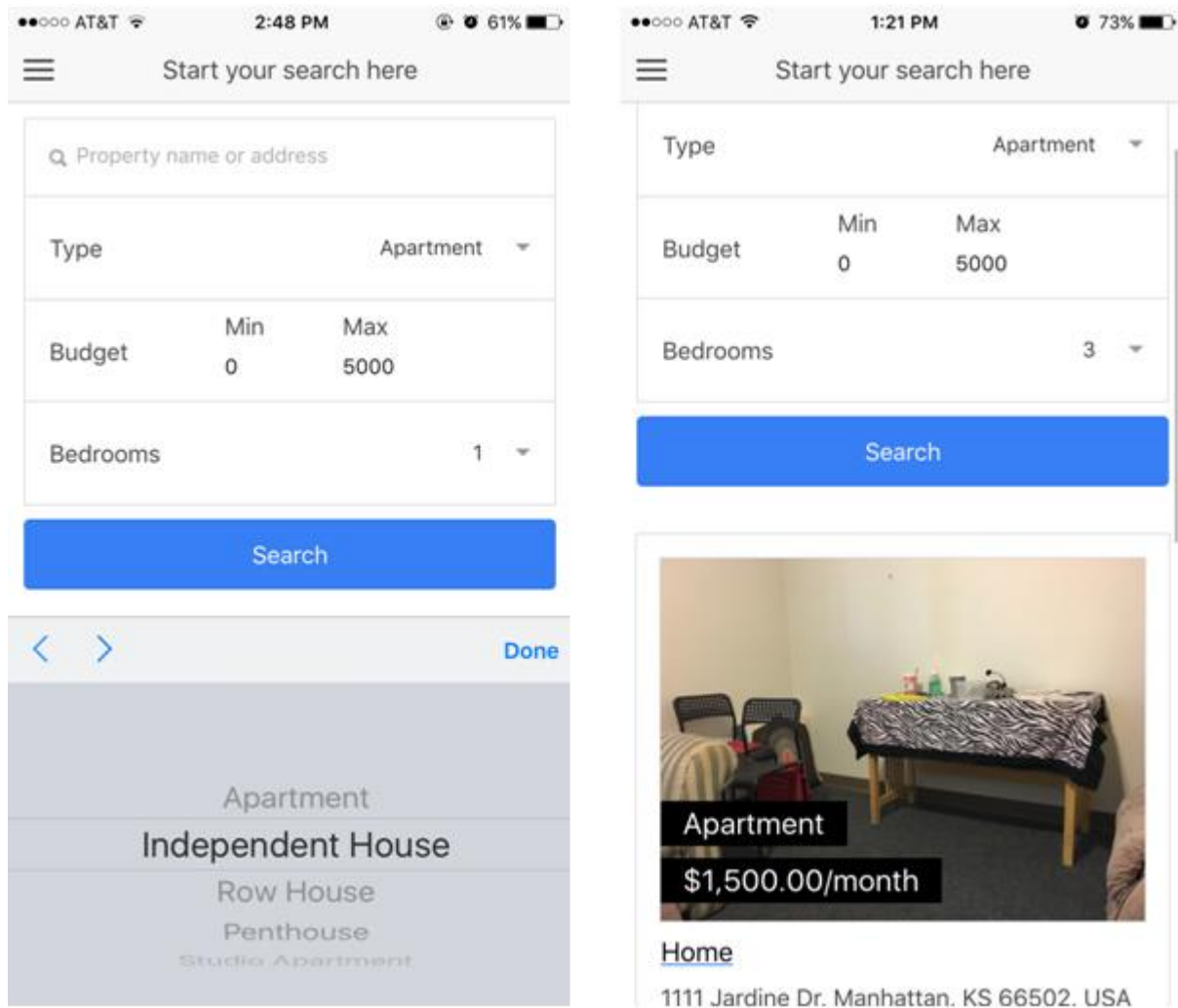


Figure 6.3 Search screen

User can search a property by providing different search criteria. There is a search box on the top where user can either enter the housing name or partial address. User can select the type of housings he/she wants in the result, budget range and number of bedrooms (left part of Figure 6.3). If there are no housings matching the criteria then a “No Results” message will be displayed at the bottom otherwise list of matching housings will be displayed (right part of Figure 6.3). The picture above shows matching result displayed at the bottom of the search. There can be more results; user can scroll down to see them.

6.1.4 Home

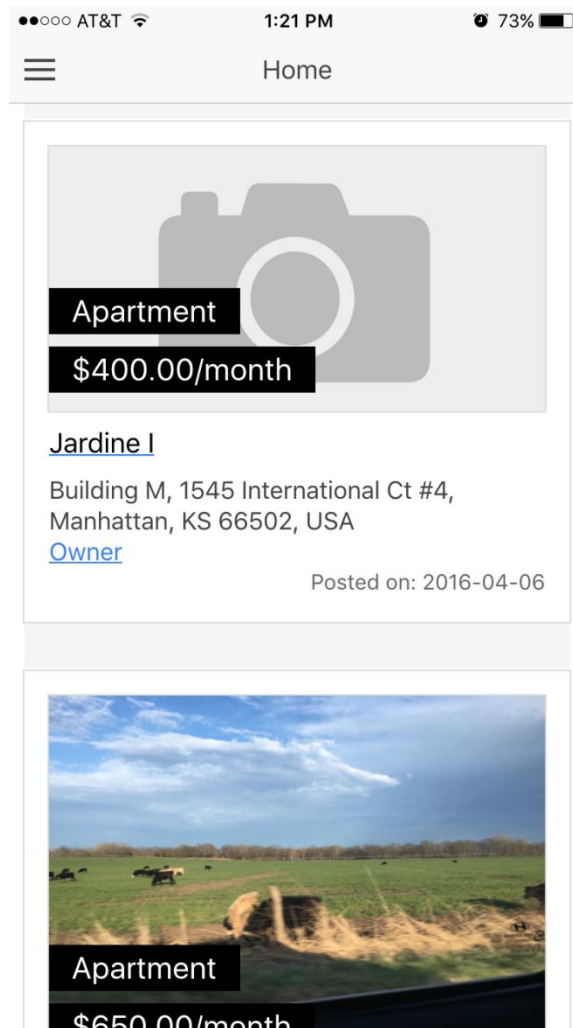


Figure 6.4 Home page

When a user lands up on the home page he/she can see all the housings which have been listed by users. To avoid overloading of this page only 20 listings are showed at one time. After scrolling at the bottom user can load the next or previous results accordingly. This list shows some basic details of the housing like rent, type and address. User can view the owner details by clicking at the Owner link. On clicking at one item, next page will open with more details about that housing.

6.1.5 Property (housing) details

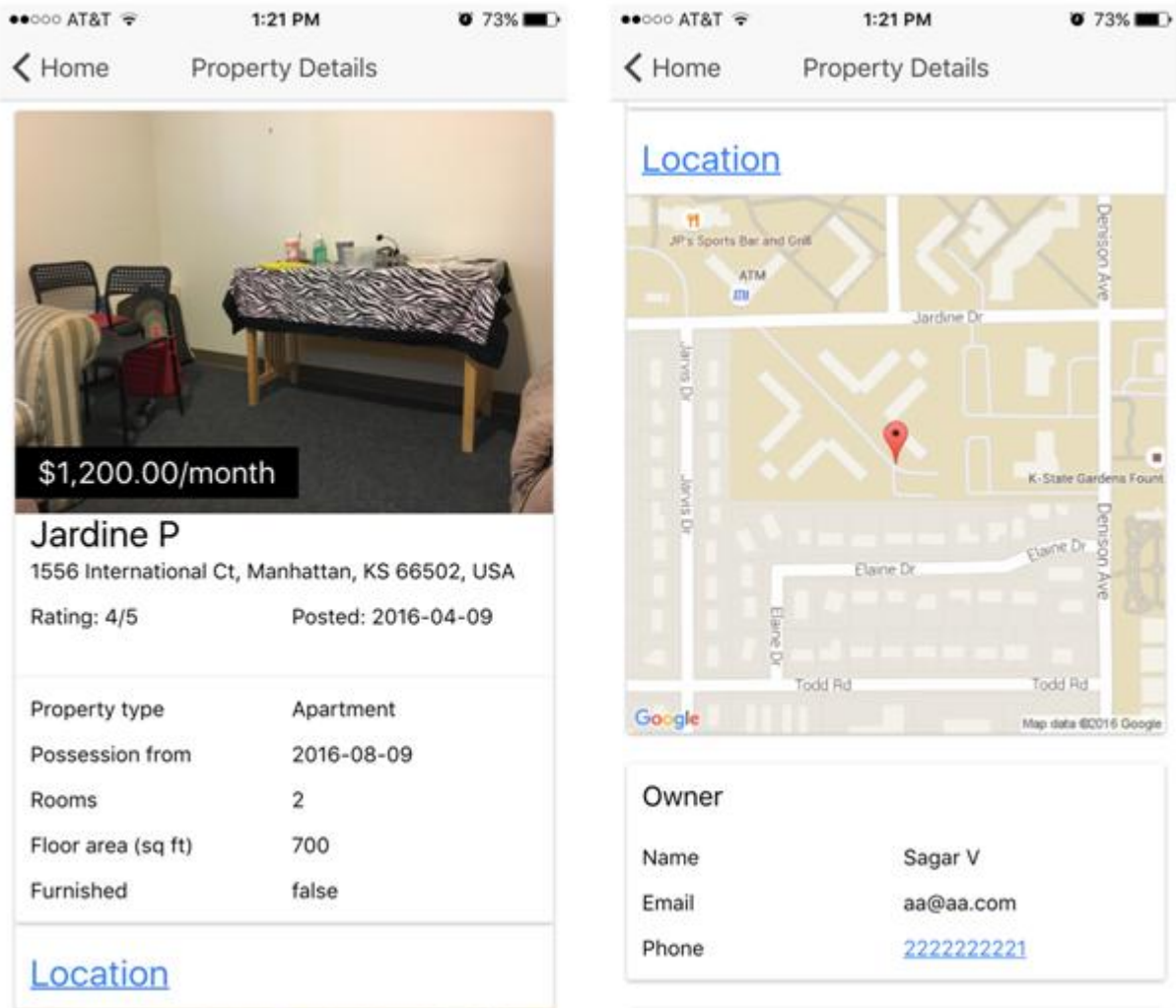


Figure 6.5 Property details page

Property details page shows an image on top with more details about the property like address, average rating, posted on, floor area etc. (left part of Figure 6.5). On scrolling down the location of the property is displayed on Google Map (right part of Figure 6.5). On clicking the map the control leaves the app and opens that location on Google Maps application. Also owner details are displayed below the map.

6.1.6 Add Review

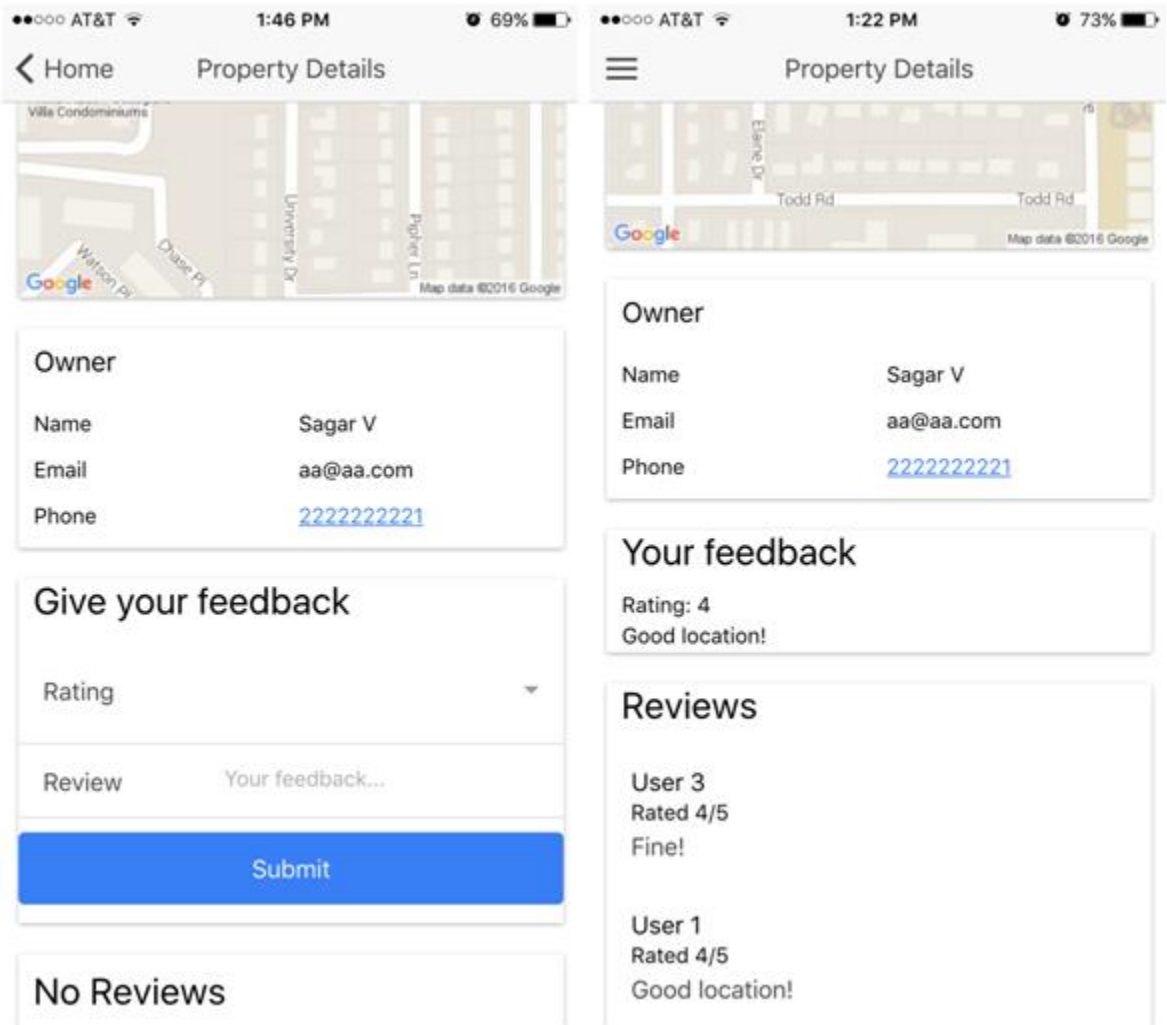


Figure 6.6 Add review

To add a review user should be logged in with his/her account. Review form and reviews of other users are at the bottom of the property details. If the user has not already provided a review, then the feedback form will be visible to the user (left part of Figure 6.6). After submitting the review for a property the user cannot submit review again. So the feedback form changes to a static block with the user's feedback in it (right part of Figure 6.6). If the user is not logged in he/she can only view reviews other users cannot perform add review action.

6.1.7 Side Menu

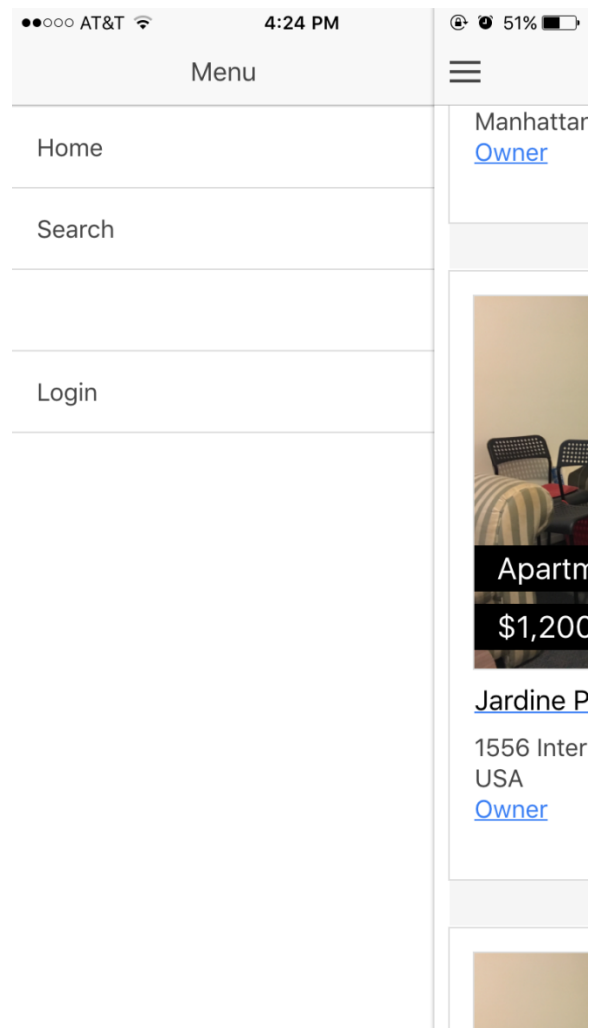


Figure 6.7 Side menu (not logged in)

Side menu helps to navigate to different pages within the application. There can be two different views of side menu. Figure 6.7 shows the side menu when the user is not logged in. So users can only view, search properties or go to login page.

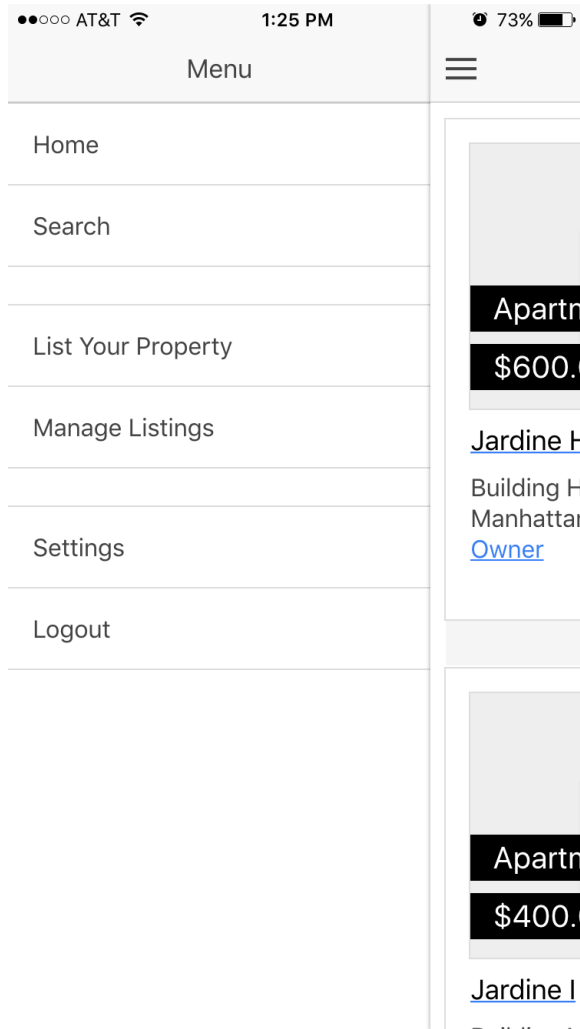
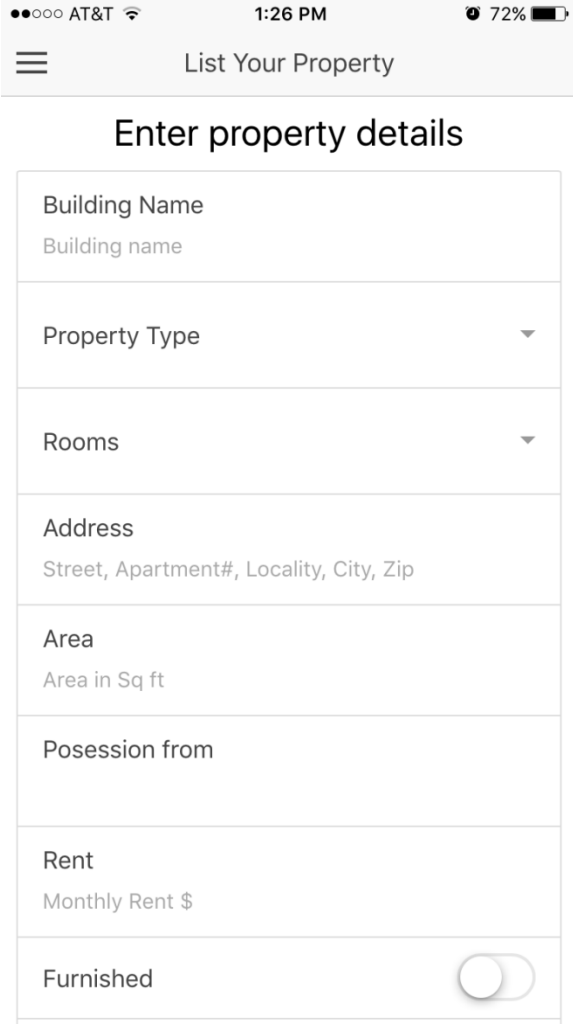


Figure 6.8 Side menu (logged in)

Figure 6.8 shows the side menu when the user is logged in. Now user can perform some additional tasks like adding a property, managing his/her properties, manage account details or logout.

6.1.8 Add a property



The screenshot shows a mobile application interface for adding a property. At the top, the status bar displays 'AT&T', '1:26 PM', and '72%' battery. Below the status bar is a navigation bar with a hamburger menu icon on the left and the text 'List Your Property' in the center. The main content area is titled 'Enter property details' and contains a form with the following fields:

- Building Name**: A text input field with the placeholder text 'Building name'.
- Property Type**: A dropdown menu with a downward arrow.
- Rooms**: A dropdown menu with a downward arrow.
- Address**: A text input field with the placeholder text 'Street, Apartment#, Locality, City, Zip'.
- Area**: A text input field with the placeholder text 'Area in Sq ft'.
- Possession from**: A text input field.
- Rent**: A text input field with the placeholder text 'Monthly Rent \$'.
- Furnished**: A toggle switch currently in the 'off' position.

Figure 6.9 Add property form

User should fill the above form to add housing. Some general details about the house is required among which only building name and furnished are not mandatory. After filling up the form user can also click or use an already clicked photo to upload with the property.

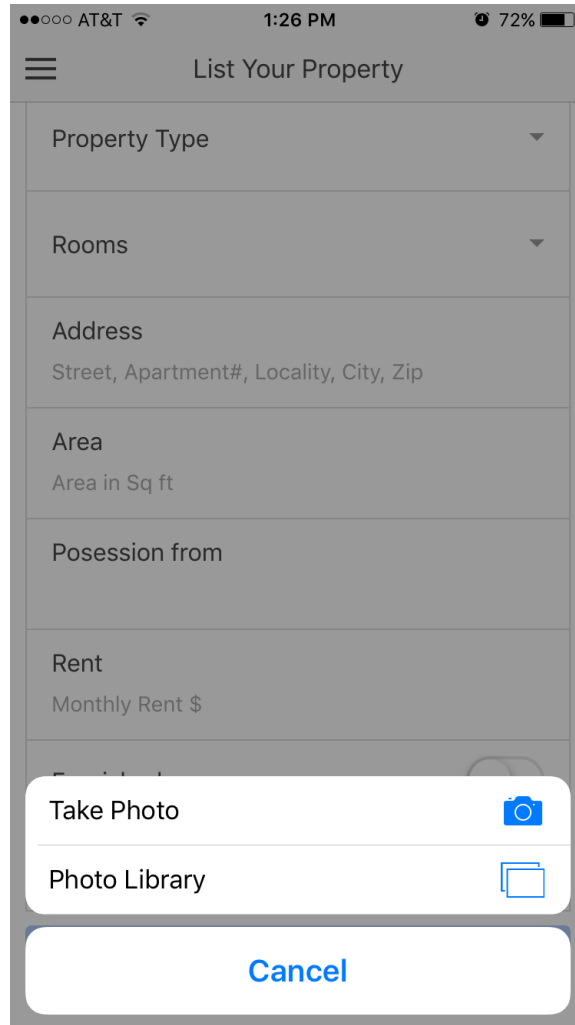


Figure 6.10 Upload picture

Above image shows the phone asking to either click a picture or use photo library to select one. On submitting the form if all the data is perfect a new property is created in the database and the picture of that property is sent to AWS S3 bucket.

6.1.9 Address verification and duplicate address detection

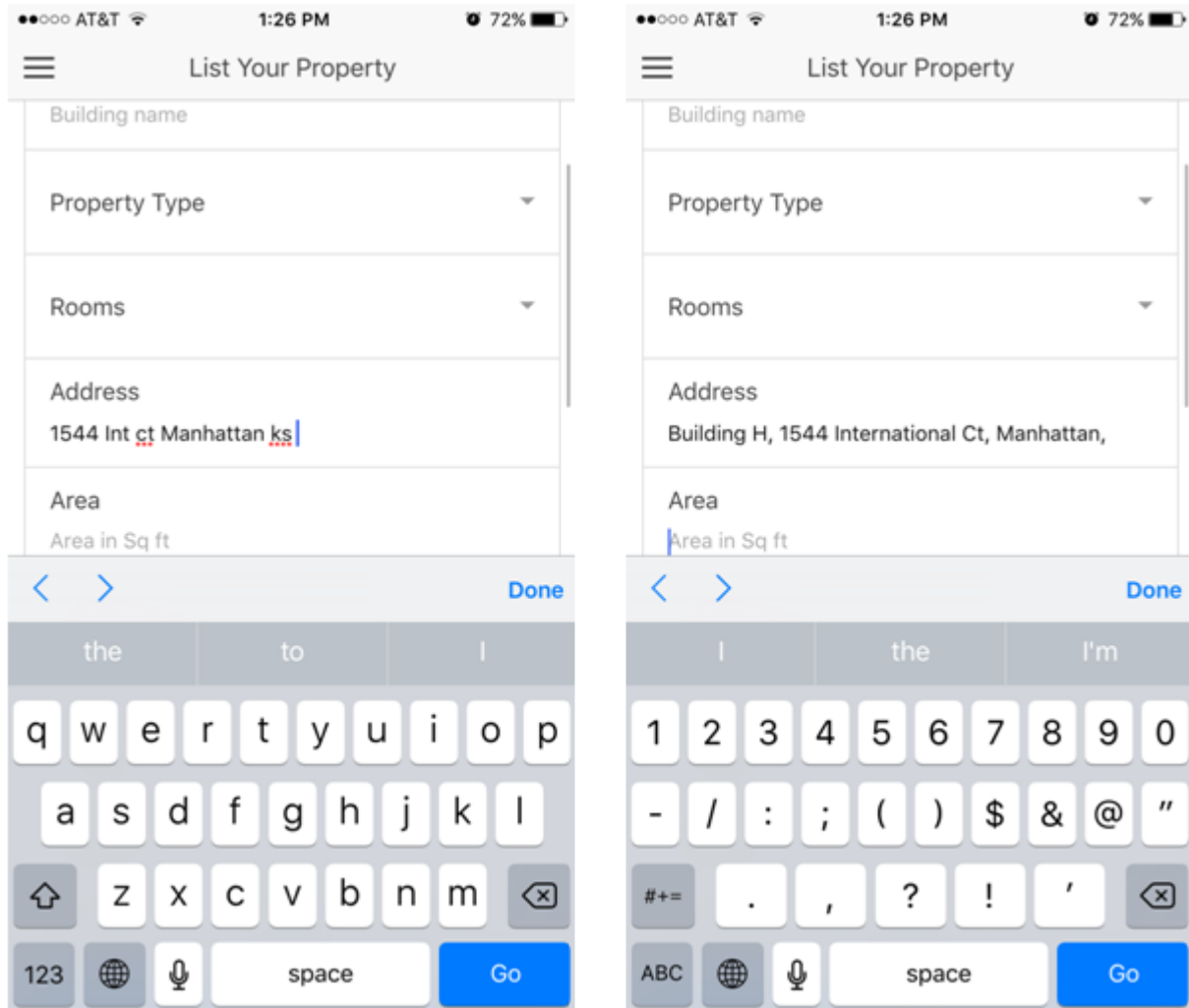


Figure 6.11 Address verification

The address verification feature does many things. One of the features is address auto completion. If users fill an address or incomplete address a request is sent to Google Maps Geocoding API. The Google server checks this address and returns the full formatted address along with more details about that place. If the address is very poorly constructed which Google cannot understand or find ambiguous, the application shows alert. See Figure 6.12.

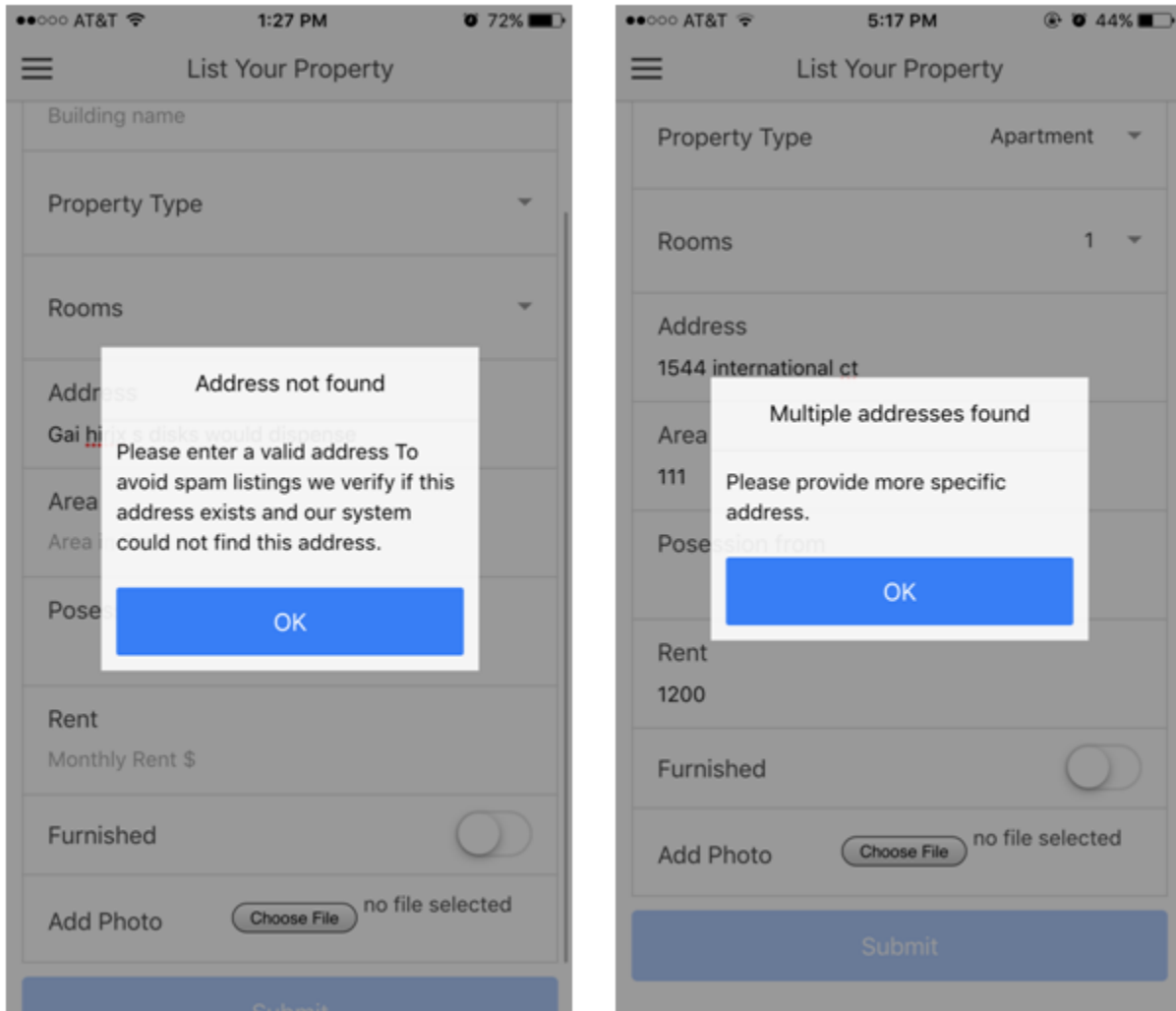


Figure 6.12 Error in address verification

There can be two cases. Either the address is not found by Google in which case system shows error that this address does not exist (see left part of Figure 6.12). For example I enter any random string “safkhjnakjfv sdiuhcius sdlni”, Google will not be able to recognize this address and our application will prevent user from entering this junk value in address.

The second case is Google finds multiple addresses for the entered value. In this case system asks user to provide more specific address (see right part of Figure 6.12). For example: “1544 International court”, Google finds multiple addresses with this input and our application will prevent user from entering ambiguous or incomplete address.

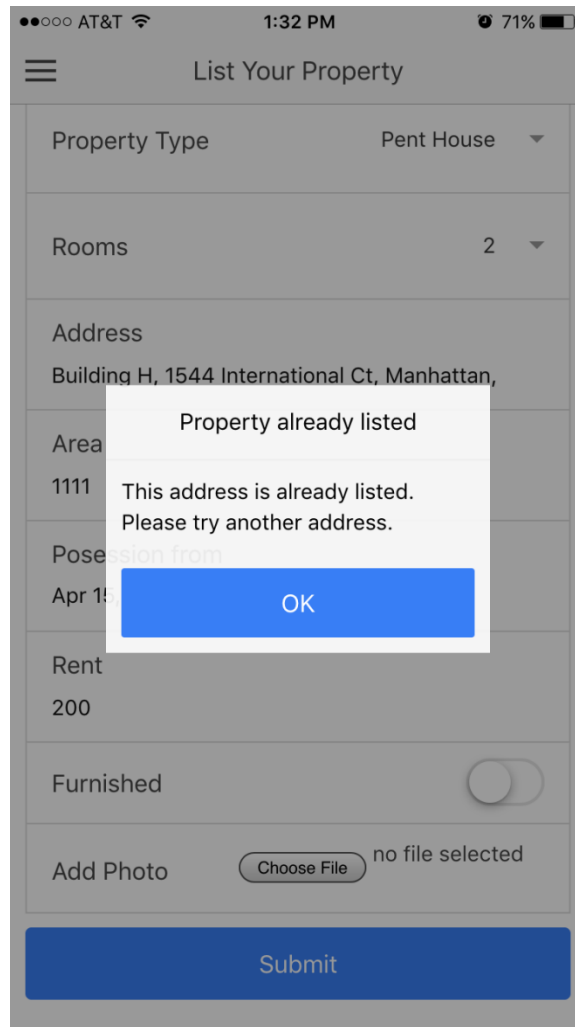


Figure 6.13 Duplicate address in the system

There is one more case in the address verification system when user submits the form. If a property is already present in the database with this address then system will show error that the address already exists. So system prevents user from adding a property twice. This address duplication check is performed using place id received from the response from Google Maps server. Google assigns a unique place id to each and every address.

6.1.10 Manage listings

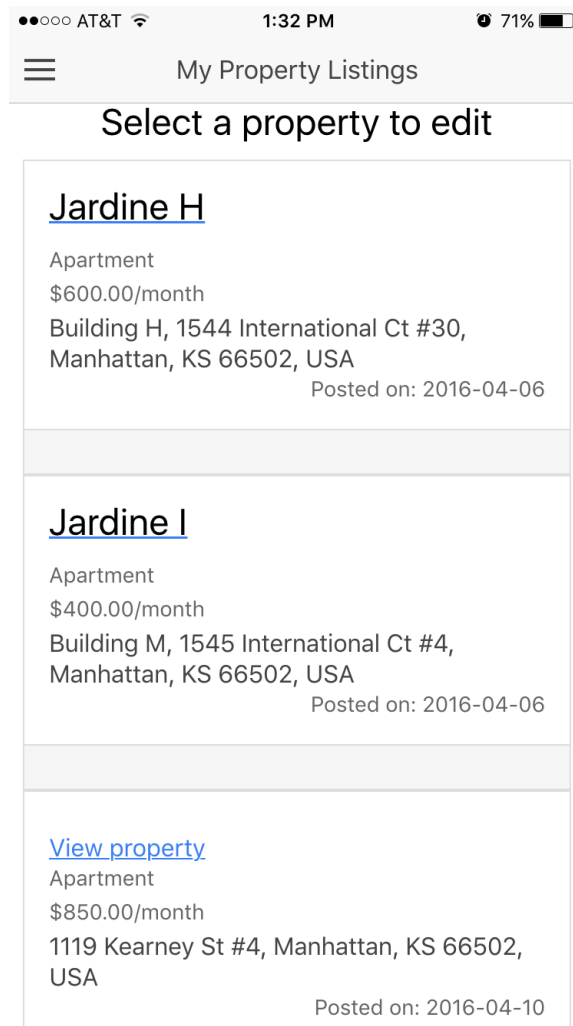


Figure 6.14 Manage listings page

This page lists all the properties or housings which a user has posted. User can select one of the property and view more details of it.

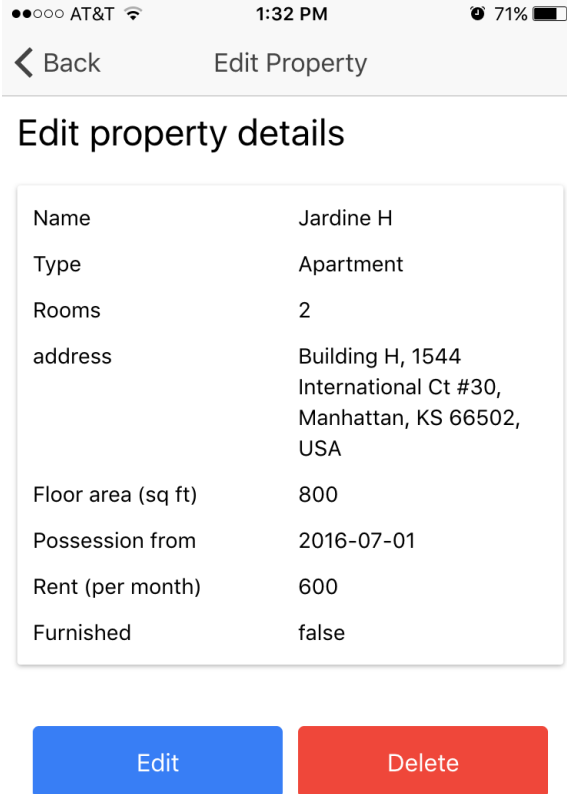


Figure 6.15 Edit property page

User can either choose to edit or delete the property. If user selects edit, then the static view converts into a form where user can make changes by editing values into the input fields. If user selects delete button, a confirmation window will ask user if he/she really wants to delete the property. See Figure 6.16

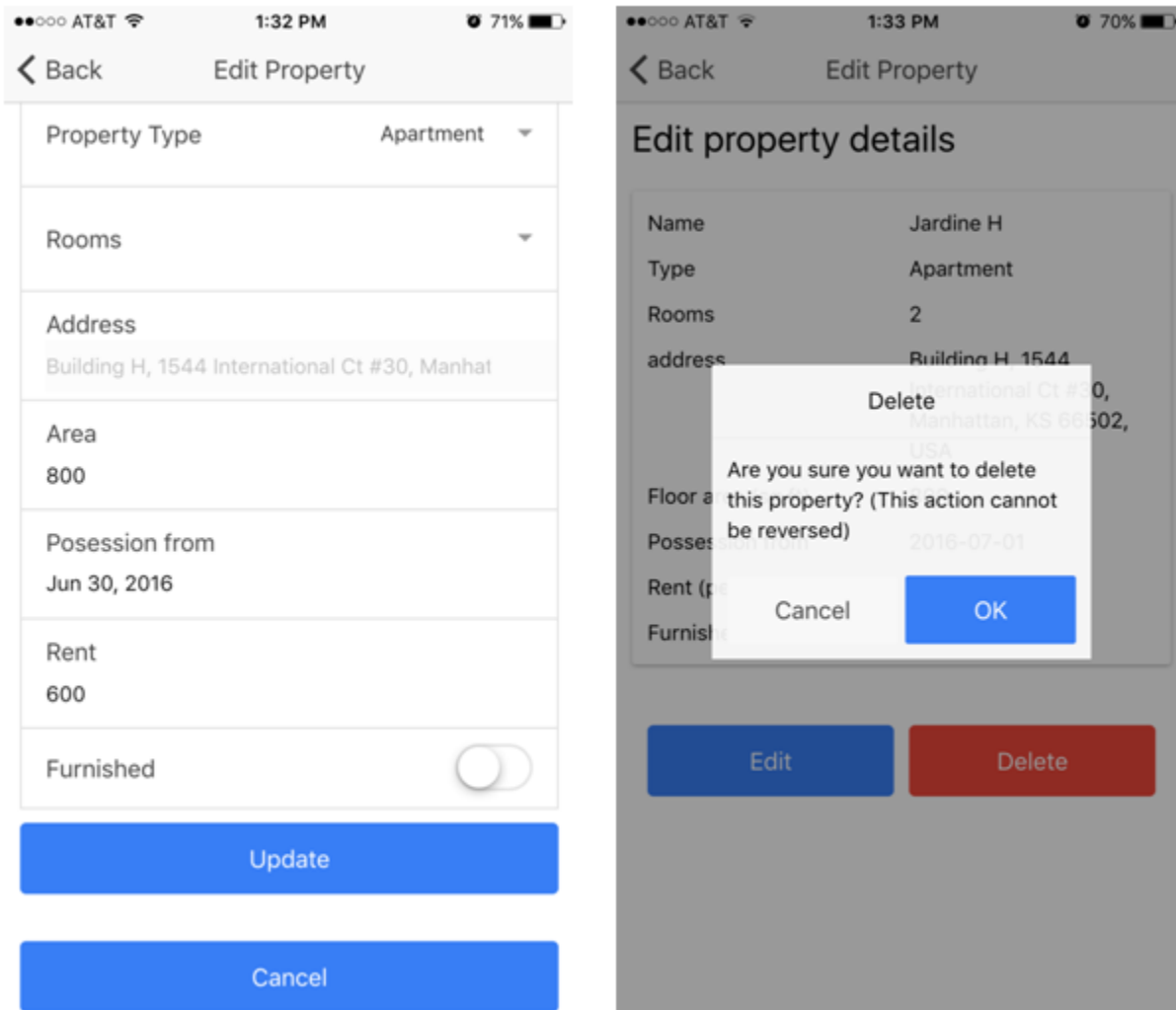


Figure 6.16 Edit and delete property

Left part of Figure 6.16 shows the edit view. User can edit all the details of the property except the address. This view useful in the sense that user might want to increase or decrease rent, change possession date etc. Selecting update button brings back the detail view with updated data. Whereas selecting cancel brings back the detailed view with previous data.

Right part of Figure 6.16 shows the delete confirmation. If user selects OK then the property is deleted from the database with all the reviews.

6.1.11 View and edit personal details

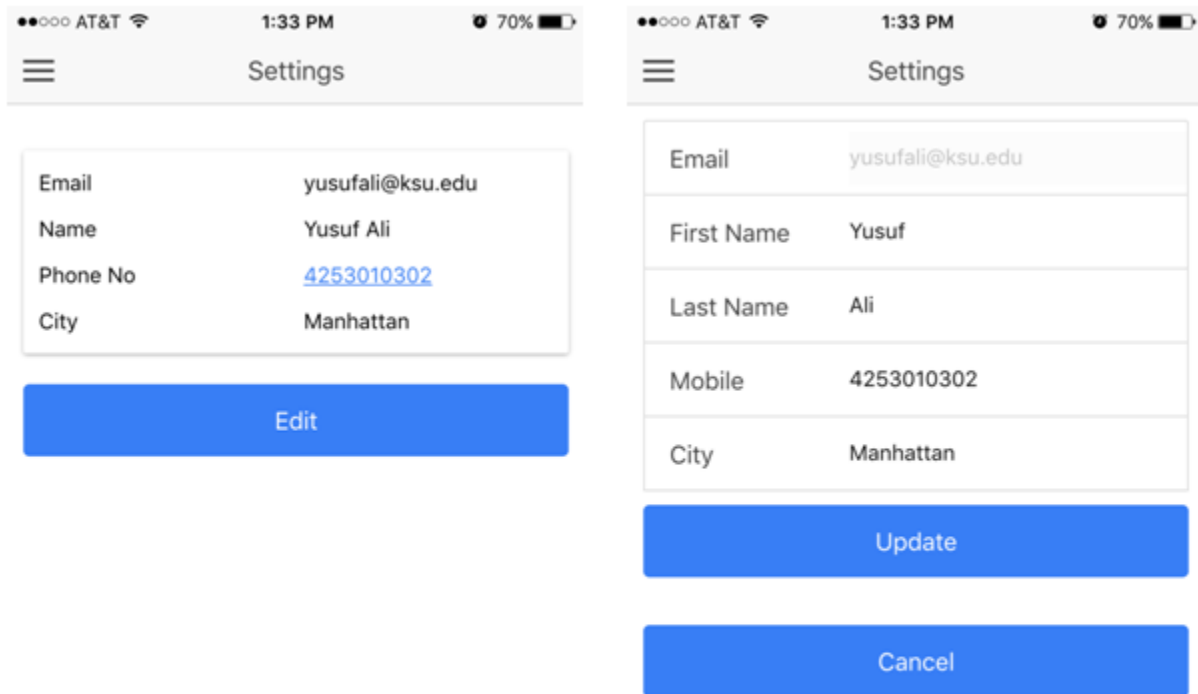


Figure 6.17 Edit personal details

Left part of Figure 6.17 shows personal details view. User can tap on the edit button to edit these details. The edit view is shown in right part of Figure 6.17. User can edit all the details except email. Selecting update button brings back the detail view with updated data. Whereas selecting cancel brings back the detailed view with previous data.

Chapter 7 - Testing

To ensure correctness and quality of the application, both manual and automated testing is done on it. Unit and integration testing is done manually using smart phones LG Nexus 5 having android v6.0.1 and iPhone 6s having iOS 9.3.1. Compatibility testing is done using an online automated tool www.testobject.com. All the tests gave expected results and no issues were reported by the automated testing tool.

4.1 Unit Testing

Unit testing is done to test each part of the application independently. The following unit tests were performed manually on LG Nexus 5 and iPhone 6s:

S. No.	Test case	Expected Result	Result
1.	On application startup	Open search page	Pass
2.	Tap menu button while on any page	Open side menu	Pass
3.	Slide finger right	Open side menu	Pass
4.	Under menu tap Home	Go to home page and display results	Pass
5.	Under menu tap Search	Go to search page	Pass
6.	Under menu tap List Your Property	Go to list property page and display form	Pass
7.	Under menu Tap Manage Listings	Go to manage listings page	Pass
8.	Under menu tap Settings	Go to settings page	Pass
9.	Under menu tap logout	Destroy user's data on phone and go to login page	Pass
10.	Under menu tap Login	Go to login page	Pass
11.	Search page: tap on search box	Open key board to allow user to type text	Pass
12.	Search page: tap on Property Type	Open options to allow user to select one option	Pass

13.	Search page: tap on Budget	Open key board to allow user to type	Pass
14.	Search page: tap on Bedrooms	Open options to allow user to select one option	Pass
15.	Search page: tap Submit	Display results at the bottom of the page. If no data related to that criteria is present then display “No results found”	Pass
16.	Home page: tap on image area or building name	Navigate to property details page and show all the property details. Hide the review form if user is not logged in	Pass
17.	Home page: tap on owner link	Open owner details page and show owner details	Pass
18.	Property details page: tap on map	Open Google maps application with that location pinned	Pass
19.	Property details page: enter rating and review then tap Submit	Store review to database and show at the bottom. If user leaved the Rating field empty then display alert	Pass
20.	List Property page: tap on any field	Open appropriate input methods	Pass
21.	List Property page: enter address and leave address field	Auto-fill address with the formatted address. If address incomplete or not found then display and alert	Pass
22.	List Property page: tap on Choose file button	Allow user to click or choose a picture from gallery	Pass
23.	List Property page: tap Submit button	Send data to database and picture to AWS S3 bucket and reset form	Pass
24.	Manage Listings page: tap on any property	Navigate to edit property page with that property in view	Pass
25.	Edit property page: tap on Edit button	Open edit property form with details of that property filled in	Pass

26.	Edit property page: click on any field	Open appropriate input methods	Pass
27.	Edit property page: tap Update button	Update property details in database and update those details in view	Pass
28.	Edit property page: tap Delete button	Show alert to confirm delete. If OK, delete the property from database	Pass
29.	Settings page: tap Edit button	Open edit user details form with details of user filled in	Pass
30.	Settings page: tap Update	Update user details in database and update those details in view	Pass

Table 7.1 Unit Test Cases

4.2 Integration Testing

There are some modules in the application which calls other components of the application or sends request to the server. The following test cases test the interaction between different modules of the application and with the server. Note that in the below tests successful result from the means integration of database with server is also tested.

S. No.	Test case	Expected Result	Result
1.	Login	Authenticate user from server. Display error if authentication fails, otherwise navigate to search page	Pass
2.	Submit signup form	Send POST request to server with form details and add user to database	Pass
3.	Navigate to home page and home page refresh	Send GET request to server to fetch the property details and fetch picture from AWS S3 bucket.	Pass
4.	Start search	Send GET request to server with filter parameters and properties from database	Pass
5.	Submit add property form	Send POST request to server with form details and add property to database. Also add picture to AWS S3 bucket	Pass

6.	Navigate to Manage Listings page	Send GET request to server to fetch the details of all the properties the user have added.	Pass
7.	Submit update property form	Send PUT request to server with form details and update that property in database.	Pass
8.	Delete property	Send DELETE request to server with property_id and delete that property from database	Pass
9.	Submit update user form	Send PUT request to server with form details and update that user in database.	Pass
10.	Logout	Send POST request to server and delete user's data from the device	Pass

Table 7.2 Integration Test Cases

4.3 Compatibility Testing

This application is developed considering diverse range of devices. This application is tested on LG Nexus 5 phone with android version v6.0.1 and on iPhone 6s with iOS version v9.3.1 on both portrait and landscape mode. Moreover using online testing tool www.testobject.com, this application is tested for compatibility and quality on Samsung Galaxy Nexus with Android v4.3.0 (Screen: 720X1280, 4.65”), Samsung Google Nexus 10 P8110 with Android v5.1 (Screen: 1600X2560, 10.1”), LG Nexus 4 E960 with Android v5.1.1 (Screen: 768X1280, 4.07”) and Asus Google Nexus 7 with Android v5.1.1 (Screen: 800X1280, 7”).

Below are the compatibility tests and their results:

S. No.	Test	Result			
		Samsung Galaxy Nexus	Google Nexus 10 P8110	LG Nexus 4 E960	Asus Google Nexus 7
1.	Install and Launch	Success	Success	Success	Success
2.	Stress Test	0 issues reported	0 issues reported	0 issues reported	0 issues reported

3.	Page transitions and button press	0 issues reported	0 issues reported	0 issues reported	0 issues reported
----	-----------------------------------	-------------------	-------------------	-------------------	-------------------

Table 7.3 Compatibility Test

Chapter 8 - Conclusion and Future Scope

Smart phones and tablets are prevalent in today's world which gives us a platform to think and implement limitless ideas. Android and iOS are the two mobile operating systems which are mostly used in the market today. The purpose of this project was to create an application for searching and rating rental housing. Having the application on both Android and iOS platform makes it easily available to most of the users. The application create fulfills this purpose and allows users to search, rate, review and add a housing using their phones.

The project meets all the functional requirements and runs on the intended platforms successfully. This project helped to learn develop cross platform mobile apps using popular frameworks like ionic and Cordova. I learnt how to design and work with a multi-tiered architecture. Also the project helped me in learning to create RESTful web service using Spring Boot framework and Java Persistent API. I also learnt how to use AWS S3 cloud to store and retrieve data that is not suitable for traditional RDBMS style database.

This application can be further enhanced in future to provide more features to the users. Some of the enhancements are:

- Merge the feature to rate and review the properties in already existing popular housing portals.
- Add recommendation feature.
- Along with the list view the search results can also be displayed on the map to allow user to get a better understanding of the area.
- Include an admin view to manually approve or reject a property listing to avoid spam listings.
- Add captcha to keep away bots from posting fake reviews.

References

1. Working of Cordova and conceptual view. [Online] [Cited: April 03, 2016]
<http://scn.sap.com/community/developer-center/mobility-platform/blog/2014/07/27/what-is-cordova-and-how-does-it-work>
2. Cordova Android Platform Guide. [Online] [Cited: March 01, 2016]
http://docs.phonegap.com/en/edge/guide_platforms_android_index.md.html#Android%20Platform%20Guide
3. Installing android SDK. [Online] [Cited: March, 28, 2016]
<http://developer.android.com/sdk/index.html>
4. Installing PhoneGap on Windows. [Online] [Cited: March 01, 2016]
<https://github.com/simnova/webdevdocs/wiki/Installing-PhoneGap-and-Android-Studio-on-Windows>
5. Using PhoneGap storage. [Online] [Cited: March 20, 2016]
http://docs.phonegap.com/en/2.1.0/cordova_storage_storage.md.html#openDatabase
6. Using Google Maps API. [Online] [Cited: Mar 23, 2016]
<https://developers.google.com/maps/>
7. Using Google Geocoding API. [Online] [Cited: Mar 23, 2016]
<https://developers.google.com/maps/documentation/geocoding/>
8. Accessing Data with JPA [Online] [Cited: Mar 08, 2016]
<https://spring.io/guides/gs/accessing-data-jpa/>
9. Spring Data JPA - Reference Documentation [Online] [Cited: Mar 07, 2016]
<http://docs.spring.io/spring-data/jpa/docs/1.10.0.M1/reference/html/>
10. Accessing JPA Data with REST [Online] [Cited: Mar 12, 2016]
<https://spring.io/guides/gs/accessing-data-rest/>
11. Uploading To S3 With AngularJS [Online] [Cited: April 04, 2016]
<http://www.cheynewallace.com/uploading-to-s3-with-angularjs/>