

Distinction Experience Report

COS30017 - Software Development for Mobile Devices

Daniel Parker 971328X

November 20, 2014

1. Introduction

Perspective: Design Communication Perspective

This document covers the experience of writing a non-trivial Android application from the perspective of communicating the design of the app, including class diagrams, navigation model and data model. It also covers limitations of API used in developing the program, as well as a short section on the experience of using RAPPT for prototyping during initial phases of development, and how that affected the continued development.

2. Project Structure

The application is structured in part as per the Android standards, and also by how RAPPT has generated package structures and various utilities such as RESTful API client, error dialogs, listview and list adapters.

2.1. Resources

Resources are separated into;

- anim
- drawable
- layout
- menu
- values

There are different styles specified for API 21 than API 15+ due to the need to use the AppCompat themes and libraries for backwards compatibility of Material Design components, visual and navigation design patterns.

2.2. Source

Java source is separated as suggested by the output of RAPPT into the following packages;

- activities
- adapters
- fragments
- interfaces
- model
- views

Some application level source files are in the root of the java source to indicate their global importance to the project.

2.3. Libraries

Most libraries are included by gradle at compile time, however one library needed modifications, and has been packaged as a jar archive and added to the libs directory and as a gradle dependency.

3. Program Structure

Much of the app is structured in the ‘predefined’ manner which is mostly dictated by the Android libraries and framework. These structural aspects can mostly be observed in the class diagram, and are highlighted by the inheritance of one of the green coloured Android classes by one of the app’s classes. Other important structural parts which aren’t shown in the class diagram due to their complexity are;

- Activities/Fragments which contain listviews in their layout will utilise one of the Adapter classes to map a Parcelable model to the correct View objects in the Activity.

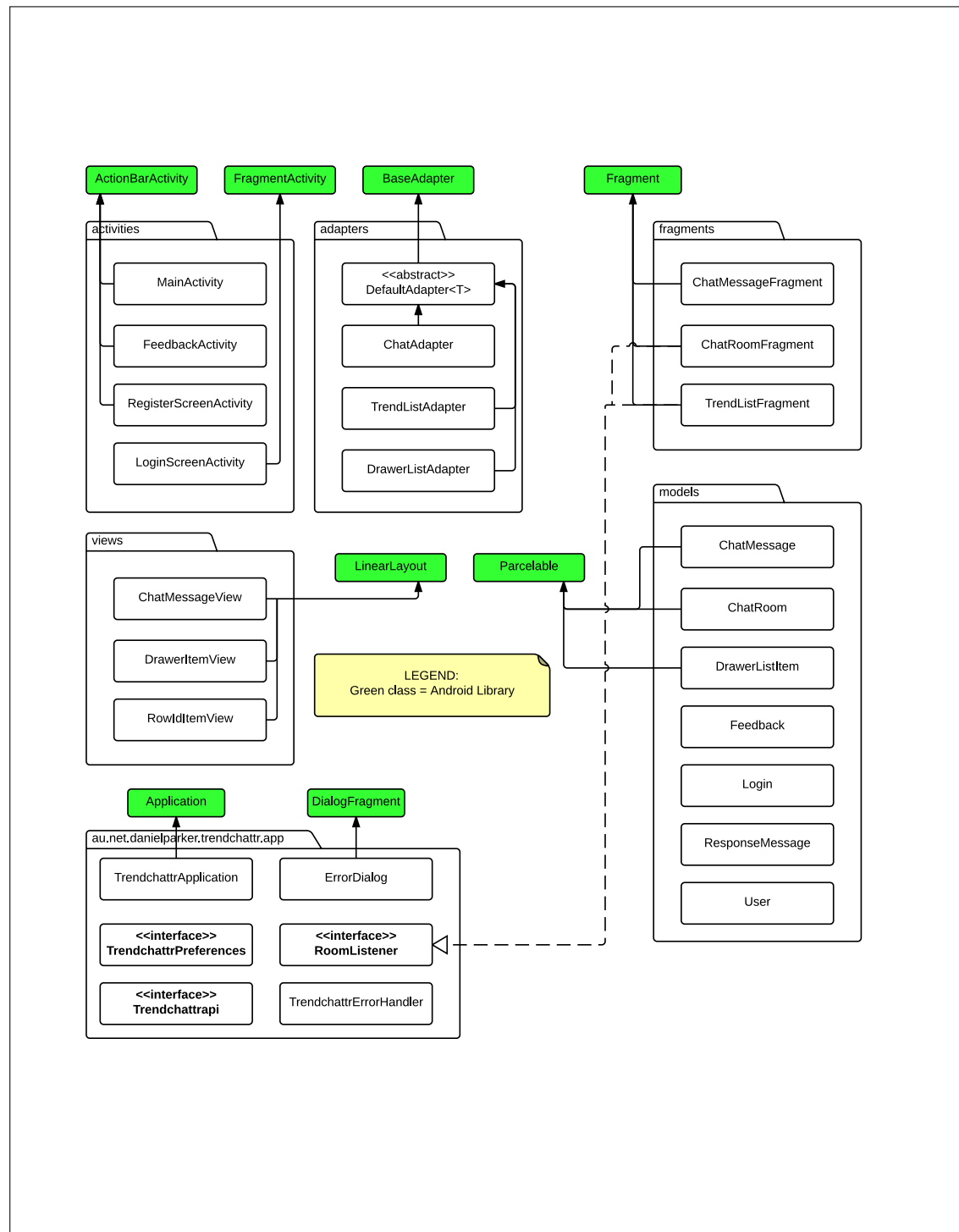
- Activities instantiate Fragments and then use the FragmentManager and FragmentTransactions to manage what fragments are in view, how they are animated onto and off the screen, and whether they are placed onto the backstack.
- The TrendchattrApplication class contains a socket.io service which will notify all registered classes that implement the RoomListener interface that a new message has arrived. It also manages the shared preferences, sending of chat messages and contains the TrendchattrApi object which executes all the RESTful API calls.

Data management and flow is particularly interesting for trendChattr in that it requires both RESTful API calls and instantaneous asynchronous chat communications. This is achieved by using a Node.js server which manages user and trend data in a Mongo database, and has a small part which manages the connecting and broadcasting of clients to socket.io rooms. The app connects to this server using the Retrofit library for REST requests (ie. Registration, Login, Chatroom list), and using the socket.io-client java library to respond to and emit socket.io events and enabling instant messenger chat communication.

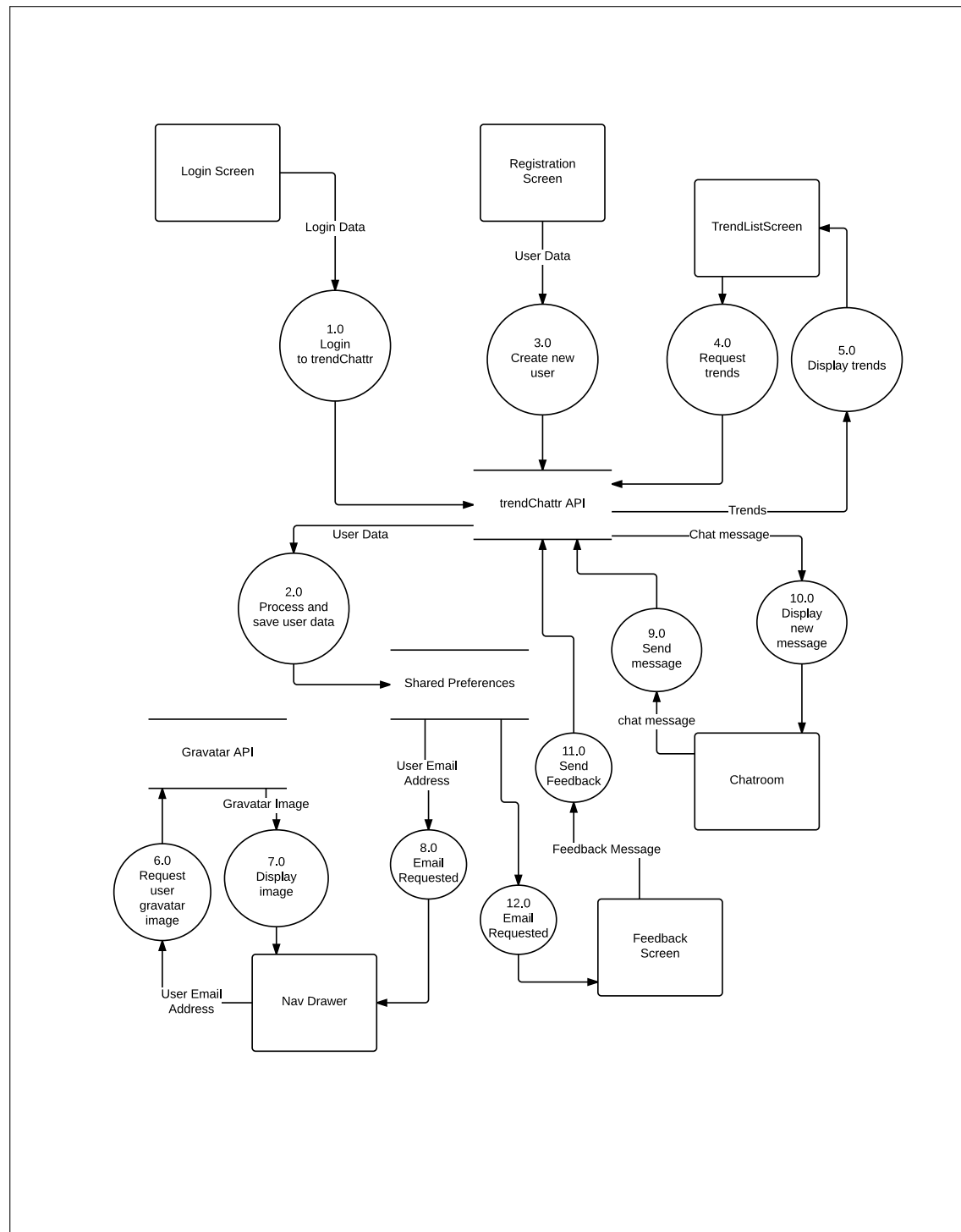
Additionally, upon logging in the server will require that all requests are accompanied by a security token which is generated and sent back as a response to the login request. Retrofit manages the inclusion of this token to all the relevant API calls by adding it to the HTTP header, and the server will ensure that the user is authenticated to perform the tasks that they try to. Tokens only become invalid if the user logs back in again, as the token will be regenerated and the old one lost.

NOTE: *The trending topics are sourced from Twitter and are hard-coded to be relevant to the Melbourne region. Due to the Twitter API limiting the frequency of requests to this endpoint, the Node.js server will update the trends itself using a cron job and always supply to the app trends relevant to the last 5 minutes.*

3.1. Class Diagram

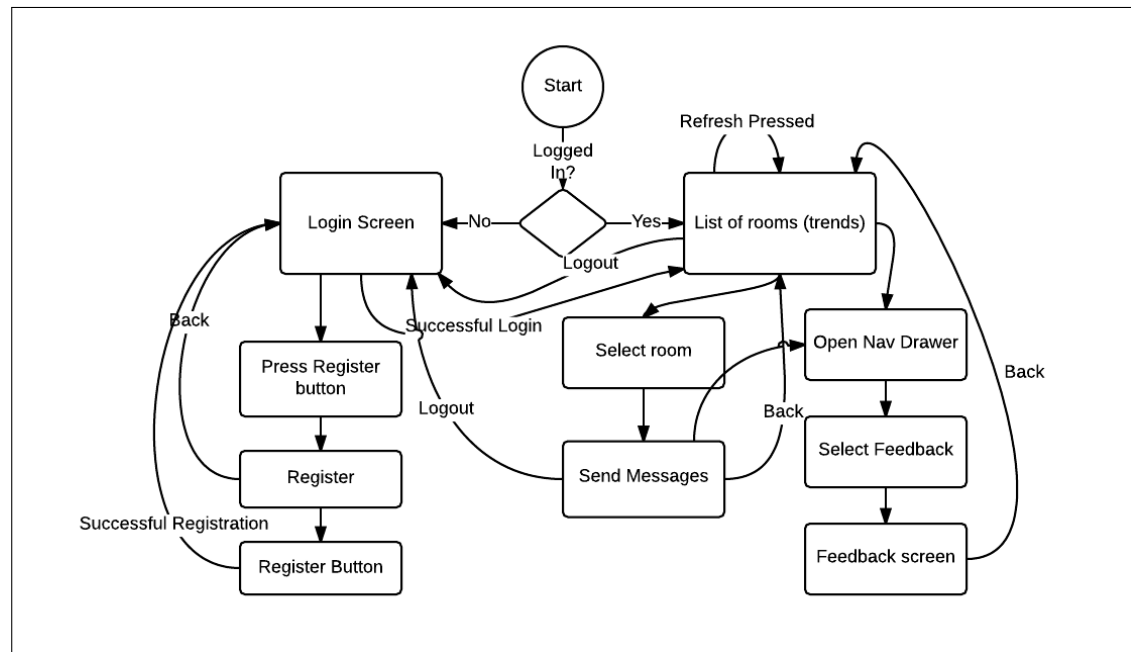


3.2. Data Model



4. Visual Structure (Navigation)

4.1. Navigation Flow Diagram



5. API Limitations

trendChattr utilises a few different library APIs, most supplied by the Android stack, and some third-party. The Android libraries while mostly stable and fully featured, do have the issue of compatibility. Unfortunately due to the requirement to have support back to API version 15, it was not possible to use the normal android libraries. Instead, the Android support libraries and appcompat libraries were used to allow for Material-like design and common navigation and visual functionality on devices running API 15+.

This caused a variety of development setbacks as some of the Activities were written before realising this, and therefor required rewriting and sometimes complete restructuring to convert them to using the appcompat libraries. Some key areas which seem to be affected by this are, styling, action bars, navigation drawers, fragments and lists.

The third-party libraries used include;

- Retrofit
- Android Annotations

- `socket.io-client`
- `cardview`
- `circularimageview`
- `jgravatar`

Most of these libraries are very stable and maintained which makes them very easy to work with. However `socket.io-client` and `jgravatar` do not seem to be heavily supported even though they do work out of the box. I would argue that in the case of `jgravatar`, the fact that it's not supported is of minor importance, as the library is trivial and due to it's open source license can be modified and maintained easily by me. In the case of `socket.io-client` it's not as simple as self-maintaining it and that is mostly due to the relative volatility of the `socket.io` server versions of which it needs to remain compatible with. The library is also a non-trivial program which could take quite a lot of time to learn. Both of these aspects combined with how crucial that library is to the chat feature in `trendChattr` makes it an issue to continue using.

6. Retrospective and Considerations

So far I am very happy with the progress I have made and the speed in which I've made it in. This is partly because of the initial gains from using RAPPT to prototype the app, but also due to the relative ease of using the libraries that RAPPT uses, Retrofit and Android Annotations, as well as the availability of the `jgravatar` and `socket.io-client` libraries. I only had to modify the source of `jgravatar` to allow me to get gravatars larger than the 512px side width maximum enforced by the library.

In general, I am happy with the structure of the app, though I would like to spend some time going back through each Activity and ensuring the consistent use of Android Annotations versus the manual procedure for certain tasks such as handling options menus. The source also lacks proper commenting and that is next on my list of things to go back and fix as I intend to continue developing this app with other developers.

There are some things which I will need to consider in future versions of the app. I need to revisit whether `socket.io` is a good solution to the implementation of instant messenger chat, and whether there are solutions that will give better support to mobile devices, in that they integrate push notifications and chat history etc. I also want to spend time researching ways to effectively show chat messages from potentially hundreds or thousands of people in a single chat window. If this is possible and can be done nicely then it may revolutionise the way that we use chatrooms in the future. I am not convinced that the old IRC way of showing chat messages is the best way to do it, especially not for chatroom based conversations.

7. Summary

In summary, the app has been designed mostly in line with the conventions and requirements of Android development in general, and also has had influence from the RAPPT generated prototype. Data-flow has been an interesting problem to solve with this app, and the solution that was found seems to work quite for the current usage levels. The experience has in general been very positive and has inspired me to continue developing this app and also an iOS and web application.

8. References

1. ralfebert 2012, *jgravatar(v1.0.0)*[Programming Library], viewed on 19 November 2014, <https://github.com/ralfebert/jgravatar>.
2. Banes, C 2014, *appcompat v21: material design for pre-Lollipop devices!*, Chris Banes, viewed 18 November 2014, <https://chris.banes.me/2014/10/17/appcompat-v21/>
3. *Creating a Navigation Drawer*, Developers, viewed 19 November 2014 <https://developer.android.com/training/implementing-navigation/nav-drawer.html>
4. Ricau, P 2013, *Android Adapter Good Practices*, viewed 19 November 2014, <http://www.piwai.info/android-adapter-good-practices/>

9. Appendix 1 (Planning and Design Document)

Custom Program

COS30017 - Software Development for Mobile Devices

Daniel Parker 971328X

November 4, 2014

1. Ideation

1.1. Rationale

I would like a way for people to be able to have instant messenger conversations around a particular topic whether that be a socially trending topic or just people in a common location such as a football game or lecture.

1.2. Vision Statement

Young adults today are socially active at all times and want to have in-depth conversations around a currently trending or popular topic. trendChattr is an instant messenger app which allows users to communicate based on a currently trending topic on social media or on room local to them. Unlike current social media platforms, which are geared towards one person stating their opinion and less on the discussion that follows, trendChattr will allow for very in-depth and focused discussion around a topic which will allow for true and extended input from all users.

1.3. Features

trendChattr will allow users to register a user handle much like on Twitter which they use to login to the app. The users can then select from a list of trending topics and local trends those which they would like to chat about and chat with other users inside those topics. Users will be able to create their own rooms based on their location and will be able to share a room on social media for others to get a link to it and join them.

1.4. Classification

Social

2. Exploration

2.1. Scenarios

1. As a social university student, I want to chat to my peers about the lecture topic during the lecture without disturbing the class.
2. As an aspiring politician, I want to have deep discussions with other Q&A viewers during the show's airing.
3. As a young person, I want to be able to talk to people outside my direct social network on certain topics as well.
4. As a young adventurous urban dweller, I'd like to be able to chat to others in the same bar that I'm at so that I can find new friends.
5. As an avid AFL fan, I would like to be able to chat to others about the game while it's being played.
6. As a news columnist, I would like to be able to create spaces for people to chat about the news that I'm reporting and to get their input on the issues at hand.

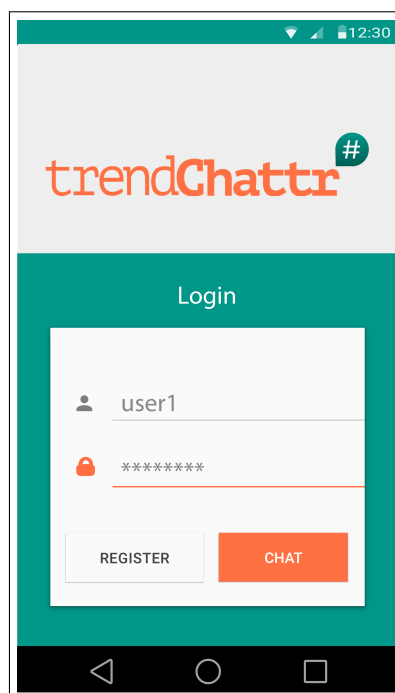
2.2. User Stories

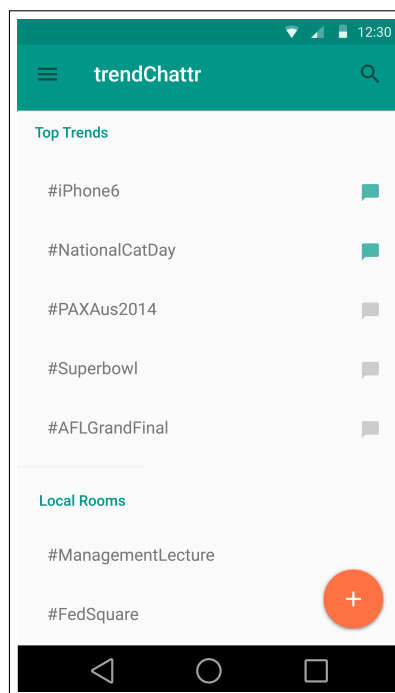
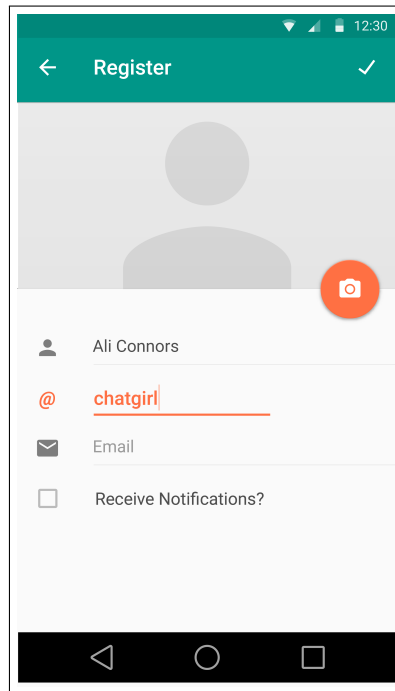
1. As a user, I would open the app and login but only the first time.
2. As a user, I would like to see a list of suggested trend chatrooms as the main screen.
3. As a user, I would like to create my own room for my current location or another location with a geographic radius of my choice.
4. As a user, I would like to see the other users usernames in the chatroom.
5. As a user, I would like to send messages to other users and see both mine and their messages in chronological order in the chatroom.
6. As a user, I would like to be able to register an account upon opening the application for the first time.

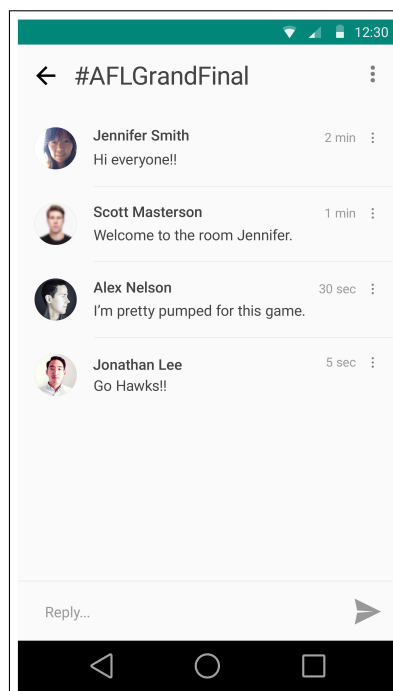
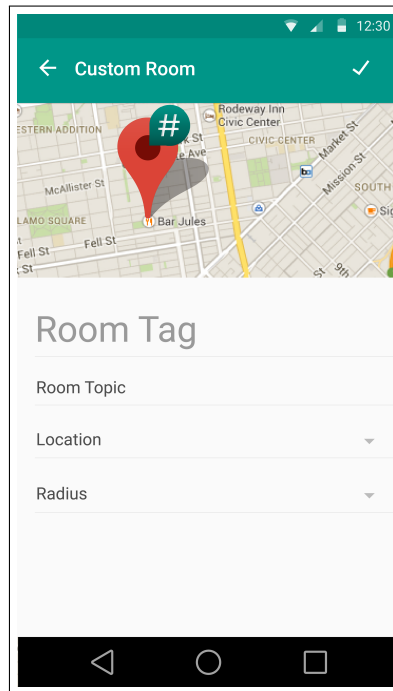
2.3. Constraints

- Chat communication over the internet
- Login as user (individual login)
- Custom rooms
- Rooms based on social trends
- Room sharing
- Location based rooms

2.4. UI Designs (Illustrator)



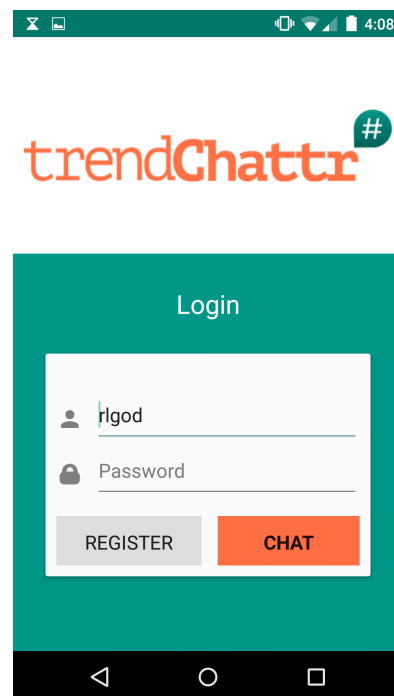
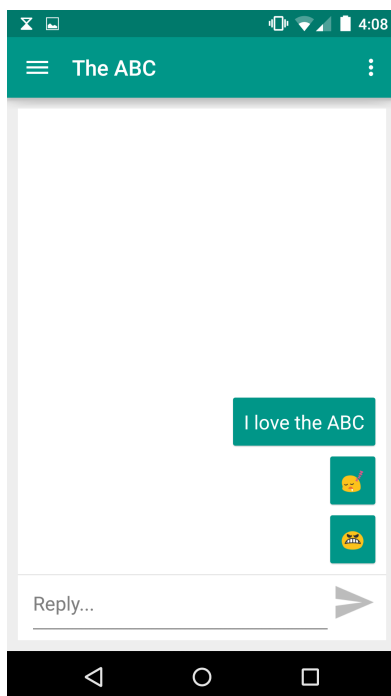
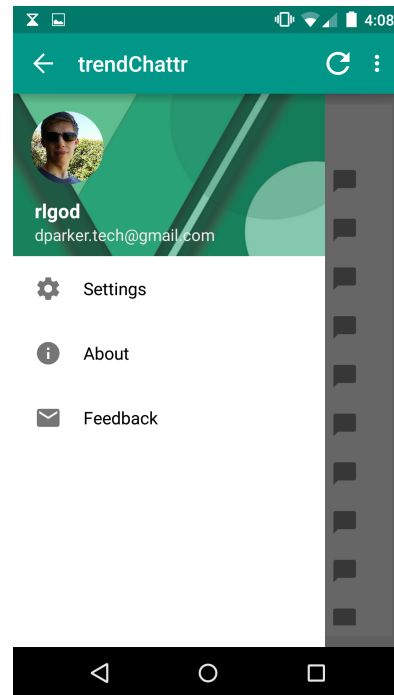
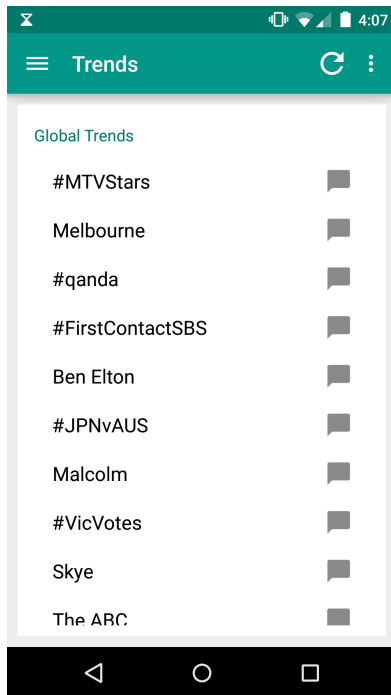


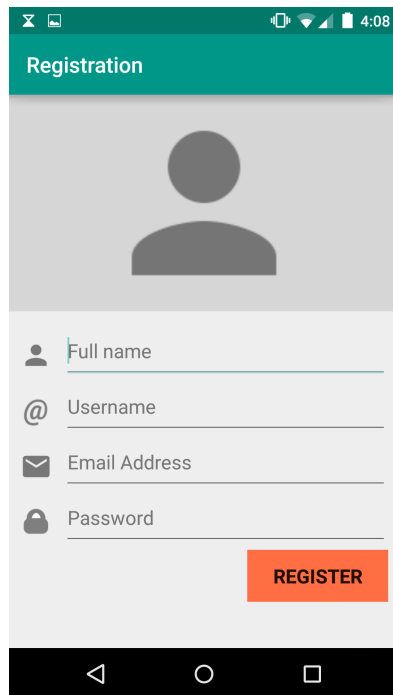


2.5. Questions

- Do the users link their accounts with twitter?

10. Appendix 2 (App Screenshots)





Registration

4:08

Full name

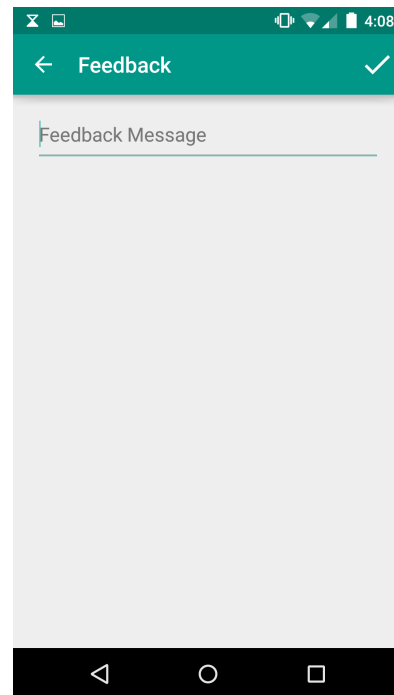
@ Username

✉ Email Address

🔒 Password

REGISTER

This is a mobile app registration screen. It features a teal header with the title 'Registration' and a status bar at the top showing the time as 4:08. Below the header is a grey placeholder for a profile picture. The main content area contains four input fields, each with an icon on the left: a person icon for 'Full name', an '@' symbol for 'Username', an envelope icon for 'Email Address', and a lock icon for 'Password'. An orange 'REGISTER' button is positioned at the bottom right of the form area. The screen is framed by a black Android navigation bar at the bottom.



Feedback

4:08

Feedback Message

This is a mobile app feedback screen. It features a teal header with a back arrow, the title 'Feedback', and a checkmark icon. The status bar at the top shows the time as 4:08. The main content area is a large grey rectangle with a 'Feedback Message' label and a horizontal line indicating where to type. The screen is framed by a black Android navigation bar at the bottom.