LiveWire Technical Manual

CA326

Andrew Cullen, Daniel Rowe, Eoin McKeever

6th March 2020

0. Table of contents

1. Introduction

1.1 Overview

The product is an Android app which notifies users of upcoming concert events and ticket releases in their local area. It's aim is to act as a social platform for users to share their tastes and experiences by giving them the ability to rate events that have happened and let their followers know they are to be attending specific events.The project was designed so users could discover new music from their prior taste and from the people they choose to follow. Using the Spotify API we will pull a user's music preferences once they link their own Spotify account to their LiveWire account. Using this information taken from their Spotify account it helps our recommender make music and artist recommendations. Each profile will have a unique scrolling feed based on what recommendations the recommender provides. Each profile will also have their own customizable profile page where they can share their tastes and preferences. The concert recommender will use the ticketmaster API to find concerts and gigs in a user local area the listings are further filtered by their preferences. The app will have 4 main pages. They are the main scrolling page, the user profile page, the feed based on a user location and a settings page.The scrolling pages consists of listings of different artists. The profile page is the users unique page that they customize themself. Finally the settings page will be where the user can change their preferences and share their location with the app. They can also link their social accounts to their LiveWire account here. The system stores users profile details and login credentials by using the service Firebase. Which safely stores user data once linked correctly. The project is made with Android Studio which makes linking these third party services quite convenient.

1.2 Glossary

API - Application Program Interface. A set of protocols to mandate how software components (usually in a client-server relationship) interact. In essence, a way to access some resource in a restricted/structured manner.

System - Defined very loosely as a set of functions, procedures, "things" and inputs interacting together towards some common goal.

Cloud Functions- Google Cloud Functions is a serverless execution environment for building and connecting cloud services.

Location Based Services - Services that make use of a user's location to provide some functionality. For example, using a phone's GPS to set their standard language on startup.

Recommender system - A way to filter information to match with some criteria that fulfills a user's expectations. In the context of LiveWire, this would be the filtering of concerts to find ones the user would enjoy going to.

Recyclerview - Gives a fixed sized window to load a large dataset into.It recycles the views it created at the beginning when the views go out of scope.

Scope - the extent of the area or subject matter that something deals with or to which it is relevant.

1.3 Starting design vs Finished design

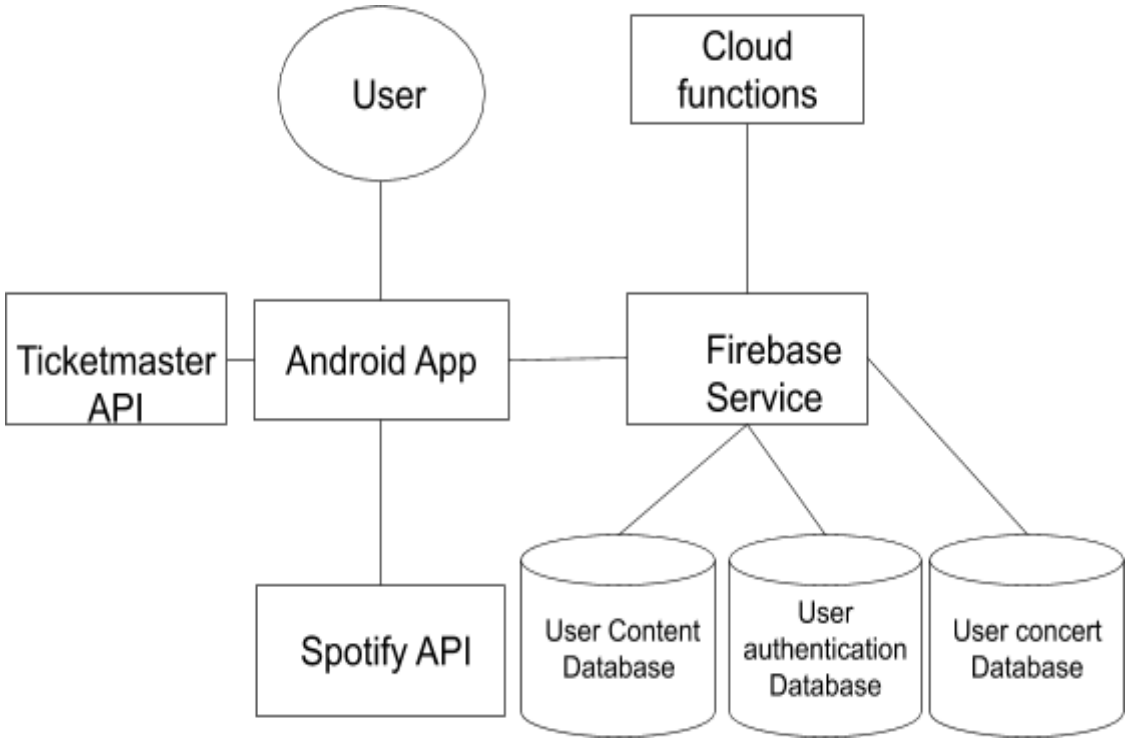
The project that we developed contains the core functionality that was specified in the functional spec. Users can make an account with us where their profile will be stored in our database. They can edit their page and scroll their feed and receive notifications and link their 3rd party accounts. Due to time constraints a follower system did not get implemented. We did not get to use the Facebook Graph API as we specified in our functional spec due to change in policy by Facebook we did not predict. We decided on the ticketmaster API but this cost us some development time. The time constraints on this project forced us to scrap a few features they include Facebook Profile Linker, Featured Song from a Concerts Artists,Follow User Feature, Concert Interactions ratings,Song and Playlist Sharing. We prioritised the implementation of the key features that are necessary to effectively demonstrate the app. Based on our work we have done throughout the year we believe if the given deadline was 1-2 weeks longer the app would have been finished in its totality.
Below is a list of system functions compared from start design to finished.

System Functions	Degree of compliance
Sign Up	Done
Sign in	Done
Concert Listings	Done
Link User Data API(s)	Done with Partial Re-designed
Profile Customization	Done
Posts	Done
Notifications	Done
User Feed	Done
Filtering Concert Listings	Partially done
Rudimentary recommender system	Partially done
Facebook Profile Linker	Not done(Due to terms of use issue)
Featured Song from a Concerts Artists	Not done
Concert Interactions ratings	Not done
Song and Playlist Sharing	Not done
Follow User Feature	Not done(relied on Facebook API)

2. System Architecture

This section describes the high-level overview of the system architecture showing the distribution functions across system modules shown using the following diagram.It consists of a user interacting with the android app. The app communicates with three 3rd party services : Firebase Service,Spotify API and TicketMaster API. The app will query the API's and pull account data. Also the app works with Firebase services to store data created from the running the app i.e user login details, user created content and user concert listings. Firebase services also include cloud based functions which updates user feed when new listings every night, when user signs up and links Spotify account finds recommended artists with help of cloud functions and spotify's API and links listings to user notifications and updates user listing feed as account develops.

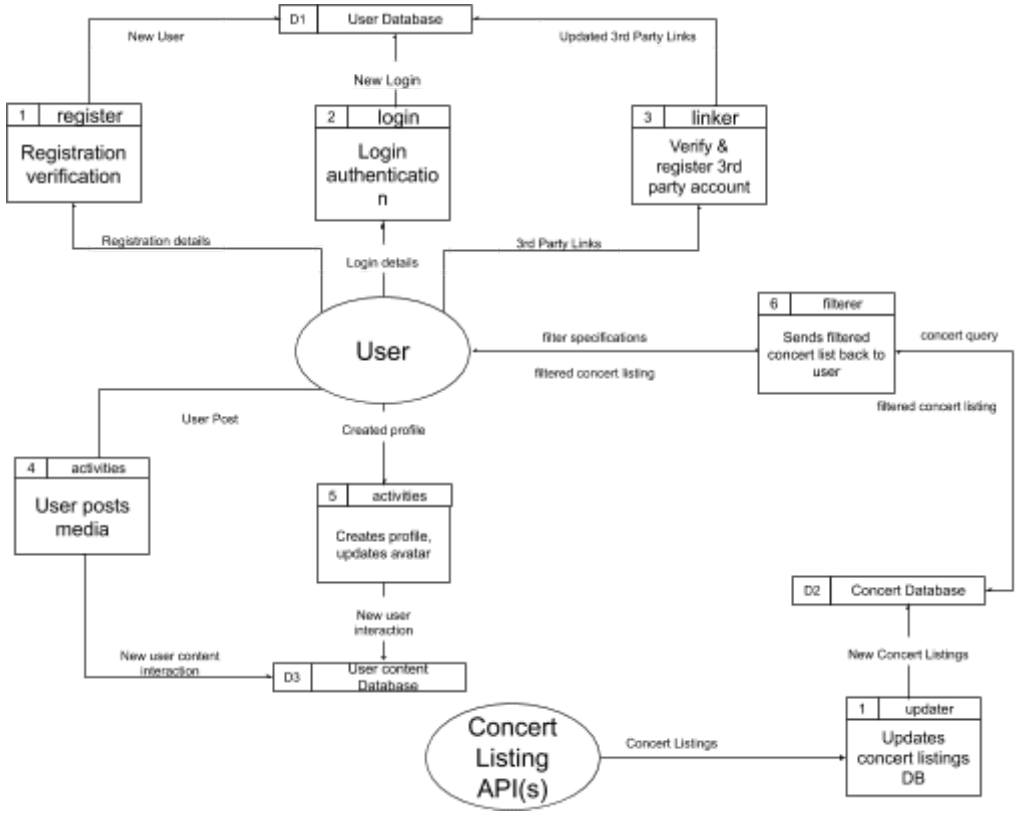
2.1 System architecture diagram



3. High-Level Design

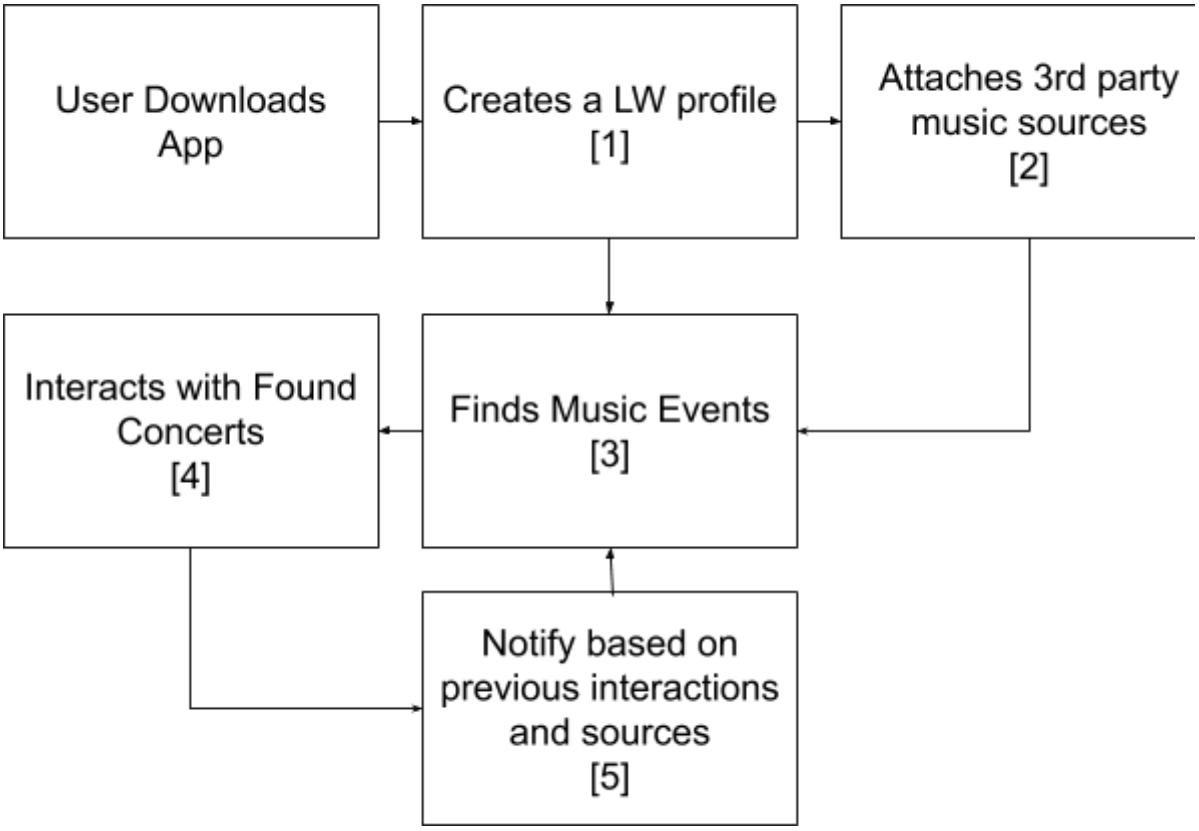
3.1 Data Flow Diagrams

This section consists of a data flow diagram that covers at a high level how the system interacts. The user interacts with D1 in three ways firstly a new user can register an account and store new login details in the database D1. Once stored the user can now enter these login details and the login details are compared to the database to see if they exist within the database. The user can also link their 3rd party accounts to the app and these can be saved to the database also. Once logged in a user gets filtered concert listings that comes from D2 the concert database which is filled by ticketmaster API these listings are filtered and sent back to the user. The user may with a created profile update their avatar and have it saved to D3 the user content Database. The user may also post to their page and have it stored in D3



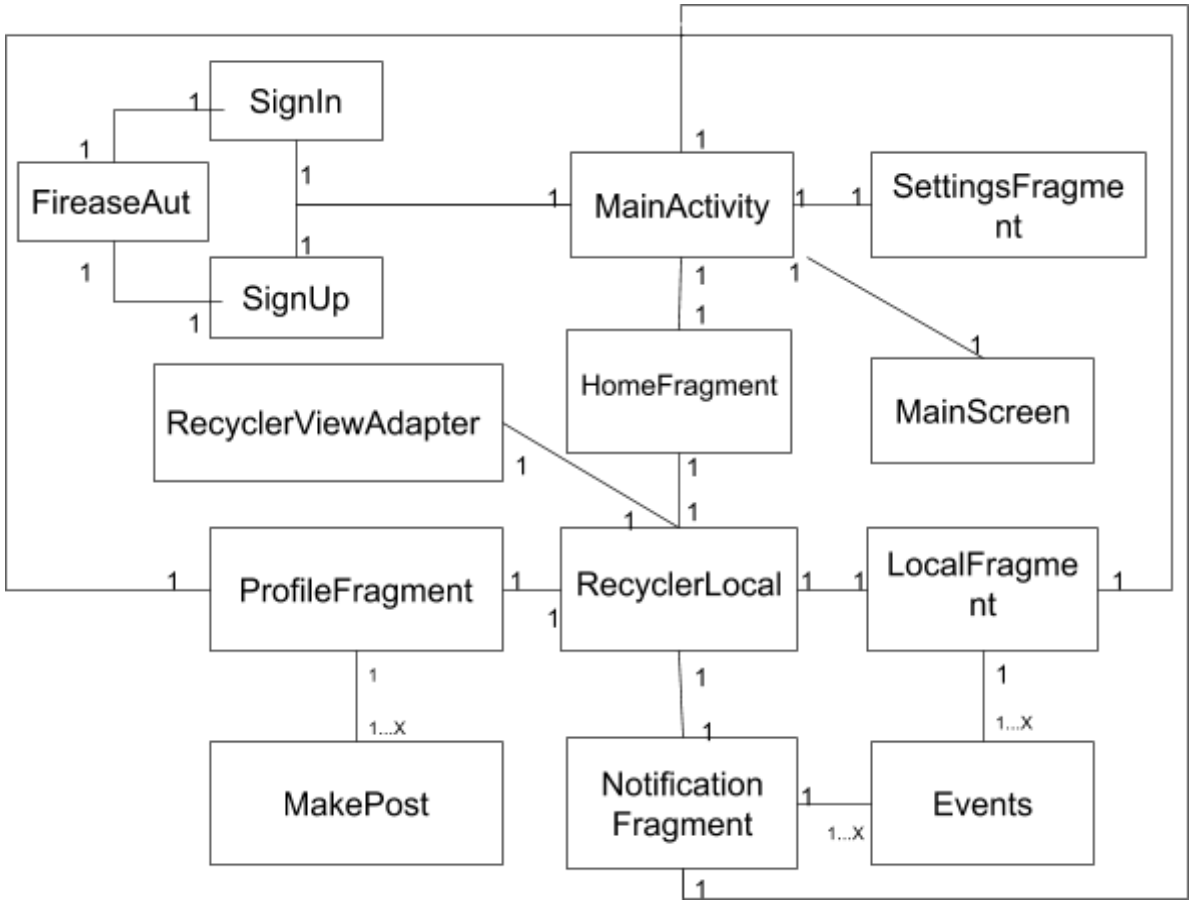
3.2 User flow diagram

This diagram shows the user flow. When the application is installed the user can make a LW profile from here they have the option to link 3rd party music sources i.e Spotify if they choose to they can continue to find music events they can also choose to not link and attempt to find music events. They will not receive an event feed tailored to their preferences this way. Users can then scroll through their feed of music events and will receive continuous notifications based off their sources and interactions.

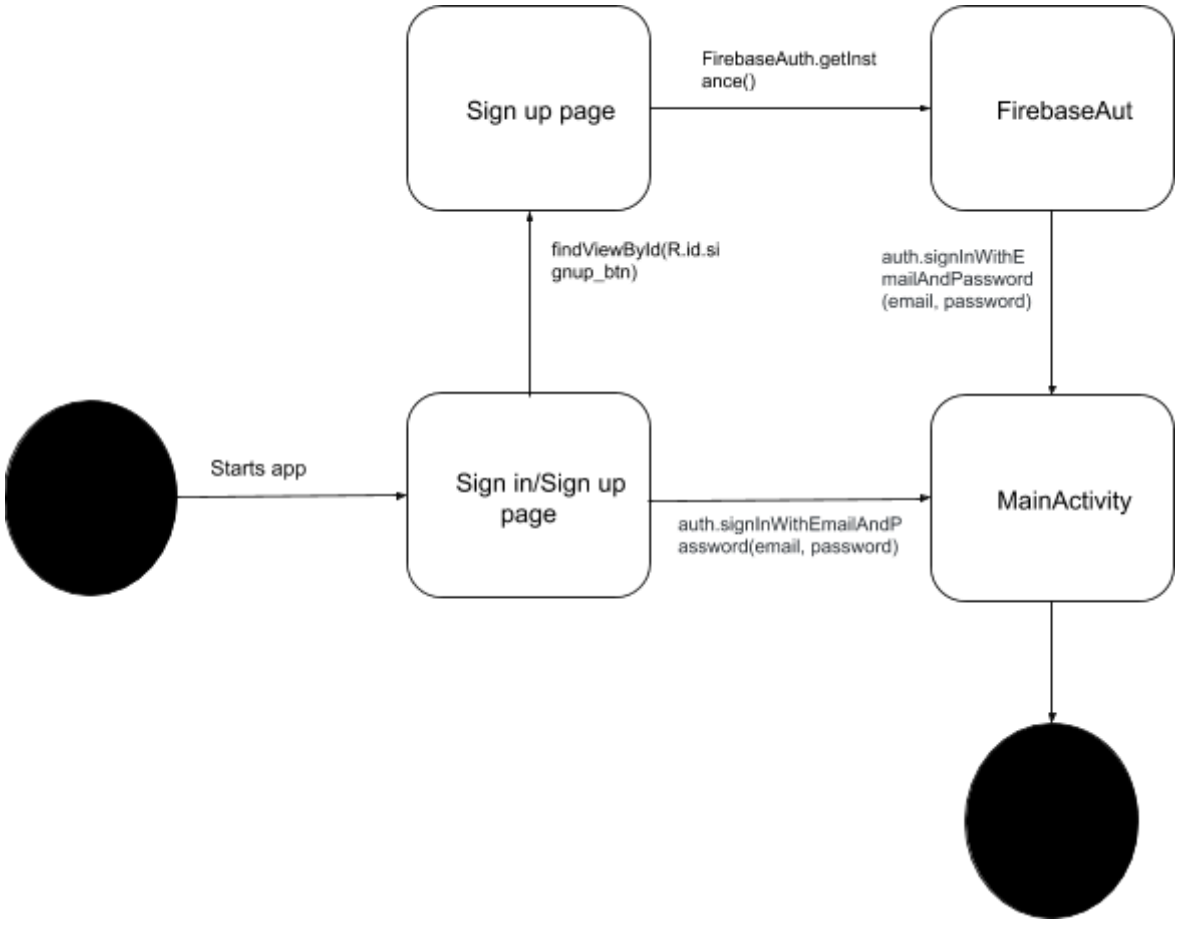


3.3 Class Diagrams

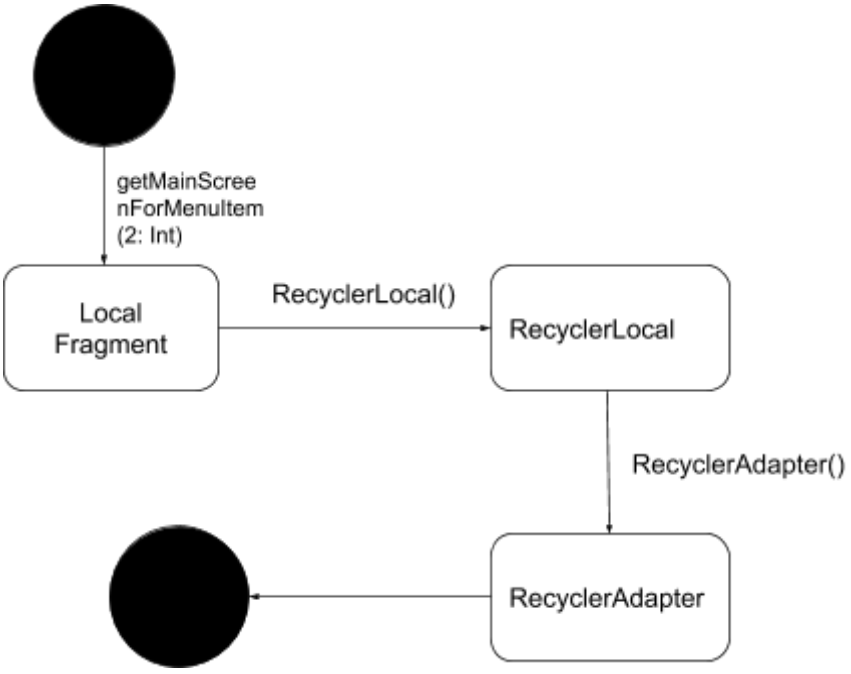
The following diagram shows classes and their relationships.



3.4 State Diagram Sign in/Signup



3.5 State diagram Local page scroll



4. Problems and Resolution

Throughout the design of this project we encountered problems like any project does.I will go into 4 of the major ones. Firstly our attempted use of the Facebook API,using FIrebase cloud functions, bugs due to overload of data and using certain features in firebase and firestore which have certain features still in beta.

4.1. Facebook Graph API

One problem was the use of the Facebook graph API. We found later in the project that the use of the Facebook API came with certain conditions that Facebook specified but we were not aware of. This meant we were forced to find another way to suggest listings in a users local area and to find new friend listings.

In future projects we will use a mix of different API's to achieve a similar effect to what Facebook API supplies. But due to time constraints we chose to use the Ticketmaster API.

4.2 Using Firebase's cloud functions

Another issue we ran into was using the Firebase in-built cloud functions. We found that on the the level of Firebase plan we had available to us we could not use these in-built cloud functions in association with our API calls because we did not own the required version of firebase.

In future projects we will avail to biting the bullet and paying to upgrade our FIrebase account to a premium plan.

4.3 Crashes due to overload of data from database

When we introduced data into our scrolling feed we were met with a crash due to the app trying to load all the data we had into the feed.

In future projects we will make sure to firstly implement a recyclerview that acts as a block to stop the app continually loading data and only loads data when requested. Which is what we ended up doing.

4.4 FireBase/Firestone beta features

We chose to use FireBase and FireStone along with their included features when dealing with data in our application. SO because of this we made the decision to work with some of their included features some of which are still in Beta version.

In future projects we will search for an alternative solution to avoid using these features by researching other services that supply these tools.

As expected from a large group project we ran into some issues. But as a whole we felt the issues were dealt with as effectively as possible within the timeframe given.

5. Installation Guide

Step 1:

To install this app a user must have access to an android device with a version of Android 9.0 or higher.

Step 2:

The app will be available to download from the google play store which is available on all devices with android OS. But until it is released the app will be available in a

APK file which can be received by email by getting in contact with any of the developers.

Step 3:

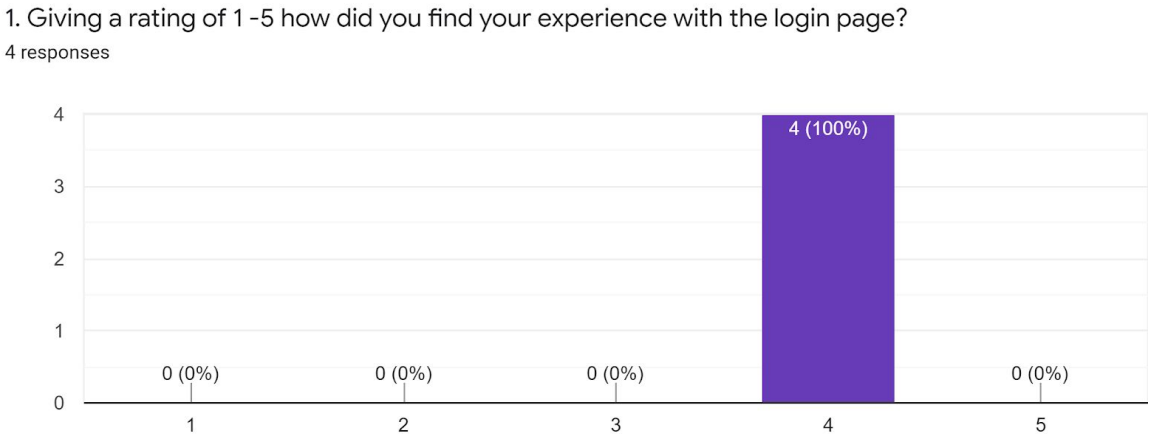
A user will simply search our apps name into the google play store and click on the install button next to our apps link. Or they will click on the executable APK file they have received by email

Step 4:

The app will then be available to open on your device from the unique icon on the menu.

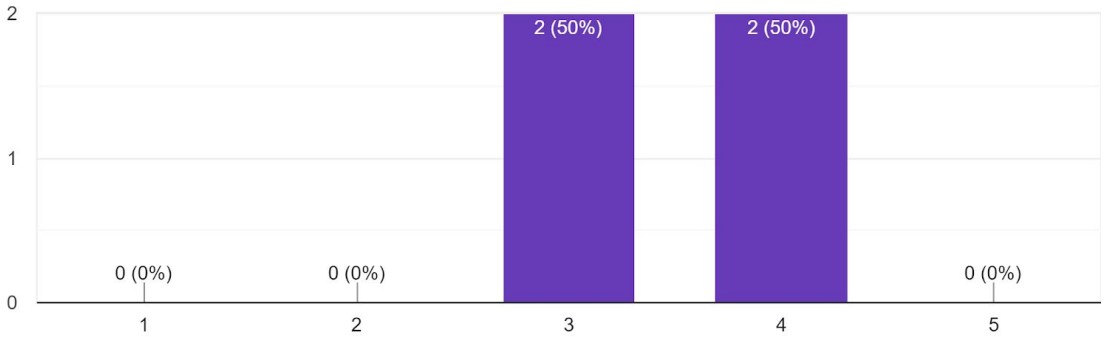
6 Testing

6.1 User Testing



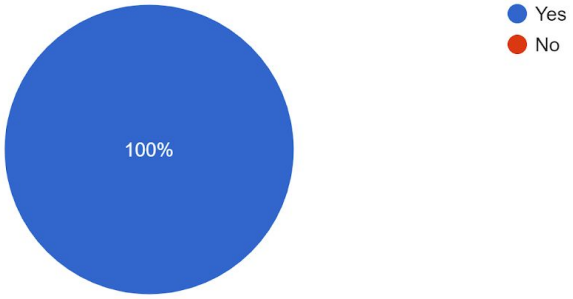
Giving a rating of 1-5 how well did you feel the information was displayed to you as the user on the scrolling page?

4 responses



Did you enjoy your use of the app?

4 responses



Why? 4 responses

It was easy to use and put everything in one place

Ui

App tries well to improve user music experience

Ui was very nice.

Spotify concert feature is very handy

Where in your opinion could this app be improved? 4 responses

Less cluttered

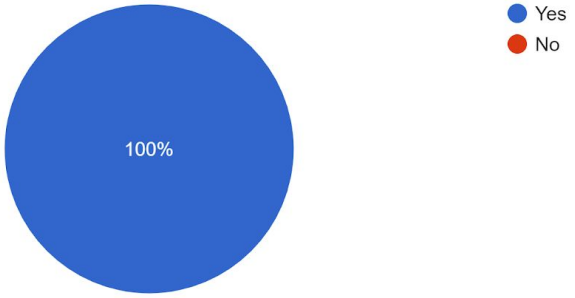
More events

Recommender systems can always be more accurate

Ability to share concerts with friends

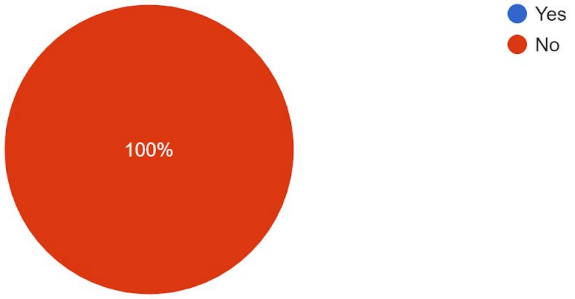
Would you like to use this system frequently?

4 responses



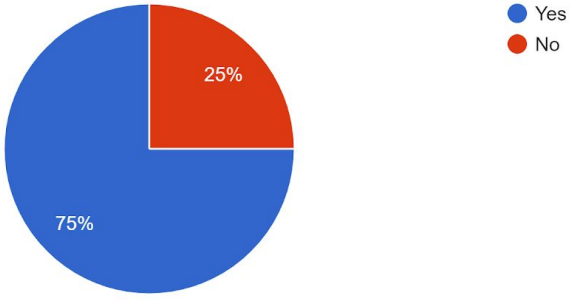
Did you find this app unnecessarily complex?

4 responses



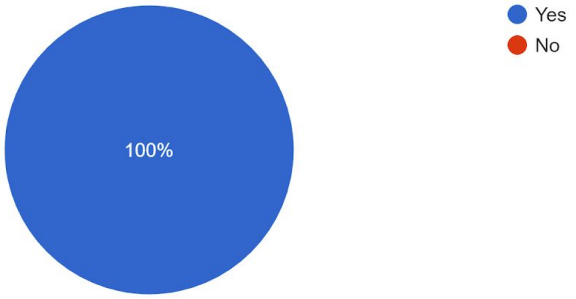
Did you find this app aesthetically pleasing?

4 responses



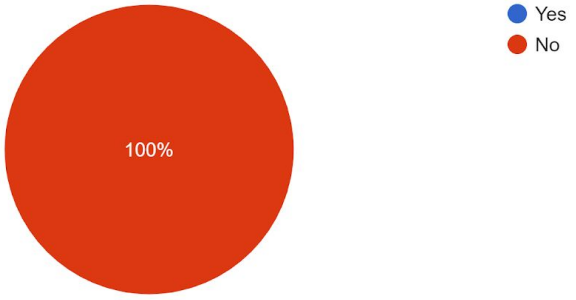
Did you find the system was easy to use?

4 responses



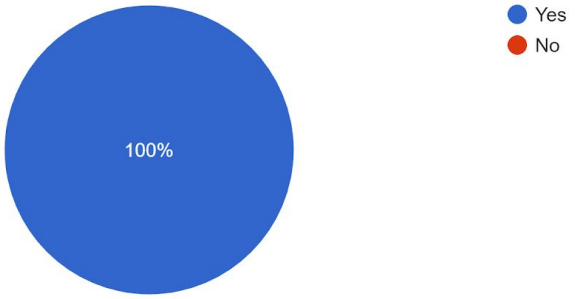
Do you think you would need the support of a technical person to be able to use this system?

4 responses



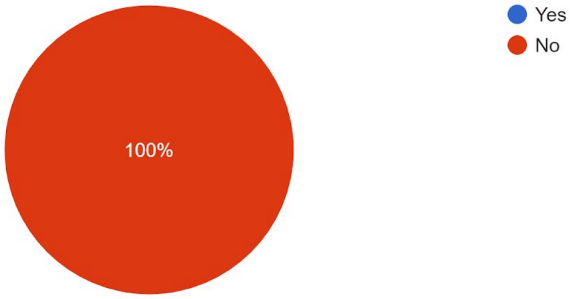
Did you find the various functions in this system well integrated?

4 responses



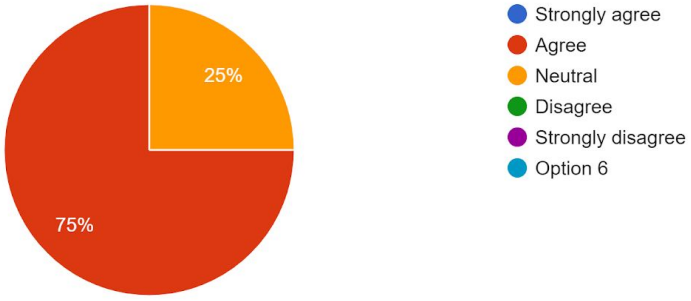
Did you think there was too much inconsistency in this system?

4 responses



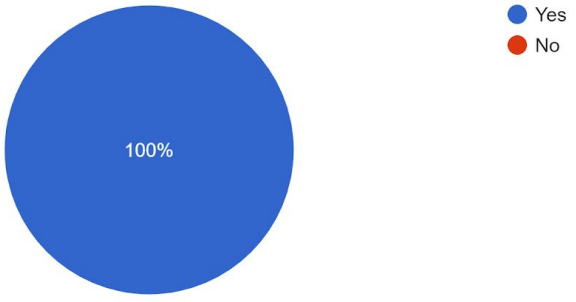
I would imagine that most people would learn to use this system very quickly, Would you agree with this statement?

4 responses



Did you feel confident using the system?

4 responses



Any further thoughts? 2 responses

No

Great app idea

6.2 Use Case Testing

No.	Job	Steps	Status
1	Sign up	1. Press “SignUp” button 2. Enter email 3. Enter password 4. Click“SignUp” button	PASS
2	Sign in	1. Enter email 2. Enter password 3. Click“Sign in” button	PASS
3	Post media	1. Click on plus on profile page 2. Enter description 3. Upload Image 4. Click “Post” button	PASS
4	Link spotify	1. Click Link Spotify button 2. Accept terms of use	PASS
5	Link Location	1. Click Link location services 2. Enable location in settings	PASS
6	Change Avatar	1. Click on avatar icon 2. Upload image	PASS
7	Search for event	1. Click on search bar 2. Type request into bar 3. View results	PASS