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
- layou
- 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;

- activites
- 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.

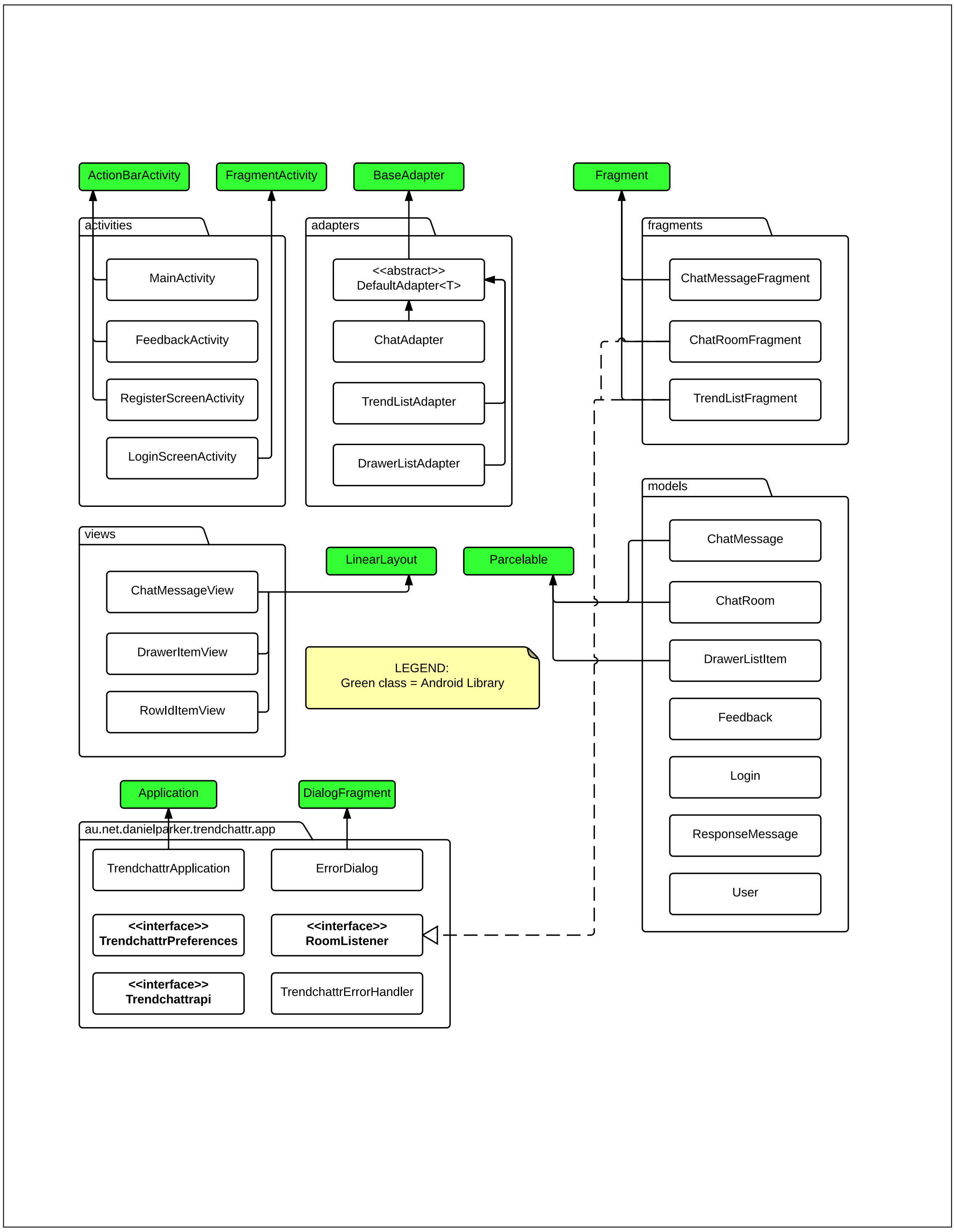
- Activities instantiate Fragments and then use the FragmentManager and FragmentTransactions to manage what fragments are in view, how they are animated onto and off the screen, and whether they are placed onto the backstack.
- The TrendchattrApplication class contains a socket.io service which will 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.

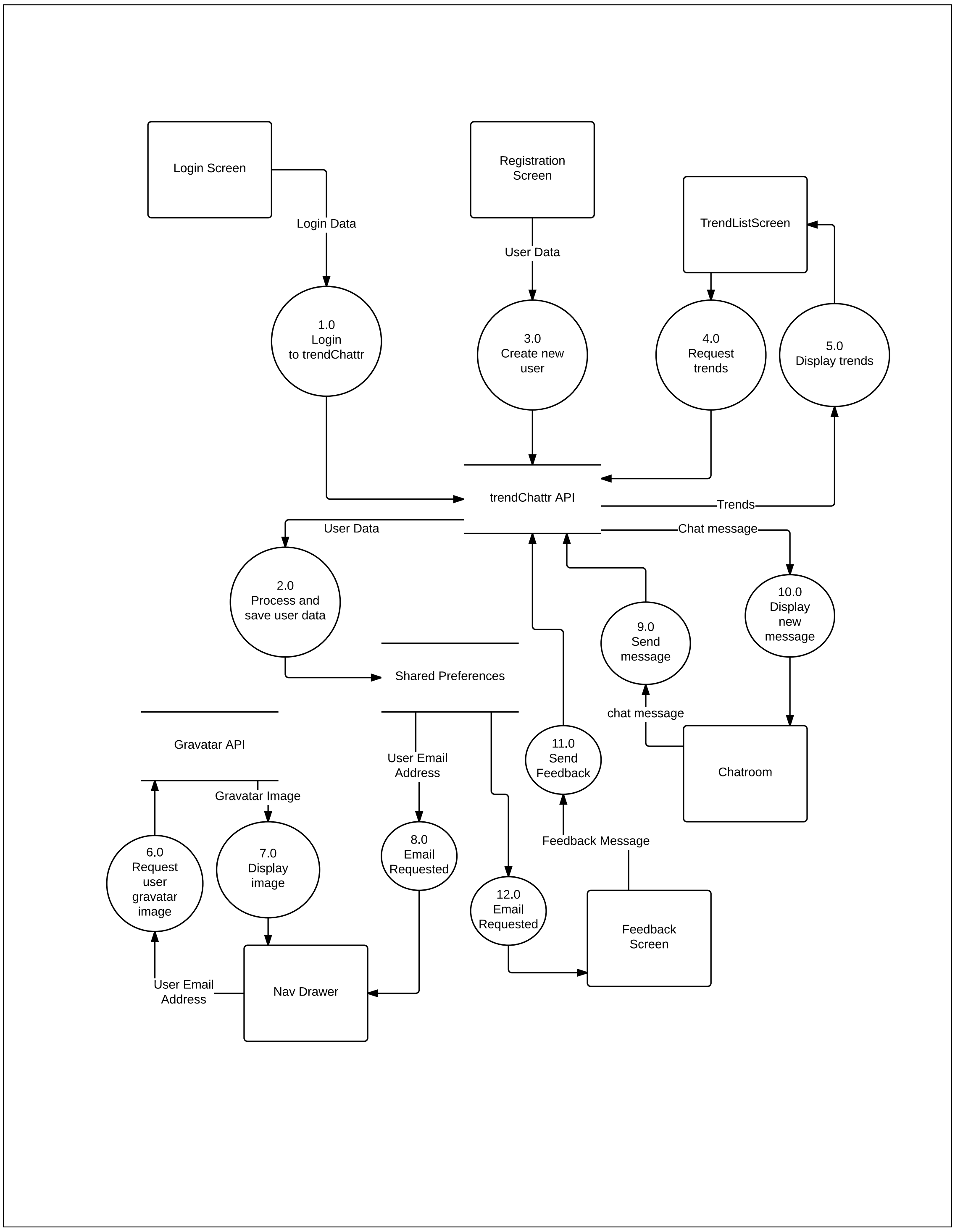
Additionally, upon logging in the server will require that all requests are accompanied by a security token which is generated and sent back as a response to the login request. Retrofit manages the inclusion of this token to all the relevant API calls by adding it to the HTTP header, and the server will 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

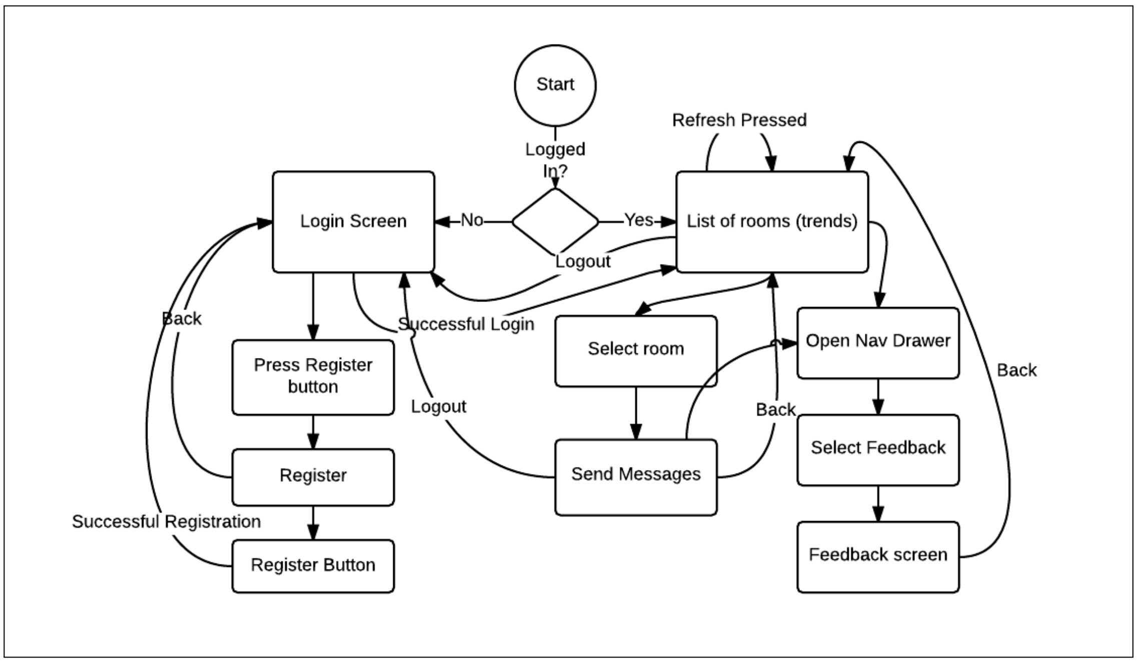


3.2. Data Model



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. Summary

7. References