Prototype Documentation

James Angel, Harrison McDonagh, Kevin Ibelgauptas, Shao Wen Young

18th Jan 2021

This documentation is a brief insight into the state of our application after the completion of the prototype. This document will describe what features our application has with some accompanying screen shots and compare these features to the user stories created at the beginning of the project life cycle. This document also shows the breakdown of contributions made by individual members of the team, as part of this we also detail what each member of the team learnt. Finally there will be a few notes made by the team on what our next steps will be.

## Feature List

A host can create a room.

A host can play, pause or skip a song.

A host can view a list of music, compiled from all the users in the room.

A host can queue a song from a list of music.

A customer can join a room (only via clicking on a button - and is only capable of joining one room).

A customer can view a list of songs, compiled from all the users in the room.

## Evidence

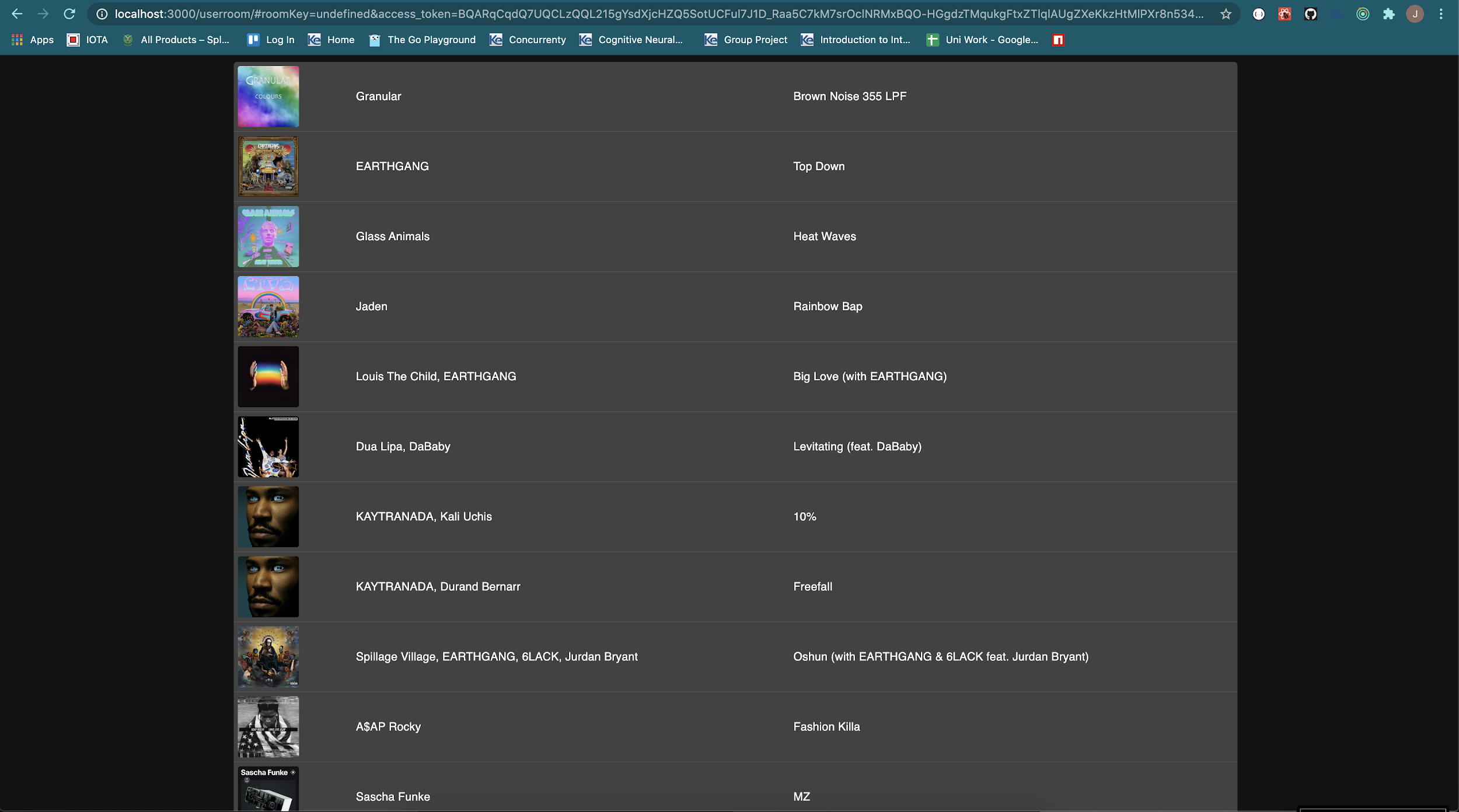
### Landing Page (User chooses to host a room or join a room).

### The host page, once a room has been created.

### The host page, once the host has started playing music.

### 

### The customer page, once a customer has joined a room.



## Uses Cases

Initial uses cases, created by the team at the beginning of the project. This can be found in more detail in the file “CO600 Group Project Use Case Scenarios v1”.

* **Host creates a virtual room**. As a host at a venue/bar (host) I want to start a virtual room, so that customers can join to share their music preferences. (**Completed**)
* **Customer joins the room**. As a customer, I want to be able to join a virtual room via QR code or URL so that I can listen to music I enjoy in the venue/bar I am at. (**Not Completed**)
* **Customers share music**. As a customer, I would like to have my Spotify data shared with AllTunes so that the host can get my music preferences. (**Completed**)
* **Host gets music from customers.**  As a host, I want to have access to a customer’s choice of genre/taste of music, so that I can queue their songs up in the Spotify queue to be played next. (**Completed**)
* **Music is shown in order of most popular.** As a host, I want to be able to see the songs listed in order of most popular, so that I can play the most popular music. (**Completed**)
* **Host Queue Music**. As a host, I want to queue songs displayed in the app so customers can listen to the songs I’ve queued. (**Completed**)

## Individual Contributions

### James Angel

#### What I Contributed

* Adding styling to the prototype to match with the spotify aesthetic. (Coloring of Buttons, correcting background colour, making the player controls appear at the bottom, and to match the style of player used in the desktop application)
* Converted a lot of already written react code to use a functional version of react rather than a OOP style, as it has a more diverse set of tools.
* I added the functionality to allow customers to be redirected to a room once clicking on the “join room” button. (with Hamish)
* I added the functionality to allow a host to create a room and be redirected to the correct page once clicking on the “host room” button.
* I created the “trackTable” component, which displayed all of the tracks to the hosts and customers. (with Kevin)
* I added code to manage the passing of variables within the front end. Mostly the room key and the users spotify access tokens.
* I created all of the API, for creating a room, adding a customer and listing the rooms details. I also wrote all of the code in the front end which makes use of these APIs.
* I added a spotify client within our backend to get all of the top tracks of a use from spotify, and added to Shao’s work (music manager)

#### What I Learnt

* A Lot of my initial work done on the project, to do with kubernetes, devops and hosting didn't make it into the prototype. I learnt a few things from this:
  + I learnt how to use the google cloud kubernetes engine, both through their user interface and their CLI.
  + I learnt how to utilise the kubernetes API to create new pods and services on demand, to allow new room-services to be created.
  + I learnt that, if you are working towards a deadline, it is better to focus on a defined set of features, ensuring that are completed, rather than trying to work on all of the features at once and arriving at the deadline with a larger amount of half completed features. (This is why the kubernetes work I did, did not make it into the prototype).
* I also learnt that, once you have started on a project and want to enforce a new coding standard. It is better to just do it moving forward and enforce “boy scouting rules”, which means you always leave a place better than how you found it. E.i, don’t bother going back and redoing old work to enforce the new coding standard, just implement it moving forward.

### Harrison McDonagh

#### What I contributed

* The front end design which must contain JS, HTML and CSS code to display the user interface for our users to navigate through the app.
* The front end pieces all the pet projects together, it allows the user to connect to spotify using Kevins code, and then returns all the needed data in order to allow the user to use our app. Once the user is signed in and onto our app it displays a room depending on the type of user, if the user is a DJ it allows them to queue the next song whilst the listeners only get to vote on which song they want next.

#### What I learnt

* How to work with ReactJS, which is a good skill as it is very different in its own way to using normal HTML JS and CSS.
* How to work with APIs and GET/POST requests in code
* Package managing inside of ReactJS, installing and managing different packages to work with ReactJS

### Kevin Ibelgauptas

#### What I contributed

My main focus was retrieving the data from Spotify and using that data to then provide functionality to our App. The data I mainly retrieved was the authentication values, where the user would be authenticated through Spotify before accessing our app, this provided us with specific data which we could control and manipulate. Below consists of what I’ve contributed;

* Playing and pausing the song, which allowed the user to control the status of the current song active in their Spotify web player.
* Skipping a song and retrieving the previous song, which provided the ability of going back and forth with songs.
* Finding the current song which was active,
* Updating the next song if the user used Spotify Web Player to change the songs.
* The device which was being used, which allowed Spotify to show that the music was playing on our All Tunes app.
* Details about the user for instance their username and display picture, which just portrayed which account you’re currently using.
* Added in the icons for the ability of pressing the buttons which change, play, pause, skip the songs. Along with the duration bar which shows the songs progress length of finishing.
* Worked on the queueing button with James.
* Set up an axios base instance, which defined a URL to retrieve data from Spotify’s SDK/API url.
* Created documentation on the Spotify front end code.
* Developed the use case, use class and sequence diagrams.
* Worked with Shao on getting the users to be pushed to an array and displayed to both host and user rooms so people know which users(customers) are in the room.

Removed duplications from the room lobby when users kept joining and leaving.

Set a limit on how many songs are displayed after retrieving each user's top tracks.

* Worked with Shao on creating our own Lobby Queue, so users can queue up the songs from the Lobby Top Tracks table/queue which resides in the middle of our app. Currently made the queue button add to the Lobby queue section on the left.

#### What I learnt

* Developing cacheable Api’s, using actions and reducers.
* Basic React
* Using react with SDK’s.
* Using react to connect an API to a React App.
* Package Manager, using Node and Npm to install software packages.
* CI/CD Pipelines for testing.
* How to use WebSockets to pass props from API’s (specifically spotify api).
* Using Websockets to send messages.

### Shao Wen Yong

#### What I contributed

* The music manager in the backend handles the incoming music from new customers. Each room has its own list of music that is updated with every user’s top tracks and this list is ordered by the number of people who have the same song as one of their top tracks.
* The application makes use of WebSockets to send and receive data. In this application it is used to update the list of songs in each room. Each customer in a room also has the list of songs of said room. The lists visible to both the hosts and customers are only updated at certain times (eg. when a new user joins a room, when a user refreshes their application, etc.) for efficiency.

#### What I learnt

* How to use JavaScript reduce.
* Basic React.
* How to create and use WebSockets.
* Good practices for project structuring and naming.

## Next Steps

* Add the ability to join rooms with QR codes. (High priority; we said we’d do it)
* Adjust the size of the room’s song list.
* Add a list of customers who are currently in the room.
* Allow customers to follow each other through the application.
* Add a voting system the customers can use to let the host know what song they want queued next.
* Add proper tests.
* Host the whole application on google cloud with kubernetes.

### Pet Projects/Individual Responsibilities

#### Kevin

Add a list of customers who are currently in the room.

Allow customers to follow each other through the application.

Look at making a queue to be displayed and create a voting system. - might need help

#### James

Hosting the application on the google cloud, using kubernetes.

#### Harrison

Add the ability to join rooms with QR codes

#### Shao Wen

Songs should be manually added to the host. If there’s no songs queued up the song at the top of the suggested tracks list will be queued.

Refactor WebSockets.

Show room key in room lobby section

Help Kevin with the WebSockets.

Fix the websocket refresh.

Create functional variation tests?

Make the guided interview questions/questionnaire.

Allow users to see what the currently playing song is and let them open it in Spotify.

Possibly also allow users to play the song on their end as well (remote joining of rooms).