CollegeThrift

Computer Science Senior Project Mid Year Report

Kalyan Parajuli
Advisor: Prof. Ewa Syta

11 December 2017

1 Introduction

CollegeThrift is an android application portal in which members of an academic community like Trinity College can list their used goods for giveaway, trade or sale. Users will also be able to make posts as well as requests for some goods which they might be in need of. The users will be able to message each other on the app for inquiries and negotiations regarding the sale of devices. This app differs from other used goods selling portal available now like Craigslist, Letgo, ebay, etc. because the users and interactions in this app will be local and limited to a certain particular institution the users are part of, rather than being world-wide or location wide.

1.1 Objectives

The objective of the app is to facilitate the transaction of the goods including books among students. The goal is to make such transactions fast, hassle-free and safe. Thus, the implementation of the app is in the mobile form - which lets users to take pictures of their items and make listings easily on the go. Additionally, since the app is local, the users will not have to seek transportation medium to sell or receive the item. This can save the cost of transaction and also make it less time-consuming.

The users are verified using their college email address to add authenticity, security and safety to the transactions. Users can use the app and feel safe - knowing that the possibility of scams and bodily harm during the transaction astronomically less than in other platforms. Similarly, the app will implement in-app messaging feature for any communications regarding the transaction. Thus, no personal contact and information will be compromised in the process. This will also allow a greater control of interactions the user have - the users can avoid spam or repeated messages by just deleting any prior conversation on the app or blocking the user, which would not be possible if contacts are shared outside of the app.

Additionally, the app will use Venmo payment API to facilitate the payment.

1.2 Development Goals

The app will have signup and login infrastructure with email verification. The verified users can then view the listings. They can make listings of their items or make request for items they need. The users should be able to take pictures of the items and add it to the posts.

The users should be able to communicate to the interested parties through the in-app messaging infrastructure. They will have the control over these messages and end the conversation at their discretion. Additionally, the users will be also manage their listings or items they are interested in by using the dashboard page. They should also be able to connect their venmo accounts to the app and make the payment for the transactions.

2 Implementation

The required infrastructure has been set up using the Firebase API. I have used firebase database API for storage of listings and user profile data. Likewise, I have used Firebase Authentication API for user enrollment, verification and authentication. The Authentication API itself also provides an interface to store some meta information on the user like user's full name and email verification status. The API also handles verification and reset-password email and the process for the app. Similarly, Firebase User Module is used in the app implementation along with these APIs. The Firebase Storage API is also connected to the app for image hosting for the listings and the users.

The infrastructure setup was itself a challenging experience at first, since my first choice of infrastructure was Amazon AWS which was not very friendly for implementation. Similarly, getting the database implementation right for the app was a steep learning experience since the firebase database is a no-SQL database and also is very different compared to other no-SQL databases.

As mentioned above, the user authentication and verification modules are complete. The grid view of the listings have also been implemented. The grid view consists of the image of the listing, the title of the listings, indication of whether the item is for sale, trade or giveaway and also the price of the item, if applicable. The create view module is also complete in its basic form. The users can make listings by inputting a title, description, type and the price of the item.

The major hurdle I am experiencing in this part of the implementation is use of Camera API. This implementation itself is a titanic task and there are not many complete resources to make this work on an app. As a result, the listings or the users in the app do not have any pictures. The view of listings uses a placeholder image for the viewing.

3 Preliminary Results and Expected Outcomes

Figure 1 shows the screenshots show the singup, login and forgotten password page of the app.

The users can only sign up and sign in using a '.edu' address. Similarly, forgotten password feature is only available for '.edu' addresses only. Signing in is only available after the verification email verification.

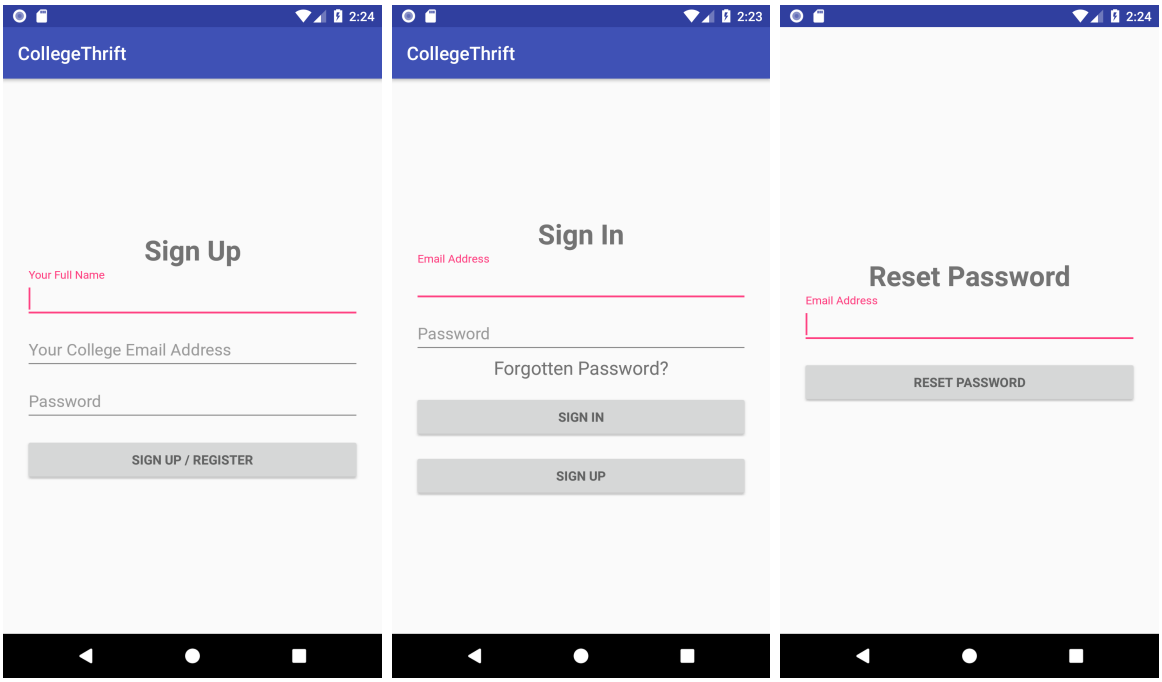

Figure 1: (i) Sign Up page (ii) Sign In page (iii) Forgotten Password

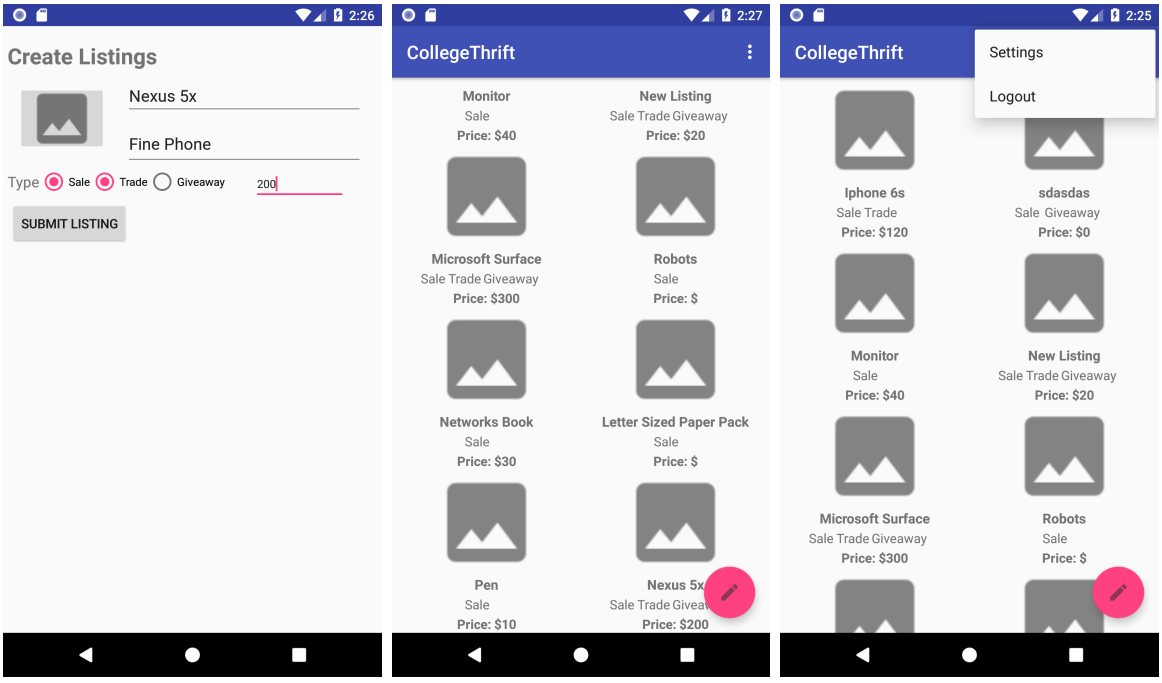

Figure 2: (i) Create Listings Page (ii) View Listings Page Showing the newly created Listing (iii) View Listing Page showing Logout Menu

Similarly, Figure 2 shows the screenshots of 'View Listings' page and the 'Create listings' page. The view listings page is the parent page of the app for the logged in users. The users can create listings using the pen icon on the bottom-right corner of the page. The users can also logout from this page.

The page dedicated to individual listings where users can see each listings in isolation is not yet done. Similarly, the Listings have no image attached to itself currently. Additionally, the listings lack search functionality and categories which is very essential for the success of the app.

4 Activity Components and Flow

Following are the different activity in the app and different components used in the app. I will give each component a distinct name so we can use those names to understand the flow of the app in the section below. The same name might not have been used in the app itself for those components.

1. **SplashScreen Activity**: This is the first activity that loads when the app starts. This activity sets up the internet connection for the app before progressing to any other activity and interaction in the app. This page sends the error message saying it could not connect to the internet and exit, if the connection us not established.
    - TextView to display the name of the app in bold, *ssAppTitle*
    - ImageView which displays a loading image, *ssLoadingIm*
2. **Login Activity**: This is the activity the SlashScreen Activity leads if the user is not signed in.
    - TextView to display the title of the page, *liHeaderTitle*
    - EditText to input user email, *liEmailTextBox*
    - EditText to input user password, *liPasswordTextBox*
    - TextView to link to forgotten password page when clicked, *liForgotPassLinkText*
    - Button to Sign In, *liSignInButton*
    - Button to Link to Sign Up Page, *liSignUpLinkButton*
3. **Signup Activity**: This is the activity which is reached when the signup button on the login page *(liSignUpLinkButton)* is pressed.
    - TextView to display the title of the page, *suHeaderTitle*
    - EditText to input the name of the user, *suNameTextBox*
    - EditText to input the email of the user, *suEmailTextBox*
    - EditText to input the user password, *suPasswordTextBox*
    - Button to Sign Up, *suSignUpButton*

4. **ResetPassword Activity**: This activity opens up when user presses forgotten password link on the login page *(suForgotPassLinkText)*.

   - EditText to input the email of the user, *rpEmailTextBox*
   - Button to submit the reset password request, *rpSubmitButton*

   The activity takes user back to Login Activity when pressed back or a when the reset password request is successful.

5. **ViewListings Activity**: This activity is the main view of the app where the listings made by the users are made visible. SplashScreen Activity leads to this activity if the user is signed in. This activity also gets started after a successful login.

   - MenuBar that contains contains the title of the app and allows user to logout, *vlMenuBar*
   - GridView to contain the listing items and manages them in a grid, *vlGrid*
   - Listing content View to wrap each listing and display them, *vlListingContent*
   - Floating Action Button to let user create a new listing, *vlCreateListingsFab*

6. **CreateListings Activity**: It is the activity where the users can create the listings.

   - TextView to display the title of the page, *clHeaderTitle*
   - EditText to input the name of the item, *clItemNameText*
   - EditText to input the description of the item, *clItemDescText*
   - RadioButtons to select whether the item is up for sale, trade or giveaway, *clSaleRadio, clTradeRadio, clGiveawayRadio*

   This page takes back to ViewListing activity on pressing back or a successful submission.

7. **Listings Adapter/Content**: This is an adapter combined with the layout which wraps every listings and feed them to ViewListing Grid View *vlGrid*

   - ImageView to display the main thumbnail assosciated with the image, *vlwImage*
   - TextView to display the title of the item, *vlwTitle*
   - TextView to display whether the item is for sale, trade or giveaway, *vlwType*
   - TextView to display the price of the item, *vlwPrice*

5 Activity and Application Logic

This section outlines what is required as input to the components of the activities and what input for a certain input block will result in a successful request.

1. **SplashScreen Activity**: This activity is the entry point of the app. This activity checks to see if the internet connection is available. If there is no connection, it will give an error message and retry or exit the app. On successful internet connection, this will start the ViewListings activity if the user is logged in or Login Activity if not.

2. **Login Activity**: The users can log into the app using this activity. The email the user provides in the *liEmailTextBox* has to be valid .edu address of the institution. Else the app will request proper .edu email address. In case of a password match and that the user's email is already verified, the login will be successful and the user will be taken to ViewListings Activity. If the user's email is not activated, it will ask for activation. If the password does not match with that of the user email, it will give and error and tell user to retry.

3. **Signup Activity**: The users can register to the app using this activity. The email the user provides in the *suEmailTextBox* has to be valid .edu address of the institution. In case of invalid email address, the app will give an error and ask user to retry. The password provided in the *suPasswordText* has to have minimum of 6 characters. The user will be asked for another password if the criteria is not met. Once the registration is successful, the app will ask the user to verify the email address by going to his/her inbox. Subsequently, it will take the user to the login page.

4. **ResetPassword Activity**: The email the user provides in the *rpEmailTextBox* has to be valid .edu address. If the email input meets the criteria, it sends the reset password email and ask user to reset the password from the email inbox. If there is no account assosciated with the email address, the app will not give any error, nor will it send the reset password error. After submission and no error, the user is taken to the Login Activity Intent.

5. **ViewListings Activity**: On clicking logout from the *vlMenuBar*, the user will be taken to the Login Activity. If the user somehow reaches this activity without being logged in, the user cannot view this activity. The user will be redirected to the login page if no logged in user is found during the check at the ViewListing Activity onStart() method call.

6. **CreateListings Activity**: The users can set any title and description to the item. The items can check any one of the *clSaleRadio, clTradeRadio* and *clGiveawayRadio* or do check the combination of *clSaleButton* and *clTradeRadio*. The price cannot be inserted at *clPriceText* if the *clGiveawayRadio* is checked. Else, it can be any float value.

6 UML diagrams

The following is the UML diagrams of the backend component of the app.

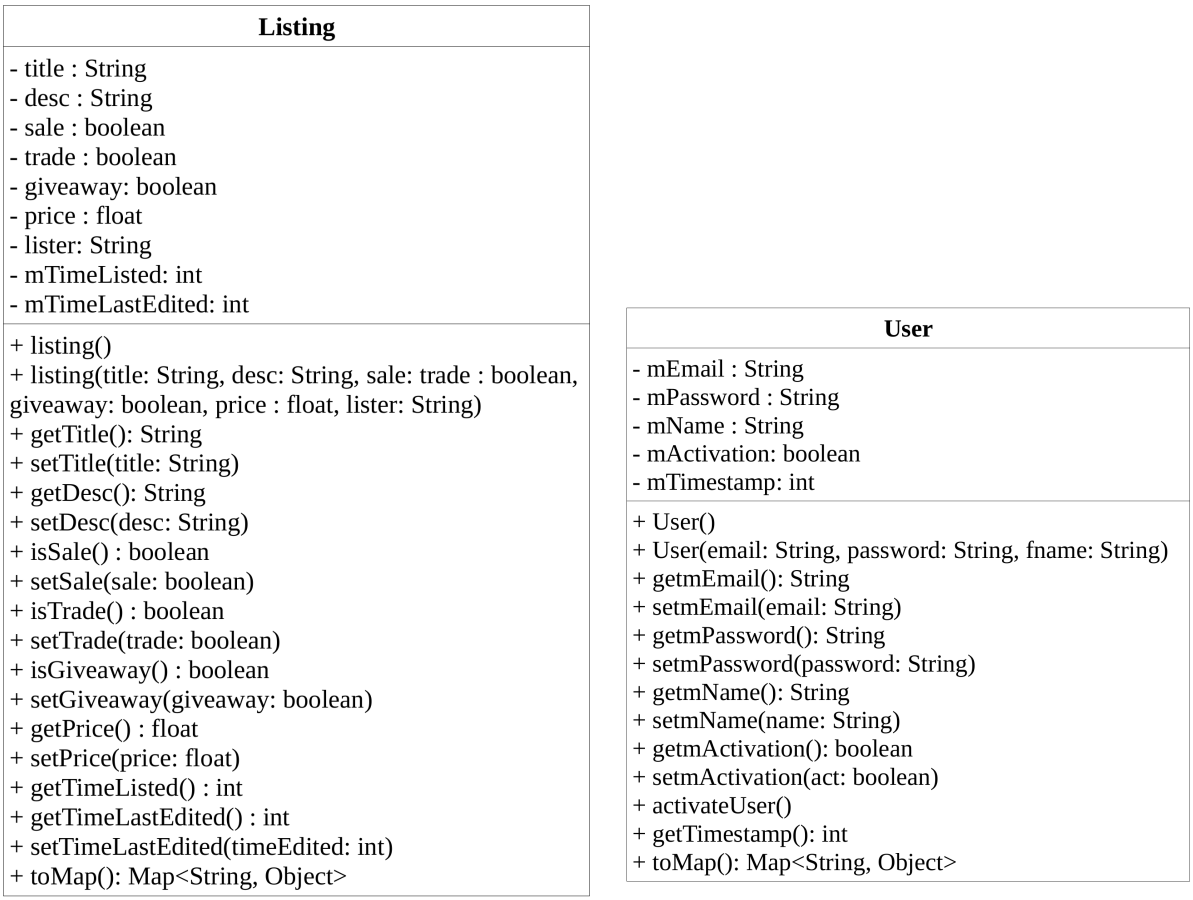

Figure 3: User class and Listing class UML diagrams

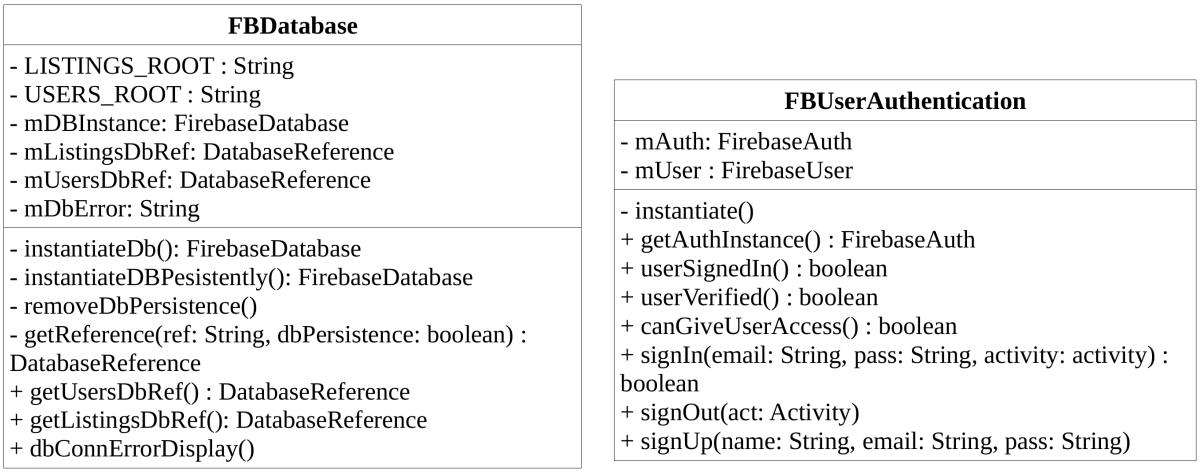

Figure 4: FBDatabase class and FBUserAuthentication class UML Diagrams

Figure 3 shows the current status of user and listing class in UML Diagram. Currently, the listing is connected to the users but not vice versa. As I look to implement the user

dashboard it will be beneficial to connect users with the list of listings they are assosciated with.

Figure 4 shows the classees used for database and authentication management. These classes act as a wrapper class for built-in Firebase database (FireBaseDatabase) and Firebase User Authentication (FirebaseAuth) API. These classes have helper methods which can be called easily for the database and authentication interactions of the app.

7 Conclusion and Future Work

The following is the projected timeline made at the start of the semester.
(MFE = Minimalistic Frond End (i.e. no effort to make it look better than required for the functionality to work)

	FALL 2017 SEMESTER
Sep 24 - Oct 08	Android Development Crash Course
Oct 09 - Oct 16	Create Listings Page w/ MFE
Oct 17 - Oct 23	Show Item Posting and Listings w/ MFE
Oct 24 - Oct 30	Show Item Requests w/ MFE
Oct 31 - Nov 6	Search Functionality for Listing and Requests w/ MFE
Nov 7 - Nov 13	User Sign-In and Binding User with Listings w/ MFE
Nov 14 - Nov 20	User Registration and Verification w/ MFE
Nov 21 - Nov 27	User Profiles and Editing w/ MFE
Nov 27 - Dec 11	User Dashboard & Linking with user activity in the App w/ MFE

The significant absences in the app from the projected features is the search functionality. Similarly, User Profiles and Dashboards have not been implemented yet. The camera functionality and pictures are missing from the listings and users. All other functionalities are functional, but would definitely require some polishing - which will be done at the refactor phase of the project later in the spring semester.

In order to make up for the lost time, I will omit some nice-to-have features from the project like Venmo API integration, if required. Similarly, I have allocated some time in the spring semester for tags and filter functionality which can be coalesced with search functionalityin some manner to reduce the time required. Similarly, the projected timeline for next semester has some allocated time for front end design - hence, I could make a reasonable compromise on the time spent on the layout and display of the app and adding the functionalities overdue this semester.

Obviously, making up for these delays needs a good understanding of my status with different tasks. I definitely should learn more about the APIs and difficulty of implementation of these features - which I plan to give some time during the winter break.