REPORT FOR SOFTWARE DEVELOPMENT GRADED UNIT 2

UNIT: H48W35

David Gemmell

May 2020

HND SOFTWARE DEVELOPMENT 2ND YEAR

COMPUTING – BUSINESS SCHOOL

PAISLEY CAMPUS, WEST COLLEGE SCOTLAND

**PAISLEY CAMPUS, WEST COLLEGE**

Contents

[1. Introduction 7](#_Toc41316305)

[2. Planning 9](#_Toc41316306)

[2.1. Initial Investigation 9](#_Toc41316307)

[Project Title 9](#_Toc41316308)

[Purpose of the Project 9](#_Toc41316309)

[Scope of the Project 9](#_Toc41316310)

[Initiator of the Project 9](#_Toc41316311)

[Names of people available for advice and assistance 9](#_Toc41316312)

[Time Limitations 9](#_Toc41316313)

[Choice and Topic for the project 10](#_Toc41316314)

[Aims and Objectives 10](#_Toc41316315)

[Similar websites to the project 10](#_Toc41316316)

[Functional Requirements 12](#_Toc41316317)

[Non-Functional Requirements 12](#_Toc41316318)

[2.2. Approach to the project 13](#_Toc41316319)

[Systems Development Life Cycle Methodology 13](#_Toc41316320)

[the analysis and design methods and/or models that will be used 14](#_Toc41316321)

[The programming languages 15](#_Toc41316322)

[2.3. Justification of Approach 16](#_Toc41316323)

[Hardware resources: 16](#_Toc41316324)

[Software resources: 16](#_Toc41316325)

[Justification 17](#_Toc41316326)

[2.4. Project Plan 19](#_Toc41316327)

[Project Plan 19](#_Toc41316328)

[Gantt Chart 21](#_Toc41316329)

[2.5. Information Sources 23](#_Toc41316330)

[2.6 Business Model 24](#_Toc41316331)

[Functional Requirements 24](#_Toc41316332)

[Use Case Diagrams 25](#_Toc41316333)

[Activity Diagrams 26](#_Toc41316334)

[Problems and Limitations 26](#_Toc41316335)

[2.7 view model 27](#_Toc41316336)

[Storyboards 27](#_Toc41316337)

[Home Page 28](#_Toc41316338)

[Results Page 29](#_Toc41316339)

[Artist Page 30](#_Toc41316340)

[Recommendation Page 31](#_Toc41316341)

[Favourite Artists Page 32](#_Toc41316342)

[Disliked Artist Page 33](#_Toc41316343)

[Navigation Map 34](#_Toc41316344)

[Structured English 34](#_Toc41316345)

[Non-functional requirements 35](#_Toc41316346)

[3. Developing 37](#_Toc41316347)

[3.1 Diagrams 37](#_Toc41316348)

[Use Case Diagram 37](#_Toc41316349)

[Use Case Descriptions 38](#_Toc41316350)

[Activity Diagrams 47](#_Toc41316351)

[View Favourites 47](#_Toc41316352)

[View Disliked 48](#_Toc41316353)

[View Recommendations 49](#_Toc41316354)

[Favourite artists 50](#_Toc41316355)

[Dislike Artists 51](#_Toc41316356)

[Search 52](#_Toc41316357)

[Navigation Map 54](#_Toc41316358)

[Structured English 55](#_Toc41316359)

[Home 55](#_Toc41316360)

[Favourites 56](#_Toc41316361)

[Dislikes 57](#_Toc41316362)

[Recommendations 58](#_Toc41316363)

[Results 59](#_Toc41316364)

[Artist 60](#_Toc41316365)

[3.2 Implementation 62](#_Toc41316366)

[Code listings 62](#_Toc41316367)

[App.JS 62](#_Toc41316368)

[APP.CSS 66](#_Toc41316369)

[Home component 69](#_Toc41316370)

[Results component 72](#_Toc41316371)

[ArtistResultSummary component 74](#_Toc41316372)

[Artist component 75](#_Toc41316373)

[ArtistTopAlbum component 84](#_Toc41316374)

[ArtistTopSong component 85](#_Toc41316375)

[Search component 86](#_Toc41316376)

[SearchedArtist component 88](#_Toc41316377)

[Recommendations component 96](#_Toc41316378)

[getRecommendation 99](#_Toc41316379)

[Favourites component 100](#_Toc41316380)

[getFavourite component 102](#_Toc41316381)

[Dislikes component 103](#_Toc41316382)

[GetDisliked component 105](#_Toc41316383)

[Testing 106](#_Toc41316384)

[Original Strategy 106](#_Toc41316385)

[Updated Strategy 106](#_Toc41316386)

[Design of the website 112](#_Toc41316387)

[Screenshots of the website 113](#_Toc41316388)

[Home page 113](#_Toc41316389)

[Search page 113](#_Toc41316390)

[Recommendations page (empty) 114](#_Toc41316391)

[Liked page (empty) 114](#_Toc41316392)

[Disliked page (empty) 115](#_Toc41316393)

[Results page 115](#_Toc41316394)

[Artist page (Foo Fighters) 116](#_Toc41316395)

[Searched artist page (Imagine Dragons) 116](#_Toc41316396)

[Liked page (filled) 117](#_Toc41316397)

[Disliked page (filled) 117](#_Toc41316398)

[Recommendations page (Filled with Foo Fighters recommendations) 118](#_Toc41316399)

[Recommendations page (Filled with Tenacious D recommendations) 118](#_Toc41316400)

[Test Logs 119](#_Toc41316401)

[Known Errors 125](#_Toc41316402)

[User Testing 126](#_Toc41316403)

[Mum’s Testing 126](#_Toc41316404)

[Dad’s Testing 127](#_Toc41316405)

[User Documentation 128](#_Toc41316406)

[Questionnaire 128](#_Toc41316407)

[Section 1 – About You 128](#_Toc41316408)

[Section 2 – The website 128](#_Toc41316409)

[Mum’s answers to the questionnaire 129](#_Toc41316410)

[Section 1 – About You 129](#_Toc41316411)

[Section 2 – The website 129](#_Toc41316412)

[Dad’s answers to the questionnaire 130](#_Toc41316413)

[Section 1 – About You 130](#_Toc41316414)

[Section 2 – The website 130](#_Toc41316415)

[Instructions on how to install and run the website 131](#_Toc41316416)

[User Manual 132](#_Toc41316417)

[4. Evaluating 137](#_Toc41316418)

[Functional requirements 137](#_Toc41316419)

[Non-functional requirements 139](#_Toc41316420)

[Strengths and weaknesses 140](#_Toc41316421)

[Recommendations 142](#_Toc41316422)

[Modiifications 144](#_Toc41316423)

[Unforseen Events 145](#_Toc41316424)

[Knowledge and Skills 146](#_Toc41316425)

[Knowledge of React Applications: 146](#_Toc41316426)

[How to use Local Storage: 146](#_Toc41316427)

[Knowledge of how to use APIs: 146](#_Toc41316428)

[Research skills: 147](#_Toc41316429)

[Development process 147](#_Toc41316430)

[5. Conclusion and Recommendations 148](#_Toc41316431)

[6. Bibliography 149](#_Toc41316432)

# 1. Introduction

This report is being written for Michelle Blair, my lecturer for my Graded Unit 2 in HND Software Development. The unit code is H48W 35 and the project I am about to undertake will allow me to develop some existing software skills, while also allowing me to practice and gain new skills whilst completing this project. This project will make me have to learn more about JavaScript, Json objects and React applications. I will also be attempting to learn and implement a bit of machine learning towards the end of the implementation. This is so I have a functioning website that can reliably give out recommendations to users by using an API. The report will cover all three of the stages that will be undertaken and completed while this project is underway.

My choice for this project is to create a music recommender. I will be making a website which will allow the user to enter their favourite band and then search for recommendations. The results will be found by using the last FM API and then stored in a Json object. The Json object will be used to display the first five results of the search. The Last FM API is pretty good as it returns the name and an image of them. I will be using both of those to display the results and I will also be running another part of the API to get the found artist’s top tracks. These will initially link to last FM’s website which has pages for all these songs but if I can, I may try and create a component of my own which shall display the track and some information about it. Once this main part of the website has been created, I will attempt to use machine learning to identify recommendations. I will do this by allowing the user to add artists to their favourites. These will be saved in a Json object which will be, in turn, read by the machine to see who they user has liked. The machine should then go and find some of the similar artists that the user could like and will display them in another component called recommendations. The reason I want to do this website in this way is because I believe machine learning is one of the biggest talking points in computing just now and it would be great to say on my cv that I have used machine learning, even in a very basic form, in a project I did. Another reason is I believe music is a big thing in most people’s day to day lives, and because of this I think some music can grow old or people can just get fed up with listening to the same old songs. That’s why I want to make this, it’s to help improve people’s taste in music. Hopefully, this can prove that someone who likes Taylor Swift can also like Frank Zappa, for example. As for what I hope to produce during this project, I want to produce a website that is to a high standard. This means that everything on the website should work, the color scheme should be consistent on all the pages and in general it should be easy to use and understand what the buttons do. I also want to be able to produce a reliable way of recommending new music to people.

When I was carrying out this project I worked independently and used the resources of the internet to help me further my knowledge. While creating the code for my website I used videos and websites to figure out how to make some things work, because of this I have put a link to the original creator of the solution and I have not plagiarized anyone’s work. I believe that my code and my report have been produced to the highest standard as possible as I worked on it to ensure every part of the code and report was the best I could do.

I got my information from the internet (see bibliography for links) and a book for HTML and CSS. My dad gave me advice on parts of the project where I got stuck. I believe the only limitations of the report would be that that of the testing stage in development as I wanted to get lots of people from the college to test however because of lockdown I was only able to get me, my mum and my dad to test the website and therefore I couldn’t get a big group of answers to the questionnaire so I could work on the website in the future.

**Business School, Paisley**

**Please Complete, Sign and Submit this Declaration with Your Assessment**

I declare that:

* the attached assessment is all my own work
* all information from other sources has been correctly cited and referenced
* I have taken reasonable care to make sure that my work has not been copied by anyone else

Name: David Gemmell

Course/Class: HND Software Development

Unit Title: Graded Unit

Assessment Title

Signature Date

**Plagiarism is “taking the work or idea of someone else and pretending it is one’s own” (Oxford English Dictionary) which includes:**

* Submitting a report/assignment written completely or partly by someone else
* Preparing a report/assignment for another student to submit
* Copying a report/assignment or allowing your own assignment to be copied by another student
* Using material sourced from the internet or textbooks without acknowledging the source
* Not labeling diagrams/illustrations etc.

# 2. Planning

## 2.1. Initial Investigation

### Project Title

Music Recommender

### Purpose of the Project

The purpose of this project is for me to teach myself about machine learning and use that as a way to help people find new music. This would also be beneficial to me as one of the big subjects in computing just now is machine learning so if I teach myself enough about it to have a good understanding it will help me stand out in interviews.

### Scope of the Project

The website will allow users to discover new music based on the music they listen to currently by getting the users to input their favourite artists and then the website will give a list of 5 new bands for them to listen to.

### Initiator of the Project

David Gemmell

### Names of people available for advice and assistance

Clare Lowe

Michelle Blair

Alan Livingston

### Time Limitations

* Planning – 7th February 2020
* Development – 15th May 2020
* Evaluation – 29th May 2020

### Choice and Topic for the project

My choice for my project is to create a music recommender. I have chosen to do a music recommender because I love music and listen to a variety of music every day. I believe that lots of people all over the world do this and music can have a really big impact on people’s lives. The impact people experience is different for every single person. For example, while Pink Floyd’s Wish You Were Here will always remind me of my grandpa every time I hear it, it could make some people be reminded of an important person in their life that they may have fallen out of contact with. I also want to make this because I know machine learning is a big subject in the computing industry just now so if I teach myself enough so I have a good understanding of it, I will stand out in interviews as I will be able to demonstrate some of my knowledge, that I have gained from doing this project, of machine learning.

### Aims and Objectives

My aims for this project are to create a react application in visual studio code which can allow users to search for similar artists, teach myself more about APIs and I will try to learn how to code machine learning so I can make a component that is dedicated to showing recommendations to the user. I believe these aims are all achievable, however I also believe that this will be learning machine learning will be challenging as it will require me to make a machine that can observe, use and learn from everything the user puts in to their favourite artists.

My main objective for this project is to make a functioning music recommender that can reliably return artists to the users. However, I have multiple smaller objectives. My first one of these objectives, is to learn how to use a web API in the code and apply this to find the similar artists. This will mean I will have to do research on how to use Get requests, how to process the responses I receive and how to display these responses in a formatted way that will appeal to people. Another objective is to learn how to code a machine that can learn from the data that the users have inputted. This will mean that I will have to do research on how machine learning works, how to code machine learning, how to connect it to a react application. Both objectives have a similar goal, to make this a functioning website. My main way of doing this will be research, on the things specified above, that will be done concurrently with the development of the website, however I am leaving the machine learning aspect of the project to the very end as I believe while it is incredibly big in the computing world just now, learning how to use APIs and applying that to make a functioning website is more important. I will still attempt to make the machine learning aspect of this project work.

Overall, most of my aims and objectives are solely focused on creating a functioning website that, at the very least, will allow the user to enter their favourite artist and receive similar artists. I would say that overall, the most important thing about the website is that the users should have a fun time discovering a new artist or artists and hopefully it’ll make people realize that there is so much music available that some people may never even hear of.

### Similar websites to the project

During the course of my initial investigation I came across a few websites that have the same function as the one I intend to create. One of these websites is called GNOOSIC (Gibney, 2002). This is probably the simplest website I’ve found. The design of the website overall is very basic as the background colour is just a plain white and the text is just the standard black. However, it does allow the user to input three of their favourite artists and then it comes up with a list of artists that the user may like. I tried this by inputting Ac/Dc, Guns’N’Roses and Kiss. The results I got were strange. I knew four of the bands it recommended me but some of the bands that it recommended to me were weird. Guns’N’Roses and Kiss are hard rock bands from America and Ac/Dc are a hard rock band from Australia. The reason the results were strange was because it recommended me a Norwegian heavy metal band and a Swiss heavy metal band. The only reason I see these as strange is because after looking at all the suggestions it gave me these are two are the only ones that did not come from America. I have not personally listened to them, so I don’t know if I like them or not, but they did stand out to me. Overall, this website was just ok. It was neither good nor bad, it was actually pretty fun to put in my favourite bands to see what it came up with, but the recommendations slightly ruined it. That was because I knew some of the recommendations, although I didn’t mind it as much but some of them were just way too strange for me.

Another website I discovered while looking for some examples was tastedive (Qloo, 2017). Comparing this tastedive (Qloo, 2017) to GNOOSIC (Gibney, 2002) showed me how basic GNOOSIC (Gibney, 2002) was. GNOOSIC (Gibney, 2002) was very easy to use but it was also very bland. Tastedive (Qloo, 2017) on the other hand was bright and colourful. It was still just as easy to use but it had a better presentation of the results. So, I typed in Tenacious D to get my recommendations and overall, I knew most of the recommendations. Some of the recommendations I hadn’t heard of, but they looked interesting, so I listened to one of them. This was a band called Beatallica and this band is a very interesting concept. The band has fused the Beatles and Metallica together to create mashups of the big songs from both of the big groups. To be frank, it wasn’t to my taste but like Tenacious D, I found it funny enough to try a couple of their songs and it grew on me. I wouldn’t go out of my way to listen to them but if they ever came on naturally, I probably wouldn’t skip them. This website, in general, was much better than GNOOSIC (Gibney, 2002) because it held my attention for much longer as it used colours and the way they displayed the results were visually appealing. However, this website really stood out to me because not only can it recommend music to the user, it also has pages to recommend Movies, Tv shows, Books, Authors, Games and Podcasts. Other than the music page, I only tried the Games page as I wanted to see if the pages worked in the exact same way.

Overall, comparing these two websites together shows a massive difference. GNOOSIC (Gibney, 2002) was extremely basic as it had no colour and no interesting way of displaying the recommendations. The only thing I could say it did well was once it was over it showed all of the recommendations it gave in the categories of You Like, You don’t know and You don’t like. On the other hand, the way tastedive (Qloo, 2017) worked was much better. It was just as easy to use but it offered more. It allowed you to create an account and add the recommendations to your favourites. Plus, it allowed for different things to get recommended to the user (see above), this website was fantastic compared to GNOOSIC (Gibney, 2002).

With this being said, I aim to make a website that drastically improves on GNOOSIC (Gibney, 2002) but I am only focusing on music so tastedive (Qloo, 2017) while good as it offers so much, it only gave me a few ideas on how to display the recommendations. Overall, this initial investigation helped me get prepared as it allowed me to get a good idea on the functional and non-functional requirements for my website. I have listed these below.

### Functional Requirements

* The user must be able to input their top three favourite artists into the search
* Once the user has searched for similar artists, they must be returned the top 2 similar artists for each artist they inputted.
* The user must be able to click on any of the results and be taken to a page for that artist.
* The results page should display the artist’s name and a picture of either the artist or their top album.
* The information page must show some general information about the artist and it should show their top 5 tracks.
* The user must be able to click on the any of the top tracks and be taken to the Last FM page for that song
* The similar artists that appear should be able to be favorited or disliked.
* The machine learning should use the favorited and disliked artists to find the recommendations.

### Non-Functional Requirements

* The website must have a consistent colour scheme throughout all the web pages. These colours should all be web safe.
* The text on the website must be readable
* All the links on the website must work and there should be no dead links
* The budget is £14000
* The deadline for this website is 15th May 2020
* The website must be secure so that no one can take over the website
* The website should have fast reaction times so the user can have a good experience with the website

## 2.2. Approach to the project

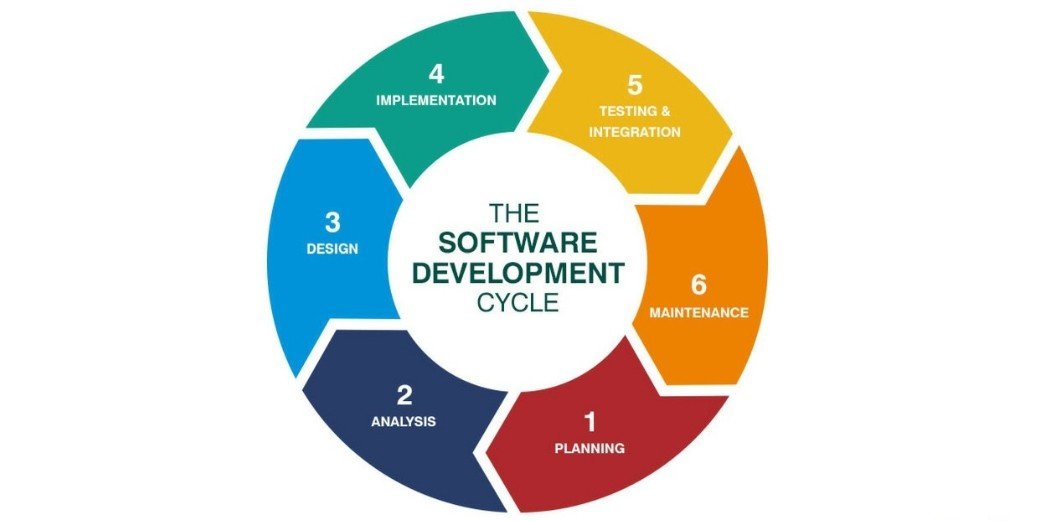
### Systems Development Life Cycle Methodology

A screenshot of a social media post

Description automatically generated

During my initial research for this project, I have researched both the Waterfall methodology and the Agile methodology. The Waterfall methodology is a well-known methodology and is one of, if not most, recognizable methodology in software development. This is because it follows a straightforward logic and makes the developer complete simple steps before moving onto the next part of it. This means that before any analysis can be done, the developer must get the requirements from the client. In the instance of my project, I acted as the client and as the developer so I will create a requirements questionnaire and complete it acting as the client. Anyway, the waterfall method is great because it allows you to follow a strict development structure but for this project it would not be suitable. I will go into more detail as to why in section 2.3.

I used these sources in this paragraph: (Codecademy, 2019), (Bowes, 2014).



The Agile methodology is a well-known methodology and is one is quickly becoming the normal methodology for developers to use. This is because they can interact with the customer at any point and this methodology allows for updates to the requirements at any stage. This means that if the client, or in the instance of my project, decides they want something different; the methodology can easily integrate the new requirements and remove the ones that are getting replaced. The agile methodology is a great methodology and it is perfect for my project. This is because it allows constant updates and it allows me to write a small block of code then test it immediately rather than waiting until I’ve created the whole program to see that the program isn’t working. I’ll go into more detail in section 2.3 as I justify why I’ve chosen this methodology in greater detail.

I used these sources in this paragraph: (Codecademy, 2019), (Kukhnavets, 2018).

### the analysis and design methods and/or models that will be used

First of all, I will be making a Use Case Diagram. Use case diagrams are a high-level diagram that allows complex ideas to shown to clients or possible users in a simplified way. Use Case Diagrams are used to show the application(in this case a website) as a whole and by doing this it allows for the person who is creating the diagram to showcase how the users can interact with the system but it also shows a very basic understanding of how the website will work. This diagram is a high-level one, but it is a very useful one as it simplifies the project into something that can be easily understood.

Once the Use Case Diagram has been completed, I will be writing the Use Case Descriptions. What this means is, I will be using the Use Case Diagram as the basis for these descriptions. The diagram shows the use cases, but the descriptions break them down further. The descriptions include the requirements for the use case to occur, the pre-conditions for it to happen, the post-conditions after the use case has happened, invariants which stop the use case from occurring and finally the scenario which is what happens during the use case. These are helpful as it helps me as a developer to discover what needs to happen during specific scenarios.

I will also be making activity diagrams. Activity diagrams are a diagram that take a part of the system being developed and break it down into how the user could go through it. This means that if the user could either choose yes or no, they diagram must show what happens in either situation. These diagrams show what the user can do and because of this it shows the flow of the program. This means that it will start with the user clicking something, show the flow to the next part of user interaction and so on until the user has finished. This diagram is useful as it helps me determine what the users should be allowed to do.

I will be making storyboards for the website. Storyboards are used to create a detailed look into what the user should see when they are on a specific page of the website. These are helpful as it allows me as the designer to plan what each page shall look like. By doing this, it gives the client a way of asking for a specific colour scheme, specific fonts and so on. The storyboards are basically a way for the client to see what the first design is and then ask for something a bit more to their liking if they so wish.

I will be making a navigation map for the website. This navigation map will be used to discover how every page should be connected to each other. This is done to ensure that pages that don’t need to be linked aren’t. It also helps as it allows the reader to see how to navigate from a certain page to another so for example, if I was on my home page and wanted to move to the results page, it would show that there is a connection from the home page to the results page. However, it does not show how to get to that page. In the same example, from my home page you need to input your favourite artist and press the search button in order to get to the results page. Navigation Maps are useful to certain extent but for my project it will be extremely useful as it will help demonstrate all the links from each page.

I used these sources in this paragraph: (Lucidchart, 2018), (SmartDraw, 2018).

### The programming languages

The website will be written in HTML (Hyper Text Markup Language). HTML is the most basic building block for creating a website. HTML is used in every website and is used for the structure of the web page. This means that HTML is used to place certain things in a certain place on the web page. However, HTML doesn’t make the website look good. CSS (Cascading Style Sheets) is used to do this. CSS is used to improve the aesthetic of the website and by doing this make it visually appealing to the users. HTML and CSS will be important in my project as they are both vital in making sure that the website looks good. It will also be used to make sure that things that should be in same place will consistently be where the user would expect them.

The website will have functions in it, and I will be coding these functions in JavaScript. The JavaScript I will be writing will be used for Get and Post requests I will be making to the Last FM API. These requests are essential to my project as I need these to be able to get the recommendations for the users. The JavaScript will be taking the input from the user and adding that to the Get request in order to get the recommendations. JavaScript is vital to my project as it will be handling all of the functionality of my website, which means it will be the base for the recommendation system. My JavaScript runtime environment is Node.js as this allows me to run my JavaScript functions in any browser. I’m also using Express.js which creates a lightweight server framework to use for my project.

The Last FM API requests I will be making will return Json objects that represent things such as artists, tracks etc. The JavaScript code that I will write will parse these responses and use the values to populate the user interface. I plan to also use Json objects to store things like the users liked and disliked artists.

I used these sources in this paragraph: (Duckett, 2011).

## 2.3. Justification of Approach

### Hardware resources:

Home PC – This PC has all the software I will need for doing this project. It also has an internet connection and I will be using it to conduct most, if not all, of my research.

Laptop – I will be using my laptop to work on my project when I am not at home. So, I will be using this at work and when I go to Denmark to ensure I stay on top of my project. This also has all of the required software.

College PC - This PC has most of the software I need for my project. It does not have Node.JS or Visual Studio code. It also has an internet connection and I will be using it to conduct research while I am using my laptop for visual studio.

College Printer – This will be used to print out all of my Documentation as I need to show my planning stage and further documentation once the Development and Evaluation stages are complete.

USB stick – This will be used to save copies of my project throughout the whole project as it will keep copies of the documentation and it will be used to bring in copies of my code once the website has been developed.

### Software resources:

Visual Studio code – I will be using this to create the react application as this allows me to write the JavaScript functions, the HTML elements and the CSS. This is a useful txt editor as I can also look at past projects I have done within Visual Studio and use these to help further my project.

Microsoft Word – This will be used to create all the documentation that is required for my project. It will also be used to put all my code and screenshots from my functioning website to showcase it working.

Microsoft Project – This will be used to create my project plan. I will be using this as it is the best software to create a plan as it allows me to allocate resources set up tasks and milestones to keep track of my progress.

Microsoft OneDrive - This will be used to save copies of my project throughout the whole project as it will keep copies of the documentation. OneDrive is another way for me to keep copies of my project so if it corrupts or if I lose it, I will always have a spare copy.

Web Browsers – These will be used for testing purposes. For example, I will need to use Chrome, Firefox and Internet Explorer to ensure that my website works on every browser the way it should. This means that the HTML elements are in the right place, the colour scheme is correct as well as the CSS and all the functionality of the website.

Lucid Chart – This will be used to create most of my diagrams and models for my Business and View Models. This is because it has lots of libraries that are used for creating certain diagrams in particular. They have a library dedicated to creating a good-looking Use Case Diagram.

### Justification

When I started my research into different methodologies to use for this project, I focused on two of them in particular. The first one I focused on was the Waterfall methodology. This is one of the biggest methodologies in software development, mostly because it offers the developers a logical guideline on how to proceed throughout the development. What that means is this methodology broke down the overall project into smaller steps which have to be completed in sequential order. This is to make sure that all the proper documentation is created at every stage, and as the developer goes through the stages, they can look over any previous documentation to ensure they stay on the correct path. However, there is a pretty big flaw with Waterfall. You must create the entire project before you can test it. This means if a vital part of the program is incorrect, the developer needs to go back to the implementation stage and try to fix the problem before moving onto the testing stage again. This is a really big disadvantage mostly because I have quite a small timeframe to create this project. I acknowledge that Waterfall has been a big methodology in software development but for my project it just wouldn’t fit well considering the timeframe and resources available to me.

Therefore, I have chosen to use agile development methodology. After whittling my list of potential methodologies, which included Extreme Programming and Spiral model, down to two I decided that I was going to choose agile development.

In order to justify using the agile development methodology, I will discuss the pros and cons of the methodology. First of all, the agile methodology can have problems when it is applied to a big project. For me, this is not so bad as I alone am undertaking this project and my project is not on a big scale. That is to say, the project has some challenges and some parts are complex, but it is not a massive project. Another con of Agile development is because of its flexibility. Now this means that because it is flexible, it’s hard to predict how far along the project will be in a few months. This means that there can be unforeseen setbacks and due to the style of the agile methodology, i.e. coding a piece of the software then testing until it works, the development may stall because of a part that won’t work. Obviously, if that happens and it’s not an essential part of the program it won’t be that annoying but in order for me to create the best possible program, I will need to ensure every part works and has been thoroughly tested.

The pros of agile methodology outweigh the cons for my project because of the resources available to me at home and at college are vast, the timeframe is small so I will need to work fast but most important of all it suits my style of programming. I prefer myself to check parts of code I have just written rather than waiting until the entire program has been written just to see that a small part of it is incorrect. The first pro of agile methodology is that it makes responding to change extremely easy. This means that if I decide to change any of the project during the development stage it will be easy for me to adapt to the change and start to update the project in accordance with the new requirement. Another pro of agile methodology is that I can approach the project whilst accepting some of the uncertainty around it. This means that I may draw up some storyboards of what the website will look like, but the finished product may not be identical. For example, my storyboards may depict a website with a pink background. Because of agile development, this can change throughout the project so it can improve the project overall. As in I could change the background to a light purple or light blue so that the text is more readable.

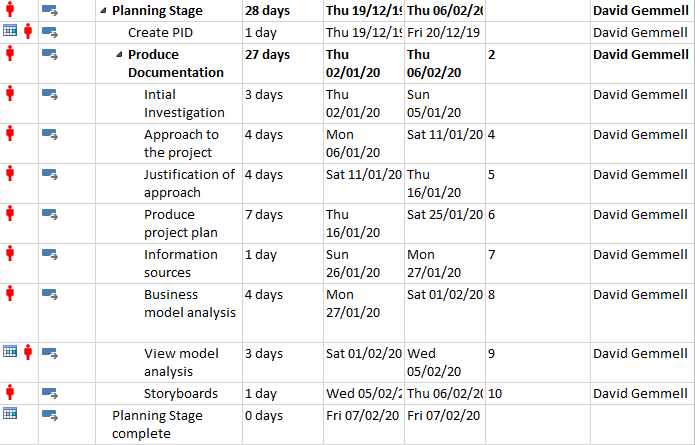
The Agile Methodology will help me create this project to the highest standard possible as it will help me make improvements during developing and testing. The flexibility is a great asset as it allows me to create pieces of the project and test them to make a functioning project. The timescale is a big constraint but because I will be using the agile methodology it is less pertinent. Overall, the Agile methodology will help me greatly in the creation, development and testing of this project. The resources I have at college are numerable, some being lecturers if I get stuck on a part and require some help to make the part work as it should, the internet to perform any research on react, APIs or anything, I have access to a lot of different programs that will be extremely helpful in the creation of my project plan and finally I’ll be able to use the computers themselves to listen to tutorials while I work on my laptop. The resources at my home are numerable as well as I have access to the internet for the same tutorials, I have a few books on HTML and CSS and I have the ability to work at home and at work as I have a laptop that has all the necessary programs on it. The only issues I will have is that I cannot create the project plan at home or at work because Microsoft Project is not included in Office 365. Given the fact that I can work on my project every day and everywhere and I have access to the internet all the time, I have almost unlimited resources. This coupled with agile development will give me the upper hand when it comes to creating my project.

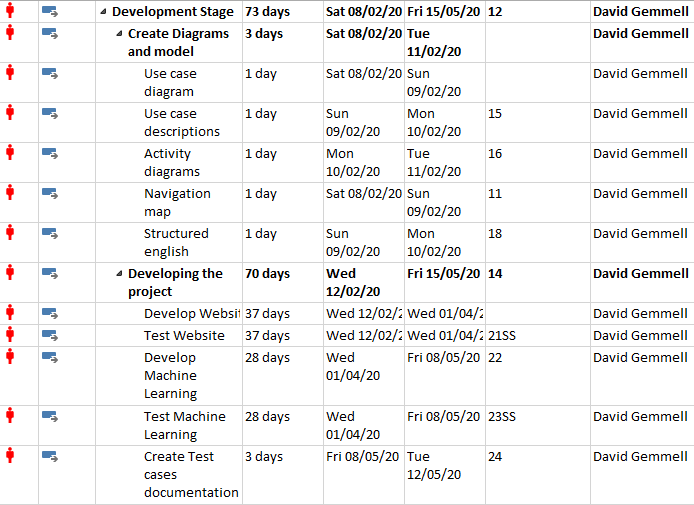
I used these sources in this paragraph: (Codecademy, 2019), (Codecademy, 2019), (Gilley, 2015), (Muslihat, 2018), (Sacolick, 2018), (Sexton, 2017) and (twproject, 2020).

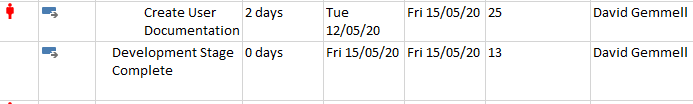
## 2.4. Project Plan

Due to the scale of this project, I require a project plan. This plan will be used to create a guideline for my project as it will have tasks and milestones that are used to track where I am at currently in the project. The plan is important as it acts as a guideline for the duration of my project. The tasks I have put into my project plan are the main tasks, that no matter what must be completed. Because I am using Agile methodology, my project plan may change as requirements for my website will change throughout the development process. The milestones are in the plan to help me track my progress.

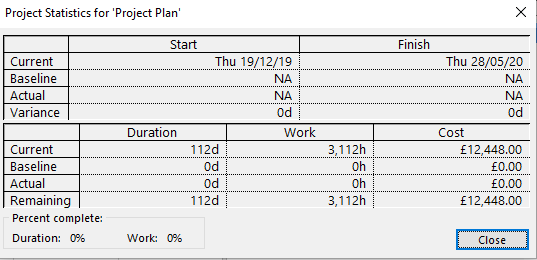
### Project Plan



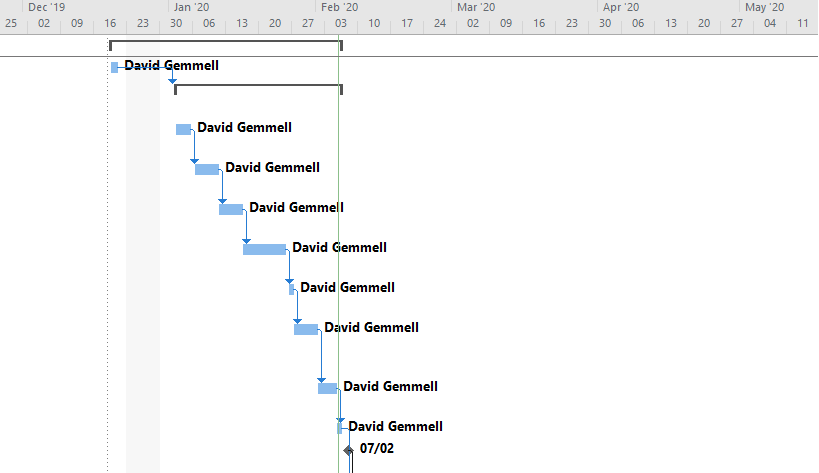


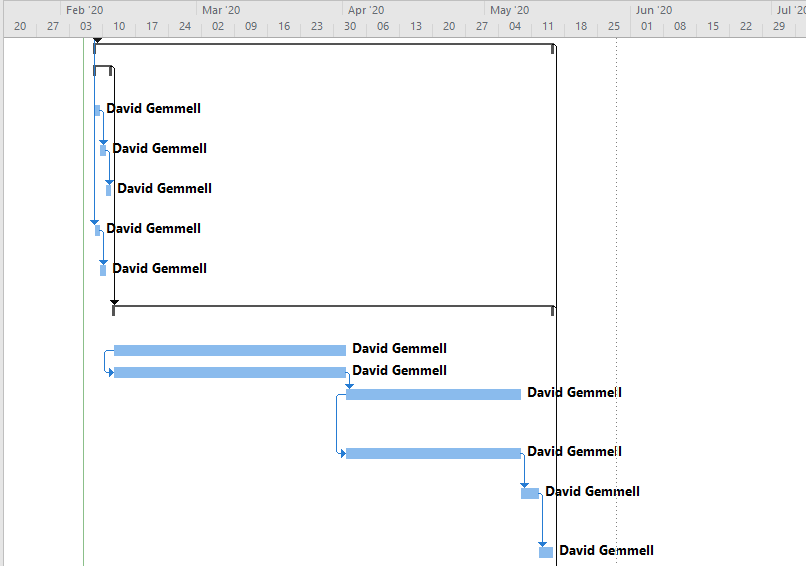


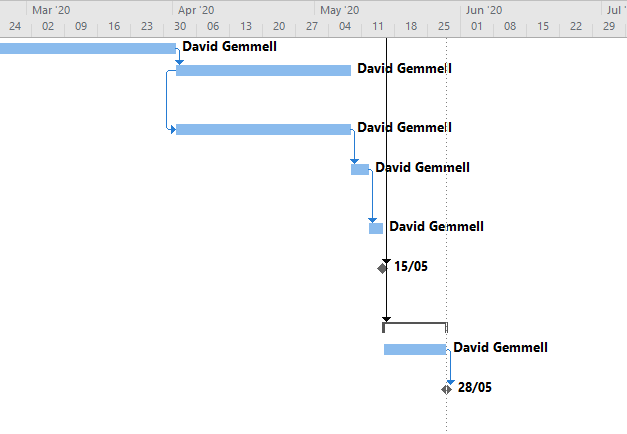




### Gantt Chart







### 2.5. Information Sources

Internet:

Gibney, M. (2002, August 03). *Gnod’s World of Music*. Retrieved from GNOOSIC: <http://www.gnoosic.com/>

Qloo. (2017, February 27). *music search.* Retrieved from TASTEDIVE: <https://tastedive.com/music>

Muslihat, D. (2018, March 02). *Agile Methodology: An Overview*. Retrieved from Zenkit: <https://zenkit.com/en/blog/agile-methodology-an-overview/>

Sexton, K. (2017, February 09). *Waterfall VS Agile.* Retrieved from YouTube: <https://www.youtube.com/watch?v=egF9-FejbsA\>

Codecademy. (2019, February 22). *Software Development Methodology: What is Waterfall?.* Retrieved from YouTube: <https://www.youtube.com/watch?v=LxEmGNgqYJA&t=329s>

Codecademy. (2019, January 06). *Software Development Methodology: What is Agile?.* Retrieved from YouTube: <https://www.youtube.com/watch?v=GzzkpAOxHXs>

Sacolick, I. (2018, March 15). *What is agile methodology? Modern software development explained.* Retrieved from InfoWorld: <https://www.infoworld.com/article/3237508/what-is-agile-methodology-modern-software-development-explained.html>

Kukhnavets, P. (2018, October 11).  *Why Agile is So Popular in Project Management World.* Retrieved from HYGGER: <https://hygger.io/blog/wp-content/uploads/2018/10/Agile-development.jpg>

Lucidchart. (2018, February 07). *UML Use Case Diagram Tutorial.* Retrieved from YouTube: <https://www.youtube.com/watch?v=zid-MVo7M-E>

SmartDraw. (2018, December 18). *All About UML Activity Diagrams.* Retrieved from YouTube: <https://www.youtube.com/watch?v=zid-MVo7M-E>

Bowes, J. (2014, July 17). *Agile vs Waterfall: Comparing project management methods.* Retrieved from Manifesto: <http://manifesto.co.uk/wp-content/uploads/2014/07/waterfall-project-management.jpg>

Kolowich, L. (2018, November 07). *Web Design 101: How HTML, CSS, and JavaScript Work.* Retrieved from HubSpot: <https://blog.hubspot.com/marketing/web-design-html-css-javascript>

twproject. (2018, August 13). *Agile methodology: advantages and disadvantages of an innovative method.* Retrieved from twproject: <https://twproject.com/blog/agile-methodology-advantages-disadvantages-innovative-method/>

tutorialspoint. (2006, September 30). *SDLC – Waterfall Model.*  Retrieved from tutorialspoint: <https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm>

Gilley, C. (2015, December 03). *The Pros and Cons of Agile Product Development.* Retrieved from Uservoice: <https://community.uservoice.com/blog/the-pros-and-cons-of-agile-product-development/>

Books:

Duckett J. (2011). *HTML & CSS design and build websites.* Indianapolis: John Wiley & Sons, Inc

## 2.6 Business Model

In this section, I will analyze the functional requirements I identified in my initial investigation. I will analyze my functional requirements first as most, if not all, of them deal with a specific function the website has to do.

### Functional Requirements

* **User artist search**

The first functional requirement I identified was that the user must be able to input their top three favourite artists into the search. What I mean is that there will be three input boxes and in each one the user will put in one of their favourite artists. The user will then press a search button which will then use a Get Request using Last FM’s API to get similar artists to the ones they have inputted.

* **View similar artist results**

The second requirement is once the user has searched for similar artists, they must be returned the top 2 similar artists for each artist they inputted. I will do this by getting the top two similar artists for each of their favourite artists and then sending these results to the results page. By doing this it should show the user some new bands and it gives them recommendations for every artist they put into the website.

* **Navigate to individual artist from results**

The third requirement is that the user must be able to click on any of the results and be taken to a page for that artist. This means that the user should be able to click on the artists name from the results page and by clicking the name, it should take them to a page that is just about them.

* **View artist results**

The fourth requirement is that the results page should display the artist’s name and a picture of either the artist or their top album. This is so the user can easily identify who the recommendation is. The image will be used so the user can easily see the artist, or the artists top album if the artist’s image is unavailable.

* **View artist information**

The fifth requirement is that the information page must show some general information about the artist and it should show their top 5 tracks. This means that once the user has clicked on the artist’s name on the results page, a new page will be loaded which will show the artist’s name, top 5 tracks and the general information I can get from Last FM’s API. The page will also have the artist’s image or top album image as well so it displays the same as the results page, but it will obviously have more data to be shown.

* **Artist’s top tracks**

The sixth requirement is that the user must be able to click on the any of the top tracks and be taken to the Last FM page for that song. What this means is that the user should be able to click on any of the top five songs and by doing this they will be taken to Last FM’s page for displaying the video of the song and it will have some general information about the song and the lyrics.

* **Favourite or dislike the artist**

The seventh requirement is that the similar artists that appear in the results page should be able to be favourited or disliked by the user. This is so that the website will know not to display certain artists that have been favourited or disliked because I want the users to find new artists not ones they already know. Plus, if the user keeps seeing artists that they don’t like it would start to get annoying because it would feel as if they are taking up a space in the results for an artist the user may like.

* **Generate recommendations based on machine learning**

The final requirement is that the machine learning should use the favorited and disliked artists to find the recommendations. This means that the machine should look at the favourited artists and find recommendations based on them. It should then cross-reference the recommendations it has made with the disliked artists and remove any that match. I.e. if I disliked Steel Panther and Steel Panther was one of the recommendations, the machine should remove Steel Panther from the recommendations and find a new one. The machine would then cross-reference the new recommendation with the disliked list until the recommendations did not have any disliked artists. The machine would also do this for the favourited artists because it makes a recommendation pointless if the user has the artist favourited already.

I will hand in the Use Case Diagram, Use Case Descriptions and Activity Diagrams at the development stage.

### Use Case Diagrams

I’m going to create a Use Case Diagram for my business model. This is because Use Case Diagrams are helpful in the development stage. What I mean by this is that the Use Case Diagram allows me to see what functions I will need to create in order for the user to have a good time whilst using the website. For example, one of the Use Cases that will be in the diagram will definitely be favourite the artist. This is an essential part of the machine learning part of the website, but it is also used to allow the user to create their own list of favourites. Basically, the Use Case Diagram is a great way for me to identify essential parts of the website that the user must be able to do, that the machine learning component must be able to do and what the website itself must be able to do. If this project was initiated by a client instead of me, I still would have used a Use Case Diagram because the diagram can be used to show the client exactly what the user can do. It also shows them a small part of what goes on behind the scenes, but it also allows the client to give feedback such as any things the user should be able to do being missed out in the first diagram.

Using the Use Case Diagram’s Use Cases, I will create Use Case Descriptions. These descriptions will be created by taking the Use Cases from the Use Case Diagram and creating a document that holds information that is specifically about that Use Case. These descriptions include the Use Case’s name, requirements, pre-conditions, post-conditions, invariants and the scenario. The requirements are just what has to have happened before the specific Use Case can happen. The pre-conditions are to show that if a certain thing has happened this Use Case can happen, the post-conditions are what can happen once the Use Case has happened. The invariants are things that if they have occurred the Use Case cannot occur. The scenario is just a basic piece of Structured English which shows what will happen during the Use Case. The Use Case Descriptions are a good pre-cursor to actual Structured English. This is because the small pieces of Structured English that I write will help when it comes to writing the bigger pieces as I can use the smaller parts as a basis for the bigger.

### Activity Diagrams

I’m also going to create Activity Diagrams for my business model. This is because the Activity Diagrams are helpful in the development stage. What I mean by this is that the Activity Diagram allows me to show exactly what the user does in a certain function. For example, one of the Activity Diagrams I will be making is getting recommendations. This means that I will show how the flow of actions that will happen from the user inputting their favourite artists to the user receiving their recommendations. These Activity Diagrams are used to clarify the specific paths users can follow when they use the website. Basically, these Activity Diagrams are used to show the flow of actions from one to another and finally it shows how the Activity ends. In the case of getting recommendations, it would end with the recommendations being sent to the results page. If I had a client, I would show them some of the activity diagrams. This is because some of them are more focused on behind the scenes rather than what the user can do. So, I would show them the getting recommendations Diagram but maybe not ones that have to do with the website calling the Last FM API. These diagrams allow the client to know the specifics of certain actions but it also helps with development as it show me as the developer exactly what the user needs to be able to do at certain stages whilst the user is using the website.

### Problems and Limitations

While I was doing this analysis a couple problems and limitations appeared to me. One problem was the timescale of the project. When doing the analysis of the functional requirements, I realized that I need to learn a lot about using APIs and machine learning. I have figured out a way to make this problem smaller and the solution is for me to learn how about APIs and machine learning while I am creating the website. Basically, I will be creating the functions and learning about APIs, how to call them (for example) at the same time.

Another problem I discovered was that because I am using the Agile methodology, the diagrams I create at this stage may not be what the finished program contains as it may contain more functions than first diagramed or it may not contain some of the first diagrammed functions. However, the Agile methodology is still suited for this as new diagrams can be created if need be to suit the new application.

A limitation I discovered was that Last FM’s API, while it is perfect for my website, it doesn’t have any images of the artists stored. This means I will have to use the artist’s top album as their image in the results page. This is minor but I would have preferred to use an image of the artist.

Another limitation I discovered was that some of the functions for the website will be more difficult than others. What I mean by this is that the machine learning aspect of this project will take me a considerable amount of time, but it will still be longer than just getting the recommendations for the results page. Because of this, the machine learning aspect may not be completed in time for the development deadline as I will be learning how to code machine learning at the same time.

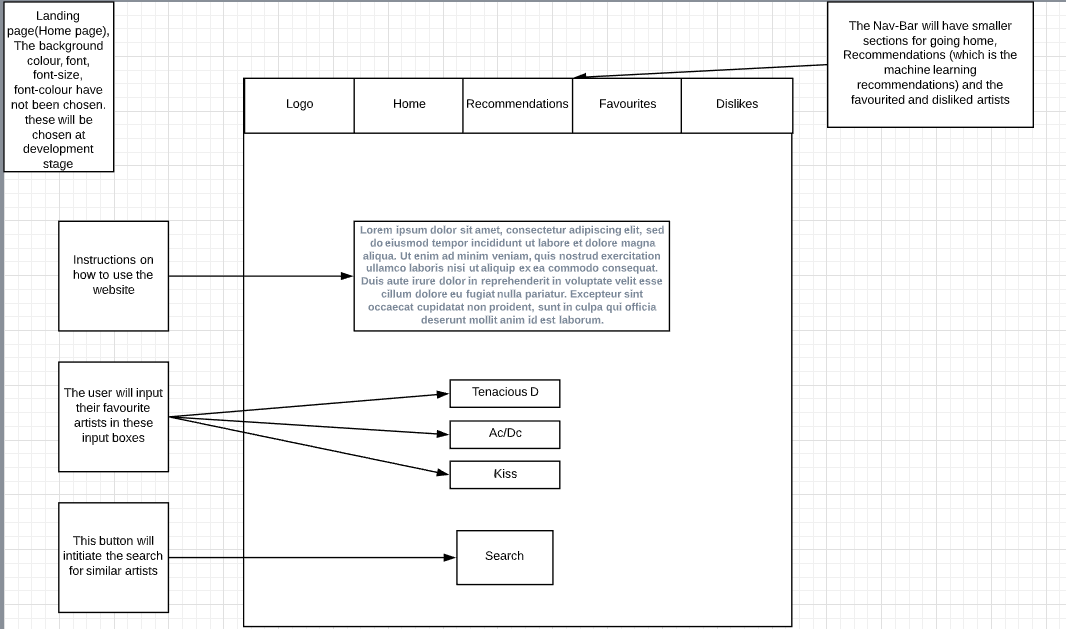
## 2.7 view model

I will hand in the Navigation Map and the Structured English at the development stage.

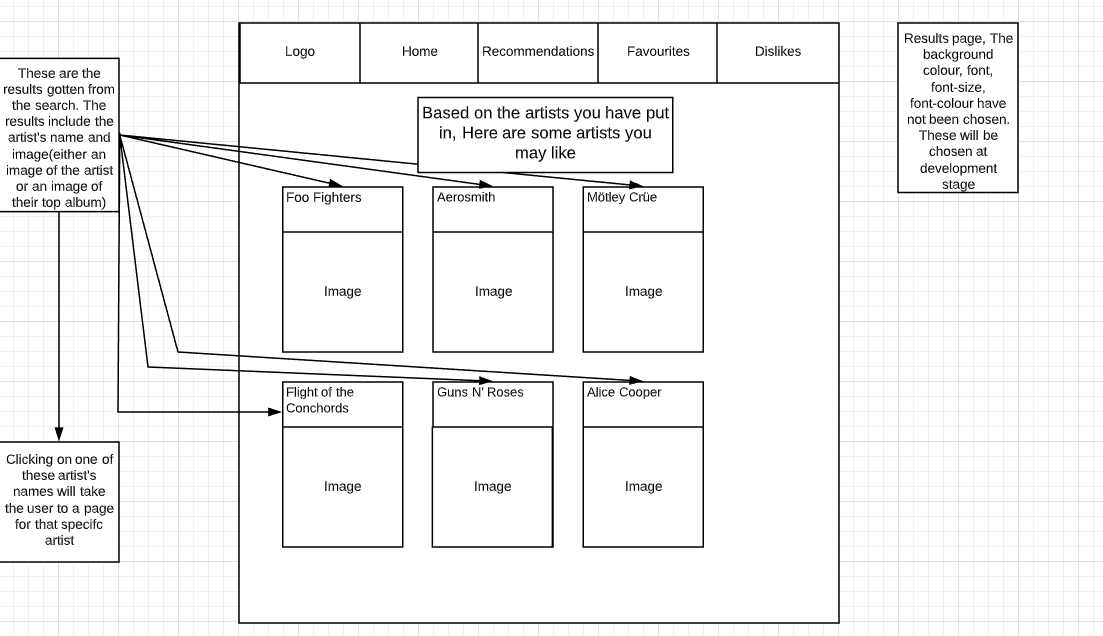
### Storyboards

For my view model, I’m going to create Storyboards. This is because Storyboards are a good way of making an idea of what the website should look like. I.e. Storyboards include the colour, general structure of the page and comments about what the text will say. For example, one of the Storyboards I will definitely be making is the home page. The home page will have a text box for the instructions on how to use the website, the input boxes for the user’s favourite artists and will have a button which will say be for starting the search. The Storyboard will also have the background colour, text colour, font and font size. The Storyboard should be accurate to the finished product as if there was a client I would be showing the client these Storyboards before I started development to make sure that I have met all of the requirements the client wants, and it allows the client to offer feedback on the general aesthetic of the website.

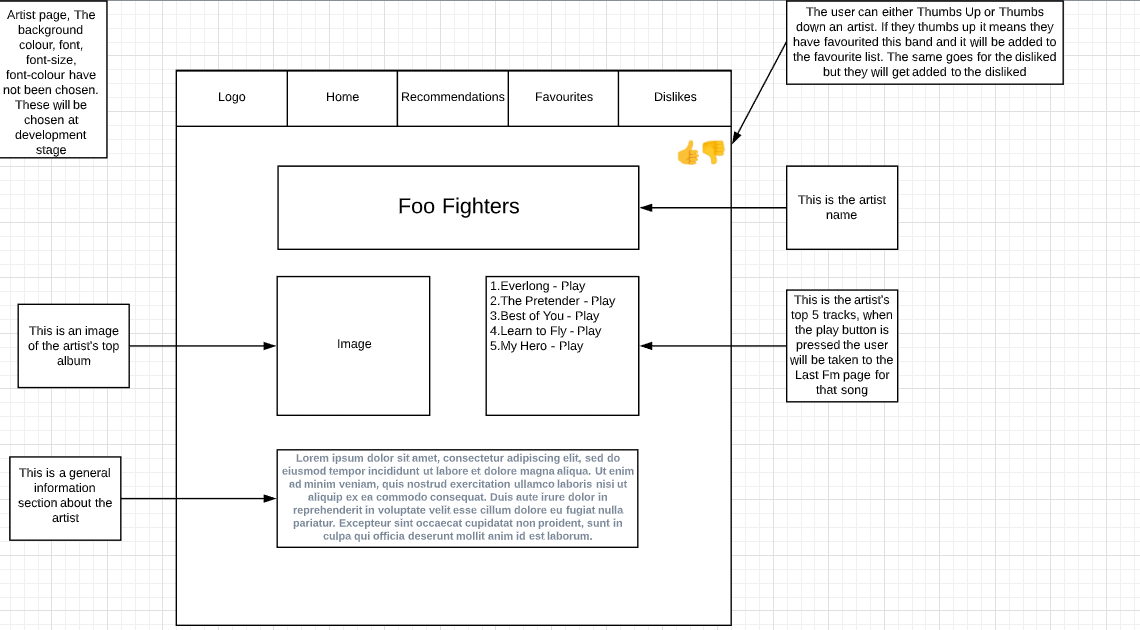
### Home Page



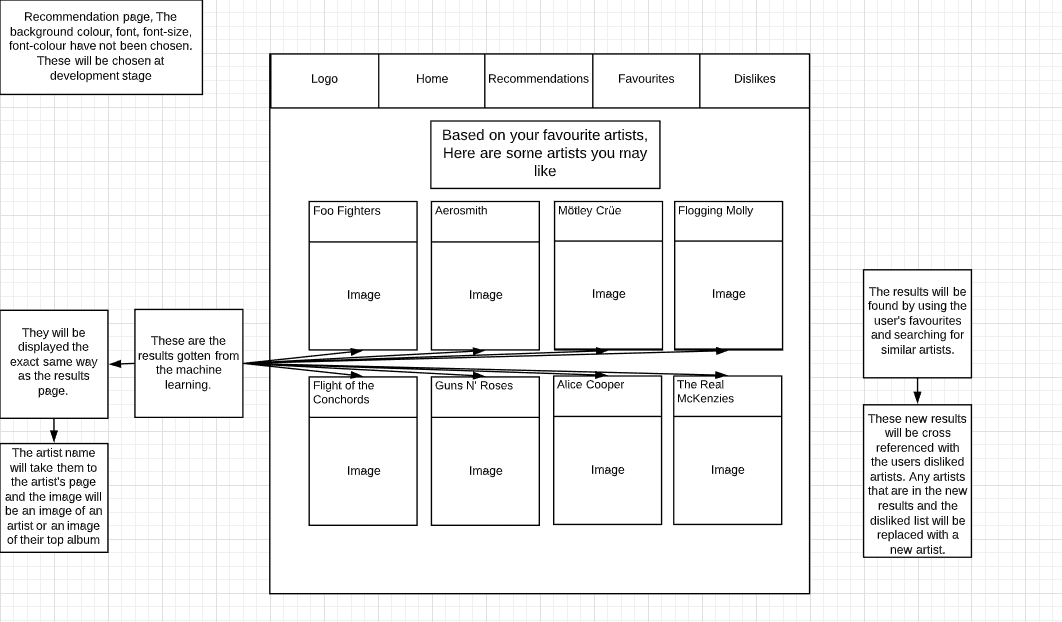
### Results Page



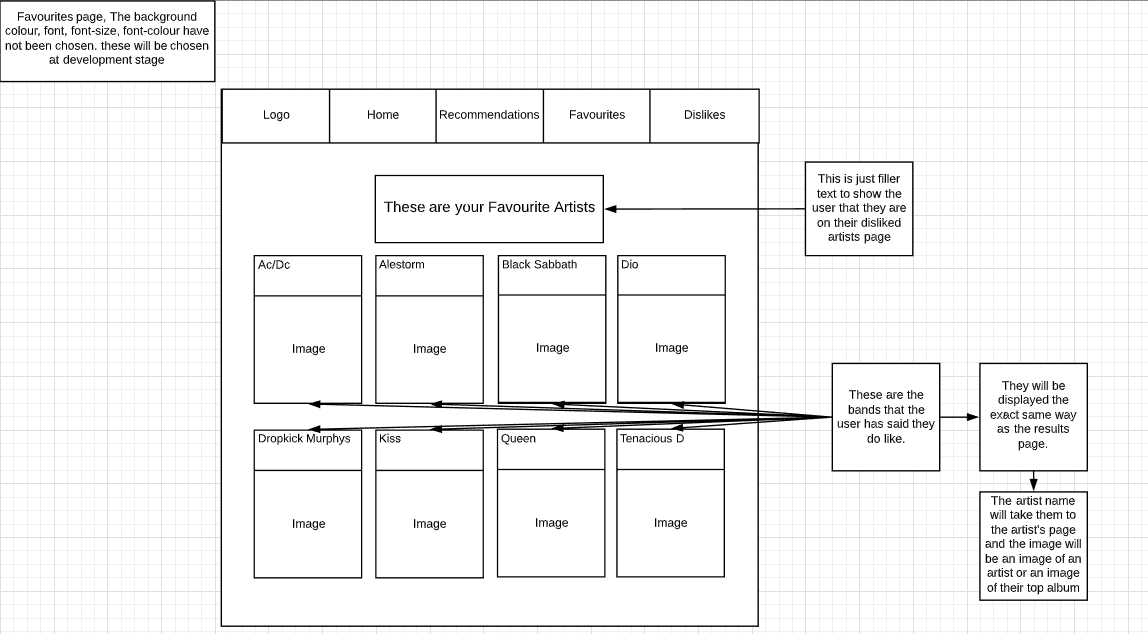
### Artist Page



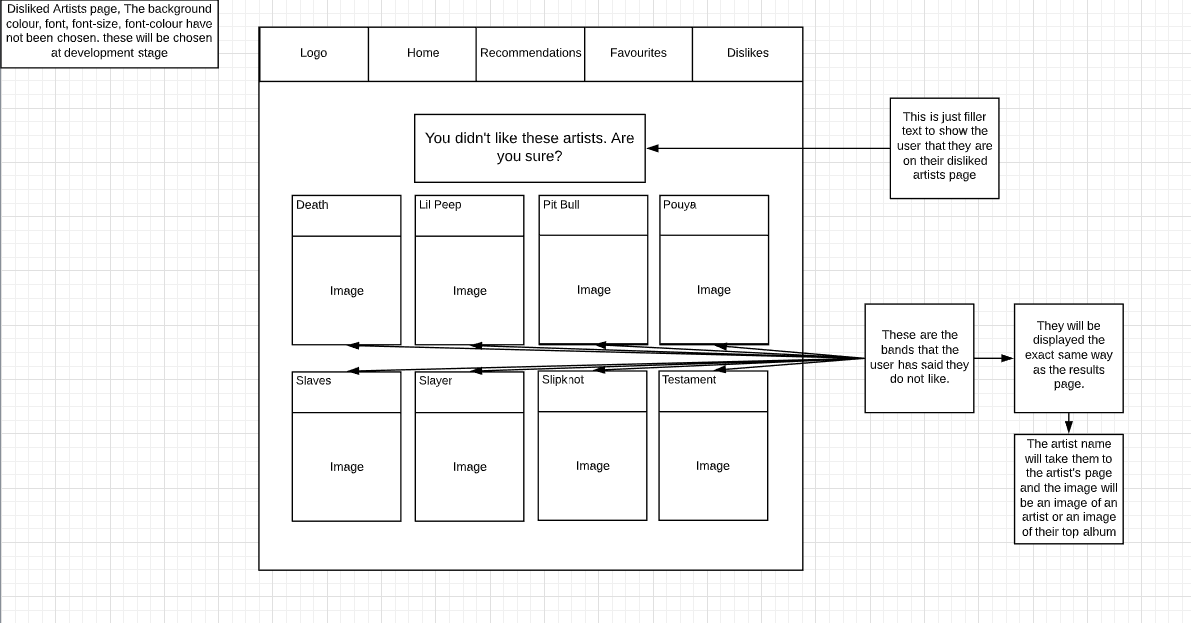
### Recommendation Page



### Favourite Artists Page



### Disliked Artist Page



### Navigation Map

I’m also going to create a Navigation Map for my view model. This is because Navigation Maps are useful when creating a website as it allows the developer to see exactly how the webpages are linked together. It can also show links to outside websites, so in my case the Navigation Map will show the link to Last FM’s page for artists songs. The Navigation Map will show each link between all the webpages in my website, but it won’t say what the link is. For example, my Navigation Map will show the connection between Artist and Last FM, but it won’t say why the connection is there. I will know why the connection is there but if anyone else looks at it on its own they will not be able to discern the connection. The Navigation Map I will create will be a simple one, but it is essential as it will help me ensure that when it comes to developing the website, all the links will be in place and that there will be no deadlinks.

### Structured English

For my view model, I’m going to create Structured English. This is because Structured English is useful when creating the JavaScript functions which will be used in my website. This is because Structured English is a very basic version of the coded functions that are required in the website. Structured English is written before the code and by doing this it serves as a precursor guide which can help developers figure out what the functions need to do. For example, the Structured English I will write for the get request will help me understand exactly what I need to get back from the get request and it will also help me come up with some variable names, so I have a good understanding of what to do when I start coding.

In this section, I will analyze the non-functional requirements I identified in my initial investigation. I will analyze my non-functional requirements because most, if not all, of them deal with how to make the website look good and feel good to use.

### Non-functional requirements

* **Website Colour scheme**

The first non-functional requirement I identified was that the website must have a consistent colour scheme throughout all the web pages. These colours should all be web safe. The reason I need to have a consistent colour scheme is to make the new webpages seem familiar to users that have only seen the home page of my website. The colours I will use will have to be web safe which means that the colour that is used must be available on every web browser. For example, if I use a certain shade of red it must be useable on chrome, Firefox and every other web browser to ensure that every user gets to experience the same website.

* **Readable text**

The second non-functional requirement is that the text on the website must be readable. What this means is that the text must be able to be read which means the font, font size and colour will be taken into consideration when deciding on the colour of the background. This is so every user can easily read the text that appears so in the case of my project, the text would be the artist’s name, the instructions on the home page for inputting the users favourite artists and the general information on the artist page. That’s just a few examples of where text will appear on my website.

* **Hyperlinks and deadlinks**

The third non-functional requirement is that all the links on the website must work and there should be no dead links. This means that all of the internal links which are links that will take the user to different web pages that are still contained within the webpage must work as well as all the external links which are links that will take the user to different webpages on different websites. A deadlink is hyperlink that should take the user to a website or another webpage, but the website or webpage has been deleted or has a new URL. If any of the hyperlinks I use in my website turn out to be a deadlink I will change them accordingly or I will remove them entirely.

* **Budget**

The fourth non-functional requirement is that the budget is £14000. I used this as in the real world, if this was requested by a client, I would need a budget as I would have been contracted to do so. This budget is made up but for the purposes of this project I have decided to give myself this budget as I believe it will cover all expenses that I would have in the real world such as minimum wage for example.

* **Deadline**

The fifth non-functional requirement is that the deadline for this website is 15th May 2020. This deadline is the deadline I have for the development part of the project. This means I will have fourteen weeks to develop and test my website. Because of this, I believe I have more than enough time to create a functioning website that can reliably recommend new artists to the user. I believe the machine learning aspect will take the longest because I will be learning how to code machine learning from scratch, but I still believe it’s possible for me to create a functioning piece of machine learning.

* **Security**

The sixth non-functional requirement is that the website must be secure so that no one can take over the website. This means that the website must have adequate security so that no one can take over the website and display anything they want. This requirement would be more important if there was a login system as I would need to protect the user’s username and password. However, security is still important as the website will need to be secure so the user can have a good experience without having to worry about any intruders or any viruses going on to their system.

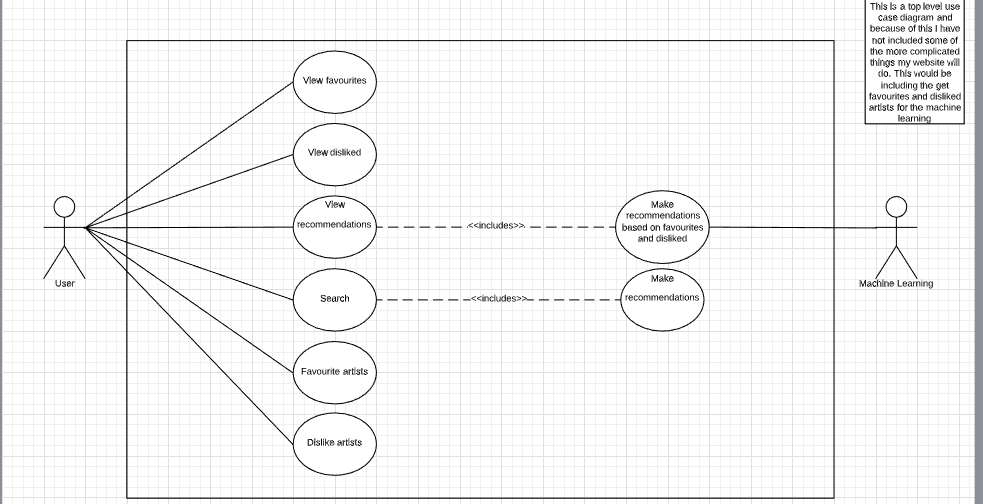
* **Reaction times**

The final non-functional requirement that I identified was that the website should have fast reaction times. This is so the user can have a good experience as the website should load quickly so the user doesn’t have to wait for anything. By doing this the user can do things on the website quicker and this makes the experience better as it would take them less time for them to correct a mistake in their favourite artists for example.

# 3. Developing

## 3.1 Diagrams

### Use Case Diagram



### Use Case Descriptions

|  |
| --- |
| **Use case:**  View favourites |
| **Requirements:**  The user must have favourited at least one artist |
| **Pre-conditions:**  One artist must be favourited |
| **Post-conditions:**  The user can see all of their favourited artists |
| **Invariants:** |
| **Scenario:**  Favourites tab clicked  Favourites page will be loaded  Favourites file will be opened  Favourite artists will be imported from Favourites file  Get Request for favourite artists  Display Artist name  Get request for Artists top album  Display top album image |

|  |
| --- |
| **Use case:**  View disliked |
| **Requirements:**  The user must have disliked at least one artist |
| **Pre-conditions:**  One artist must be disliked |
| **Post-conditions:**  The user can see all of their disliked artists |
| **Invariants:** |
| **Scenario:**  Disliked tab clicked  Disliked page will be loaded  Disliked file will be opened  Disliked artists will be imported from Disliked file  Get Request for Disliked artists  Display Artist name  Get request for Artists top album  Display top album image |

|  |
| --- |
| **Use case:**  Favourite artists |
| **Requirements:**  The user must have clicked on to an artist page |
| **Pre-conditions:**  The webpage must be an artist page |
| **Post-conditions:**  The user can add to their favourited artists |
| **Invariants:** |
| **Scenario:**  Search results have been displayed  User clicks on an artist  User clicks favourite button  Artist name gets added to Favourites file  Artist can now be seen on the favourites page |

|  |
| --- |
| **Use case:**  Dislike artists |
| **Requirements:**  The user must have clicked on to an artist page |
| **Pre-conditions:**  The webpage must be an artist page |
| **Post-conditions:**  The user can add to their disliked artists |
| **Invariants:** |
| **Scenario:**  Search results have been displayed  User clicks on an artist  User clicks dislike button  Artist name gets added to Disliked file  Artist can now be seen on the disliked page |

|  |
| --- |
| **Use case:**  View recommendations |
| **Requirements:**  The user must have one favourited artist |
| **Pre-conditions:**  One artist must be favourited |
| **Post-conditions:**  The user can view recommendations without searching |
| **Invariants:**  No artists have been favourited |
| **Scenario:**  Recommendations tab clicked  Recommendations page will be loaded  <<includes>> Import recommendations from the machine learning component  Get request for recommended Artists top album  Display Recommendations (These include the artist name and top album image) |

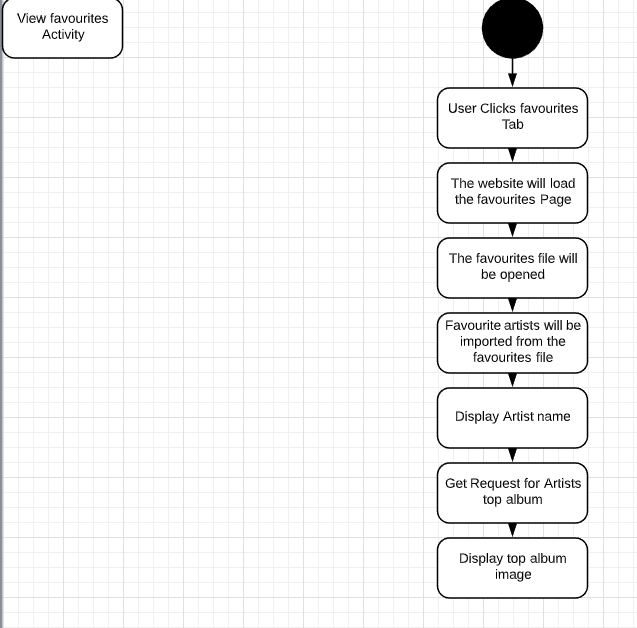
|  |
| --- |
| **Use case:**  Make recommendations based on favourites and disliked |
| **Requirements:**  At least one artist must be favourited |
| **Pre-conditions:**  One artist must be favourited |
| **Post-conditions:**  The user can view recommendations without searching |
| **Invariants:**  No artists have been favourited |
| **Scenario:**  The machine learning component will import the Favourites and Disliked Files  The machine learning component will open the Favourites and Disliked Files  The machine learning component will perform a get request for similar artists using artist names from the Favourites file  If any of the returned artists from the get request are in the Disliked file then  Remove those results and perform another get request on the artist the results came from until all of the results are not found in the recommendations  End if  The recommendations will be exported to the Recommendations page |

|  |
| --- |
| **Use case:**  Search |
| **Requirements:**  Three artists must have been inputted |
| **Pre-conditions:**  Three artists must have been put into the input boxes |
| **Post-conditions:**  The user can view recommendations tailored to a specific set of artists |
| **Invariants:**  Less than three artists have been inputted |
| **Scenario:**  Artists have been inputted  Search button clicked  Get Request using inputted artists for similar artists  Get Request using found artists to get the found artist’s top album image  Export Results |

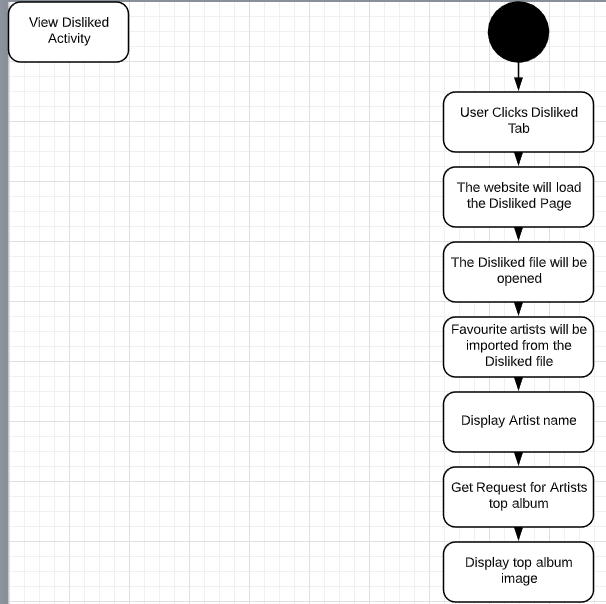
|  |
| --- |
| **Use case:**  Make Recommendations |
| **Requirements:**  Search button must have been clicked |
| **Pre-conditions:**  The search button on the home page must have been clicked |
| **Post-conditions:**  The user can view recommendations tailored to a specific set of artists |
| **Invariants:**  The search button hasn’t been clicked |
| **Scenario:**  Import results from home page to results page  Display results page  Display Artist name and Top album image |

### Activity Diagrams

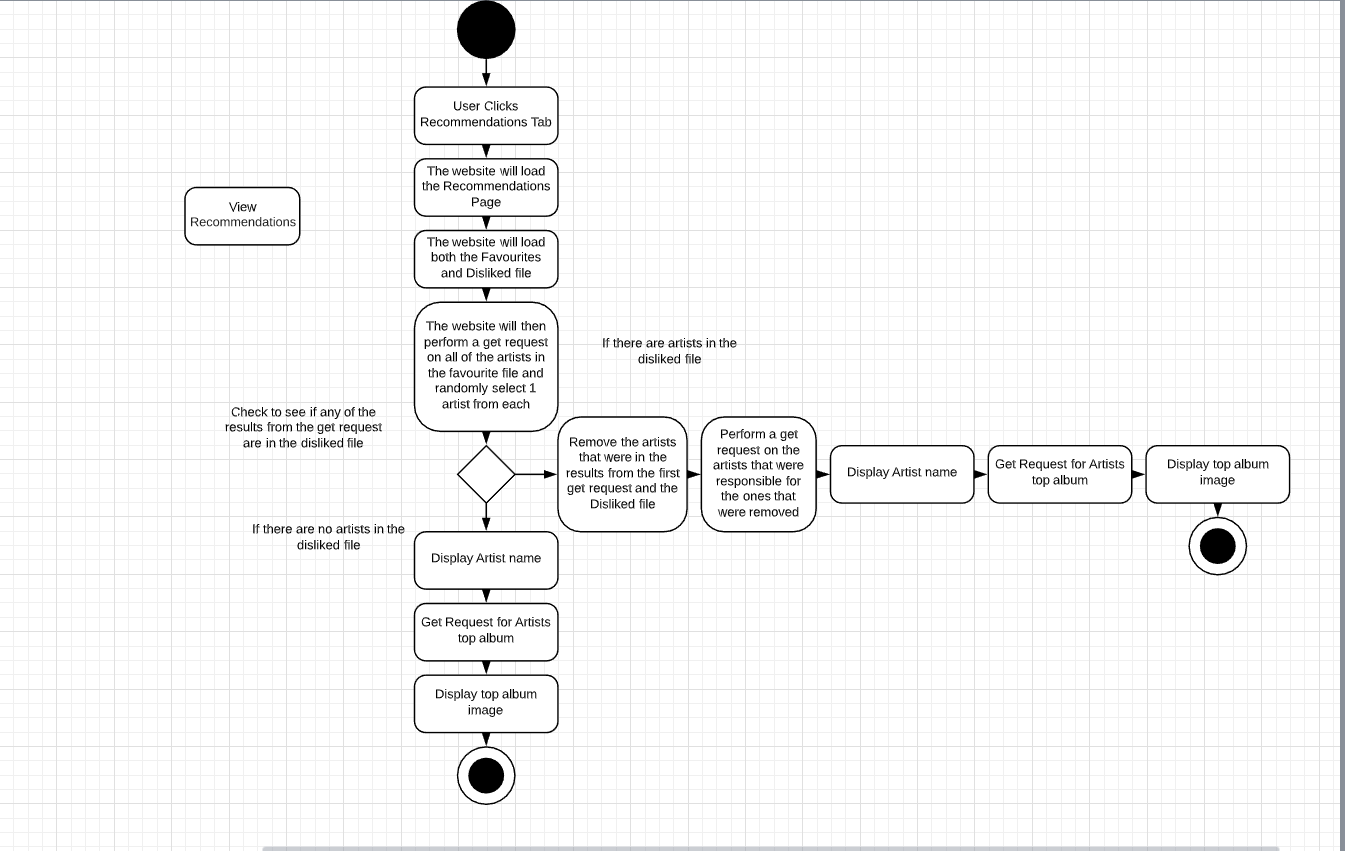
### View Favourites



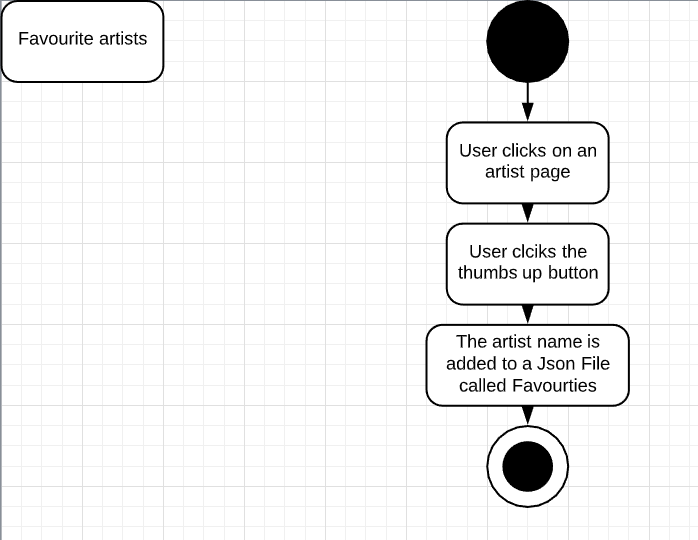
### View Disliked



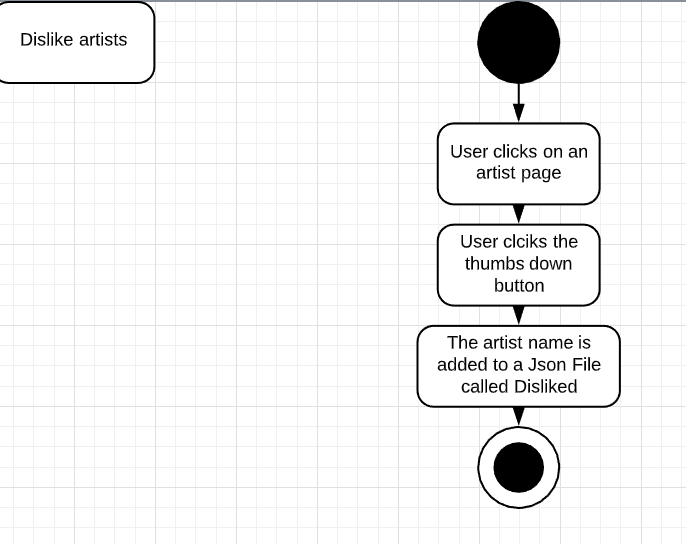
### View Recommendations



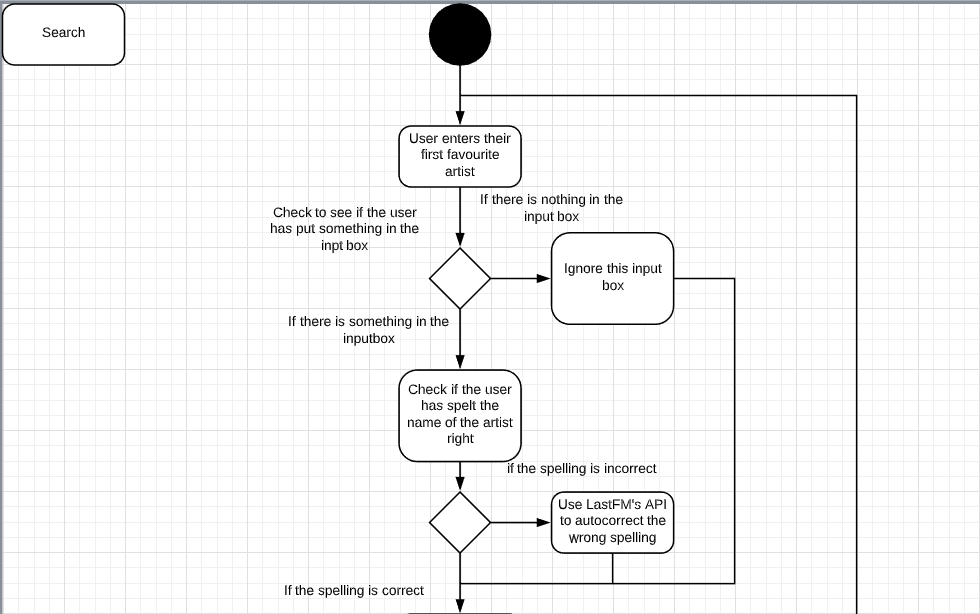
### Favourite artists

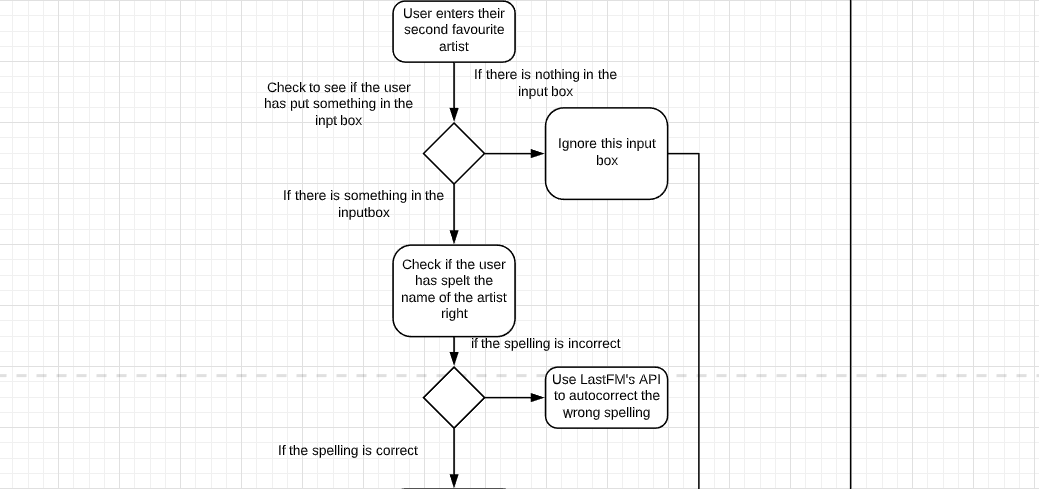


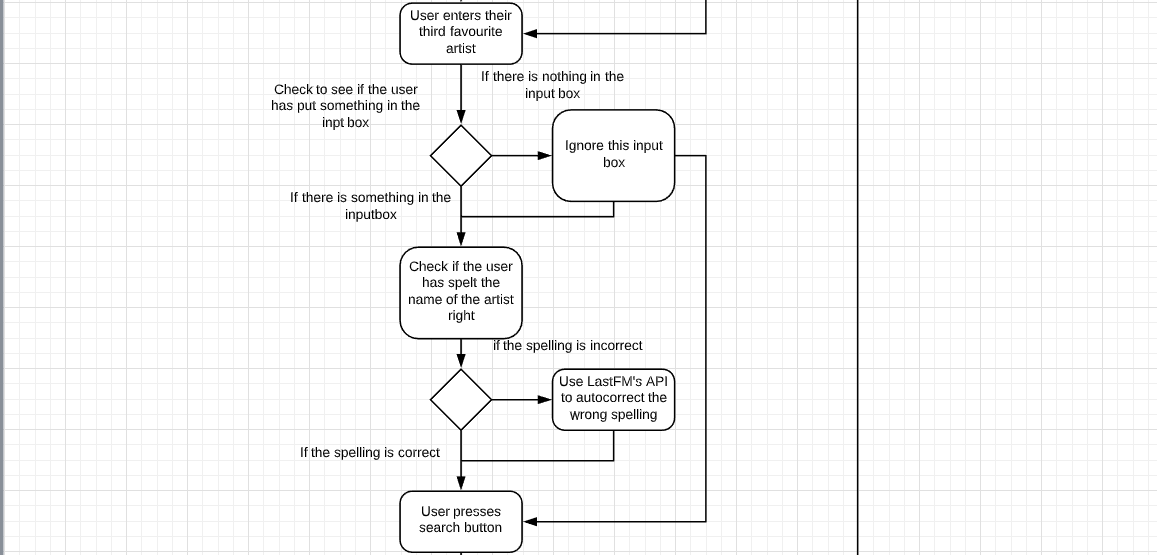
### Dislike Artists

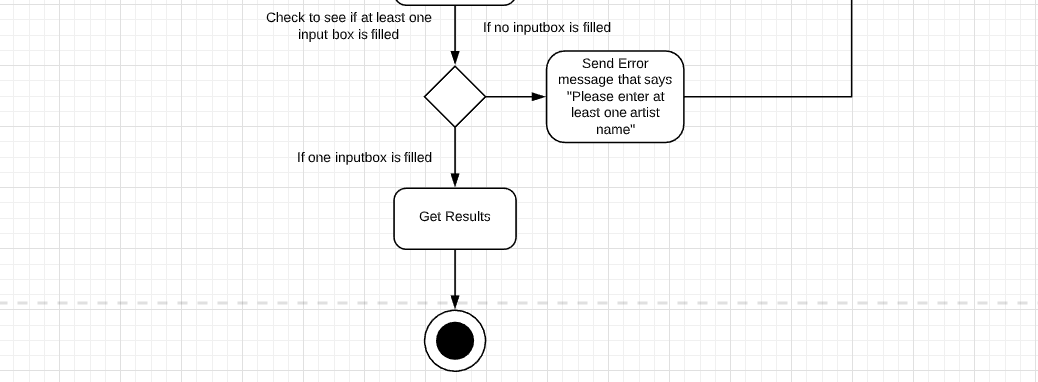


### Search

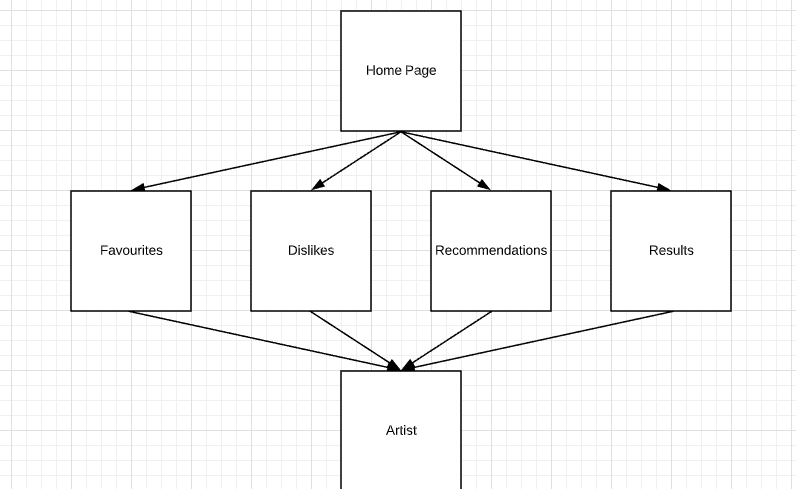








### Navigation Map



### Structured English

### Home

OUTPUT “Please enter your artist”

INPUT = choice1

IF choice1 ==”” THEN

searchButton is Disabled

ELSE IF choice1 !=”” THEN

searchButton is Enabled

EndIF

IF searchButton is clicked THEN

Get Request Last FM API (getsimilar)

IF resultsFromRequest ==”” THEN

OUTPUT “There is no similar artists”

ELSE

similarArtists = resultsFromRequest

GO TO REUSLTS COMPONENT

### Favourites

CALL getLikedArtists

READ likes.JSON from local directory

IF likes.JSON ISEMPTY THEN

OUTPUT “You have not liked any artists”

ELSE

SET favourites to likes.JSON file data

OUTPUT “Here are your favourite artists”

OUTPUT favourites

### Dislikes

CALL getDisikedArtists

READ dislikes.JSON from local directory

IF dislikes.JSON ISEMPTY THEN

OUTPUT “You have not disliked any artists”

ELSE

SET dislikes to dislikes.JSON file data

OUTPUT “Here are your disliked artists”

OUTPUT dislikes

### Recommendations

CALL getLikes

READ likes.JSON from local directory

SET likes to likes.JSON file data

IF likes.length > 0 THEN

SET userFavourites to likes

END IF

CALL getDislikes

READ dislikes.JSON from local directory

SET dislikes to dislikes.JSON file data

IF dislikes.length > 0 THEN

SET userDislikes to likes

END IF

CALL getRecommendtions

SET rand to a random number between 0 AND the userFavourites.length

SET randomArtist to userFavourites[rand].name

Get Request Last FM API (getsimilar using randomArtist)

SET recommendations to resultsFromRequest

CALL checkRecommendations

IF any of recommendations == any of userDislikes

SET checkedRecommendations to recommendations that != userDislikes

END IF

OUTPUT “Your Recommendations”

OUTPUT checkedRecommendations

### Results

READ choice1 from Home Component

CALL getSimilar

Get REQUEST Last FM API (getsimilar using choice1)

IF resultsFromRequest ==”” THEN

OUTPUT “You have beat me”

ELSE

similarArtists = resultsFromRequest

### Artist

GET mbid FROM URL

SET artistId to mbid

CALL getMainArtistInfo

GET REQUEST Last FM API (getInfo using artistId)

SET artistName = resultsFromRequest.artist.name

SET artistBio = resultsFromRequest.artist.bio.summary

SET similarArtists = resultsFromRequest.artist.similar.artist

CALL getTopFiveAlbums

GET REQUEST Last FM API (getTopAlbums using artistName)

PUSH the first five results onto topFiveAlbumsArray

SET topAlbums = to topFiveAlbumsArray

CALL getTopFiveSongs

GET REQUEST Last FM API (getTopTracks using artistName)

PUSH the first five results onto topFiveSongsArray

SET topSongs = to topFiveSongsArray

CALL getSimilarArtistInfo

FOR (I=0, I < 5, I++)

GET REQUEST Last FM API (getInfo using similarArtist[I])

SET SimilarArtistInfo = resultsFromRequest

END LOOP

IF Like\_Btn clicked THEN

CALL addToLikes

READ Likes file

READ Dislikes file

IF artistName IS IN Likes file THEN

OUTPUT “You have already liked this artist”

ELSE IF artistName IS IN Dislikes file THEN

OUTPUT “You have disliked this artist, they will be added to your likes”

REMOVE artistName AND artistId FROM Dislikes file

PUSH artistName AND artistId to Likes file

ELSE IF artistName IS NOT IN Likes file AND artistName IS NOT IN Dislikes file THEN

PUSH artistName AND artistId to Likes file

END IF

END IF

IF Dislike\_Btn clicked THEN

CALL addToDislikes

READ Dislikes file

READ Likes file

IF artistName IS IN Dislikes file THEN

OUTPUT “You have already disliked this artist”

ELSE IF artistName IS IN Likes file THEN

OUTPUT “You have liked this artist, they will be added to your dislikes”

REMOVE artistName AND artistId FROM Likes file

PUSH artistName AND artistId to Dislikes file

ELSE IF artistName IS NOT IN Dislikes file AND artistName IS NOT IN Likes file THEN

PUSH artistName AND artistId to Dislikes file

END IF

END IF

## 3.2 Implementation

### Code listings

### App.JS

// Main app entry point that loads the basic framework and this also defines the routes so the components can be used.

//This is where I learned about react and how to use it:

//https://codingthesmartway.com/the-mern-stack-tutorial-building-a-react-crud-application-from-start-to-finish-part-1

//My Dad and I worked on a react project before I started this so I used that experience to help me with this project as I was able to look back

//And it helped me with some parts of this such as the routes and reusable components

//Import and Export statements

//where I learned about browserify-fs

//https://www.npmjs.com/package/browserify-fs

//This is used for local storage as browserify-fs makes it eaier to read nad write to

//Local storage is storage that is local to the browser

//Fore example, if I liked Ac/Dc on this PC and then went to a laptop, Ac/Dc would not be liked

//The LAST\_FM\_KEY is the API key I was given when I created my API account with Last FM

//This key is needed whenever I use one of the methods from Last FM

//For example, I will need it when I call getInfo in artist component and when I call

//getSimilar in the Home component

import React, { Component } from "react";

import { BrowserRouter as Router, Route, Link } from "react-router-dom";

import "bootstrap/dist/css/bootstrap.min.css";

import Home from "./components/home.component";

import results from "./components/results.component";

import artist from "./components/artist.component.js";

import recommendations from "./components/recommendations.component";

import favourites from "./components/favourites.component";

import dislikes from "./components/dislikes.component";

import Logo from "./Logo.png";

import Search from "./components/search.component";

import searchedArtist from "./components/searchedArtist.component";

import fs from "browserify-fs"

export const LAST\_FM\_KEY = '686acc4c7ab472a332f7944062df7518'

export const LIKES\_AND\_DISLIKES\_DIR = '/LikesAndDislikes';

export const DISLIKES\_JSON = LIKES\_AND\_DISLIKES\_DIR + '/Dislikes.json';

export const LIKES\_JSON = LIKES\_AND\_DISLIKES\_DIR + '/Likes.json';

class App extends Component {

  //The constructor: this creates the likes and dislikes files in the LikesAndDislikes directory in local storage

  //This means that if a user closes the original page, and then goes back to it later,

  //The likes and dislikes files are still in local storage

  //fs.stat is used to check if the file exist within the LikesAndDislikes folder

  //If they do exist then nothing happens

  //But if the files don't and empty array gets pushed onto them so the files have been intialised

  constructor(props) {

    super(props);

    fs.stat(LIKES\_AND\_DISLIKES\_DIR, (err) => {

      // Intialise the files if they don't exist

      if (err) {

        fs.mkdir(LIKES\_AND\_DISLIKES\_DIR, () => {

          let artists = {

            artists: []

          }

          fs.writeFile(LIKES\_JSON, JSON.stringify(artists), (err) => {

            if (err) {

              console.log("Failed to create Likes file");

            }

          });

          fs.writeFile(DISLIKES\_JSON, JSON.stringify(artists), (err) => {

            if (err) {

              console.log("Failed to create Dislikes file");

            }

          });

        });

      }

    })

  }

//This is what gets shown on the website at all times.

//This is the Navbar and this shows the different pages users can go to from the start

//Some will be empty(Reccommendations, Favourites and dilikes) because the user will not have liked/disliked anyone

//So no reccomendations can be created, and the favourites/dilikes cannot be shown

//Based on the route, it will load the relevant components

  render() {

    return (

      <Router>

        <div className="container">

          <nav class="col-sm-12" className="navbar navbar-expand navbar-light navbar-center">

            <a className="navbar-item">

              <img className="Logo-spin" src={Logo} width="40" height="40" alt="Logo" />

            </a>

            <div className="collpase navbar-collapse">

              <ul className="navbar-nav mr-auto">

              <li className="navbar-item">

                  <Link to="/" className="nav-link">Home</Link>

                </li>

                <li className="navbar-item mr-auto">

                  <Link to="/search" className="nav-link">Search</Link>

                </li>

                <li className="navbar-item mr-auto">

                  <Link to="/recommendations" className="nav-link">Recommendations</Link>

                </li>

                <li className="navbar-item mr-auto">

                  <Link to="/favourites" className="nav-link">Liked</Link>

                </li>

                <li className="navbar-item mr-auto">

                  <Link to="/dislikes" className="nav-link">Disliked</Link>

                </li>

              </ul>

            </div>

          </nav>

          <br />

          <br />

          {/\* This sets up the routes. These routes tell the website, what component to load when they are on a specific route.

          For example, if i was on the route "/" the Home component would be loaded onto the screen for the user to use \*/}

          <Route path="/" exact component={Home} />

          <Route path="/recommendations" component={recommendations} />

          <Route path="/favourites" component={favourites} />

          <Route path="/dislikes" component={dislikes} />

          <Route path="/results" component={results} />

          <Route path="/artist/:id" component={artist} />

          <Route path="/searchedArtist" component={searchedArtist} />

          <Route path="/search" component={Search} />

        </div>

      </Router>

    );

  }

}

//Allows other components to import this one

export default App;

### APP.CSS

.App {

  text-align: center;

}

.App-logo {

  height: 40vmin;

  pointer-events: none;

}

@media (prefers-reduced-motion: no-preference) {

  .App-logo {

    transform: rotate(90deg);

  }

}

/\* https://codepen.io/WhirlwindRhyme/pen/PwrMYE \*/

.Logo-spin {

  -webkit-animation: spin 5s infinite;

  animation: spin 5s infinite;

}

@-webkit-keyframes spin{

  from {

    -webkit-transform: rotateY(0deg);

  }

  to {

    -webkit-transform: rotateY(-360deg);

  }

}

@keyframes spin{

  from {

    -moz-transform: rotateY(0deg);

    -ms-transform: rotateY(0deg);

    transform: rotateY(0deg);

  }

  to {

    -moz-transform: rotateY(-360deg);

    -ms-transform: rotateY(-360deg);

    transform: rotateY(-360deg);

  }

}

body {

  background-color: beige;

}

#Like {

  margin-right: 1rem;

  background-color: green;

}

#Dislike {

  background-color: red;

}

.btn-float-right {

  float: right;

}

.users-choice {

  margin-right: 1rem;

}

#SearchArtists {

  margin-left: 1rem;

  background-color: dodgerblue;

}

h1, h2, p {

  text-align: center;

}

.lists {

  text-align: center;

  list-style-type: none;

}

.center-data {

  text-align: center;

}

.btn-center {

  float: right;

}

#btn-search {

  background-color: dodgerblue;

}

.navbar-center {

  position: absolute;

  left: 50%;

  transform: translatex(-50%);

}

.App-header {

  background-color: #282c34;

  min-height: 100vh;

  display: flex;

  flex-direction: column;

  align-items: center;

  justify-content: center;

  font-size: calc(10px + 2vmin);

  color: white;

}

.App-link {

  color: #61dafb;

}

@keyframes App-logo-spin {

  from {

    transform: rotate(0deg);

  }

  to {

    transform: rotate(360deg);

  }

}

### Home component

//Import statements

//where I learned about axios

//https://www.npmjs.com/package/axios

import React, { Component } from 'react';

import "../App.css"

import axios from 'axios';

import { LAST\_FM\_KEY } from '../App'

// Allows other components to import this one

export default class Home extends Component {

    //The constructor: Just intialises the variable "choice1" on the state

    //It's called choice 1 because originally I had planned for the user to put in 3 selections and get a certain amount of results

    //from each of them. However, when I started developing this, I chose to start off with one choice and then go to two and then three

    //I have ran out of time to add the other two choices but I implemented a search function which allow the user to search for a specific

    //and be taken to their page if it's available

    constructor(props) {

        super(props);

        //This was used to test the API key: console.log(LAST\_FM\_KEY)

        //I did so to ensure that the API key was getting imported correctly

        this.state = {

            choice1: ''

        }

    }

    //This updates the state every time the user types in the input box

    handleChoice1Change = (event) => {

        this.setState({

            choice1: event.target.value

        })

    }

    //This is the first instance of me using Last FM's API. This calls the API and uses the getSimilar method to find similar artists

    //Using the user's input from the input box.

    //Once the API call has been used the results are set equal to a constant variable(These can't be changed)

    //Then the url is updated and the const is pushed through to the results component

    getSimilarArtists = () => {

        axios.get('http://ws.audioscrobbler.com/2.0/?method=artist.getsimilar&artist=' + this.state.choice1 + '&api\_key=' + LAST\_FM\_KEY + '&format=json')

            .then(res => {

                //console.dir - redirects the JSON object to the chrome console so you can look into it

                //console.dir(res);

                if (!res.data.similarartists) {

                    alert("There is no similar artists for the artist you searched");

                } else {

                    const similarArtists = res.data.similarartists;

                    this.props.history.push({ pathname: '/results', state: { similarArtists: similarArtists } })

                }

                // console.log - prints out a simple attribute value

                //console.log(res.data.similarartists.artist[0].name);

            });

    }

    //This renders a small bit of information for the user to read and know what the website is about

    //There is an instruction as well so the user knows how to operate the website as well

    //Underneath the instruction there is an input box for the user to type in their artist

    //And to the side of that there is a button which will perform the getSimilarArtists method once it has been clicked

    render() {

        return (

            <div>

                <h1>Welcome to music recommender!!</h1>

                <p>Please put in an artist you like. This will get you some artists that you may like!</p>

                <div className="center-data">

                    <label className='users-choice'>Your artist:</label>

                    <input

                        type="text"

                        value={this.state.choice1}

                        onChange={this.handleChoice1Change}

                    />

                    {/\* This button is always disabled if the input box is empty and the only way to enable it is if the input box gets a single character in it \*/}

                    <button id="SearchArtists" className="btn btn-primary" type="submit" onClick={this.getSimilarArtists} disabled={!this.state.choice1}>Search</button>

                </div>

            </div>

        )

    }

}

### Results component

//Import statements

import React, { Component } from 'react';

import "../App.css"

import ArtistResultSummary from './artistResultSummary.component';

//Allows other components to import this one

export default class Results extends Component {

    //The constructor: This intialises the variables on the state

    //One of them is the similar artists from the home component

    //The other is mainMessage which will always be one of two things

    //The first is an error which is "You have beat me! I can't think of anyone to recommend, please go home and search again!!"

    //The other is the actual message which should always be displayed unless there is no similar artists

    //The message is "Here are some artists you may like!!"

    //The statement "this.goHome = this.goHome.bind(this);" is used so "this" can be accessed within the method

    constructor(props) {

        super(props);

        this.state = {

            similarArtists: this.props.location.state ? this.props.location.state.similarArtists : '',

            mainMessage: 'Here are some artists you may like!!'

        }

        this.goHome = this.goHome.bind(this);

    }

    //This is the first method that is accessed after the constructor

    //This checks if there similarArtists exists

    //Then it checks if there similarArtists is populated with data

    //If it's not then the mainMessage gets set to the error message

    //Otherwise nothing happens

    componentDidMount() {

        if (this.props.location.state && this.props.location.state.similarArtists) {

            if (this.state.similarArtists.artist.length === 0) {

                this.setState({

                    mainMessage: 'You have beat me! I can\'t think of anyone to recommend, please go home and search again!!'

                })

            }

        }

    }

    //This just allows the user to go to the home page on a button click

    goHome() {

        this.props.history.push('/');

    }

    //First off the mainMessage will be displayed for the user to see.

    //Second this renders the similarArtists found on the Home page by using the slice and map function

    //Where i learned about map: https://reactjs.org/docs/lists-and-keys.html & https://www.youtube.com/watch?v=It9iL4EXFWc

    //All Slice does is it take the results and(what I have done) starts at 0 and Slices everything off after the 10th result

    //Map is used to create the list items. In this instance, the artist and the key gets passed in to ArtistResultSummary

    //And returns that component 10 times because of the slice function

    //Finally the button uses the goHome method to allow the user an easy way to go back to the home page rather than pressing the back button

    render() {

        return (

            <div>

                <p>{this.state.mainMessage}</p>

                <ul className="lists">

                    {this.props.location.state.similarArtists.artist.slice(0, 10).map((artist, key) => {

                        return <ArtistResultSummary artist={artist} />

                    })}

                </ul>

                        {<button id="btn-search" className="btn btn-primary btn-center " onClick={this.goHome} type="submit">Search again!</button>}

                </div>

        )

    }

}

### ArtistResultSummary component

//Import statements

import React, { Component } from 'react';

import { Link } from 'react-router-dom';

import "../App.css"

// Allows other components to import this one

export default class ArtistResultSummary extends Component {

    render() {

        return (

            //This creates the list item that is used in the unique lists in the results and artist component(it is used for the similar artist map)

            //All it does is print out the artists' name which links to /artist/the artists' mbid

            //However, if the artist that should be passed in doesn't exist, it outputs "This artist can't be displayed. Sorry!"

            <li key={this.props.artist.mbid}>

               {this.props.artist ? <Link to={"/artist/" + this.props.artist.mbid}>{this.props.artist.name}</Link> : 'This artist can\'t be displayed. Sorry!'}

            </li>

        )

    }

}

### Artist component

//Import statements

import React, { Component } from 'react';

import "../App.css"

import { LAST\_FM\_KEY, LIKES\_JSON, DISLIKES\_JSON } from '../App'

import axios from 'axios';

import ArtistResultSummary from './artistResultSummary.component';

import ArtistTopAlbum from './artistTopAlbum.component';

import ArtistTopSong from './artistTopSong.component';

import fs from "browserify-fs";

import "bootstrap/dist/css/bootstrap.min.css";

// Allows other components to import this one

export default class Artist extends Component {

    //The constructor: This intialises the variables on the state

    //All of these are created in this component

    //The statement "this.myMethod = this.myMethod.bind(this)"" is used so "this" can be accessed within that method

    constructor(props) {

        super(props);

        this.state = {

            mainArtist: {},

            artistName: '',

            artistBio: '',

            similarArtists: [],

            similarArtistInfo: [],

            artistId: '',

            topAlbums: [],

            topSongs: [],

            isInLikes: false,

            isInDislikes: false

        }

        this.getMainArtistInfo = this.getMainArtistInfo.bind(this);

        this.getSimilarArtistInfo = this.getSimilarArtistInfo.bind(this);

        this.getArtistId = this.getArtistId.bind(this);

        this.getTopFiveAlbums = this.getTopFiveAlbums.bind(this);

        this.getTopFiveSongs = this.getTopFiveSongs.bind(this);

        this.goHome = this.goHome.bind(this);

        this.addToLikes = this.addToLikes.bind(this);

        this.addToDislikes = this.addToDislikes.bind(this);

        this.checkIfInLikes = this.checkIfInLikes.bind(this);

        this.checkIfInDislikes = this.checkIfInDislikes.bind(this);

    }

    //This is the first method that is accessed after the constructor

    //Calls getArtistID first so getMainArtistInfo can be called

    componentDidMount() {

        this.getArtistId();

        this.getMainArtistInfo();

    }

    //This is used if the route is updated

    //It checks to see if the updated id is not the same as the artistId

    //If not it changes the artistId to be the updated id

    //And then calls getMainArtistInfo and updates the entire component

    //It is used in conjunction with getDerivedStateFromProps to update the component

    componentDidUpdate(nextProps) {

        if (nextProps.match.params.id !== this.state.artistId) {

            this.setState({

                artistId: nextProps.match.params.id

            })

            this.getMainArtistInfo();

        } else {

            return null;

        }

    }

    //This is used if the route is updated

    //It checks to see if the updated id is not the same as the artistId

    //if not it changes the artistId to be the updated id

    //It is used in conjunction with componentDidUpdate to update the component

    static getDerivedStateFromProps(nextProps, prevState) {

        if (nextProps.match.params.id !== prevState.artistId) {

            return { artistId: nextProps.match.params.id }

        }

    }

    //This is used to get the artistId from the URL parameters

    //The Id is put in after the artist part of the URL

    //For example, here is ozzy Osbourne's URL : "http://localhost:3000/artist/'8aa5b65a-5b3c-4029-92bf-47a544356934'"

    //The part in single quotes is the artistId

    getArtistId = () => {

        let url = window.location.href;

        let artistUrl = url.split('/');

        this.state.artistId = artistUrl[4];

    }

    //This is Last FM's API getting called again

    //However, I am using a different method from Last FM

    //I am using the getInfo which requires the artist's name or their mbid

    //Once the get request has occured I am setting some of the things I can onto the state

    //So I can use their values elsewhere

    //After the state gets set, getTopFiveAlbums and getTopFiveSongs get called as well

    //It performs a check before calling getSimilarArtistInfo because some artist's don't have any that are similar to them

    getMainArtistInfo = () => {

        axios.get('http://ws.audioscrobbler.com/2.0/?method=artist.getinfo&api\_key=' + LAST\_FM\_KEY + '&format=json&mbid=' + this.state.artistId)

            .then(res => {

                this.setState({

                    artistName: res.data.artist.name,

                    artistBio: res.data.artist.bio.summary.replace("<a href", '<a target="\_blank" href'),

                    similarArtists: res.data.artist.similar.artist

                })

                this.checkIfInLikes();

                this.getTopFiveAlbums();

                this.getTopFiveSongs();

                if (this.state.similarArtists.length > 0) {

                    this.getSimilarArtistInfo();

                }

            })

    }

    //This checks the likes file for the current artist's Id

    //If the current artist Id is in the likes file then the Like btn will be disabled so the user cannot like them again

    //If it is not in the file though it will make sure the button is enabled

    checkIfInLikes = () => {

        var i;

        if (this.state.artistId) {

            fs.readFile(LIKES\_JSON, 'utf-8', (err, likeData) => {

                let likes = JSON.parse(likeData);

                for (i = 0; i < likes.artists.length; i++) {

                    if (likes.artists[i].mbid === this.state.artistId) {

                        this.setState({

                            isInLikes: true,

                            isInDislikes: true

                        })

                        break;

                    }

                }

                if (this.state.isInLikes === false) {

                    this.checkIfInDislikes();

                }

            })

        }

    }

//This checks the dislikes file for the current artist's Id

//If the current artist Id is in the dislikes file then the Dislike btn will be disabled so the user cannot dislike them again

//If it is not in the file though it will make sure the button is enabled

checkIfInDislikes = () => {

    var i;

    if (this.state.artistId) {

        fs.readFile(DISLIKES\_JSON, 'utf-8', (err, dislikeData) => {

            let dislikes = JSON.parse(dislikeData);

            for (i = 0; i < dislikes.artists.length; i++) {

                if (dislikes.artists[i].mbid === this.state.artistId) {

                    this.setState({

                        isInLikes: true,

                        isInDislikes: true

                    })

                    break;

                }

            }

        })

    }

}

//This is Last FM's API getting called again

//However, I am using a different method from Last FM

//I am using the getTopAlbums which requires the artist's name or their mbid

//Because of getMainArtistInfo, I am able to use the artist name because getInfo gave that information

//In order for me to get the top five albums I used a for loop to push the first five results on to an

//Array. This array is then set to topAlbums on the state so it can be accessed elsewhere in the component

getTopFiveAlbums = () => {

    var i;

    var topFiveAlbums = [];

    axios.get('http://ws.audioscrobbler.com/2.0/?method=artist.gettopalbums&artist=' + this.state.artistName + '&api\_key=' + LAST\_FM\_KEY + '&format=json')

        .then(res => {

            for (i = 0; i < 5; i++) {

                topFiveAlbums.push(res.data.topalbums.album[i])

            }

            this.setState({

                topAlbums: topFiveAlbums

            })

        })

}

//This is Last FM's API getting called again

//However, I am using a different method from Last FM

//I am using the getTopTracks which requires the artist's name or their mbid

//Because of getMainArtistInfo, I am able to use the artist name because getInfo gave that information

//In order for me to get the top five tracks I used a for loop to push the first five results on to an

//Array. This array is then set to topSongs on the state so it can be accessed elsewhere in the component

getTopFiveSongs = () => {

    var i;

    var topFiveSongs = [];

    axios.get('http://ws.audioscrobbler.com/2.0/?method=artist.gettoptracks&artist=' + this.state.artistName + '&api\_key=' + LAST\_FM\_KEY + '&format=json')

        .then(res => {

            for (i = 0; i < 5; i++) {

                topFiveSongs.push(res.data.toptracks.track[i])

            }

            this.setState({

                topSongs: topFiveSongs

            })

        })

}

//This is Last FM's API getting called again

//However, I am using a different method from Last FM

//I am using the getInfo which requires the artist's name or their mbid

//Because of getMainArtistInfo, I was able to set an array up of the similar artists

//I am using getInfo so I can set an array up of all the similar artists' info

getSimilarArtistInfo = () => {

    var i;

    var similarArtistData = [];

    for (i = 0; i < this.state.similarArtists.length; i++) {

        axios.get('http://ws.audioscrobbler.com/2.0/?method=artist.getinfo&artist=' + this.state.similarArtists[i].name + '&api\_key=' + LAST\_FM\_KEY + '&format=json')

            .then(res => {

                similarArtistData.push(res.data.artist)

                this.setState({

                    similarArtistInfo: similarArtistData

                })

            })

    }

}

//This just allows the user to go to the home page on a button click

goHome() {

    this.props.history.push('/');

}

//This is used to add the artits(whose page you are on) to the likes file

//So they will appear in the likes component and will be used to find recommendations in then recommendations component

//It checks first to see if the artist has an Id because some artists do not

//If they don not have an Id an errorr message will tell the user that the artist could not be added to the likes file

//It then reads the likes file into an array, which allows the variable currentArtistId to be pushed on to it

//It then writes this array back to the likes file

addToLikes = () => {

    if (this.state.artistId) {

        let currentArtistId = { "mbid": "" + this.state.artistId + "", "name": "" + this.state.artistName };

        fs.readFile(LIKES\_JSON, 'utf-8', (err, likeData) => {

            let likes = JSON.parse(likeData);

            likes.artists.push(currentArtistId)

            fs.writeFile(LIKES\_JSON, JSON.stringify(likes), (err) => {

                if (err) {

                    console.dir(err);

                } else {

                    this.setState({

                        isInLikes: true,

                        isInDislikes: true

                    })

                }

            })

        });

    } else {

        alert("This artist can't be added to your likes. Sorry for this, please enjoy searching the music reccomender!!");

    }

}

//This is used to add the artits(whose page you are on) to the dislikes file

//So they will appear in the dislikes component

//It checks first to see if the artist has an Id because some artists do not

//If they don not have an Id an errorr message will tell the user that the artist could not be added to the dislikes file

//It then reads the dislikes file into an array, which allows the variable currentArtistId to be pushed on to it

//It then writes this array back to the dislikes file

addToDislikes = () => {

    if (this.state.artistId) {

        let currentArtistId = { "mbid": "" + this.state.artistId + "", "name": "" + this.state.artistName };

        fs.readFile(DISLIKES\_JSON, 'utf-8', (err, dislikeData) => {

            let dislikes = JSON.parse(dislikeData);

            dislikes.artists.push(currentArtistId)

            fs.writeFile(DISLIKES\_JSON, JSON.stringify(dislikes), (err) => {

                if (err) {

                    console.dir(err);

                } else {

                    this.setState({

                        isInDislikes: true,

                        isInLikes: true

                    })

                }

            })

        });

    } else {

        alert("This artist can't be added to your dislikes. Sorry for this, please enjoy searching the music reccomender!!");

    }

}

//This is what the user will see

//First off, in the top and middle of the screen there will be the artists' name and just underneath is the like and dislike buttons

//The buttons use the onClick methods of addToLikes and AddToDislikes respectively

//Underneath that there is a short paragraph that is about the artist

//I learned about dangerouslySetInnerHTML here: https://zhenyong.github.io/react/tips/dangerously-set-inner-html.html

//After this there is a row with 3 columns. The columns are the artist's top albums, top songs and then their similar artists

//Map is used to create the list items. In this instance, the topAlbum gets passed in to ArtistTopAlbum

//In this instance, the topSong gets passed in to ArtistTopSong

//In this instance, the similarArtist gets passed in to ArtistResultSummary. If there is no similar artist at all it just displays:

//"This artist has no similar artists, sorry!!"

//Finally the button uses the goHome method to allow the user an easy way to go back to the home page rather than pressing the back button

render() {

    return (

        <div>

            <div class="col-sm-12">

                <h1>{this.state.artistName}</h1>

                <div className="center-data">

                    <button id="Like" className="btn btn-primary" onClick={this.addToLikes} type="submit" disabled={this.state.isInLikes}>Like!</button>

                    <button id="Dislike" className="btn btn-primary" onClick={this.addToDislikes} type="submit" disabled={this.state.isInDislikes}>Dislike!</button>

                </div>

            </div>

            <p dangerouslySetInnerHTML={{ \_\_html: this.state.artistBio }} />

            <div class="row">

                <div class="col-sm-4">

                    <h3 className="center-data">Top five albums:</h3>

                    <ul className="lists">

                        {this.state.topAlbums.map((topAlbum) => {

                            return <ArtistTopAlbum topAlbum={topAlbum} />

                        })}

                    </ul>

                </div>

                <div class="col-sm-4">

                    <h3 className="center-data">Top five songs:</h3>

                    <ul className="lists">

                        {this.state.topSongs.map((topSong) => {

                            return <ArtistTopSong topSong={topSong} />

                        })}

                    </ul>

                </div>

                <div class="col-sm-4">

                    <h3 className="center-data">Similar artists:</h3>

                    {

                        this.state.similarArtistInfo.length > 0 ?

                            <ul className="lists">

                                {this.state.similarArtistInfo.map((artist, key) => {

                                    return <ArtistResultSummary artist={artist} />

                                })}

                            </ul>

                            : 'This artist has no similar artists, sorry!!'

                    }

                </div>

            </div>

            <div class="row">

                <div class="col-sm-12">

                    <span>

                        <button id="btn-search" className="btn btn-primary btn-float-right " onClick={this.goHome} type="submit" >Search again!</button>

                    </span>

                </div>

            </div>

        </div>

    )

}

}

(Last FM , 2013) (LevelUpTuts, 2016)

### ArtistTopAlbum component

//Import Statements

import React, { Component } from 'react';

import "../App.css"

// Allows other components to import this one

export default class ArtistsTopAlbum extends Component {

    render() {

        return (

            //This creates the list item that is used in the unique lists in the artist component and the searchedArtist component

            //All it does is print out the album's name which links to the Last FM page for the album

            //It also shows the album cover

            //However, if the artist that should be passed in doesn't exist, it outputs "This album can't be displayed. Sorry!"

            <li>

                {this.props.topAlbum ?

                    <div>

                        <div>

                            <img src={this.props.topAlbum.image[1]["#text"]}></img>

                        </div>

                        <div>

                            <a href={this.props.topAlbum.url} target="\_blank">{this.props.topAlbum.name}</a>

                        </div>

                    </div>

                    :

                    'This album can\'t be displayed. Sorry!'}

            </li>

        )

    }

}

### ArtistTopSong component

//Import statements

import React, { Component } from 'react';

import "../App.css"

// Allows other components to import this one

export default class ArtistsTopSong extends Component {

    render() {

        return (

            //This creates the list item that is used in the unique lists in the artist component and the searchedArtist component

            //All it does is print out the album's name which links to the Last FM page for the song

            //However, if the artist that should be passed in doesn't exist, it outputs "This song can't be displayed. Sorry!"

             <li>

                {this.props.topSong ? <a href = {this.props.topSong.url} target="\_blank">{this.props.topSong.name}</a>: 'This song can\'t be displayed. Sorry!'}

            </li>

        )

    }

}

### Search component

//Import Statements

import React, { Component } from 'react';

import "../App.css"

import axios from 'axios';

import { LAST\_FM\_KEY } from '../App'

//Allows other components to import this one

export default class Search extends Component {

    //The constructor: This intialises the variables on the state

    //All of these are created in this component

    //The statement "this.myMethod = this.myMethod.bind(this)"" is used so "this" can be accessed within that method

    constructor(props) {

        super(props);

        this.state = {

            choice: '',

            foundArtist: ''

        }

    }

    //This updates the state every time the user types in the input box

    handleChoiceChange = (event) => {

        this.setState({

            choice: event.target.value

        })

    }

    //This creates a local variable called foundArtist which will then be set on the state

    //It then performs a get request to see if that artist exists or not

    //If the artist doesn't exist then an error message will appear

    //Otherwise the searchedArtist component will be loaded

    getArtist = () => {

        const foundArtist = this.state.choice

        this.setState({

            foundArtist: this.state.choice

        })

        axios.get('http://ws.audioscrobbler.com/2.0/?method=artist.getinfo&artist=' + foundArtist + '&api\_key=' + LAST\_FM\_KEY + '&format=json')

            .then(res => {

                if (!res.data.artist) {

                    alert("The artist you searched for could not be found. Please search again!");

                } else {

                    this.props.history.push({ pathname: '/searchedArtist', state: { foundArtist: foundArtist } })

                }

            }

            )

    }

    //This renders a small bit of information for the user to read and know what the website is about

    //There is an instruction as well so the user knows how to operate the search as well

    render() {

        return (

            <div>

                <h1>Search!</h1>

                <p>Please search for an artist!</p>

                <div className="center-data">

                    <label className="users-choice">Your artist:</label>

                    <input

                        type="text"

                        value={this.state.choice}

                        onChange={this.handleChoiceChange}

                    />

                    <button id="SearchArtists" className="btn btn-primary " type="submit" onClick={this.getArtist} disabled={!this.state.choice}>Search</button>

                </div>

            </div>

        )

    }

}

### SearchedArtist component

//Import statements

import React, { Component } from 'react';

import "../App.css"

import { LAST\_FM\_KEY, LIKES\_JSON, DISLIKES\_JSON } from '../App'

import axios from 'axios';

import ArtistResultSummary from './artistResultSummary.component';

import ArtistTopAlbum from './artistTopAlbum.component';

import ArtistTopSong from './artistTopSong.component';

import fs from "browserify-fs";

//Allows other components to import this one

export default class SearchedArtist extends Component {

    //The constructor: This intialises the variables on the state

    //All of these are created in this component

    //The statement "this.myMethod = this.myMethod.bind(this)" is used so "this" can be accessed within that method

    constructor(props) {

        super(props);

        this.state = {

            foundArtist: this.props.location.state ? this.props.location.state.foundArtist : '',

            mainArtist: {},

            artistId: '',

            artistName: '',

            artistBio: '',

            similarArtists: [],

            similarArtistInfo: [],

            topAlbums: [],

            topSongs: [],

            isInLikes: false,

            isInDislikes: false

        }

        this.getMainArtistInfo = this.getMainArtistInfo.bind(this);

        this.getSimilarArtistInfo = this.getSimilarArtistInfo.bind(this);

        this.getTopFiveAlbums = this.getTopFiveAlbums.bind(this);

        this.getTopFiveSongs = this.getTopFiveSongs.bind(this);

        this.goHome = this.goHome.bind(this);

        this.addToLikes = this.addToLikes.bind(this);

        this.addToDislikes = this.addToDislikes.bind(this);

        this.checkIfInLikes = this.checkIfInLikes.bind(this);

        this.checkIfInDislikes = this.checkIfInDislikes.bind(this);

    }

    //This is the first method that is accessed after the constructor

    //Calls getMainArtistInfo

    componentDidMount() {

        this.getMainArtistInfo();

    }

    //This is Last FM's API getting called again

    //However, I am using a different method from Last FM

    //I am using the getInfo which requires the artist's name or their mbid

    //Once the get request has occured I am setting some of the things I can onto the state

    //So I can use their values elsewhere

    //After the state gets set, getTopFiveAlbums and getTopFiveSongs get called as well

    //It performs a check before calling getSimilarArtistInfo because some artist's don't have any that are similar to them

    getMainArtistInfo = () => {

        axios.get('http://ws.audioscrobbler.com/2.0/?method=artist.getinfo&artist=' + this.state.foundArtist + '&api\_key=' + LAST\_FM\_KEY + '&format=json')

            .then(res => {

                this.setState({

                    artistName: res.data.artist.name,

                    artistId: res.data.artist.mbid,

                    artistBio: res.data.artist.bio.summary.replace("<a href", '<a target="\_blank" href'),

                    similarArtists: res.data.artist.similar.artist

                })

                this.checkIfInLikes();

                this.getTopFiveAlbums();

                this.getTopFiveSongs();

                if (this.state.similarArtists.length > 0) {

                    this.getSimilarArtistInfo();

                }

            }

            )

    }

    //This checks the likes file for the current artist's Id

    //If the current artist Id is in the likes file then the Like btn will be disabled so the user cannot like them again

    //If it is not in the file though it will make sure the button is enabled

    checkIfInLikes = () => {

        var i;

        if (this.state.artistId) {

            fs.readFile(LIKES\_JSON, 'utf-8', (err, likeData) => {

                let likes = JSON.parse(likeData);

                for (i = 0; i < likes.artists.length; i++) {

                    if (likes.artists[i].mbid === this.state.artistId) {

                        this.setState({

                            isInLikes: true,

                            isInDislikes: true

                        })

                        break;

                    }

                }

                if (this.state.isInLikes === false) {

                    this.checkIfInDislikes();

                }

            })

        }

    }

//This checks the dislikes file for the current artist's Id

//If the current artist Id is in the dislikes file then the Dislike btn will be disabled so the user cannot dislike them again

//If it is not in the file though it will make sure the button is enabled

checkIfInDislikes = () => {

    var i;

    if (this.state.artistId) {

        fs.readFile(DISLIKES\_JSON, 'utf-8', (err, dislikeData) => {

            let dislikes = JSON.parse(dislikeData);

            for (i = 0; i < dislikes.artists.length; i++) {

                if (dislikes.artists[i].mbid === this.state.artistId) {

                    this.setState({

                        isInLikes: true,

                        isInDislikes: true

                    })

                    break;

                }

            }

        })

    }

}

    //This is Last FM's API getting called again

    //However, I am using a different method from Last FM

    //I am using the getTopAlbums which requires the artist's name or their mbid

    //Because of getMainArtistInfo, I am able to use the artist name because getInfo gave that information

    //In order for me to get the top five albums I used a for loop to push the first five results on to an

    //Array. This array is then set to topAlbums on the state so it can be accessed elsewhere in the component

    getTopFiveAlbums = () => {

        var i;

        var topFiveAlbums = [];

        axios.get('http://ws.audioscrobbler.com/2.0/?method=artist.gettopalbums&artist=' + this.state.artistName + '&api\_key=' + LAST\_FM\_KEY + '&format=json')

            .then(res => {

                for (i = 0; i < 5; i++) {

                    topFiveAlbums.push(res.data.topalbums.album[i])

                }

                this.setState({

                    topAlbums: topFiveAlbums

                })

            })

    }

    //This is Last FM's API getting called again

    //However, I am using a different method from Last FM

    //I am using the getTopTracks which requires the artist's name or their mbid

    //Because of getMainArtistInfo, I am able to use the artist name because getInfo gave that information

    //In order for me to get the top five tracks I used a for loop to push the first five results on to an

    //Array. This array is then set to topSongs on the state so it can be accessed elsewhere in the component

    getTopFiveSongs = () => {

        var i;

        var topFiveSongs = [];

        axios.get('http://ws.audioscrobbler.com/2.0/?method=artist.gettoptracks&artist=' + this.state.artistName + '&api\_key=' + LAST\_FM\_KEY + '&format=json')

            .then(res => {

                for (i = 0; i < 5; i++) {

                    topFiveSongs.push(res.data.toptracks.track[i])

                }

                this.setState({

                    topSongs: topFiveSongs

                })

            })

    }

    //This is Last FM's API getting called again

    //However, I am using a different method from Last FM

    //I am using the getInfo which requires the artist's name or their mbid

    //Because of getMainArtistInfo, I was able to set an array up of the similar artists

    //I am using getInfo so I can set an array up of all the similar artists' info

    getSimilarArtistInfo = () => {

        var i;

        var similarArtistData = [];

        for (i = 0; i < 5; i++) {

            axios.get('http://ws.audioscrobbler.com/2.0/?method=artist.getinfo&artist=' + this.state.similarArtists[i].name + '&api\_key=' + LAST\_FM\_KEY + '&format=json')

                .then(res => {

                    similarArtistData.push(res.data.artist)

                    this.setState({

                        similarArtistInfo: similarArtistData

                    })

                })

        }

    }

    //This just allows the user to go to the home page on a button click

    goHome() {

        this.props.history.push('/');

    }

    //This is used to add the artits(whose page you are on) to the likes file

    //So they will appear in the likes component and will be used to find recommendations in then recommendations component

    //It checks first to see if the artist has an Id because some artists do not

    //If they don not have an Id an errorr message will tell the user that the artist could not be added to the likes file

    //It then reads the likes file into an array, which allows the variable currentArtistId to be pushed on to it

    //It then writes this array back to the likes file

    addToLikes = () => {

        if (this.state.artistId) {

            let currentArtistId = { "mbid": "" + this.state.artistId + "", "name": "" + this.state.artistName };

            fs.readFile(LIKES\_JSON, 'utf-8', (err, likeData) => {

                let likes = JSON.parse(likeData);

                likes.artists.push(currentArtistId)

                fs.writeFile(LIKES\_JSON, JSON.stringify(likes), (err) => {

                    if (err) {

                        console.dir(err);

                    } else {

                        this.setState({

                            isInLikes: true,

                            isInDislikes: true

                        })

                    }

                })

            });

        } else {

            alert("This artist can't be added to your likes. Sorry for this, please enjoy searching the music reccomender!!");

        }

    }

//This is used to add the artits(whose page you are on) to the dislikes file

//So they will appear in the dislikes component

//It checks first to see if the artist has an Id because some artists do not

//If they don not have an Id an errorr message will tell the user that the artist could not be added to the dislikes file

//It then reads the dislikes file into an array, which allows the variable currentArtistId to be pushed on to it

//It then writes this array back to the dislikes file

addToDislikes = () => {

    if (this.state.artistId) {

        let currentArtistId = { "mbid": "" + this.state.artistId + "", "name": "" + this.state.artistName };

        fs.readFile(DISLIKES\_JSON, 'utf-8', (err, dislikeData) => {

            let dislikes = JSON.parse(dislikeData);

            dislikes.artists.push(currentArtistId)

            fs.writeFile(DISLIKES\_JSON, JSON.stringify(dislikes), (err) => {

                if (err) {

                    console.dir(err);

                } else {

                    this.setState({

                        isInDislikes: true,

                        isInLikes: true

                    })

                }

            })

        });

    } else {

        alert("This artist can't be added to your dislikes. Sorry for this, please enjoy searching the music reccomender!!");

    }

}

//This is what the user will see

//First off, in the top and middle of the screen there will be the artists' name and just underneath is the like and dislike buttons

//The buttons use the onClick methods of addToLikes and AddToDislikes respectively

//Underneath that there is a short paragraph that is about the artist

//After this there is a row with 3 columns. The columns are the artist's top albums, top songs and then their similar artists

//Map is used to create the list items. In this instance, the topAlbum gets passed in to ArtistTopAlbum

//In this instance, the topSong gets passed in to ArtistTopSong

//In this instance, the similarArtist gets passed in to ArtistResultSummary. If there is no similar artist at all it just displays:

//"This artist has no similar artists, sorry!!"

//Finally the button uses the goHome method to allow the user an easy way to go back to the home page rather than pressing the back button

render() {

    return (

        <div>

            <div class="col-sm-12">

                <h1>{this.state.artistName}</h1>

                <div className="center-data">

                    {<button id="Like" className="btn btn-primary" onClick={this.addToLikes} type="submit" disabled={this.state.isInLikes}>Like!</button>}

                    {<button id="Dislike" className="btn btn-primary" onClick={this.addToDislikes} type="submit" disabled={this.state.isInDislikes}>Dislike!</button>}

                </div>

            </div>

            <p dangerouslySetInnerHTML={{ \_\_html: this.state.artistBio }} />

            <div class="row">

                <div class="col-sm-4">

                    <h3 className="center-data">Top five albums:</h3>

                    <ul className="lists">

                        {this.state.topAlbums.map((topAlbum) => {

                            return <ArtistTopAlbum topAlbum={topAlbum} />

                        })}

                    </ul>

                </div>

                <div class="col-sm-4">

                    <h3 className="center-data">Top five songs:</h3>

                    <ul className="lists">

                        {this.state.topSongs.map((topSong) => {

                            return <ArtistTopSong topSong={topSong} />

                        })}

                    </ul>

                </div>

                <div class="col-sm-4">

                    <h3 className="center-data">Similar artists:</h3>

                    {

                        this.state.similarArtistInfo.length > 0 ?

                            <ul className="lists">

                                {this.state.similarArtistInfo.map((artist) => {

                                    return <ArtistResultSummary artist={artist} />

                                })}

                            </ul>

                            : 'This artist has no similar artists, sorry!!'

                    }

                </div>

            </div>

            <div class="row">

                <div class="col-sm-12">

                    <span>

                        <button id="btn-search" className="btn btn-primary btn-float-right " onClick={this.goHome} type="submit" >Search again!</button>

                    </span>

                </div>

            </div>

        </div>

    )

}

}

### Recommendations component

//Import statements

import React, { Component } from 'react';

import "../App.css";

import { LAST\_FM\_KEY, LIKES\_JSON, DISLIKES\_JSON } from '../App';

import fs from "browserify-fs";

import axios from 'axios';

import GetRecommendation from '../components/getRecommendation.component';

//Allows other components to import this one

export default class Recommendations extends Component {

    //The constructor: This intialises the variables on the state

    //All of these are created in this component

    //The statement "this.myMethod = this.myMethod.bind(this)"" is used so "this" can be accessed within that method

    constructor(props) {

        super(props);

        this.state = {

            userFavourites: [],

            userDislikes: [],

            recommendations: []

        }

        this.getLikes = this.getLikes.bind(this);

        this.getDislikes = this.getDislikes.bind(this);

        this.getRecommendations = this.getRecommendations.bind(this);

        this.checkRecommendations = this.checkRecommendations.bind(this);

        this.getRandomNumber = this.getRandomNumber.bind(this);

    }

    //This is the first method that is accessed after the constructor

    //Calls getLikedArtists

    componentDidMount() {

        this.getLikes();

    }

    //This reads in the likes file

    //Once it's read in the data gets put into an array which is then set on the state

    //so it can be accessed anywhere in this component

    //It then calls getDislikes

    getLikes = () => {

        fs.readFile(LIKES\_JSON, 'utf-8', (err, LikedArtists) => {

            let likes = JSON.parse(LikedArtists);

            if (likes.artists.length > 0) {

                this.setState({

                    userFavourites: likes.artists

                })

                this.getDislikes();

            }

        })

    }

    //This reads in the dislikes file

    //Once it's read in the data gets put into an array which is then set on the state

    //so it can be accessed anywhere in this component

    //It then calls getRecommendations

    getDislikes = () => {

        fs.readFile(DISLIKES\_JSON, 'utf-8', (err, DislikedArtists) => {

            let dislikes = JSON.parse(DislikedArtists);

            if (dislikes.artists.length > 0) {

                this.setState({

                    userDislikes: dislikes.artists

                })

            }

            this.getRecommendations();

        })

    }

    //https://stackoverflow.com/questions/45175836/random-number-using-react-js This is where i learned how to create a random number generator

    //This creates a random rumber between 0 and whatever the userFavourites array length is

    getRandomNumber = () => {

        let min = 0;

        let max = this.state.userFavourites.length - 1

        let rand = min + Math.random() \* (max - min);

        return Math.round(rand);

    }

    //Calls getRandomNumber

    //This random number is used to determine which artist from the userfavourites array gets used

    //Last FM's API is then used

    //I use the getSimilar method here to find similar artists to the random one that has been chosen with the random number

    //The results from the method are set in an array on the state so it can be accessed elsewhere in this component

    getRecommendations = () => {

        let rand = this.getRandomNumber();

        let randomArtist = this.state.userFavourites[rand].name;

        axios.get('http://ws.audioscrobbler.com/2.0/?method=artist.getsimilar&artist=' + randomArtist+ '&api\_key=' + LAST\_FM\_KEY + '&format=json')

            .then(res => {

                let similarArtists = res.data.similarartists.artist

                this.setState({

                    recommendations : similarArtists

                })

            })

    }

    //This is supposed to check and see if any of the recommendations array is the same as the dislikes array

    //If there is any overlap these should be taken out and either removed entirely from the recommendations array

    //Or replaced with a new artist

    checkRecommendations = () => {

    }

    //This is what the user will see

    //Firstly they will see the text which will just explain what this page is about

    //Underneath that they will see a list of the reccommendations

    render() {

        return (

            <div>

                <h1>Your Recommendations!!</h1>

                <p>This will show you an assortment of artists for you to browse without seraching!!<br />Hopefully you'll find the next band you love!!</p>

                <ul className="lists">

                    {this.state.recommendations.slice(0, 30).map((artist) => {

                                return <GetRecommendation artist={artist} />

                            })}

                </ul>

            </div>

        )

    }

}

### getRecommendation

//Import statements

import React, { Component } from 'react';

import { Link } from 'react-router-dom';

import "../App.css"

//Allows other components to import this one

export default class GetRecommendation extends Component {

    render() {

        return (

            //This creates the list item that are used in the unique lists in favourites component

            //All it does is print out the artist's name and links to the artist's page on the website

            <li>

               {<Link to={"/artist/" + this.props.artist.mbid}>{this.props.artist.name}</Link>}

            </li>

        )

    }

}

### Favourites component

//Import statements

import React, { Component } from 'react';

import "../App.css";

import { LIKES\_JSON } from '../App';

import GetFavourite from "./getFavourite.component";

import fs from "browserify-fs";

//Allows other components to import this one

export default class Favourites extends Component {

    //The constructor: This intialises the variables on the state

    //All of these are created in this component

    //The statement "this.myMethod = this.myMethod.bind(this)"" is used so "this" can be accessed within that method

    constructor(props) {

        super(props);

        this.state = {

            userFavourites: [],

            mainMessage: 'Here are your favourite artists!!'

        }

        this.getLikedArtists = this.getLikedArtists.bind(this);

    }

    //This is the first method that is accessed after the constructor

    //Calls getLikedArtists

    componentDidMount() {

        this.getLikedArtists()

    }

    //This method reads in the Likes file from local storage

    //Once it's been read, if the Likes file is bigger than 0(this was added to stop errors)

    //The data from the file is put into an array

    //And the array gets put onto the state so it can accessed elsewhere

    //Or is there is nothing in the file, the mainMessage will be updated to :

    //"You have not liked any artists. Please like some so you can easily find them here!!"

    getLikedArtists = () => {

        fs.readFile(LIKES\_JSON, 'utf-8', (err, LikedArtists) => {

            let likes = JSON.parse(LikedArtists);

            if (likes.artists.length > 0) {

                this.setState({

                    userFavourites: likes.artists

                })

            }else{

                this.setState({

                    mainMessage: 'You have not liked any artists!!'

                })

            }

        }

        )

    }

    //This is what the user will see

    //Firstly, they will see either "Here are your favourite artists!!"

    //or

    //"You have not liked any artists. Please like some so you can easily find them here!!"

    //This depends if the user has liked an artist

    //If the user has liked artists, the artists they have liked will appear underneath the message in a list

    //Map is used to create the list items. In this instance, the userFavourite gets passed in to GetFavourite

    //They will all be links to their specific artist page so the user can see their page again and

    //Find people Similar to that particular artist

    render() {

        return (

            <div>

                <h2>{this.state.mainMessage}</h2>

                <ul className="lists">

                    {this.state.userFavourites.map((userFavourite) => {

                        return <GetFavourite userFavourite={userFavourite} />

                    })}

                </ul>

            </div>

        )

    }

}

### getFavourite component

//Import Statements

import React, { Component } from 'react';

import { Link } from 'react-router-dom';

import "../App.css"

//Allows other components to import this one

export default class GetFavourite extends Component {

    render() {

        return (

            //This creates the list item that are used in the unique lists in favourites component

            //All it does is print out the artist's name and links to the artist's page on the website

            <li>

               {<Link to={"/artist/" + this.props.userFavourite.mbid}>{this.props.userFavourite.name}</Link>}

            </li>

        )

    }

}

### Dislikes component

//Import statements

import React, { Component } from 'react';

import "../App.css";

import { DISLIKES\_JSON } from '../App';

import GetDisliked from "./getDisliked.component"

import fs from "browserify-fs";

//Allows other components to import this one

export default class Dislikes extends Component {

    //The constructor: This intialises the variables on the state

    //All of these are created in this component

    //The statement "this.myMethod = this.myMethod.bind(this)"" is used so "this" can be accessed within that method

    constructor(props) {

        super(props);

        this.state = {

            userDislikes: [],

            mainMessage: 'Here are your disliked artists!!'

        }

        this.getDislikedArtists = this.getDislikedArtists.bind(this);

    }

    //This is the first method that is accessed after the constructor

    //Calls getDislikedArtists

    componentDidMount() {

        this.getDislikedArtists()

    }

    //This method reads in the Dsisikes file from local storage

    //Once it's been read, if the Dislikes file is bigger than 0(this was added to stop errors)

    //The data from the file is put into an array

    //And the array gets put onto the state so it can accessed elsewhere

    //Or is there is nothing in the file, the mainMessage will be updated to :

    //"You have not disliked any artists. Your disliked artists will appear here for you to see and possibly change!!"

    getDislikedArtists = () => {

        fs.readFile(DISLIKES\_JSON, 'utf-8', (err, DislikedArtists) => {

            let dislikes = JSON.parse(DislikedArtists);

            if (dislikes.artists.length > 0) {

                this.setState({

                    userDislikes: dislikes.artists

                })

            }else{

                this.setState({

                    mainMessage: 'You have not disliked any artists!!'

                })

            }

        }

        )

    }

    //This is what the user will see

    //Firstly, they will see either "Here are your disliked artists!!"

    //or

    //"You have not disliked any artists. Your disliked artists will appear here for you to see and possibly change!!"

    //This depends if the user has disliked an artist

    //If the user has disliked artists, the artists they have disliked will appear underneath the message in a list

    //Map is used to create the list items. In this instance, the userDislike gets passed in to GetDisliked

    //They will all be links to their specific artist page so the user can see their page again and

    //Find people Similar to that particular artist

    render() {

        return (

            <div>

                <h2>{this.state.mainMessage}</h2>

                <ul className="lists">

                    {this.state.userDislikes.map((userDislike) => {

                        return <GetDisliked userDislike={userDislike} />

                    })}

                </ul>

            </div>

        )

    }

}

### GetDisliked component

//Import statements

import React, { Component } from 'react';

import { Link } from 'react-router-dom';

import "../App.css"

//Allows other components to import this one

export default class GetDislike extends Component {

    render() {

        return (

        //This creates the list item that are used in the unique lists in dislikes component

            //All it does is print out the artist's name and links to the artist's page on the website

            <li>

                {<Link to={"/artist/" + this.props.userDislike.mbid}>{this.props.userDislike.name}</Link>}

            </li>

        )

    }

}

### Testing

### Original Strategy

In the design stage of my project I said I was going to be using the agile methodology to develop my website, Music Recommender. Because I said I was using the agile methodology, I never created any test logs as the agile methodology is all about testing as the program is being developed. The downfall of this, for me personally, was that even though I had designed the program extensively, I had less of an idea on how to test the website. However, this downfall went away quickly after I tested the first page of my website. I’ll explain how later but because of this it allowed me to think up of ways to test the website better and it made the testing more comprehensive as I tested more parts of the website.

### Updated Strategy

In this section, I will talk about all the functional parts of my website.

As I was developing my website, my test strategy never really changed. I was always making sure the part of the website I was working on, the home page, for example, was working and there were no errors to be found. By doing this, it made me feel like I was making real progress on my website.

The first thing I tested was the file App.js, and this was because it is used to show the user one of the most important things on my website, the navbar. The navbar is essential for my website because not only is it the way users move around the website; it also shows off some of the features and it tries to grab the user’s attention. For example, if a user loads onto any page on the website, at the top and centre of the page, the navbar is there. The navbar holds links to the Home page, Recommendations, Favourites, Dislikes and a page I decided to add while developing my website, the Search.

I decided to add the Search because I realized when developing and testing the Home page, I would not have enough time to allow the user to enter 3 artists and get a mix of results that are similar to those three. So, I decided to add a feature, which when I look back on the design stage should have been there from the beginning, the Search page. The Search page wasn’t created until near the end of development so I will talk about it later in more detail but the basis of it is that a user can put in their favourite artist, for example, Queen, and the user will be taken to the artist's page directly rather than having to hope for them to appear in the Results page.

App.js is fundamental because it holds the navbar, but it also sets up the routes I use throughout the website. The routes tell the website that when the URL is, for example, “/” that the Home component will be loaded onto the website. This means that on a fundamental basis App.js is the most important piece of this website because if it did not exist then the website would not work, and everything would be broken.

The next thing I created and tested was the Home component. This is the landing page for the website and because of this, its main job is to welcome the user and allow them to jump straight into using the website to find some new artists. In my original design documents, I said I was going to have the user be able to put in three artists and then once they click search, they will be taken to a page where a list of results will be displayed. The results would be a mix of similar artists from all three/two of the inputted artists or one artist. This was because of the websites I looked at during the design stage but when I was creating the home component, I decided to just allow the user to input one artist. This reason for this was I wanted the project to be functional by the deadline and I wanted to build up the project. I decided to start with just implementing the user inputting one artist, and if I had time I was going to go back and add in the ability to put in a second artist and once everything was working with that, I would then put in the ability to input a third. Sadly, I never got the chance because of the time constraints but if I had more time, I would have at least tried to implement this.

The testing for this component was not that much because it was a very basic component compared to some of the later ones, but the most important part was to check and ensure that the user could put their artist into the input box, have the website recognize their input and then carry out a get request using the API. I tested the website recognizing the user’s input by using the command console.log(choice1). By using this it allowed me to see what the website had stored the user’s choice as before carrying out the get request. I tried it three times, the first time I put in Dio, the second Tenacious D and the third Foo Fighters. The console showed me these three when I called the method, so this test was successful. The next and last test for the home component was the get request using Last FM’s API. This was more difficult to test as I had to rely on Last FM’s API working but the API worked perfectly. On the first test, I couldn’t get the results I was wanting, this was because I got an error saying similarArtists was undefined. This was because I had not properly implemented the results of the get request which was working fine. After fixing the results, I checked using the three artists and used the console.log command to log out the results to the console in chrome. This showed me that I was getting back a JSON object which contained an array of one hundred similar artists. Once this test was complete, I made sure that once the user clicks the Search button they get taken to the Results page.

The Results component was the next component I made. The basis of this component is to show the results from the get request made in the Home component. In my design documents, I said I was going to show six artists. However, when developing I decided to show ten instead because I was not going to use images in the Results component like I said I would in the storyboards. This was simply because I felt showing more names was better than showing six artists with an image, I also decided to use images in the Artist and SearchedArtist component and I didn’t want to use the same image in two different components.

The Results component had slightly more in-depth testing as I had to pass through the results from the Home component and make sure these were here otherwise I was going to have to show an error message saying the band they search for didn’t have any artists similar to them. I made sure the data from the Home component was coming through by going to the results page three times using Dio, Tenacious D and Foo Fighters. I used the code console.log () and logged out the results. Once that test was completed, I was able to build the rest of the page.

The very first thing the Results component does is it checks if the similar artists are in the Results component and if they aren’t it shows a generic error message saying that the recommender has been beaten and to try again. If the similar artists exist, they will be displayed in a unique list and every list item should be the artist’s name and it should also be a link. The way I did this was by using a function called map. The map function allows an array to be rendered until it has been completely rendered. In my case, I used the slice function to cut the array down to the first ten similar artists. I had to create another component called ArtistResultSummary because this is the component that was getting rendered by the map function. I tested this by going through the home page putting in three artists and making sure that the list was rendering correctly. Once I was happy that was working, I added a button that when clicked allows the user to go back to the home page instantly. I tested that a couple of time to ensure it worked and once it worked, I moved onto the next component.

The next component is one of the biggest, and because of that the development and testing of this page took me longer than I expected. The biggest reason for this was because I first had to ensure that the artist’s mbid, this is a unique identifier in Last FM’s API which I later found out was not included with every artist, was able to be read in. I got this to work but unfortunately, I was unable to ensure these artists, who do not have a mbid, would not appear. This leads to a bug in the program which will break the project, however, it is only on a few artists that do not have mbids so while this does break the website I think if I had longer to do this project, I would try and fix this immediately. For example, an artist named Intervals has no mbid and because of that, the website will break. The user can still use the website as long as they close the error and go back to the previous page.

The next thing I did was using the URL parameters to get the artist’s mbid. I made it so the mbid came after artist/ so I could easily get access to it. I made this the first method after the componentDidMount method. I used the mbid, so I was able to use Last FM’s API to perform a get request. This method is called after getting the mbid and is called getMainArtistInfo. The method I was using is getInfo and this gives me back a set of information about the artist. I used the console.log() line to ensure I got the data and once I did, I started setting some of the info on to the state so I could use it elsewhere in the component. For example, I store the artist’s name, their biography and the similar artists that Last FM supplies. After these have been set, I call the next two methods, getTopFiveAlbums and getTopFiveSongs, the names spell out exactly what the methods have to do. Once those methods have been called and done, I call the last one and it only gets called if the artist has more than 0 similar artists because some artists do not have anyone similar to them according to Last FM’s API. Because I have now set the artist’s name on the state, I use this rather than the mbid in the remaining calls to Last FM.

The method getTopFiveAlbums uses Last FM’s API again but calls the getTopAlbums method. This method returns an array of the artist’s albums, in order of how many times the album has been played. By using this method, I use a for loop to push the top five albums into an array which then gets set on the state so I can use it elsewhere.

The method getTopFiveSongs is largely the same as getTopFiveAlbums, however, it is getting the top five songs. It uses the Last FM API to call the method getTopTracks. I do the same hereby pushing the top five results into an array using a for loop, then I set it in the state so I can use it elsewhere. In both these methods, the testing was really simple as by this point, I knew how to use Last FM’s API so all I needed to check was that I was getting the right results. I used the console.log() line to output the response from the API to check that I was getting everything I needed.

The next method that gets called is getSimilarArtistInfo. This method is also self-explanatory but the point of it is to get an array that holds the information about the similar artists. This information can then be used to display the artist’s name and also make the artist’s name a link to another version of the artist component, although that version will be about the new artist. Like in the getMainArtistInfo method, this method uses the getInfo method from Last FM but now it is used five separate times. This is because Last FM usually returns five similar artists in a getInfo search. This is called inside a for loop and the result of this get request is pushed into an array and then the array is set on the state.

The next method in the code is the method goHome. This is used in the same way as the goHome method is used in the results component. It is used by a button in its onClick handler so the user can quickly go back to the home page and start searching again.

The next two methods are practically the same so I’ll only describe one of them. The two methods are addToLikes and addToDislikes. I’ll describe addToLikes because when looking at the component in the code it comes before addToDislikes.

So addToLikes was implemented after I had set up the likes and dislikes component. This was one of the hardest things in the whole project as I had initially designed that the likes and dislikes were going to be stored in a JSON file on the PC. This doesn’t work as react is used for the browser and therefore cannot write to a local file. However, after looking at how to implement some sort of like system, I discovered local storage in the browser. Local storage is something used by the browser, I used local storage by creating the Likes and Dislikes files in the LikesAndDislikes folder in the local storage. What this means is that the user can add Queen, for example, to their likes and they can see it in the favourites component. However, because I am using local storage, if the user went onto another machine and tried to see Queen in the favourites component it would not be there as Queen is in the local storage of the first machine.

So, the first thing I had to do was I had to check and make sure that the current artist has and artistId which was set on the state to be equal to their mbid. This is because it will break the favourites component as it will display artists that do not have a mbid but if the user clicks on them it will take them to a page with a bug (I will have a list of the bugs after this section). If the artistId doesn’t exist a generic error message will appear. So, if the artistId exists, the Likes file will be read into the component and then the data from this will be put into an array, then the artistId and the artist’s name will be pushed onto the array and then the array will be written back to the likes file.

The addToDislikes method is the same although it reads the dislikes file and writes to the dislikes file. After doing this testing I discovered that it was possible to like artists multiple times and it was possible to like and dislike artists at the same time. Because of this, I realized I had to create new methods to check the two methods checkIfInLikes and checkIfInDislikes.

These next two methods are basically the same so I will only discuss checkIfInLikes. This method checks the likes file and then goes through every individual like and compares the liked artists Id with the current artistId. If any of these matches each other then a variable called isInLikes will be updated to true. If none of the likes from the file match the current artist then isInLikes gets set to false. This determines whether the like button will be enabled or disabled. If it is true then the like button will be disabled so the user cannot like them again. The only issue I ran into and could not fix is that I could not get the dislike button to be disabled as well if the artist was liked from the beginning. I got it to be disabled if the user likes the artist while on the page.

The last method in the artist component is the render and this shows all of the data I got from the methods I have just discussed except the addToLikes and addToDislikes methods. Like in the results component, I used the map function to display the top five songs, top five albums and the similar artists. The similar artists map component was the exact same one as the results component, whereas two new ones were created for the top five albums and songs. These were called artistTopAlbum and artistTopSong. The artistTopAlbum displays the album name but above the name, it displays the album cover. Originally, the top album cover was supposed to be used in the results component to add an image to the results component, but I decided to show the images here. The artistTopSong just displays the top five songs but the names of the albums and songs are links. These links are direct links to Last FM’s page for that specific song or album. And finally, the similar artists link to an internal page which just refreshes the component. It does this by using two different methods.

The first method is componentDidUpdate and what this does, is it checks the new properties that have been set and sees if the URL parameter id is the same as the old artistId. If it’s not, then the artistId will be set to the URL parameter id and then the getMainArtistInfo is called. However, this next method is used in order to compare the next props and the previous state. It is called getDerivedStateFromProps, all it does is return artistId being equal to the next props id. But it only does this if the next props id is not equal to the previous artistId. These two methods are called when the URL is updated so that whenever the user clicks on a similar artist link, the page will update quickly without having the user have to manually refresh the page every time.

At the very top of the page and in the centre, the artist’s name is displayed and just underneath that the like and dislike buttons are displayed. These buttons have an onClick handler which uses addToLikes and addToDislikes respectively. The artist’s bio is underneath the buttons and I decided to use the small version as I did not want to have a wall of text on the page. In the smaller version of the bio, there is a tag which I had to set as being active. I did this by replacing the actual tag with the tag again, but I added target= “blank” so the link opened on another page. I also had to use an attribute in a p tag, this was called dangerouslySetInnerHTML and it’s called that because the website will render whatever HTML is in the text. The reason it is dangerous though is because you have to trust that the HTML is safe to use and not malicious. I trusted it because the HTML in the paragraph is a link to Last FM’s page about the artist.

After the biography, I used the column system from bootstrap in order to move the top five albums, songs and the similar artists into three columns. The top five albums are on the left, the top five songs in the middle and the similar artists are on the right. Underneath the similar artists is the last thing on the page, which is the search again button which takes the user back to the home page so they can quickly search for another artist.

The testing for this page was mixed, for the most part after completing one of the methods I would console out the state to make sure it was updated. I would do this on a couple of different artists to ensure it worked properly. Because I had already used the map function for the results page, I did the same type of testing for all three of the maps I used in the render method. I found that some artists do not have any artists similar to them, so I put the similar artists map function in an if statement. As long as similarArtistInfo is bigger than 0 the list can be rendered.

I am not going to discuss the development and testing of the searchedArtist component because it is the exact same as the artist component. I will discuss the one slight difference between the searchedArtist and Artist component once I have talked about how the actual search component works.

The search component was not originally meant to be in my project. I added it after I realized I would not have enough time to add in the ability for the user to put two artists into the website and for the website to go and get a mix of results for. The search is a really simple idea as the user will just put in who they want to see. For example, if I put in Alice Cooper, I would be taken to Alice Cooper’s page and could then explore Alice Cooper’s albums, songs and similar artists. When I click on one of the similar artists it will take me to the artist component as the map function uses the artistResultSummary component to generate the links to the similar artists and because it is used in the results component it will always go to the route /artist/mbid.

The search component is essentially the same as the home component except it goes straight to an artist’s page. I think this is a good addition to the website that should have definitely been in the design documents, but I just overlooked the fact that I could have done this. The component that gets loaded after the search has been done is the searchedArtist component. The one difference I mentioned earlier is that it does not use the artist’s mbid to perform the getMainArtistInfo search. It uses the user’s input and performs the search with that. Other than that, the two components are virtually identical. They show the same data, but I feel this is something that could keep a user entertained and interested while they try and think of artists to search for.

The next component I will discuss is the favourites component. This component’s purpose is for it to show all the artists the user has liked so far. Because I am using local storage if the user switches the website off and on again the artists they have liked will still be there, although if they change machine they will have to start over again. There is a known bug with this, and I just did not have the time to fix it. The bug is that the user can like an artist more than once and they can also dislike the artist at the same time. For example, the user could add The Beatles twice to their likes, but they could also dislike them by accident. It doesn’t break the website, but I find it annoying and I’m sure that users would be confused at the fact they can add an artist an infinite amount of times to either likes or dislikes or both.

The first thing I had to do in both of the favourites and dislikes components, was I had to read in the likes and dislikes files respectively and then put the data from these files into an array which is stored on the state. At this point in the development, I had learned how to use the map function well so for both the favourites and dislikes component I used a map function to display the list of artists in the render method and I tested these both by putting some artists in both. This list is also a list of links, so the user will see the artists name and be able to click on their name. Once they have clicked, they will be taken to the artist component for them. For the map function, I used the components getFavourite and getDislike respectively.

The last components I made were the recommendations and get Recommendation components. The recommendations component is used to display artists that you may like based on who you have liked and disliked so far. The recommendations component does show the artists based on what the user has liked but I was unable to make any of the disliked artists not appear. I did leave in where I was going to check the recommendations in the code, and I go into more detail about what it was supposed to do there.

### Design of the website

In this section I will talk about the CSS I have done for my website.

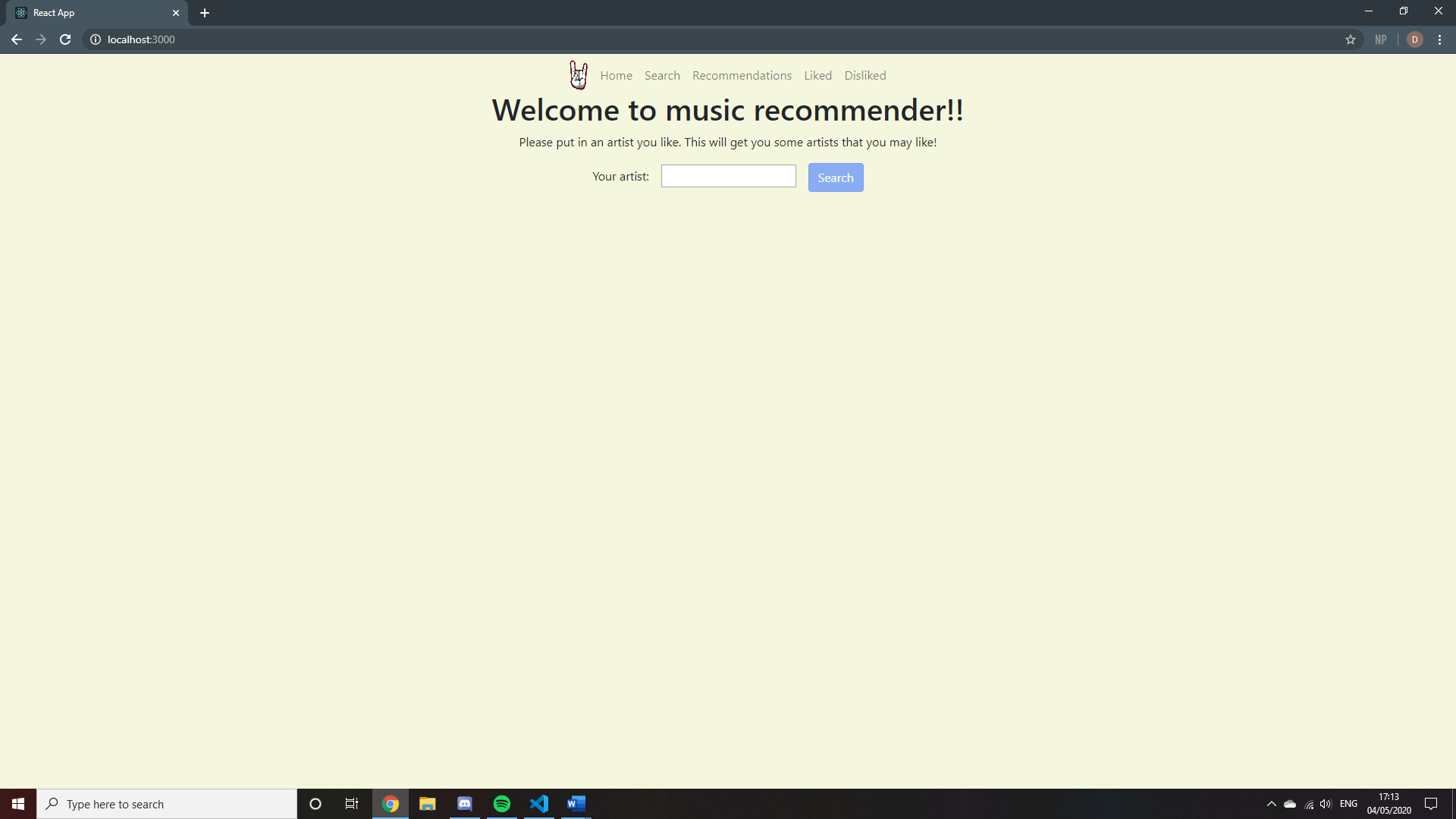
The main bit of CSS in my website is the background colour and this is a consistent colour of beige as I feel this is a nice and calm colour and allows the user to relax when using the website. I also made most of the things on my website in the center of the page as it allows the user easily to read it and it allows me to branch out in the future and possibly add images down the side or more things to the page. I made my logo in the navbar spin as I thought some people might see it and think that it was quite cool. I also made all of the lists not have any bullet points or be numbered as I think it looks better when you just see a list of the names of artists you may like.

In the artist and searchedArtist component everything is put into columns. I did this by using bootstrap and using the classes called “col-sm”. This allowed me to make the title be in right in the center and have nothing in the same row as it. Although, the like and dislike buttons are in the same column but are displayed underneath the artist’s name. The colour of the buttons was chosen by a friend who actually helped me use my website as a new user by telling me about some new artists.

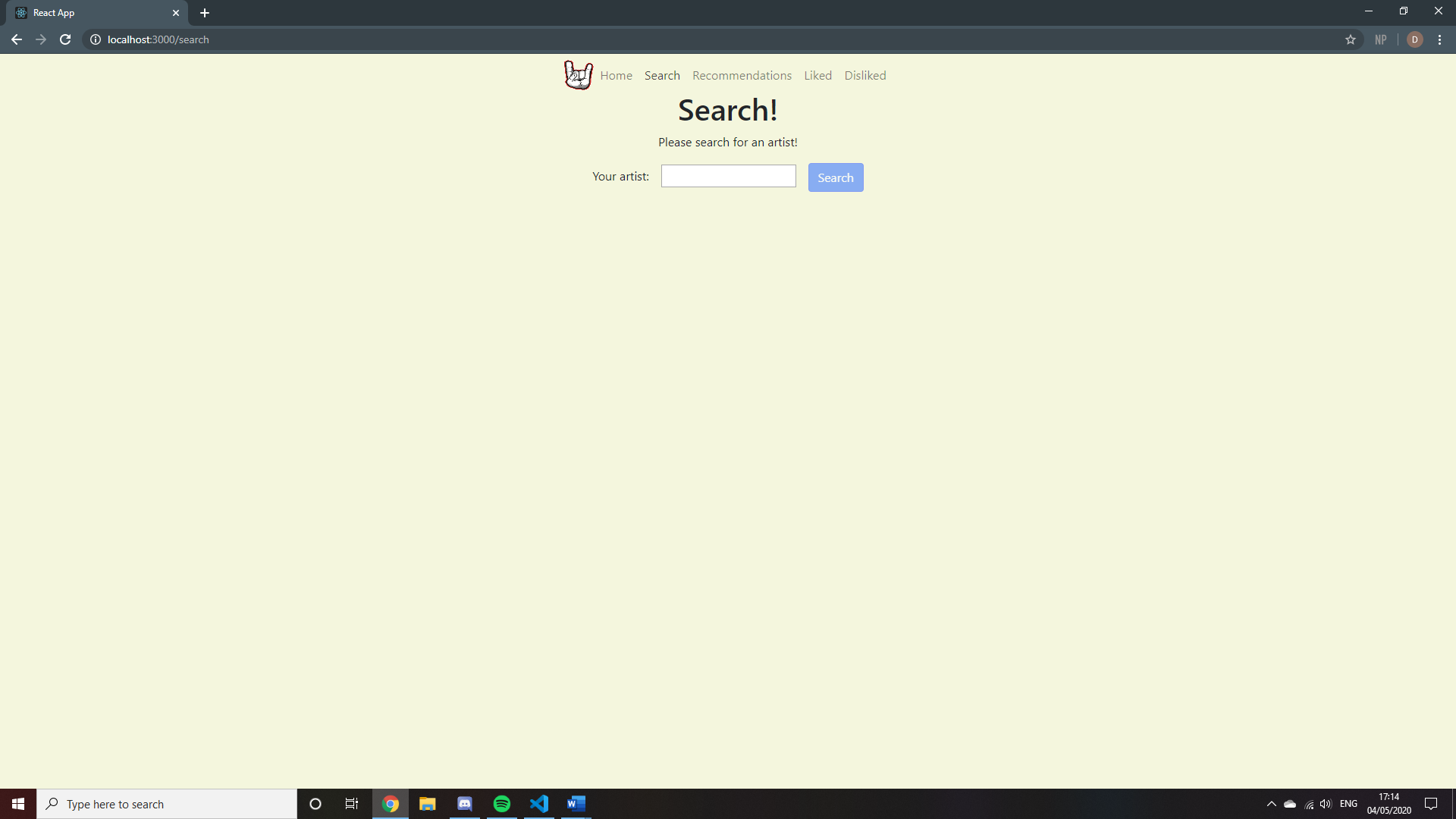
I decided to go from right to left the top five albums, top five songs and then the similar artists because I believe that albums are more important than one song as a great album is always better than one great song. That’s why the songs are in the middle because I wanted the user to see them and they could go and look at them on Last FM and then come back and see the similar artists and look through their pages.

### Screenshots of the website

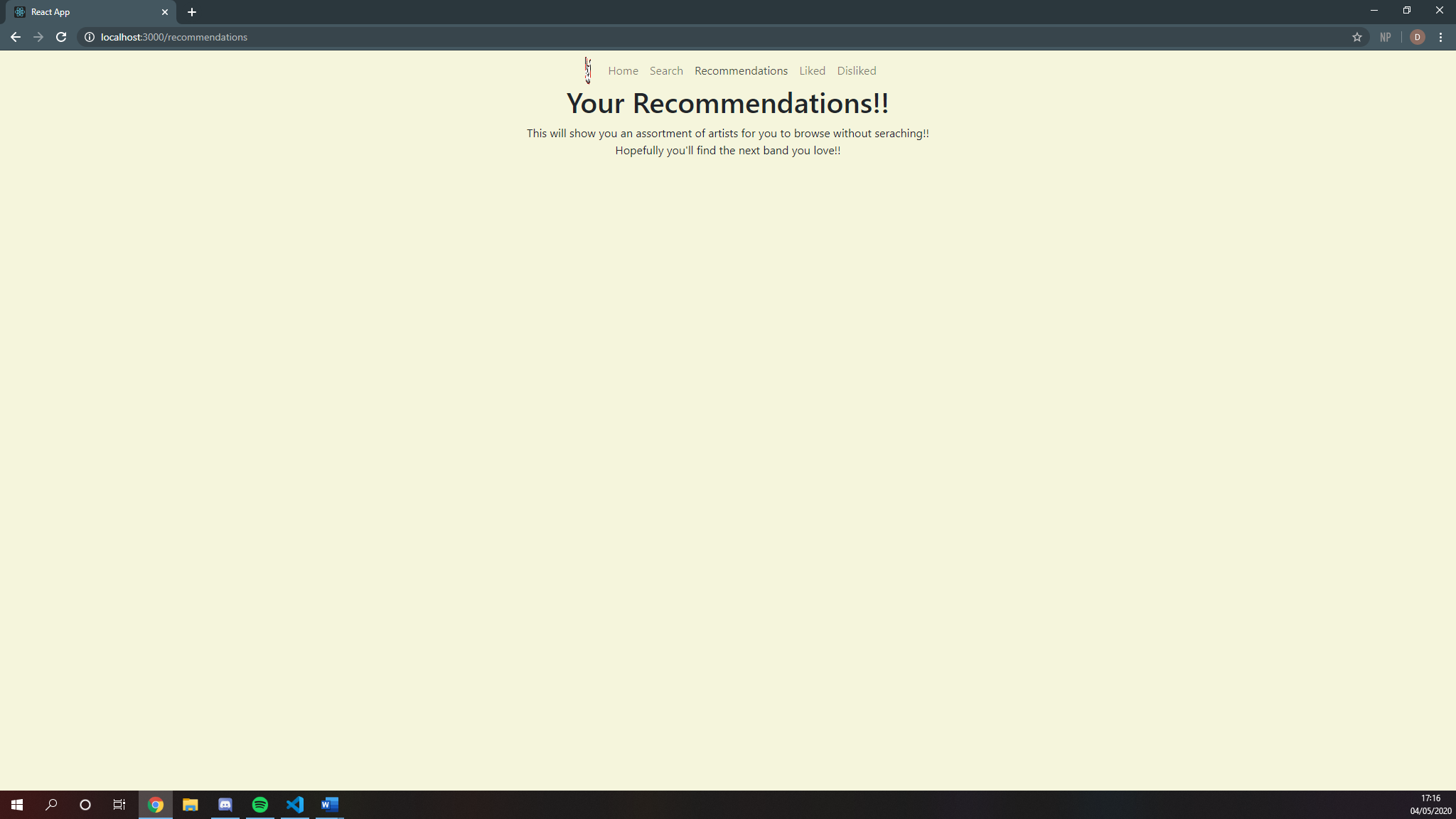
### Home page



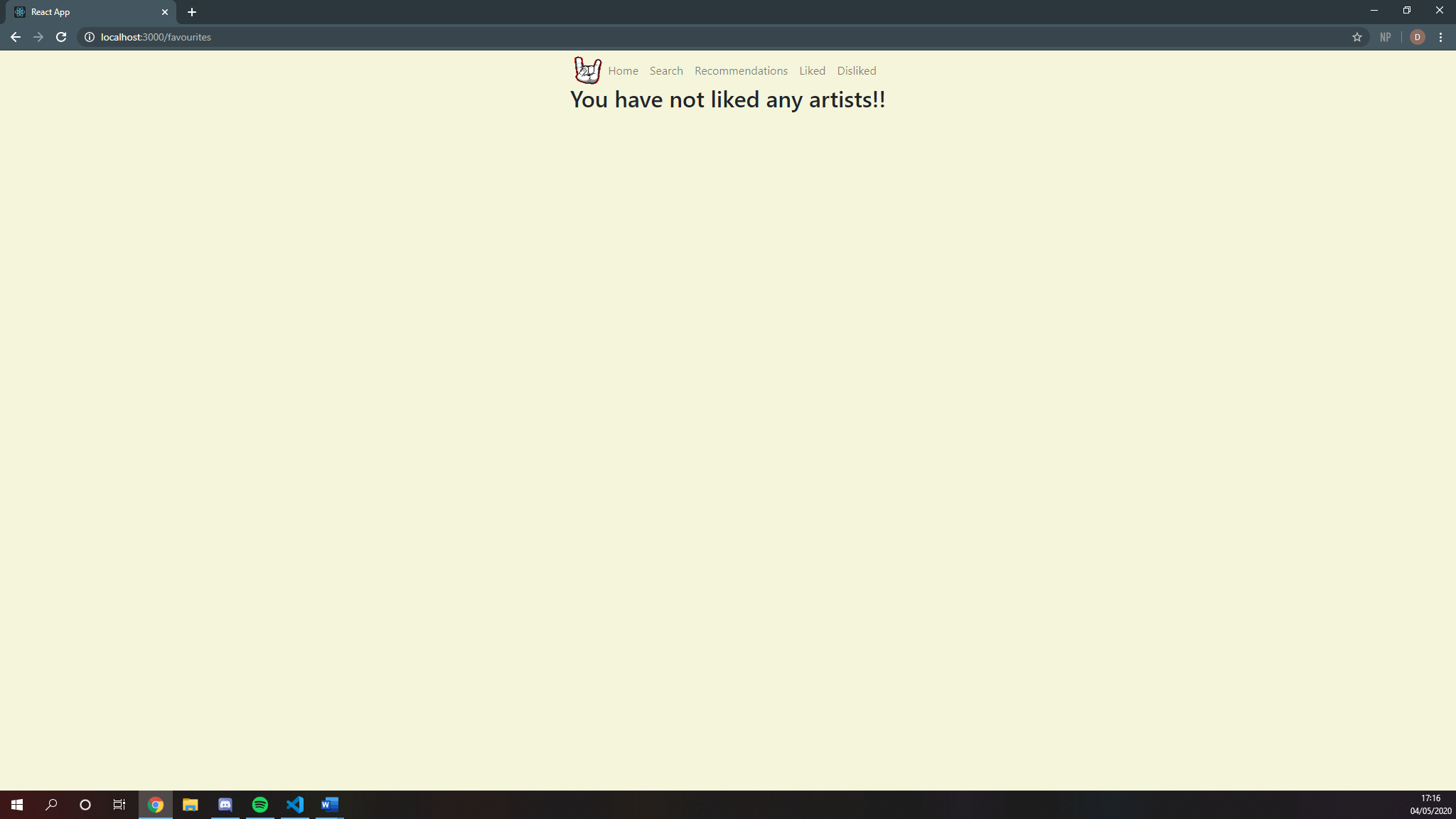
### Search page



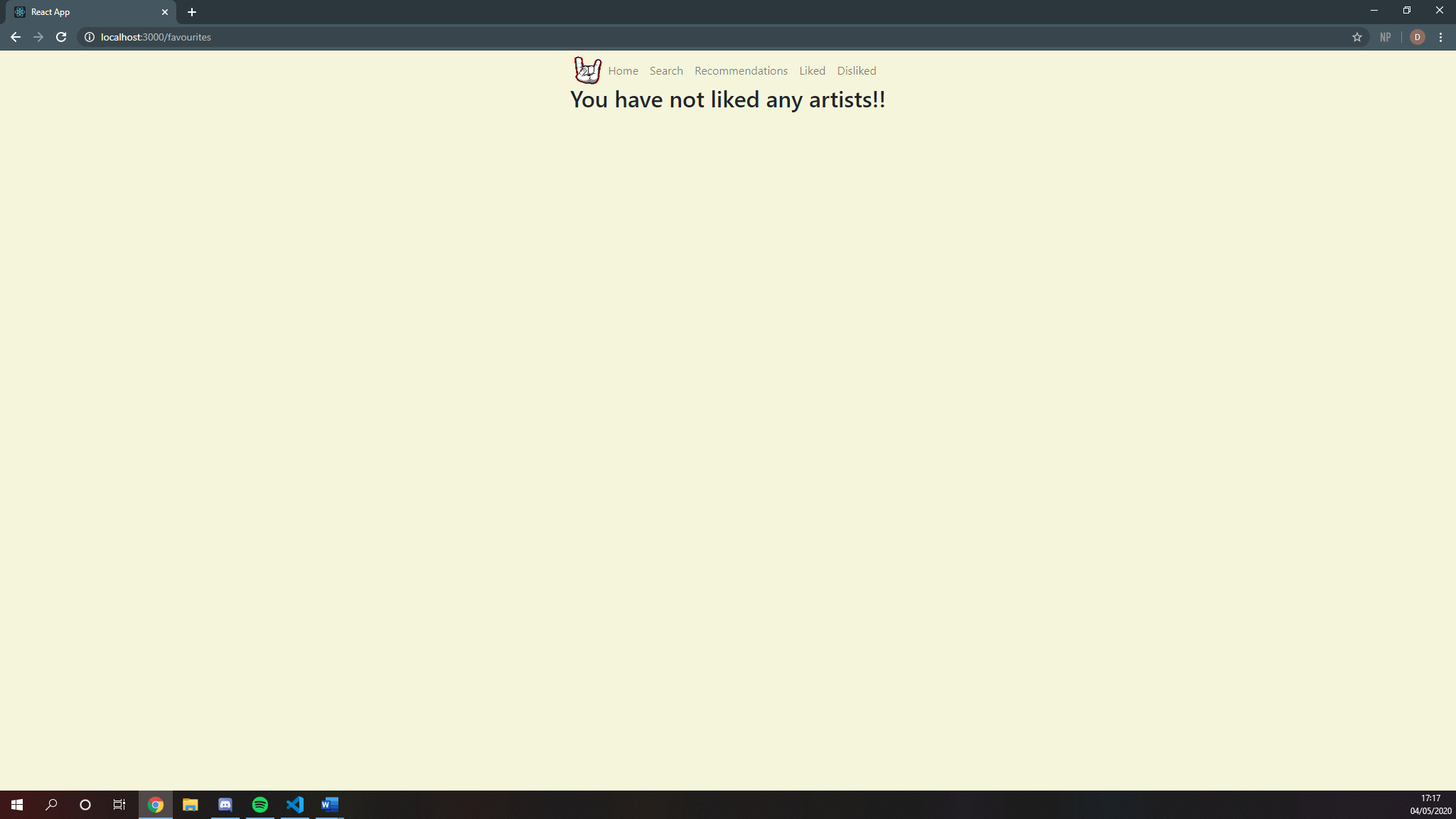
### Recommendations page (empty)



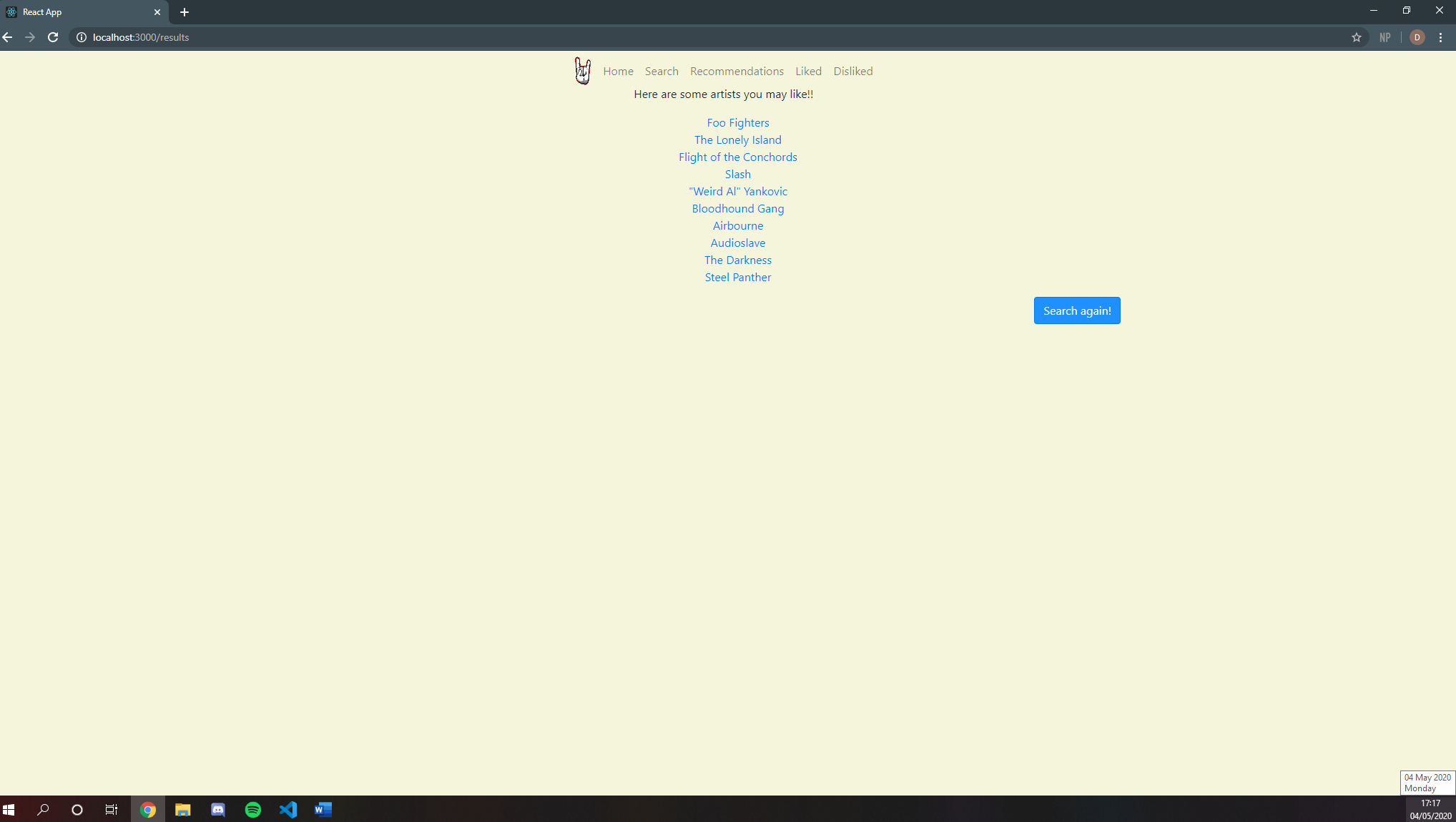
### Liked page (empty)



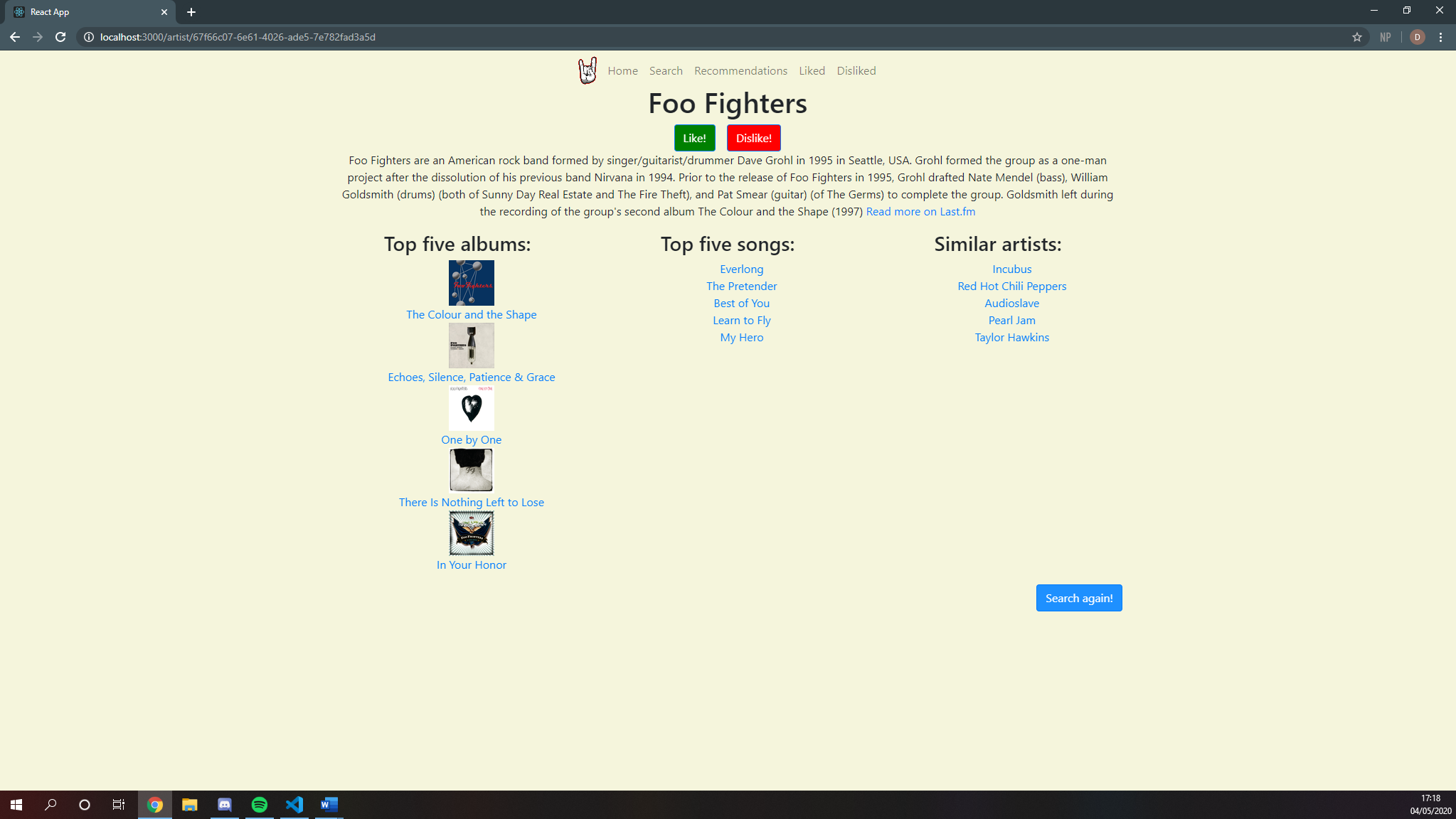
### Disliked page (empty)



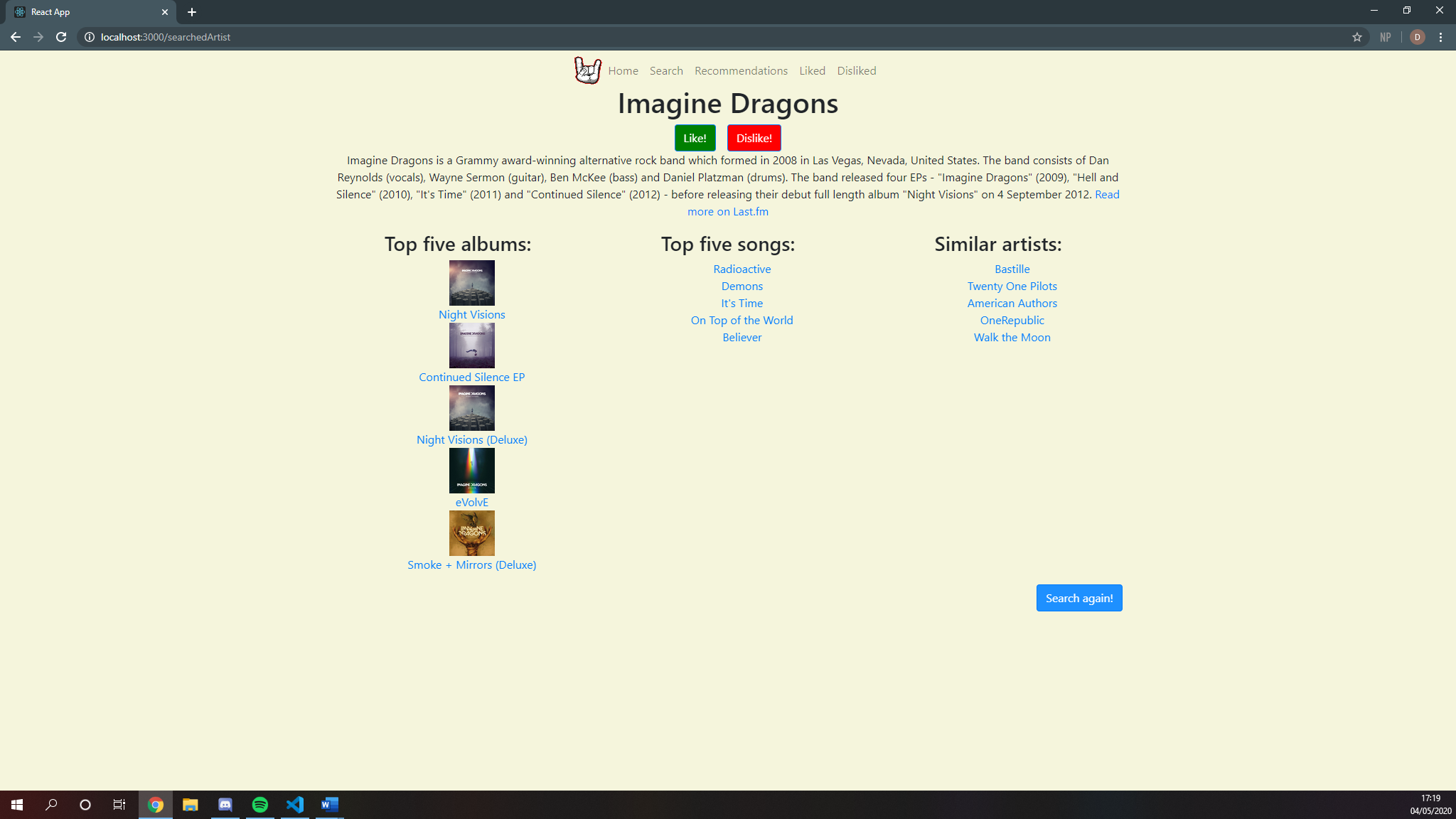
### Results page



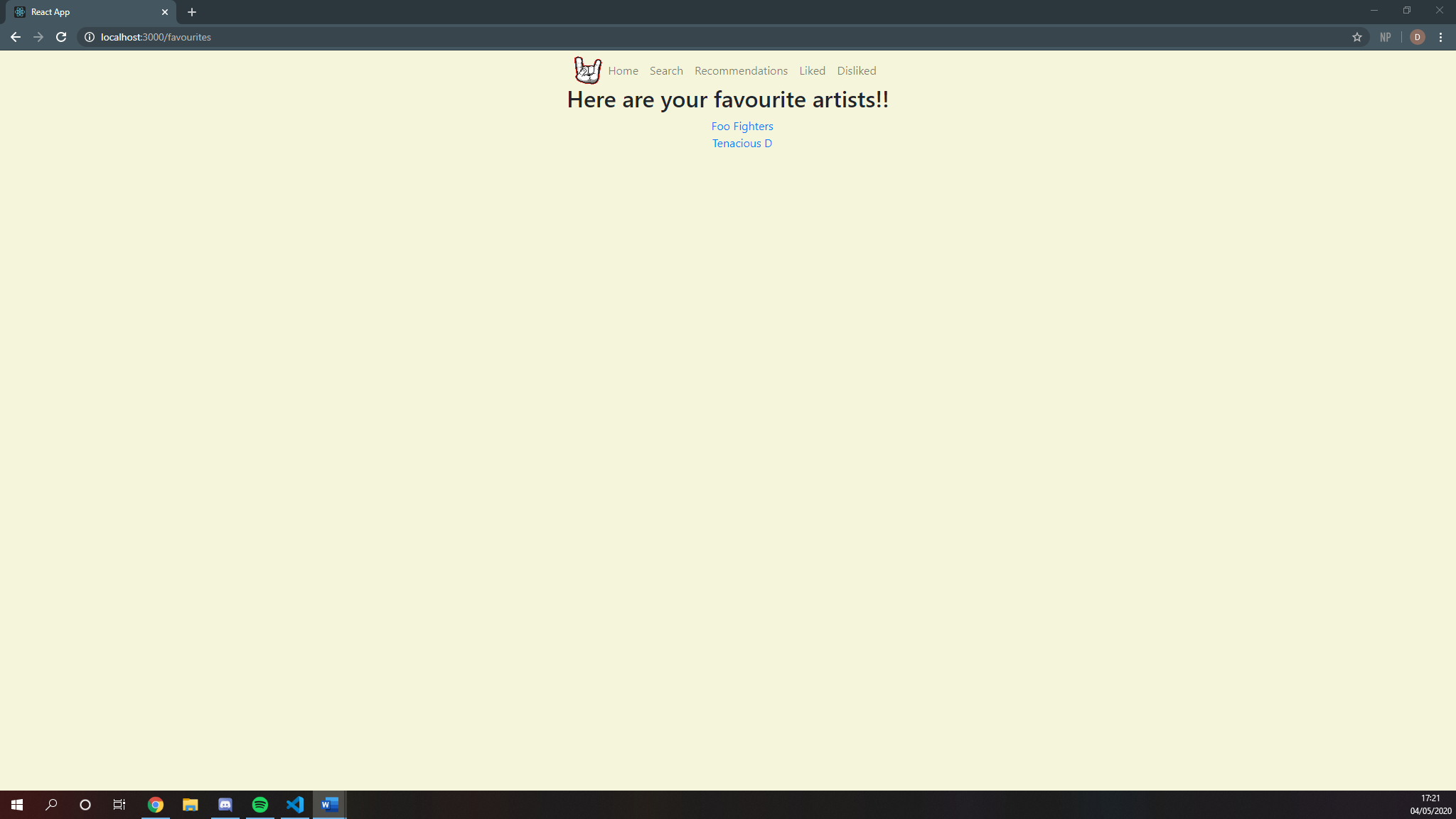
### Artist page (Foo Fighters)



### Searched artist page (Imagine Dragons)

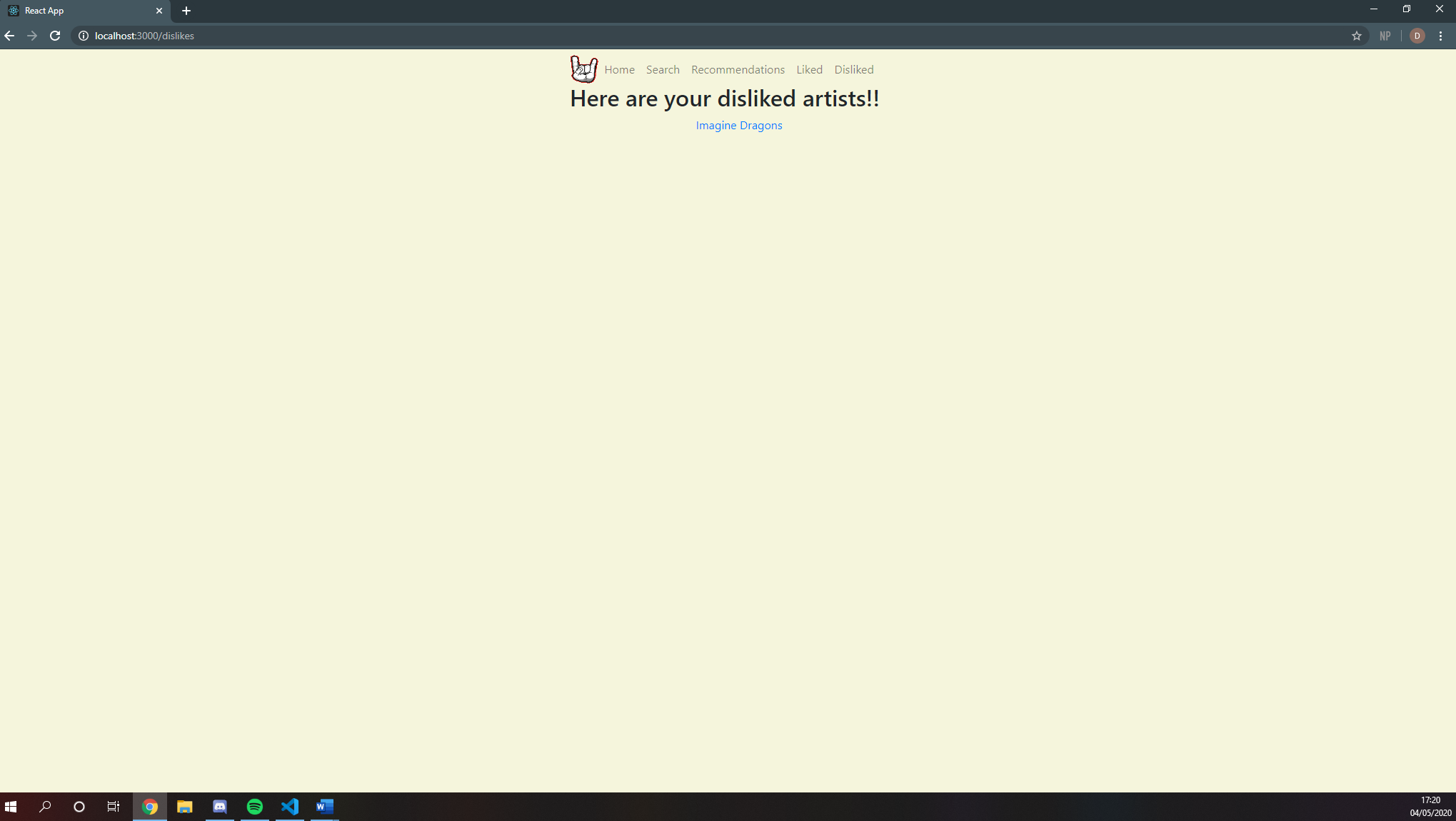


### Liked page (filled)



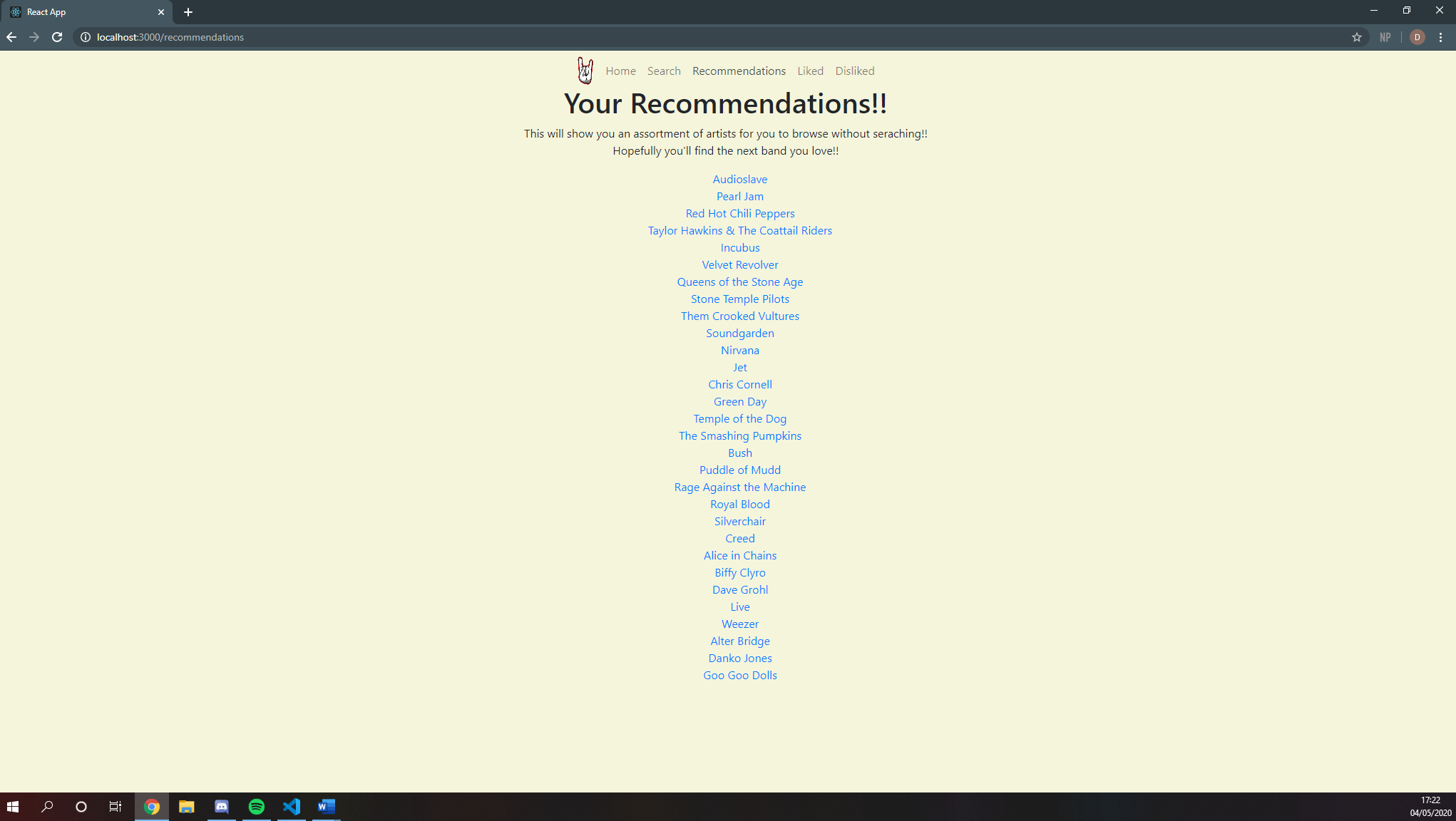
More artists can be added

### Disliked page (filled)

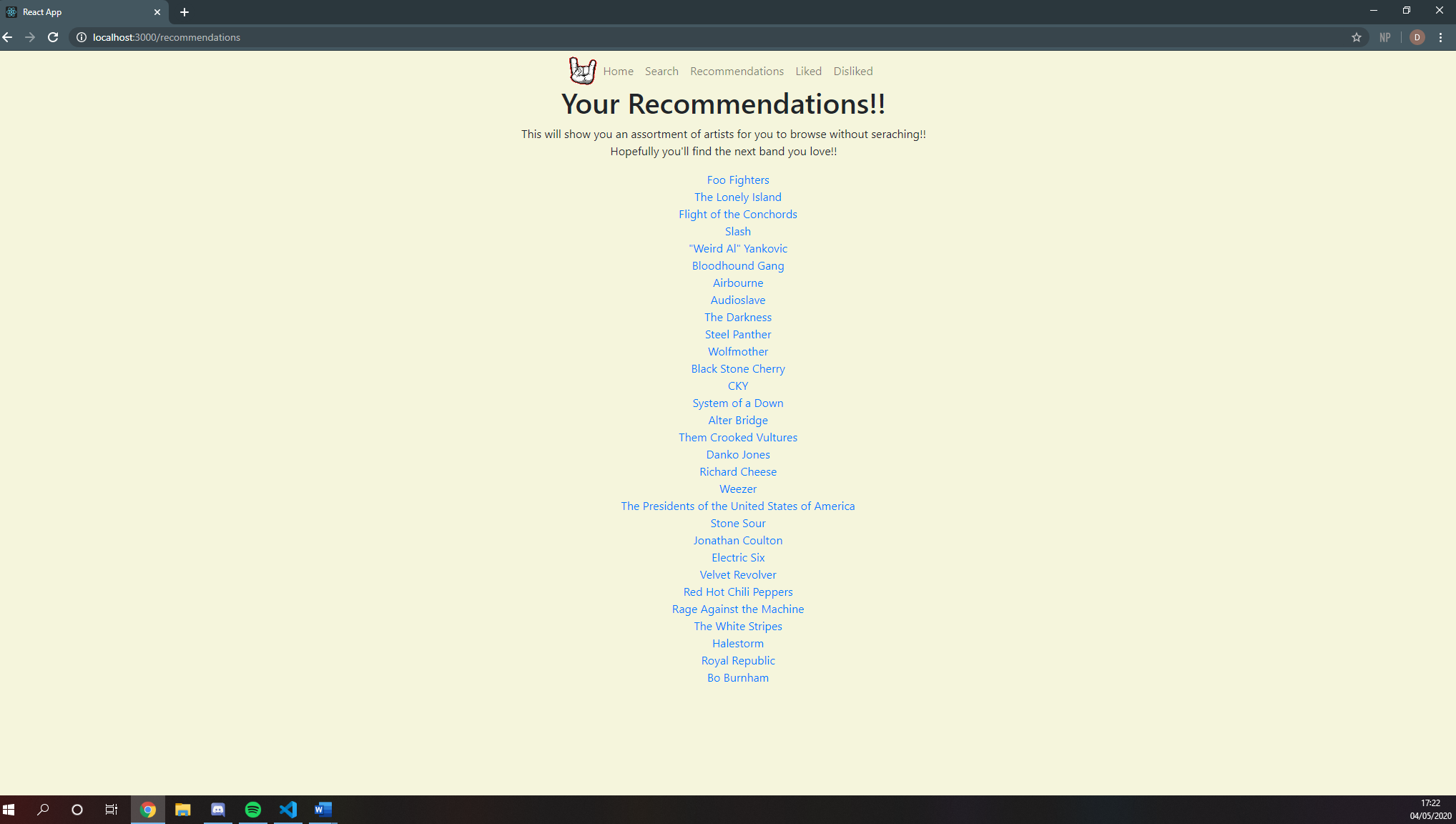


More artists can be added

### Recommendations page (Filled with Foo Fighters recommendations)



### Recommendations page (Filled with Tenacious D recommendations)



### Test Logs

My personal test:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Test | Expected Result | Actual Result | Passed Y/N |
| 1 | Start the website | Home Page loads | Home Page loads | Y |
| 2 | Type in Ac/Dc and search | Get taken to the results page with artists similar to Ac/Dc | Get taken to the results page with artists similar to Ac/Dc | Y |
| 3 | Click on top result | Get taken to the artist page for the top result | Get taken to the artist page for the top result | Y |
| 4 | Does the artist’s name load | Motörhead loaded | Motörhead loaded | Y |
| 5 | Does the artist’s bio load | Motörhead’s bio loaded | Motörhead’s bio loaded | Y |
| 6 | Does the like, dislike and search again button load | Buttons loaded | Buttons loaded | Y |
| 7 | Does the link in the bio work | Get taken to Last FM page for Motörhead | Get taken to Last FM page for Motörhead | Y |
| 8 | Does the top five albums and album covers show | The albums and album covers show | The albums and album covers show | Y |
| 9 | Do the album names link to the Last FM page for that album | The album names load and they link to the Last FM page. | The album names load and they link to the Last FM page. | Y |
| 10 | Do the song names load and do they link to the Last FM page for that song | The song names load and they link to the Last FM Page | The song names load and they link to the Last FM Page | Y |
| 11 | Do the similar artists load and do they link to their artist page | The similar artists load and they link to their artist page | The similar artists load and they link to their artist page | Y |
| 12 | Click the top similar artist | Get taken to Judas Priest’s artist page | Get taken to Judas Priest’s artist page | Y |
| 13 | Like Judas Priest | Judas Priest is liked | Judas Priest is liked | Y |
| 14 | Dislike Judas Priest | A message should appear saying they are now disliked | Judas Priest is now disliked | N |
| 15 | Go to Liked page | Judas Priest should not appear | Judas Priest Appears | N |
| 16 | Go to Disliked page | Judas Priest should appear | Judas Priest Appears | Y |
| 17 | Go to Recommendations | A message should appear saying go like some artists | Judas Priest recommendations appear | N |
| 18 | Go to Search | Search loads | Search loads | Y |
| 19 | Search Billy Squier | Billy Squier’s artist page loads | Billy Squier’s artist page loads | Y |
| 20 | Like Billy Squier | Billy Squier added to likes | Billy Squier added to likes | Y |
| 21 | Go back to search and search for Galactic Empire | Galactic Empire’s artist page loads | Galactic Empire’s artist page loads | Y |
| 22 | Like Galactic Empire | Error message should appear | Error message appears | Y |
| 23 | Dislike Galactic Empire | Error message should appear | Error message appears | Y |
| 24 | Go to recommendations | Judas Priest or Billy Squier recommendations should appear | Billy Squier recommendations appear | Y |
| 25 | Go to Likes | Billy Squier should be the only artist displayed | Judas Priest and Billy Squier are displayed | N |
| 26 | Go to Dislikes | Judas Priest should be displayed | Judas Priest are displayed | Y |

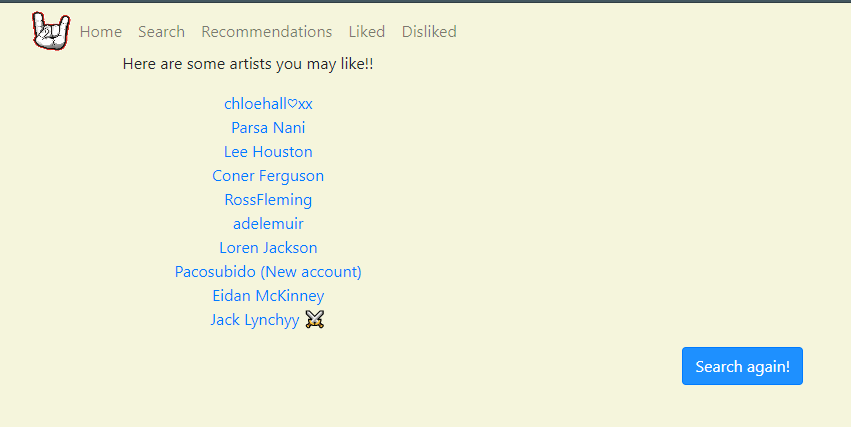
My Mum performing the same test:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Test | Expected Result | Actual Result | Passed Y/N |
| 1 | Start the website | Home Page loads | Home page loads | Y |
| 2 | Type in Ac/Dc and search | Get taken to the results page with artists similar to Ac/Dc | Results loaded | Y |
| 3 | Click on top result | Get taken to the artist page for the top result | Taken to page for Motorhead | Y |
| 4 | Does the artist’s name load | Motörhead loaded | Motorhead loaded | Y |
| 5 | Does the artist’s bio load | Motörhead’s bio loaded | Bio loaded | Y |
| 6 | Does the like, dislike and search again button load | Buttons loaded | Buttons loaded | Y |
| 7 | Does the link in the bio work | Get taken to Last FM page for Motörhead | Taken to last.fm page | Y |
| 8 | Does the top five albums and album covers show | The albums and album covers show | Yes both were shown | Y |
| 9 | Do the album names link to the Last FM page for that album | The album names load and they link to the Last FM page. | Taken to last.fm for album | Y |
| 10 | Do the song names load and do they link to the Last FM page for that song | The song names load and they link to the Last FM Page | Song laded and the link worked | Y |
| 11 | Do the similar artists load and do they link to their artist page | The similar artists load and they link to their artist page | Artists loaded and got taken using the link | Y |
| 12 | Click the top similar artist | Get taken to Judas Priest’s artist page | Taken to judas priest page | Y |
| 13 | Like Judas Priest | Judas Priest is liked | Judas priest is liked | Y |
| 14 | Dislike Judas Priest | A message should appear saying they are now disliked | No message appeared but they were disliked | N |
| 15 | Go to Liked page | Judas Priest should not appear | Judas priest appeared | N |
| 16 | Go to Disliked page | Judas Priest should appear | Judas priest appeared | Y |
| 17 | Go to Recommendations | A message should appear saying go like some artists | Message appeared | Y |
| 18 | Go to Search | Search loads | Page loaded | Y |
| 19 | Search Billy Squier | Billy Squier’s artist page loads | Page loaded | Y |
| 20 | Like Billy Squier | Billy Squier added to likes | Added to liked page | Y |
| 21 | Go back to search and search for Galactic Empire | Galactic Empire’s artist page loads | Page loaded | Y |
| 22 | Like Galactic Empire | Error message should appear | Error message appeared | Y |
| 23 | Dislike Galactic Empire | Error message should appear | Error message appeared | Y |
| 24 | Go to recommendations | Judas Priest or Billy Squier recommendations should appear | Recommendations loaded | Y |
| 25 | Go to Likes | Billy Squier should be the only artist displayed | Judas priest and Billy Squier appeared | N |
| 26 | Go to Dislikes | Judas Priest should be displayed | Judas priest displayed | Y |

### Known Errors

* Last FM user accounts will appear if the user searches for a user account

Searched for Jamie McInnes and got back these user accounts:



* The recommendations component takes a while to load the map function part of the render method
* Some artists do not have a mbid and because of this they cannot be accessed unless it is through the search
* The artists without a mbid can appear in the similar artists section of the artist/searchedArtist component
* If the user likes/dislikes an artist and then goes onto another artist page from the similar artist section the like and dislike buttons will be disabled. To fix this the page just needs refreshed.

### User Testing

In this section I will be getting my mum and dad to test the website and I will share any errors they find.

### Mum’s Testing

This is a list of what my mum did when testing the website. Any errors will be pointed out and explained why they happened.

* Searched for Adele in the home page
* Results page loaded with results on the screen
* Went to the search and searched for Adele
* Adele (searchedArtist component) loaded
* Clicked on Make you feel my love and was taken to Last FM page for the song
* Clicked on Hello single (in the Top five albums) and was taken to Last FM page for the album
* Adele liked
* Went back to the search and searched for Bananarama
* Bananarama (searchedArtist component) loaded
* Clicked on Kim Wilde from similar artists section
* Kim Wilde (artist component) loaded
* Clicked on Belinda Carlisle from similar artists section
* Belinda Carlisle (artist component) loaded
* Clicked on one of Belinda Carlisle’s song links and was taken to Last FM page for the song
* Searched for Seal in the home page
* Results page loaded with results on the screen
* Simply Red clicked from results
* Simply Red (artist component) loaded
* Simply Red disliked
* Tina Turner clicked similar artists section
* Tina Turner (artist component) loaded
* Tina Turner disliked
* Annie Lennox clicked from similar artists section
* Annie Lennox liked
* Annie Lennox link in bio clicked and was taken to Last FM page for her
* Likes in Nav-bar clicked
* Bananarama clicked from likes
* Bananarama (artist component) loaded
* Sonia clicked from similar artists section
* Sonia liked
* Recommendations clicked from the navbar
* Recommendations component loaded
* The recommendations themselves took a while to load in but did load in
* K.D. Lang clicked from recommendations
* K.D. Lang (artist component) loaded

### Dad’s Testing

This is a list of what my Dad did when testing the website. Any errors will be pointed out and explained why they happened.

* Searched in 123214 on the home page
* Searched 123214 on the search page
* Went back to the home page
* Searched right said Fred
* Results page loaded
* Clicked on Scatman John
* Clicked on Scatman John’s top song and was taken to the Last FM page for that song
* Clicked on Rednex from the similar artist section
* Liked Rednex four times
* Clicked on recommendations from the navbar
* DJ Bobo clicked on from the recommendations
* Captain Hollywood clicked on from the recommendations
* Clicked on Liked
* Rednex appeared four times in the Liked pages

### User Documentation

### Questionnaire

Name:

Date:

# Section 1 – About You

1. How often do you listen to music?
2. How do you listen to music (i.e. Spotify, prime music, Radio)?

1. Do you listen to the same type of music or do you listen to lots of different types?
2. How do you find out about new artists?
3. Have you discovered any new artists recently (without the use of the website)?
4. Would you use a recommender to find new artists to try?

# Section 2 – The website

1. What was your first impression of the website?
2. Did you like the colour scheme?
3. If not, what could be changed?
4. Was the website easy to use?
5. Did you understand the website’s purpose?
6. Did all the links you clicked work?
7. Is there anything in the website you believe could be better?
8. Did you find any new artists to listen to?
9. Did you find any artists you had forgotten about?

### Mum’s answers to the questionnaire

Name: Alison Gemmell

Date: 06/05/20

# Section 1 – About You

1. How often do you listen to music?

Daily

1. How do you listen to music (i.e. Spotify, prime music, Radio)?

Radio/Spotify

1. Do you listen to the same type of music or do you listen to lots of different types?

Different types

1. How do you find out about new artists?

Radio

1. Have you discovered any new artists recently (without the use of the website)?

No

1. Would you use a recommender to find new artists to try?

Yes

# Section 2 – The website

1. What was your first impression of the website?

Easy and straight forward to use

1. Did you like the colour scheme?

Yes

1. If not, what could be changed?

n/a

1. Was the website easy to use?

Yes

1. Did you understand the website’s purpose?

Yes

1. Did all the links you clicked work?

Yes

1. Is there anything in the website you believe could be better?

No

1. Did you find any new artists to listen to?

Yes

1. Did you find any artists you had forgotten about?

Yes

### Dad’s answers to the questionnaire

Name: Stephen Gemmell

Date: 06/05/2020

# Section 1 – About You

1. How often do you listen to music?

Every day

1. How do you listen to music (i.e. Spotify, prime music, Radio)?

Spotify, radio and youtube

1. Do you listen to the same type of music or do you listen to lots of different types?

Different types

1. How do you find out about new artists?

Friends or by internet suggestions like similar youtube videos

1. Have you discovered any new artists recently (without the use of the website)?

Yes – John Mayer

1. Would you use a recommender to find new artists to try?

Yes

# Section 2 – The website

1. What was your first impression of the website?

Good functionality but the layout is quite basic. Not snazzy.

1. Did you like the colour scheme?

Yeah it was neutral enough and easily readable.

1. If not, what could be changed?

nothing

1. Was the website easy to use?

Yes

1. Did you understand the website’s purpose?

Yes

1. Did all the links you clicked work?

Yes

1. Is there anything in the website you believe could be better?

Embedding the music inside the website rather than going to lastFM would be good but this might be a licensing issue.

1. Did you find any new artists to listen to?

Yes – John Mayer Trio

1. Did you find any artists you had forgotten about?

Yes – James Morrison

### Instructions on how to install and run the website

Pre-Requisite

Install Node.js

1. Download music-recommender folder to PC

2. Open a cmd window in the music-recommender folder

3. npm install

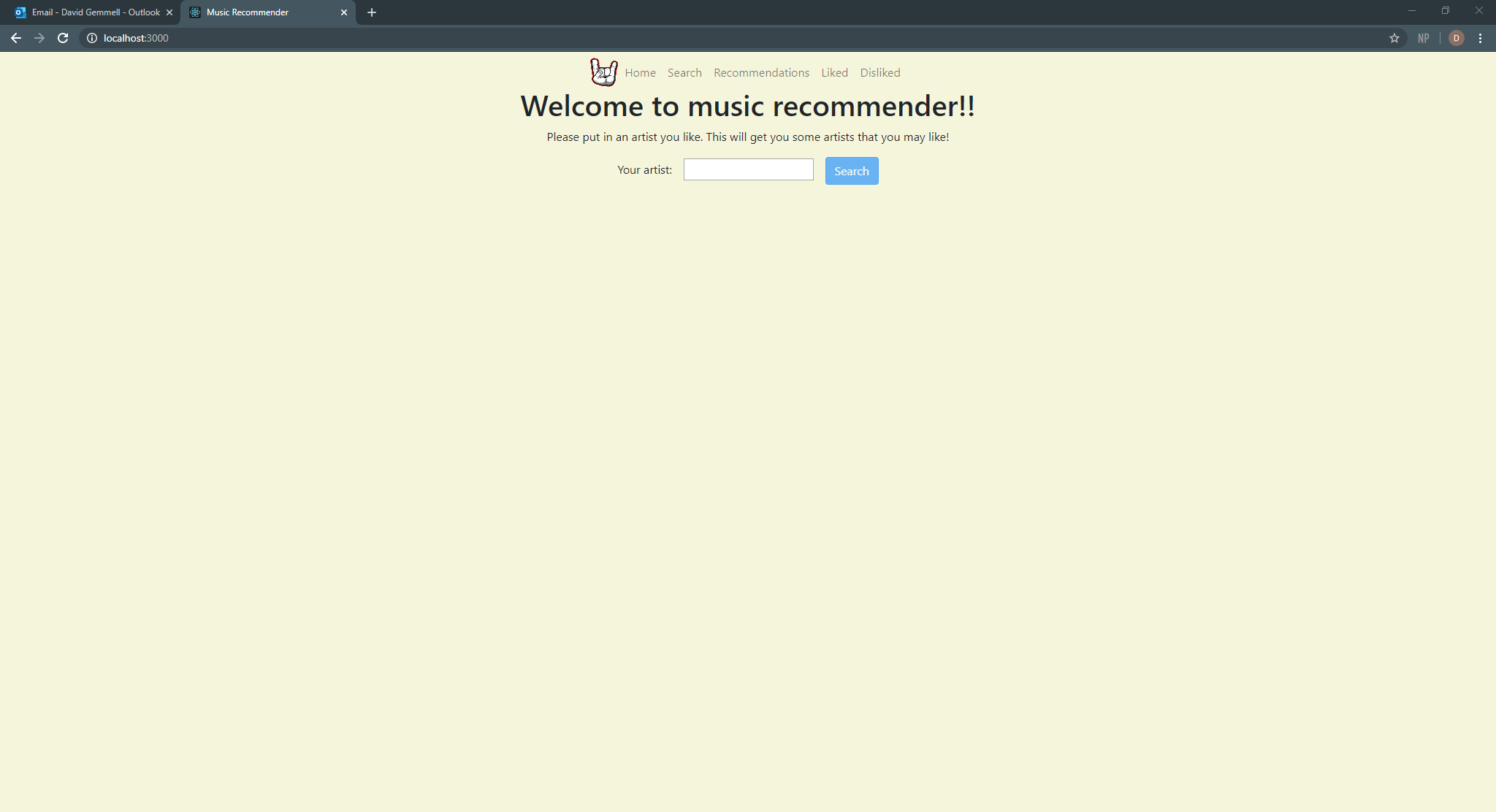
4. npm start

Have fun!

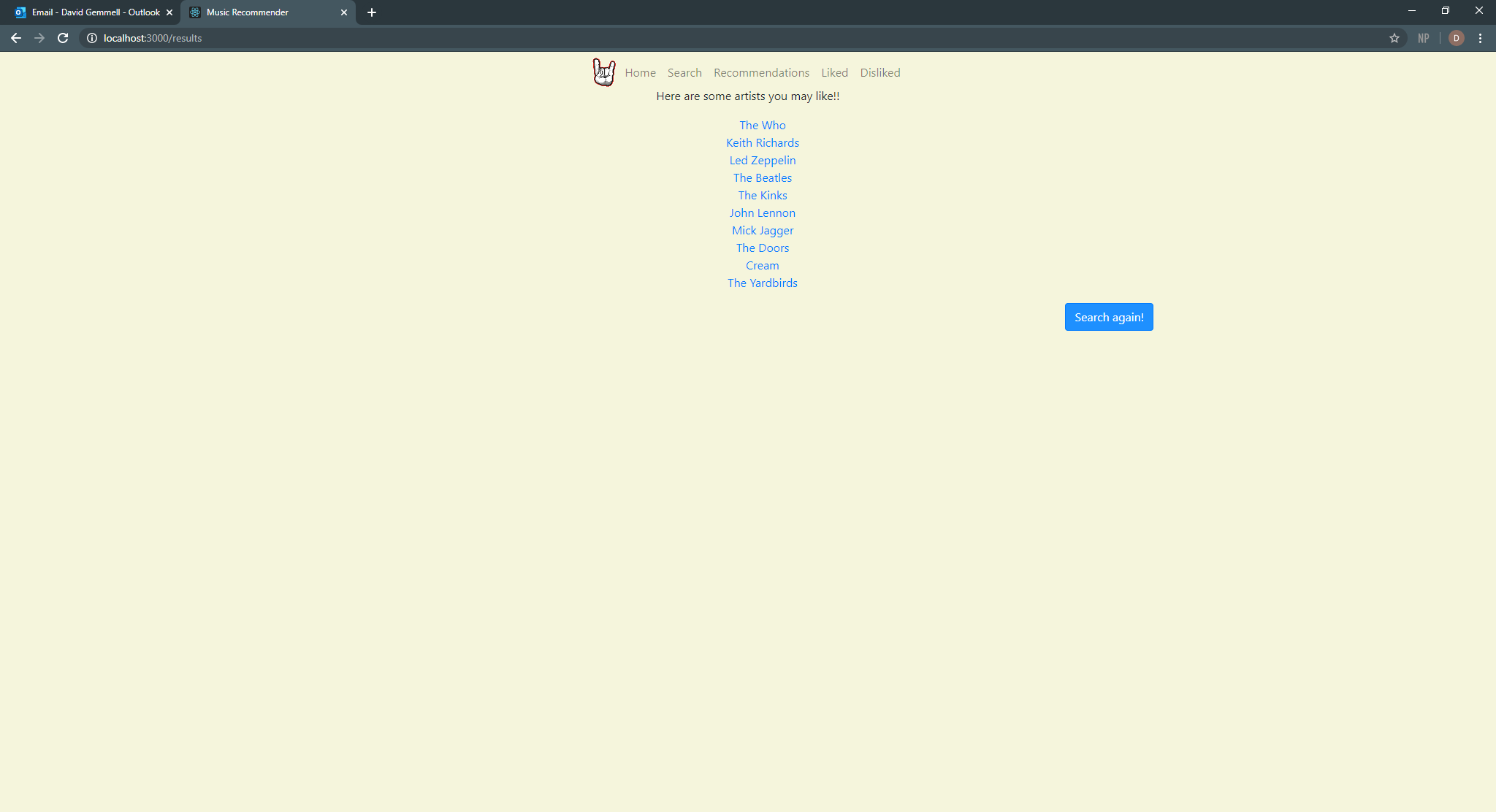
### User Manual

This is a quick guide on how to use music recommender!!

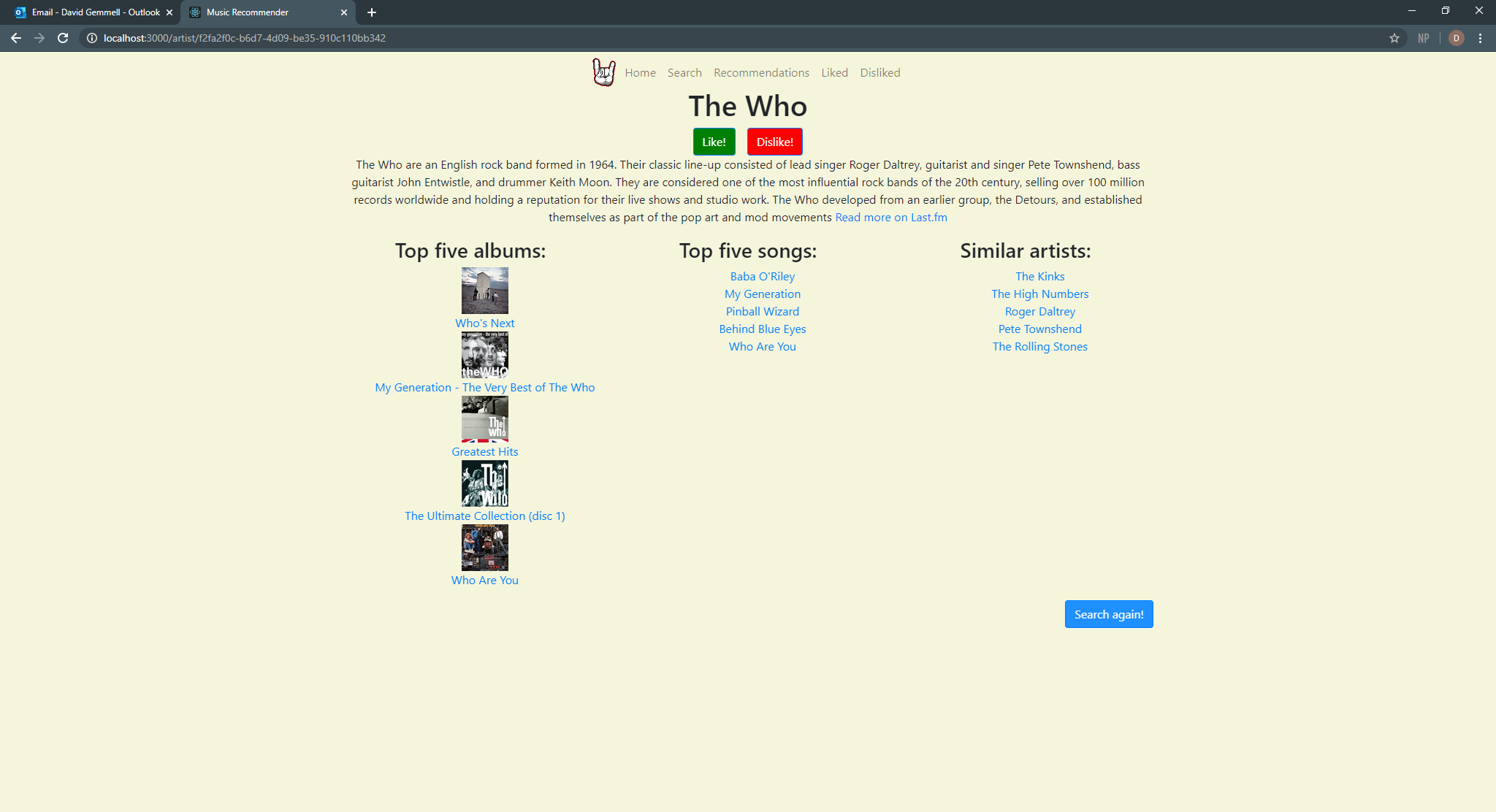
So, the first thing you will see when the website opens is the home screen. There is a navbar on every page so the user can easily and quickly move around the website. On this page, it gives the user an instruction on what to do with the input box and search button.

Home Page - 

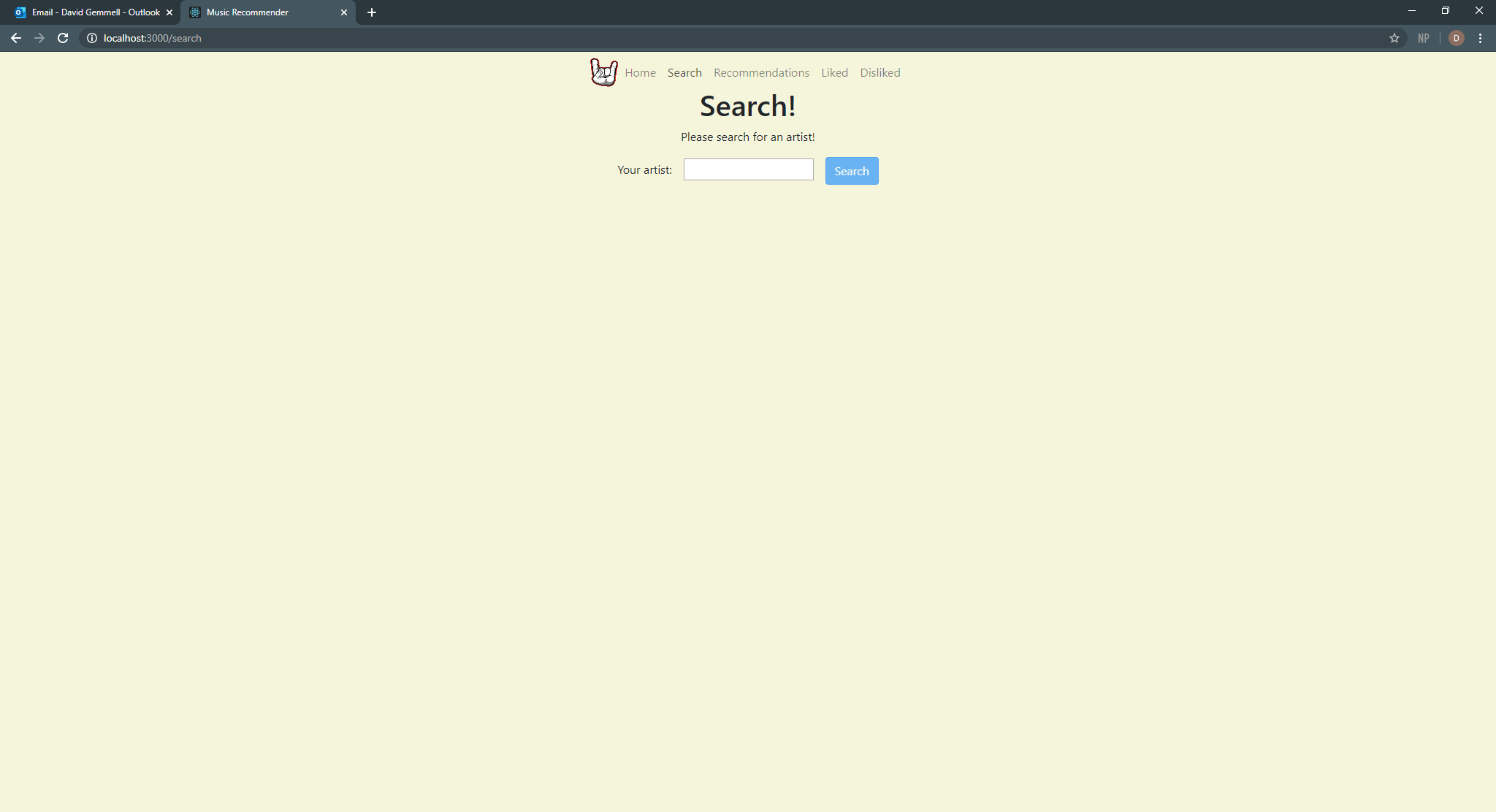
Once the user has searched for an artist, they will be taken to the results page. This page will display an error message if the artist the user searched for has no similar artists to be displayed. There is a button on the results page that allows the user to be taken back if this occurs or if they decide to change when they see the results. The results are displayed in a list that the user can see and click on anyone of them to be taken to a page that is dedicated to that artist.

Results page (Rolling stones searched for) - 

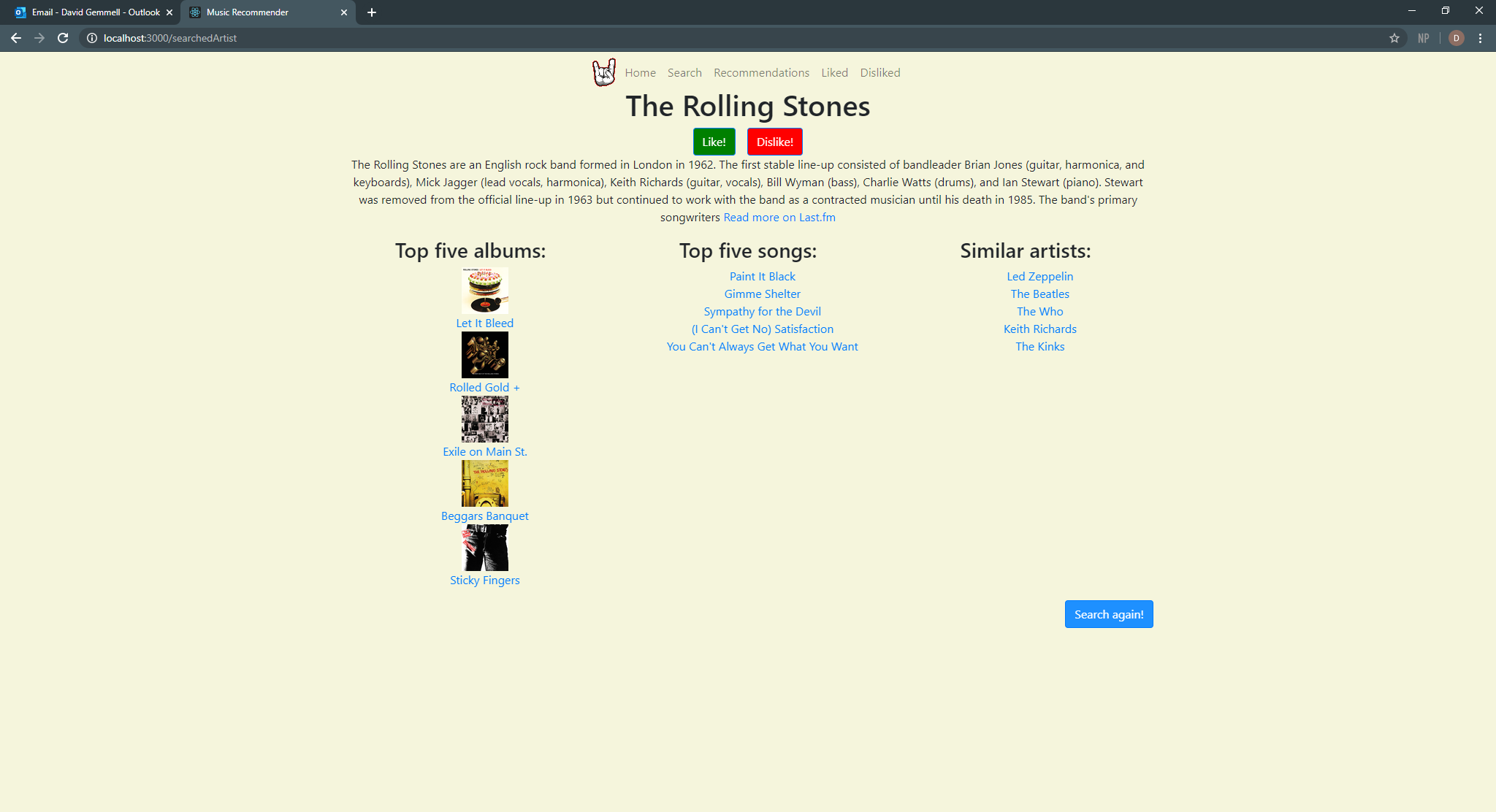
Once the user has clicked on a result, they will be taken to that artist’s page. The user will be able to read a small bit about the artist, and they will see their top five songs and albums along with a list of artists that is similar to the main artist. The user can also like or dislike the artist. They do this by pressing the like or dislike button at the top of the page. The one thing to watch out for is that once the artist has been liked or disliked they cannot be removed from that list. The button from the results page is also on this page as it provides a quick way for the user to go back to the home page.

Artist page (The who clicked) - 

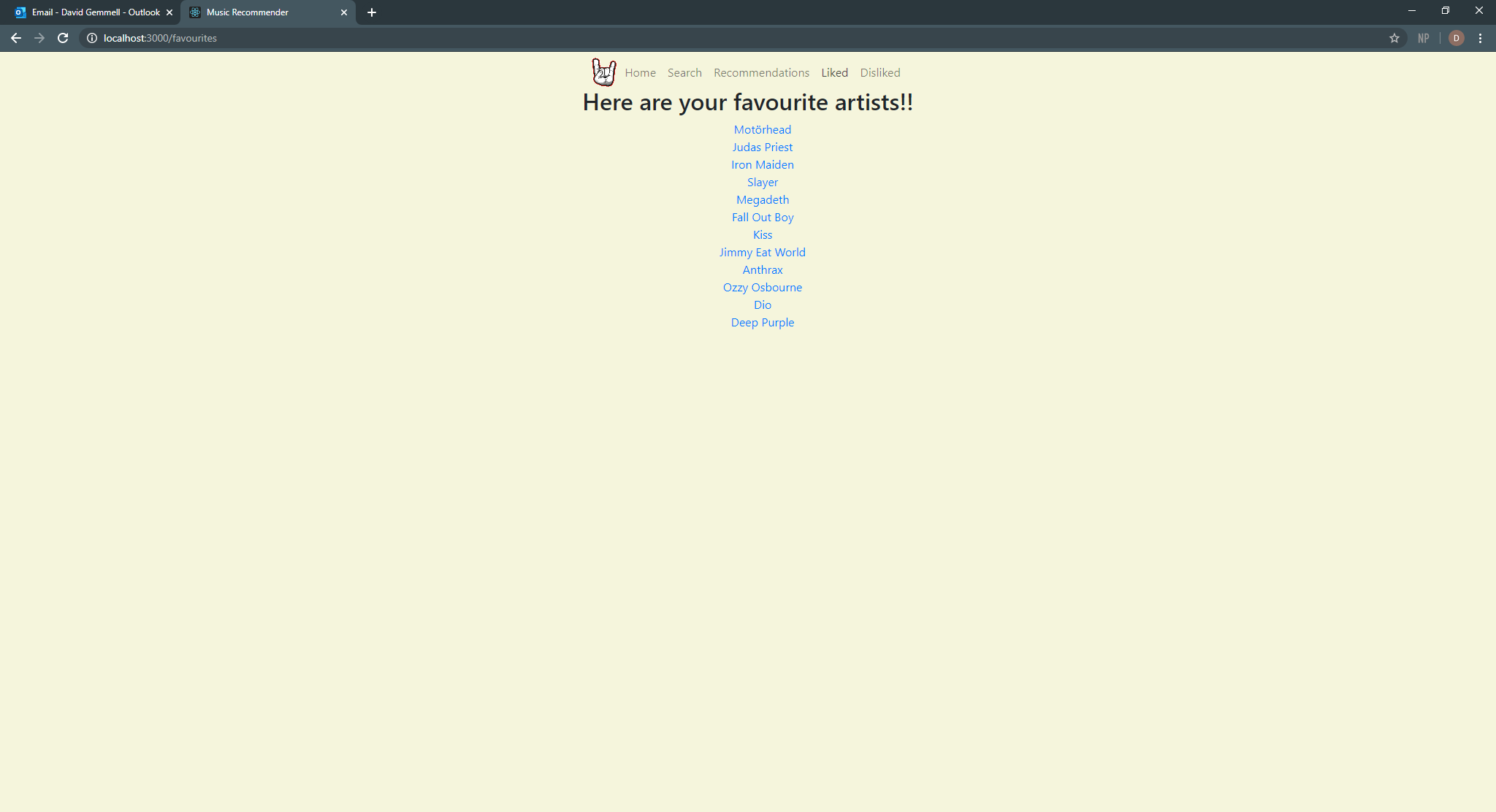
The search page is self-explanatory as all it does is allow a user to search for an artist and then be taken directly to that artist’s page.

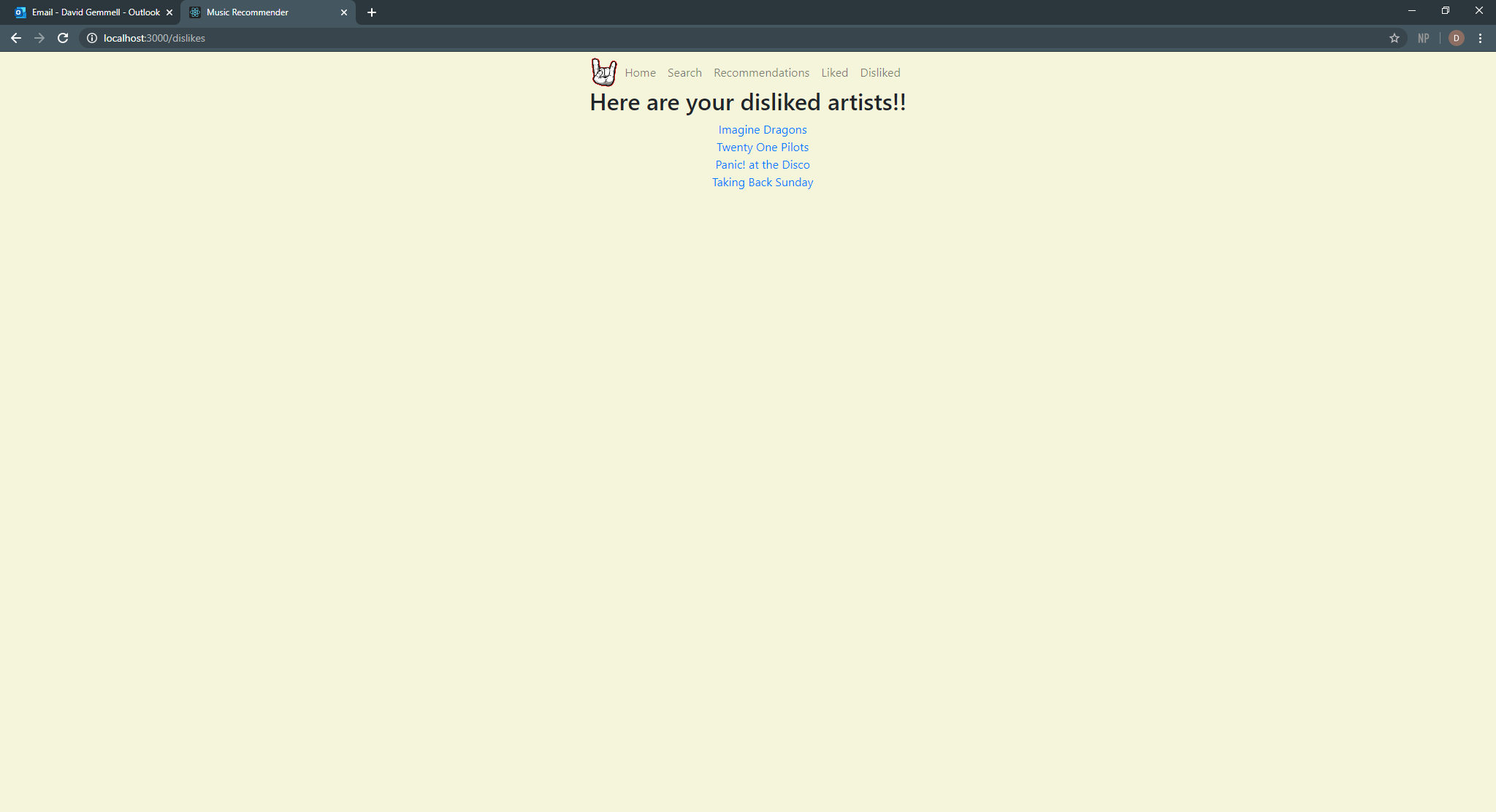
Search page - 

The searched artist page is the exact same as the artist page that the user can get to by using the home page and then the r (Sebastian, 2018)esults page. It displays the same data and can do all the same stuff as the normal artist page.

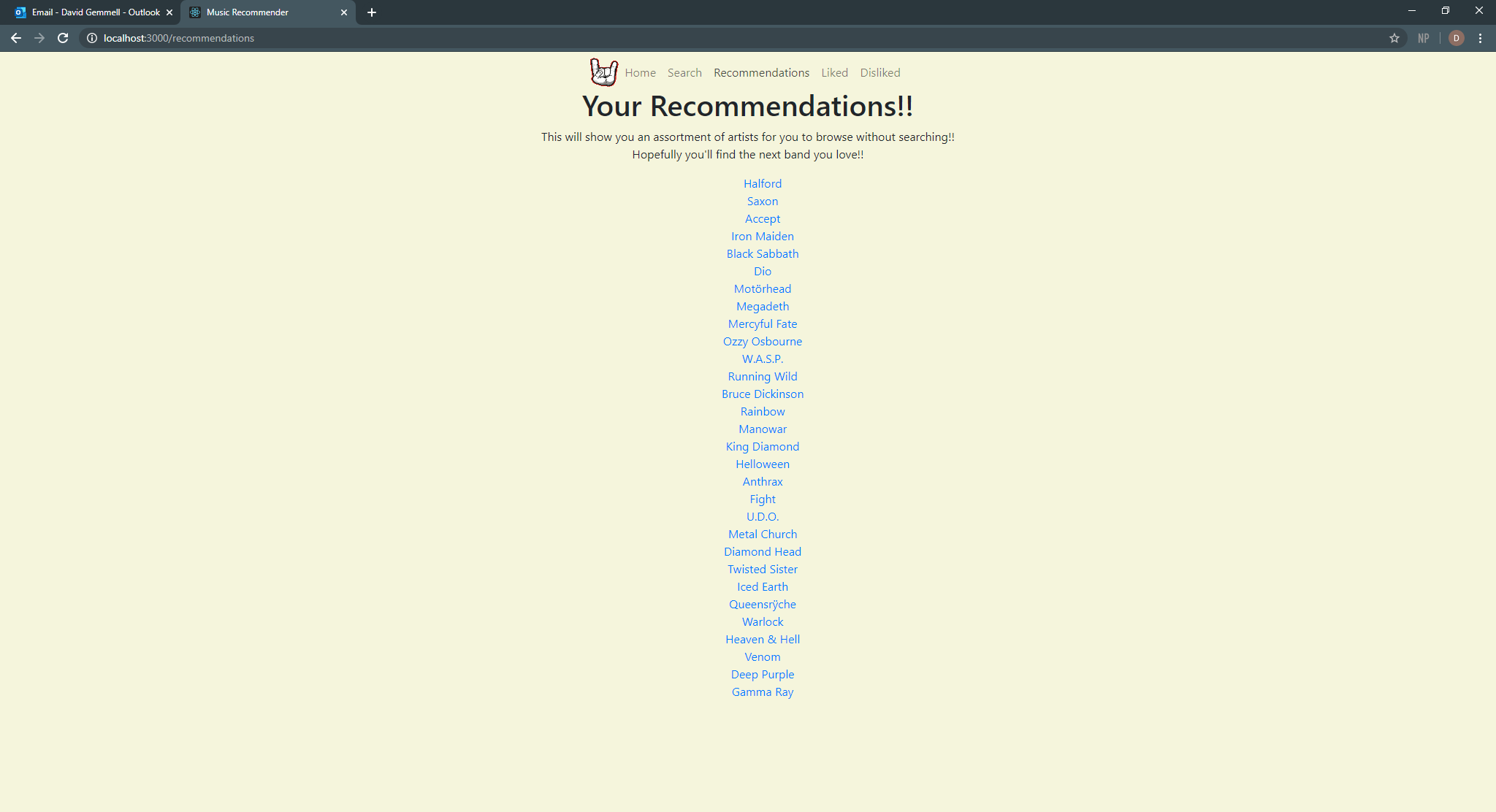
Search artist page (Rolling stones searched for) - 

The liked and disliked page do the same thing and that is they both display a list of the liked or disliked artists respectively. This is so the user can get to one of their liked/disliked artist’s pages quickly in case they want to see who is similar to them.

Liked page – 

Disliked page - 

The recommendations page is for users that have liked some artists and they just want to find artists without having to think about an artist. The page will give the user a list of thirty random artists for the user to look through and click on. Once they have clicked on one of the artists, they will be taken to that artist’s page to look at. Every time the user goes to the recommendations page, they will get new recommendations so they can find new artists quickly and easily.

Recommendations page - 

# 4. Evaluating

This project started with an idea to bring new music into people’s lives. At the time I got this Idea I was listening to a band I had just discovered called RMB. I found them through YouTube and that got me thinking about a recommender system. At that point I didn’t really realize the size of this project but I was certain I wanted to create a website that would allow users to go to the website and search for their three favourite bands. I had gotten the idea for the searching of three bands from GNOOSIC (Gibney, 2002). This website, to me anyway, had a good basis for what could be a great website. It had a good idea, the fact the user could put in three of their favourite artists and then be recommended a list. The issues with the website though were just too much. For example, the website was bland, the colour scheme is very basic with black and white, and the only two other colours are used when you click a button. The other major issue for me was the fact that the website allowed the user to like artists but once you had completed a run through of the website it forgot what artists the user had liked. I decided to add a liking system because of the lack of this feature in this website and this concept I had had become my project. I do say I ran out of time a lot but this is because when I handed the project in, I did not have enough time to do anything else to my website with the time I had remaining before the deadline.

In the planning stage, I created a list of functional and non-functional requirements. I made these to give myself a guideline on what must be done for the development deadline. Most of these requirements were fulfilled however when I started working on the website, I realized that it was going to take me much longer to build my ideal version of the website so I decided to make some changes. I will now go through each requirement and explain why it was or was not completed.

### Functional requirements

**The user must be able to put their top three favourite artists into the search: wasn’t completed.**

This requirement was scrapped very early on. This was because to save time, I decided to start off by allowing the user to enter their favourite artist and then once everything was working with the one, I was going to allow a second artist to be searched for and then a third once everything was working. This was so at the very least my website would be fully functional for the deadline however over time I realized even if I finished everything and made sure nothing was wrong with the program I wouldn’t have had enough time to implement a way to allow the user to put two artists into the search. However, I did use the remaining time I had left once most of the original program was finished to create a normal search that allows users to go straight to an artist’s page.

**Once the user has searched for similar artists, they must be returned the top 2 similar artists for each artist they inputted: wasn’t completed.**

This requirement was changed when I changed the first requirement. I decided to just instead show a list of artists so the user would have more artists to look through and be able to go to. This was going to get replaced with the original requirement once the ability to input three artists was implemented however I wasn’t able to add that ability so I left it as it is and I think it works just as well as the original as even though it’s only for one artist, the user gets a lot more artists back than they would from the original design.

**The user must be able to click on any of the results and be taken to a page for that artist: was completed.**

This was a big part of the website, so it was one of the first things that was developed. I made sure this was incorporated into the website as I knew just seeing a list of people you may like wouldn’t be good enough as if I was the user, I would want to click on someone and learn more about them just out of curiosity. I think this was a key feature that under no circumstance should have even been considered getting removed and as such it wasn’t even put under consideration from getting removed.

**The results page should display the artist’s name and a picture of either the artist or their top album: was and wasn’t completed.**

This requirement was partially completed. This is because the results page shows a list of artist names however, I did not add the images in this component. I decided to move the images into the artist’s personal page and from there I showed their top five album covers. I still think it would have looked good if the image was in the results page, I think it makes more sense to showcase it in their personal page.

**The information page must show some general information about the artist and it should show their top 5 tracks: was completed.**

This requirement was completed and actually improved upon. I say this because I had initially only wanted to show the artist’s top five songs but as I got more used to Last FM’s API, I found out I could get a lot more information about an artist. This made the artist’s personal page much better and it allows for the user to learn more about the artists when they go their personal page.

**The user must be able to click on the any of the top tracks and be taken to the Last FM page for that song: was completed.**

This requirement was completed and as with the last requirement, it was improved on. This was because I made the track names links to the Last FM web page and I did the same with the album names as well. Meaning that the user can find out more on Last FM is they want to about one of their songs or about one of their albums. This means that the user can easily spend more time finding out about new music while going to Last FM’s website and can even listen to the music from Last FM.

**The similar artists that appear should be able to be favorited or disliked: was completed.**

This was completed and the point of this feature is for the user to be able to go to their favourite artist’s page without having to perform a search to get to them. It was really implemented because the websites I looked at during the planning stage didn’t have this feature so I wanted to add this to make my website stand out from the rest of the music recommendation websites.

**The machine learning should use the favorited and disliked artists to find the recommendations: was and wasn’t completed.**

This is a weird requirement as I do use the favourited artists to get the recommendations however I did not use the disliked artists. The way I had envisioned this was that using the favourite artists I would get a list of recommended artists and then using the disliked artists I would check and then either remove or replace those artists with ones that haven’t been disliked. I would say this requirement was completed to a certain extent however I just ran out of time and was unable to figure out how to perform the necessary checks and removal or replacement in time.

### Non-functional requirements

**The website must have a consistent colour scheme throughout all the web pages. These colours should all be web safe: was completed.**

The website sticks to the same colour scheme and the main colour for the website is beige as it’s meant to be a relaxing website to use and I felt that the colour beige was the perfect fit for that purpose. I used the colours green and red to signify if the user likes or dislikes an artist by making the like and dislike buttons green and red respectively. This was to show the user that there was a difference between the two fundamentally.

**The text on the website must be readable: was completed.**

The text in my website is readable as the black colour of the text easily stands out from the beige background as well as the blue for the links. During my testing neither my mum or dad said anything about the text on the website and they were able to read the instructions and easily interpret them and use the website.

**All the links on the website must work and there should be no dead links: was completed.**

All the links on my website are either links to webpages from the website or they are links to Last FM’s website. During my testing I didn’t find any dead links however, if anything happens to Last FM’s website or for certain artists then certain links would become dead. This is a slight issue but I would not be able to fix the links to Last FM as I didn’t develop Last FM.

**The website must be secure so that no one can take over the website: was completed.**

Due to the nature of my website, being started from a local machine and hosted on localhost the website is secure from everything except the kernel and other applications running in the localhost. (user207421, 2013)

**The website should have fast reaction times so the user can have a good experience with the website: was completed.**

For the most part the website has quick reaction times the only place I noticed some issues with was the recommendations page however, it does load the recommendations it just takes slightly longer as it needs to perform quite a lot of things to ensure that the website runs smoothly.

**The deadline for this website is 15th May 2020: was completed.**

I completed my development stage for this deadline and was able to hand it in a few days before the deadline.

### Strengths and weaknesses

While working on this project I found that I was better at writing the documentation rather than the website itself. I believe this is because I have had experience in the past writing technical documents and some user documents but I have not had that much experience creating react applications from scratch. However, I believe that I started out with the weaknesses of the website creation I got better at creating it as I worked on it more and more. I still would say it’s more of a weakness but I feel more confident in my ability to create react applications.

One of my strengths was the planning of this project. What I mean by that is, I was able to set a clear plan of action for myself to follow. Because this plan was set up from the beginning, I was able to follow it well, however once the lockdown started, I had to adapt the plan. I was able to quickly change the original plan so it would better suit me while the country is in lockdown. I changed the plan by making myself work on the website for at least three hours a day on Mondays to Fridays. By setting this new plan for myself it allowed me to really get stuck into my project and I believe lockdown actually helped me. I say this because if lockdown had not happened, I would not have been able to put as much time into the development as I would still be going to college and I would also have to go to work. I would have been able to do some of my project work in work however my work is customer based and so I would be getting pulled away from it every five minutes.

One of my strengths is definitely creating the technical documentation. I find creating the technical documentation easier and because of this I can create it faster while keeping it to a good standard. For example, my use case diagram I submitted in the planning stage only took me a few rough sketches and then two attempts at creating it on lucid chart to make the submitted piece. The fast pace can be a slight downfall but to counter this I always make sure to stop after creating a sketch and look over what I have just done, to check if I have missed something and to check if anything doesn’t need to be there.

Obviously, the technical documentation doesn’t just cover the use case diagrams. It covers all of the diagrams I used in the planning stage. I would consider most of these diagrams strengths as I can usually create them quickly and to a good standard however, I found that creating the user documentation was tougher than I expected.

The reason for this was because I didn’t really know what I was supposed to create for my user documentation. In the end I went for install instructions and a user manual as they were the only two things I could think of. The install instructions are very basic and only have four instructions for the user to actually follow but I think my biggest let down was the user manual.

I believe I didn’t do a good job of my user manual and looking back at it, I could have made it so much better. I feel like I didn’t do a good job explaining the purpose of each page well enough and I used visual aids to showcase what the website looked like. I think the user manual would have been better if I spent more time working on the descriptions of what each page’s purpose was and how the user could use the website as the website can be used in a multitude of ways. For example, one user could might not like or dislike any artists meaning they won’t get any recommendations on the recommendations page. Whereas another user could use this specifically for that purpose of liking a few artists and then going to the recommendations page for them to browse for their leisure.

My biggest weakness during all of this project was definitely my lack of experience with working with react. However, this was a good weakness to have as it forced me to learn more and more about react so I could build a react application to a good standard. The issue with this weakness definitely impacted my project significantly because at the beginning I had barely any experience working with react so it took me a while to learn and get the hang of it. I had to watch tutorials on YouTube (Mosh, 2018)to learn the basics of react just so I could create a basic react application.

Another part of this weakness was my lack of experience using API’s. I had used APIs before but it was very limited use. So, I had to watch a couple of videos to learn how to do this (Sebastian, 2018), (Last FM , 2013).However, I also had to make a decision on which API to use. Originally, my plan had been to use Spotify’s API however once I looked at what that API could do, I realized what I needed wasn’t there. That’s when I had to look for another API. I found Last FM’s API and this API had everything I needed. I ended up using Last FM’s API for my project and this was a real benefit as the API had every method, I could have wanted.

My lack of experience weakness has pretty much disappeared now because I spent so much time and effort working on this project so the website would be good. Obviously, I still have this weakness because making one website isn’t that much experience. However, I still gained a lot of knowledge creating this website and I will be able to take that knowledge and apply it elsewhere later on, on other projects.

### Recommendations

Here is a list of my recommendations for future development on website:

* A login feature should be added
* The ability to remove artists from liked or disliked
* I would make the layout of the website more appealing
* I would ensure the user’s disliked artists can’t appear in the recommendations page

I will now discuss these recommendations and why they should be implemented in the future.

The first recommendation “A login feature should be added” is recommended as originally, I wanted to add this in but I decided against it because of the time constraints. However, in the future when I can work on it at my own pace without any time constraints, I would want to add a login function so multiple users can use the same browser on the same machine and use it as a personal website. I think that if the user can login it would make the website much better as then different users would have different lists for their favourites and their disliked. But this change would also mean that the recommendations would really become unique to the users themselves as one user may like the Beatles, Elton John and Queen but dislike Led Zeppelin David Bowie so those two won’t appear in their recommendations whereas a completely different user could like Alestorm, Dropkick Murphys and Dio so they would get completely different recommendations. This change would take this website to the next level as if I host the website then multiple users could use it at the same time while logged in to make a collection of their favourite artists and constantly discover new or old forgotten artists to listen to. Plus, while I am thinking about adding a login for users, I could also add an account page where users can make their favourite artist shown their favourite album and their favourite song. I would let them write a bit about themselves and I would allow other users to view their favourites and dislikes.

The second recommendation “The ability to remove artists from liked or disliked” is recommended as I had planned to implement this but I just ran out of time. The user can still like or dislike artists however once they have liked or disliked an artist, they cannot undo that choice. This can obviously lead to some frustrating choices as well as looking back and wondering why you disliked an artist when you now love them. I would add this ability so the users can have more control over their favourites and dislikes lists so they can constantly update their favourites and dislikes accordingly. The reason I would want this if I was a user is because my music taste is always changing, one month I will be really loving a band and then the next I will get slightly bored of them and move onto another band or artist. Because of this being unable to like an artist I formerly disliked or the opposite would be annoying and it would probably make me wonder why I can’t remove them from liked or disliked. As I am thinking about this I realize another feature that could come under this recommendation would be a way to clear all of the user’s favourites or disliked or both of them so the user can start afresh and select new favourites, new dislikes and get, possibly, completely different recommendations from their previous recommendations.

The third recommendation “I would make the layout of the website more appealing” is recommended because the layout is very basic. The reason it is basic is because I wanted to make sure the website functioned well rather than looked good when I handed it in. I understand that I could have made it much better looking if I used bootstrap as bootstrap not only makes the website look good but I can also use the same piece of code to make something and apply it somewhere else in the website. For example, the liked and disliked pages. These two pages are the exact same with the only difference being what they display, the user’s likes and dislikes respectively. If I used bootstrap, I could have made these lists look better by maybe putting them into rows of their own and highlighting the rows when the artist name has been clicked. That’s just an example of what I could have done but I could also use it for designing the login and registration forms if I create the login system from my first recommendation. A saying that works with food and with websites is “the first taste is always with the eyes” and obviously with food if the dish looks good you have high expectations for it to taste good as well. The same goes for websites, if the website doesn’t look good and presentable to the user then the user will get bored and not really pay attention. I know and understand at this point that my website is not the most visually striking and it’s not the most well laid out. That is why I would start to implement bootstrap into the lists, the forms I would make for the login and registration pages, even the artist pages could use a bit of sprucing up to make it more visually appealing.

The last recommendation I would make is “I would ensure the user’s disliked artists can’t appear in the recommendations page”. I recommend this because it would be a bit weird to see an artist that the user has disliked be recommended to them. It would be like someone hating the film “Rocketman” but then getting recommended it by their friends. It doesn’t make sense for that to happen to the user. I would definitely recommend this to be one of the first things that should be fixed however, I can see a benefit of being recommended one or two artists that you don’t like. This is assuming that the second recommendation has been put into the website as then when users see someone they don’t like they could have a small reflection as in “Do I actually dislike them?” and then they can go that page and either leave them as disliked or add to them to likes. However, while that could be a good thing, I think most people know what type of music/artists they do and don’t like. So, I would probably end up making sure no disliked artists can be recommended.

Overall, I would want to implement all of these recommendations in future development because each one adds to the users experience by either updating existing features and making them better or by adding completely new features and adding even more content for the user’s to digest and use to whatever they want to.

### Modiifications

My original project plan was followed for the most part. Most of the plan I had created originally was put to good use as I was able to follow the designs and the documents to create a website that met most of the requirements I had originally outlined. The one thing I did change was the design of the website as the storyboards I had originally created did not fit the finished website so the overall design of the website was changed to a more suitable design. I also created a new feature I did not intend to add in the original design documents. This is the search function. This function was implemented as once the main functionality of the website was created and tested and worked, I felt that there weren’t enough things for the users to do. That’s why I added it and I feel that it adds to the overall user experience as it allows users to go straight to an artist, of their choice, page. By adding this it makes the website better as it allows the user’s to quickly find artists that they want to, and then learn about them. This didn’t really impact the development as the search and the searchedArtist page were made quickly as the basics for the page were already created in other pages, the home page and artist page to be specific. The only thing it did impact was the amount of testing I had to do, but even at that it was mostly the same as the home page testing and the searchedArtist testing was the same as the artist testing.

Another thing I planned to do was to allow the user to login, I didn’t really show this in the storyboards as at the time of creating them I hadn’t really thought about how to make a user’s favourites and dislikes unique to them. I ended up using local storage and not creating a login feature as I ran out of time to implement it. The login feature was supposed to be added, and I talked about it in my recommendations, however because I left it for once the website was working as it should I had ran out of time to even attempt to make it a feature of the website. It is a shame this wasn’t implemented as this would have made the website better with users being able to use the same machine and browser as each other but in the current state of the website using local storage works well and to implement this would have taken too long with the remaining time I had left.

Another thing that I had planned to do was to show the artist’s top album cover in the results page so the users could recognize some of their work. I originally intended to showcase the artists top album image to make the results page more visual but as time went on, I decided to move these images to the artists page as I wanted this to be where the user can learn more about the artist and it didn’t make sense to me to show the same image twice. I ended up showing the top five album covers on the artist page so the user can have a visual aid to go along with the album names. This didn’t really impact development as I had not implemented the images when I decided to move the images to the artist page. If anything, this actually sped up development as I did not have to spend a lot of time on the results page as this was simply displaying a list rather than having to get the images as well.

### Unforseen Events

During the course of my project I did run into a couple of unforeseen events. These were the time constraint and me running out of time and obviously, Lockdown. I will talk about the time constraint first and then the Lockdown.

At the beginning of development, I had not anticipated the time constraint being such a big influence on my project. About halfway through the development period, I realized that I did not have enough time to do some of the original features so I had to change my plans accordingly. I had to take some of the features I had originally planned to be in it, such as the login feature. The login feature was mentioned in my recommendations as I wanted to add it in, but I still feel that it should be implemented in the future.

At the beginning of the development phase, lockdown was not in place. Obviously, the coronavirus pandemic forced most countries to go into lockdown and this actually benefitted me as it allowed me to spend much more time working on the development which led to a much better end project. This being said, obviously the pandemic is horrific but, and I’m sad to say this, without it, my project would be nowhere near how good it is without the pandemic.

### Knowledge and Skills

I will now list the knowledge and skills I have learned over the course of my project and then I will discuss how and why these have been achieved. The list:

* Knowledge of React Applications
* How to use Local Storage
* Knowledge of how to use APIs
* Research Skills

### Knowledge of React Applications:

Over the course of this project I gained a lot of knowledge whilst doing it. Most of the knowledge, if not all, was about React Applications. Going into this project I knew a tiny bit about React however I was not confident in my ability and I wanted to learn more about React. I started out making a very basic React Application and over the course of the development, I had to learn more difficult techniques for the website. The first difficult technique was the map function. The map function is quite easy to do now that I have done it multiple times but the first time trying to get it to work was difficult I had to watch a couple of YouTube videos to teach myself how to use it (LevelUpTuts, 2016) and once I had learned how to do it, it became simple and I implemented it many times in the website. It is an essential piece of the website so it was critical that I learned how to do it quickly and well. Once I had learned it, I quickly implemented it and made sure it worked. Most of the other knowledge I had to learn had to do with Local Storage or APIs so I will talk about that knowledge in their own sections. However, I gained a basic understanding of React and because of this I can now confidently say that I can code in react and produce a website that is a good standard using react.

### How to use Local Storage:

I realized pretty late on into the development stage that I had to use a storage system that wasn’t on my PC as react doesn’t support this type of file storage. So, I had to learn how to use Local Storage. Local Storage is local to the browser that is opened, so I figured that this was a pretty good substitute for a file system on my PC (X, 2019). This video helped me learn how to use Local Storage however I did still have difficulty learning about the callback function it utilizes. It basically says do all the other code first and once that’s done do this code inside the local storage function. I learned it once I had done it multiple times however it still trips me up now and again. I would be able to use it again if need be but I would need to refamiliarize myself with it before trying as it still trips me up even after finishing the project.

### Knowledge of how to use APIs:

The biggest part of this project was definitely the use of the Last FM API. This API made the project possible as it had all the methods I needed and more in case I want to expand it in the future. I learned how to call API’s with the use of axios from (Sebastian, 2018).This was my first exposure to React and I followed that tutorial completely, this not only taught me the basics but it also taught me how to use axios. In that tutorial axios was used for calling a database but I learned that calling to the database could be changed to calling an API. This allowed the website to become what I wanted it as I could now use the API in whatever way I wanted. Understanding axios was essential as it allowed the API to be called and made the website into what it is, luckily, I knew axios from the tutorial but I did gain more of an understanding when using it in the website.

### Research skills:

This is hard to narrow down as for this project I had to do a lot of research. For instance, I had to research each one of the afore mentioned new knowledge so I could make the best website possible. The only way I got this new knowledge was with my research. I think my research skills have gotten substantially better as the project got on as I was able to find what I needed at the time I needed it which is a great skill to have as it will help me no matter where I go in life as I may need it in work, university or even just in life if I need to research things that are trivial such as finding out if an area is good to live in or even flight prices for a holiday. It is a lifelong skill that is a great one to have and I have to thank this project in allowing me to better not only my research skills but my own knowledge and how to perform research. I discovered techniques that probably only I use which will make it unique to me.

### Development process

The only thing I really think could have been improved about the development process is how I managed my time. I know that at the beginning I did not put as much work in as I should have. What I mean is that before I went to Denmark with the college, I had basically just made the basic react app and nothing much else. While I was away in Denmark, I barely touched it. I did the remaining diagrams I was handing in at development stage and chose my image for the navbar and that was it. It was really when I came back, and all the news about coronavirus came out that I actually took a step back and realized how important this project was. If it wasn’t for that and my mum and dad pushing me to be the best I can, I would have stayed the same and the website wouldn’t be as good as it is now. Once I got back and had this moment of self-reflection, I realized I had wasted so much time and made myself work for at least three hours a day (excluding weekends) and I was determined to make this project the best I could make. I had a lot of time to make up for and because of lockdown I was more than able to make up for it.

Possibly the only other thing I can think of that really impacted my development of this website was me breaking up with my girlfriend during the lockdown. It happened quite early in lockdown so it was a blow before I had really even started the project properly as I said above. This was devastating to me but I got over it quickly by diving into my work, it was healthy for me and I used this to my advantage as it made me spend more time working on the project as I wanted to lessen the blow as much as possible. It still hurts but the project got me through because of all the things I had to do. It’s genuinely been one of the best experiences I’ve had as a programmer although lockdown did impact the experience since I worked on it every day but I will always remember spending the time everyday creating this website.

# 5. Conclusion and Recommendations

In conclusion, over the course of this project I had ups and downs but I will always remember getting up in the morning having breakfast, a shower and then getting ready to work on the project. It helped me deal with some personal stuff and it helped me avoid boredom from lockdown. Most people could look at doing a massive project in lockdown as a curse but I think it was a blessing in disguise as I was able to learn so much from so many different sources and that will help me in the future I can guarantee it.

This was my project music-recommender and to whoever uses it I hope they have fun with it as it was intended for users to have fun finding new music they can share with their friends, or music they like and music they have special memories linked with. Music is special for a different reason to everyone, for me it brings back amazing memories of my grandpa Dave who I never ever talked to about music but I will always think about him when I hear wish you were here by pink Floyd. He had a big impact on my life without me even realizing it and it just makes me realize how much I miss him every time I hear it. It’s bittersweet since he’s gone now but I have perfect memories of him.

Thanks for reading this.

# 6. Bibliography

Bowes, J., 2014. *Agile vs Waterfall: Comparing project management methods.* [Online]   
Available at: http://manifesto.co.uk/wp-content/uploads/2014/07/waterfall-project-management.jpg   
[Accessed 24 January 2020].

Codecademy, 2019. *Software Development Methodology: What is Agile?.* [Online]   
Available at: https://www.youtube.com/watch?v=GzzkpAOxHXs  
[Accessed 30 January 2020].

Codecademy, 2019. *Software Development Methodology: What is Waterfall?.* [Online]   
Available at: https://www.youtube.com/watch?v=LxEmGNgqYJA  
[Accessed 30 January 2020].

Duckett, J., 2011. *HTML & CSS design and build websites.* N/A ed. Indianapolis: John Wiley & Sons, Inc.

Gibney, M., 2002. *Gnod's World of Music.* [Online]   
Available at: http://www.gnoosic.com/

Gilley, C., 2015. *The Pros and Cons of Agile Product Development.* [Online]   
Available at: https://community.uservoice.com/blog/the-pros-and-cons-of-agile-product-development/  
[Accessed 26 January 2020].

GitNation, J. M. -. R. C. b., 2019. *How Many Ways to Say I'm Sorry, Error Handling in React - Jesse Martin.* [Online]   
Available at: https://www.youtube.com/watch?v=ExC0N1XHaRQ  
[Accessed 18 February 2020].

Kolowich, L., 2018. *Web Design 101: How HTML, CSS, and JavaScript Work.* [Online]   
Available at: https://blog.hubspot.com/marketing/web-design-html-css-javascript  
[Accessed January 2020].

Kukhnavets, P., 2018. *Why Agile is So Popular in Project Management World.* [Online]   
Available at: https://hygger.io/blog/wp-content/uploads/2018/10/Agile-development.jpg  
[Accessed 25 January 2020].

Last FM , 2013. *Last FM API.* [Online]   
Available at: https://www.last.fm/api/  
[Accessed 8 May 2020].

LevelUpTuts, 2016. *React For Everyone #12 - Iterating Over Data With JSX.* [Online]   
Available at: https://www.youtube.com/watch?v=It9iL4EXFWc  
[Accessed 6 April 2020].

Lucidchart, 2018. *UML Use Case Diagram Tutorial.* [Online]   
[Accessed January 2020].

Manorisms, 2018. *Getting Started with Create React App.* [Online]   
Available at: https://www.youtube.com/watch?v=eCz3rhsDG5s  
[Accessed 20 April 2020].

Mosh, P. w., 2018. *React Tutorial - Learn React - React Crash Course [2019].* [Online]   
Available at: https://www.youtube.com/watch?v=Ke90Tje7VS0&t=891s  
[Accessed 20 January 2020].

Muslihat, D., 2018. *Agile Methodology: An Overview.* [Online]   
Available at: https://zenkit.com/en/blog/agile-methodology-an-overview/

Qloo, 2017. *music search.* [Online]   
Available at: https://tastedive.com/music

Sacolick, I., 2018. *What is agile methodology? Modern software development explained..* [Online]   
Available at: https://www.infoworld.com/article/3237508/what-is-agile-methodology-modern-software-development-explained.html  
[Accessed 24 January 2020].

Sebastian, 2018. *codingthesmartway.* [Online]   
Available at: https://codingthesmartway.com/the-mern-stack-tutorial-building-a-react-crud-application-from-start-to-finish-part-1/  
[Accessed 20 03 2020].

Sexton, K., 2017. *Waterfall VS Agile.* [Online]   
Available at: https://www.youtube.com/watch?v=egF9-FejbsA\

SmartDraw, 2018. *All About UML Activity Diagrams.* [Online]   
Available at: https://www.youtube.com/watch?v=zid-MVo7M-E   
[Accessed 26 January 2020].

Tuan, Q. N. T., 2017. *For Loops in React Render() — No You Didn’t!.* [Online]   
Available at: https://blog.cloudboost.io/for-loops-in-react-render-no-you-didnt-6c9f4aa73778  
[Accessed 6 April 2020].

tutorialspoint, 2006. *SDLC – Waterfall Model.* [Online]   
Available at: https://www.tutorialspoint.com/sdlc/sdlc\_waterfall\_model.htm   
[Accessed 26 January 2020].

twproject, 2020. *2018.* [Online].

user207421, 2013. *StackOverflow.* [Online]   
Available at: https://stackoverflow.com/questions/17816577/are-tcp-connections-on-localhost-secure  
[Accessed 19 May 2020].

X, p., 2019. *React Local Storage | Store React Form State in Local Storage.* [Online]   
Available at: https://www.youtube.com/watch?v=2kTCztIhuXg  
[Accessed 17 April 2020].