Student Housing

Joe Tom Job
jxj2428@rit.edu
&
Sikha Rani
sxr3531@rit.edu

Final Project Report for Mobile Application Development I

Information Sciences and Technologies
Rochester Institute Of Technology
Rochester, USA, 14623

Contents

1 Application Definition Statement

This application acts as a platform for students looking for housing which includes rent, sublet and roommate requests. Currently there is no such existing housing application which acts as a common platform for students from a particular university.

2 Implemented Features

- **Database**
  The extra feature which we implemented in our project was using "Firebase" [1] as the database for the application. "Firebase" is a mobile and web application development platform. "Firebase" provides a lot of features significant for application development. The following features from "Firebase" were used in our application:
  - **Authentication**
    The login/register feature for the application mentioned in section 2 uses the Authentication feature provided by the "Firebase".
  - **Database**
    The data required for the application is saved in "Firebase" database. All the data about the apartment listings are saved in the database. The data is stored in JSON format.
  - **Storage**
    The images for the apartments are stored in "Storage" feature of Firebase. Firebase provides a download URL for the images saved in Storage. This can be used to retrieve the images from the database.

- **Login/Register**
  The users can login/register using the Login page. For registering, the user needs to provide the email id and password. The login system for the project was implemented using the "Email and password" based authentication provided by Firebase. The login page prompts the user to enter the user name and password for login. A new user needs to register in the application using email id and password. Once logged in, the user will be redirected to the initial page discussed in the next section 2. The username and password are stored in the userdefaults so that the user does not need to enter the login credentials again. The screenshot of the login page can be seen in 1.

- **Initial Page**
  The initial page has two text fields which displays the country and the college name. The country and college name will be auto-populated. The country is obtained from the current location. The college text will be auto populated by the nearest college. The college details are saved in the database 2. The users current location is identified by the *CoreLocation* feature available in Swift. The university details is saved in the database. The university location is identified from the university address stored using *geocoder* feature available in *CoreLocation*. By calculating the distance between the university and current location, the nearest university is identified. There is a search button available in the page. On clicking on the search button, the application is redirected the Listing page 2. The Initial page has the option to logout from the system. Once the user clicks the Logout button, the user will be logged out and redirected to the Login/Register page. The screenshot of initial page can be seen in 2.

- **Listing Page**
  The listing page lists all the listings/apartments available near the user. Apartments within 25 mile range will only be displayed. The apartment name, address, rent, distance from the current location and an image of the apartment can be seen in the Listing page. The user will have the provision to sort the listing based on two parameters which are price and listing.

The list can be sorted in ascending and descending order based on price and distance. The distance is computed from by calculating the distance between the current location and the location obtained from the address provided in the listing. Clicking on any row will redirect the user to a listing detail page which shows all the details related to the selected apartment 2. The screenshot of listing page can be seen in 3.

- **Listing Detail page**
  The specific details about the listing/apartment is listed in this page. All the details including price, description, contact details etc will be shown in this page. The user can contact the user via various means. The user can also navigate to the apartment location from the current location using the from this page. The apartment listing can be saved to favorites from this page. The features available in this page are discussed in detail below:

  – **Contact**
    The user can email the owner by clicking on the "Email" button available next to the email id. The button click will pop up the mailing page. The phone number of the user is also listed in the listing detail page. The user can call/message the user by clicking on the CALL/MESSAGE button available next to the contact number. The application throws an alert if the user is not capable of doing any of these(call/message/email).

  – **Maps and Navigation**
    A map showing the location of the apartment is available in the listing detail page. A pin is placed on top of the apartment listed. On clicking on the pin, the user will be prompted to navigate to the apartment location. On clicking, the navigation page will be opened driving directions from current location to apartment location will be shown.

  – **Favorites**
    If the user wants to save an apartment listing for future, he/she can add it to his favorites. There is an add to Favorites button available in the page. An alert message saying "Added to Favorites" will be popped while clicking on the button so that the user gets the notification that listing is added to his favorites.

  The images in the listing detail page are shown using a "ImageSlideShow". This allows to show multiple images of the listing as a slide show. The user can swipe left or right to see the next images [4]. The images are fetched from the url using *Alamofire* [2]. The screenshots of listing detail page can be seen in 4 and 5.

- **Favorites Page**
  The Favorites tab will show all the listings which are marked as favorite by the user. On clicking a row/apartment listing from this page, the application redirects to the Listing detail page discussed in section 2. The user also has the option to delete the listings from his favorites. On clickin on the edit button available at top right corner of the favorites page, the user can delete the listings from his/her favorites. The screenshot of listing page can be seen in 7.

- **Posting Page**
  The user will be able to post details about the apartment to be rented. The username will be auto populated. The user can enter the rent details, address, availability date, contact details and many more details. The user can also upload pictures of the apartment being posted. Once posted, the user can navigate to My Listings page to view his/her listings 2. There are different UI controls which we have used for user's ease of use like switch, slider , UIPickerView and DatePickerView.We have also included validaton so that the user can't post a posting with empty values.Also Once a user posted his posting and in future he wants to edit that posting he can always go to the mylistings page and edit any of his postings and after submitting the firbase database will be update accordingly. The user will not be able to post a listing if he enter invalid address. The application checks the address

provided and verify whether it is a valid one with the help of geocoder feature available within CoreLocation.

- **My Listings page**.
  The user can see the postings made by him in the My Listings page. The user can edit his/her post from this page. Clicking on a row/listing will redirect to the "Edit post" page. The user can make any changes for his post in this page. The user can also reaarange his/her mylisting sequence in the table.He can also delete a listing and it will also permanently delete that listing from the firebase database and reload the listings and mylistings page consequently.

- **Edit post page**
  The Edit post page will be same as the Post page. If we are redirecting from the mylistings page, all the controls will be auto populated. The existing post can be reordered/deleted on clicking the edit button.We can also go inside each of the mylisting values and even edit it to update the database simultaneously.

3 Features for future versions

- **Mapview page**
  A new mapview page can be included which shows all the apartment listings near the user in mapview. The user will be able to see the details of the listing and navigate to the page from the mapview page. The mapview page can have standard, satellite and hybrid view. The mapview will be focused to a 25 mile range from current location.

- **Prevent invalid values to be posted**
  The Before posting, the system needs to check whether the provided address is valid. This can be done with the help of *geocoder* feature available in CoreLocation.

- **Sharing a Post**
  The user will have the ability to share a posting among friends through applications Facebook or Whatsapp.

- **List nearby attractions, restaurants, parks and schools**
  Once the user select the apartment listing, all the nearby attractions, schools, parks, restaurants will be listed in the mapview available in the Listing detail page 2, 5.

- **Admin user**
  The will be one or more admin users who will have the privilege to delete irrelevant or scam posts. Admin user will not be allowed to edit. He will have the sole option to delete a post or block a user from making further posts.

- **Report Invalid Posts**
  The user will have an option to report invalid posts. The admin will get notifications and he can take actions after observing the scenario.

- **Notifications**
  If the user has posted an apartment listing and if others try to connect to the owner, notifications should be sent to the owner of the post.

- **Ratings** The users will have an option to give ratings for the owners of the post. Thus user gets to know the credibility of the post owners using their ratings.

4 Self Evaluation and Documentation

We have implemented all the features mentioned in the proposal document. We assess ourselves as follows:

- **Data: A**
  The data is saved in the using "Firebase" [1]. The user details, posting details and the university details are saved in the Firebase database. The images are stored in the Storage feature of Firebase.

- **Design: A**
  - The design of the application is good with a lot of option for building new features on top of it. We have followed MVC pattern to build our application.
  - "Listing" class has been created to store all the data related to a listing. Objects of this class is being across the pages. The "Listing" page, "Posting" page, "Listing Detail" page, "My Listings" page and "Favorites" page.
  - A "University" class has been created to save the details of the universities. University details are being saved in the objects of this class.
  - The application is a tabbed application with different tabs suiting to the convenience of the user.
  - We have used delegates from "CoreLocation" to get the current location. Delegates from "Mapkit" has been used to navigate to the apartment location from current location. "UIPickerview' delegate has been used in Posting page. Delegates has been used to populate the table view. cell size, count of rows in a section and number of sections in the table view are also controlled using the delegates available in the "TableViewController".
  - Images are being lazy loaded into the application. The images are fetched only when required. *Alamofire* [2] has been used to fetch the images from Firebase storage using the image URL.
  - The application is pretty fast. Expect from posting and login, user do not need to type much data. The basic filters for listing like Distance and Price are added in the UI.
  - The user can logout from the application from the search page.
  - The user will not be able to post a listing with invalid address. The application checks the address provided and verify whether it is a valid one with the help of geocoder feature available within CoreLocation.
  - The username and password are stored in the userdefaults so that the user does not need to enter the login credentials again.

- **User Interface: A**
  The user interface for the application is simple and user friendly. All the pages was well organized. There are five tabs for the application. The first tab shows the country and the nearby college. Posting can be done using the second tab. All the apartment listings within 25 mile range from user location will be displayed in the third tab. The fourth tab shows the Favorite listings and the fifth tab shows the listings posted by the user.

- **External Frameworks: A** The application has used external frameworks for its development. The list of external frameworks used are described in section 6. The third party frameworks used were "Firebase", "Alamofire" and "ImageSlideShow".

5 Classes

- **LoginViewController**
  The login view controller handles the login page. The login button redirects the user to the Initial page where college name and country is displayed. The LoginViewController is a normal UIViewController.

- **InitialVC**
  The initial page is handled by the InitialVC. InitialVC is a normal UIViewController. The initial page has two text fields which are country and college. CLLocacationManager delegate from the CoreLocation framework is used to get the current location of the user and populate the country and the nearby college. Clicking on search button available in the initial page will navigate to the Listing page.

- **ListingTableVC**
  The Listing page is controlled using the ListingTableVC. ListingTableVC is a UITableViewController. All apartment listings within 25 mile range of the user location are displayed in the listing page. The CLGeocoder delegate from the the CoreLocation framework is used to get the location coordinates of the listing from the address provided in the listing. Once the location is calculated, distance from user location to the listing location is calculated. The distance of the apartment listing from the current location is also shown in the ListingTable page. The user has the option to sort based on distance and the price. Clicking on a row redirects the user to Listing Detail page where all details of the apartment listing are shown.

- **ListingDetailVC**
  The Listing Detail page is controlled using the ListingDetailVC. ListingDetailVC is a UITableViewController. The Listing Detail page displays all the details about the apartment listing. The user can see the apartment location in the mapview provided in the Listing Detail page. The user can navigate the apartment location form the current location from the mapview. The user has the provision to call/sms/email the posting owner from this page. The listong can be added to users favorites from this page. The Favorites are saved using the UserDefaults.

- **PostingVC**
  Posting page is controlled using 2 view controllers. The first one is PostingVC. All the data entered in this page is sent to the second view controller which is PostingVC2. The data is entered using textbox, sliders, switches, pickerview and datepickerview. Once data is entered in this page, the user can go to PostingVC2 by clicking on the Continue button.

- **PostingVC2**
  All the data from the PostingVC is received in PostingVC2. Uploading the picture of the apartment is done using this page. Once the post button is clicked, all the data will be saved in firebase database and the image will be saved in Firebase Storage. The link to the downloadURL for the image stored in the storage will be saved as the imageName field in the database.

- **FavoritesTableVC**
  The Favorites page is controlled using the view controller FavoritesTableVC. The user can delete a listing from his Favorites. The favorites details are fetched from the UserDefaults.

- **MylistingsTableVC**
  My listings page is controlled by the MylistingsTableVC. The user can delete his listings from this page. On clicking a row, the user will be redirected to his listing. Internally the page is redirected to PostingVC. Here all the details of the listings are pre-populated and user can edit any change and post it. The existing post will be edited. The entry in the database will be updated.

- **Models**
  - **Listing**
    The Listing class stores all the data related to a listing. This class contains the constructor, getter and setter methods. Objects of this class is used throughout the application. The data related to an apartment is loaded, saved and updated using this methods declared inside the class.

- **Listings**
  Listings class contains an array of listings.
- **ListingCell**
  Listing cell is the class used for the custom cell used for the table view cell. The Listing cell contains the listingname, listingrate, distance, listingplace and and imageview.
- **University**
  The University class stores all the data related to a university. This class contains the constructor, getter and setter methods. Objects of this class is used to get and set university details.
- **Universities**
  Universities class contains an array of Universities.

6 Third party frameworks

- **Firebase:** The detailed explanation of firebase has been given in section 2 [1]
- **AlamoFire:** Alamofire is used to show images from the URL. [2]
- **Imageslideshow:** ImageSlideShow is used to show images in a slideshow. [4]
- **IQKeyboardManager** IQKeyboardManager was used to take care of the keyboard. While clicking on a textview, the keyboard was covering the textview and there was no option to the hide the keyboard. On using IQKeyboardManager, the whole view was shifted up and there was an option to hide the keyboard. [3]

The screenshots of the application are shown in the appendix page.

References

[1] Firebase. Firebase. https://firebase.google.com/docs/ios/setup.

[2] C. Noon. Alamofireimage. https://github.com/Alamofire/AlamofireImage.

[3] M. I. Qurashi. Iqkeyboardmanager. https://github.com/hackiftekhar/IQKeyboardManager.

[4] P. Zvoníček. mageslideshow. https://github.com/zvonicek/ImageSlideshow.

A APPENDIX

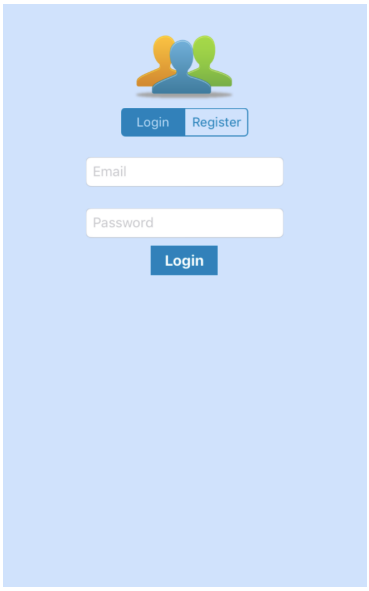

Figure 1: Login Page

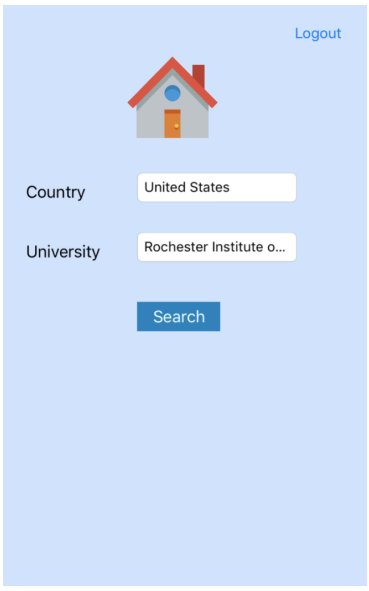

Figure 2: Initial Page

Figure 3: Listing page

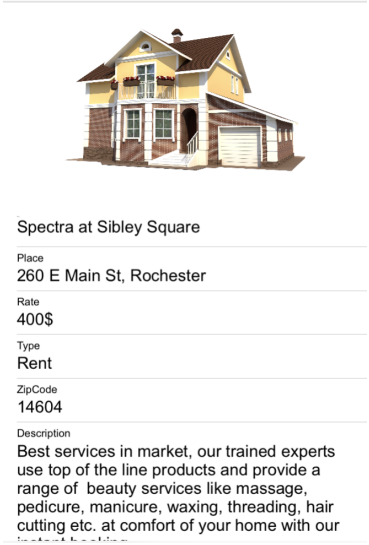

Figure 4: Listing Details (1)

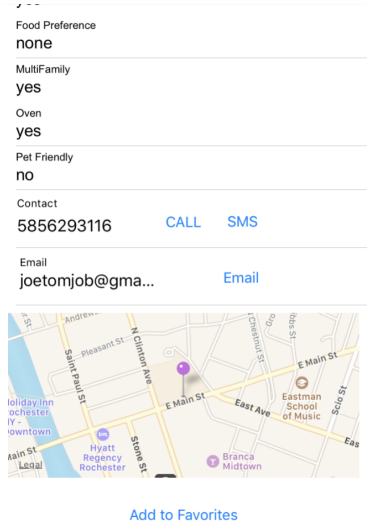

Figure 5: Listing Details (2)

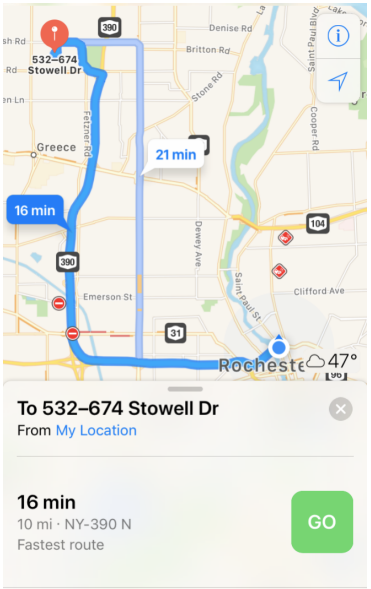

Figure 6: Navigation page from listing detail page

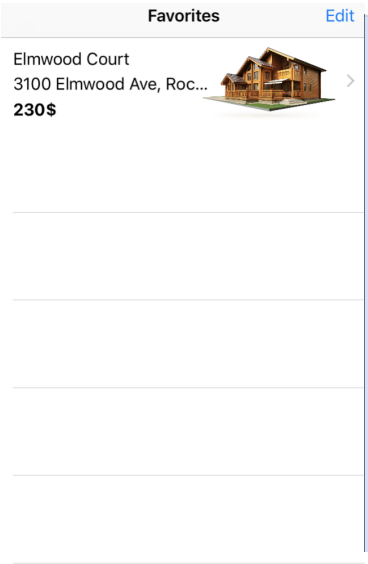

Figure 7: Favorites page

Figure 8: Posting (1)

Figure 9: Posting (2)