Project Document

Team members: Hebaq Hassan, Manal Adam, Shirley Man.

TopSpot.

How well do you know your music taste?

Table of contents

1 Introduction

TopSpot is a react web application that uses the Spotify API to get data from a user's Spotify account. The app will get the user's top 10 tracks and artists and use that data to create a quiz where the user guesses their top tracks and artists for the last 4 weeks, 6 months or of all time. The aim of this project is to give the user an overview of their listening activity and provide a fun and interactive quiz to test how well they know their own music taste.

Since all team members are Spotify users, we really wanted to make use of the Spotify API to explore our own listening activity and add an interactive element to our music listening experience. Currently, Spotify has a feature called 'Spotify Wrapped' that gives the user a yearly roundup of their listening activity at the end of the year. Our goal with TopSpot is to create an improved version of this feature that encourages user engagement and provides monthly roundups instead of yearly, giving the user an activity to look forward to at the end of each month. This report will discuss the development process we undertook in working on this project and creating our app.

Target audience: Spotify users.

1.1 Roadmap

1. **Sprint 1** - Ideation, requirements gathering & and system design

   The first step in the development process was deciding what kind of app to create.we brainstormed and after several discussions agreed on a direction for the app and all the functionalities it would have. We then worked on producing a list of requirements and created a system design for our app, illustrating the different components of our system and how they interact.

2. **Sprint 2** – Development

   We created a shared GitHub repository to facilitate successful collaboration on the code. We then divided the tasks between us, each team member focusing on a particular task.

3. **Sprint 3** – Testing and bug fixing

   In this sprint we focused on testing our app by carrying out unit tests and useability testing.

2 Background

2.1 Application details

As a hybrid of a Spotify data checking and quiz app, the key features will be the following:

- Users can login to the app with their Spotify username and password.
- Users can take a quiz to guess their top songs and artists for the month.
- Users can view their top artists for the past 4 weeks, 6 months or their all-time period
- Users can view their top songs for the past 4 weeks, 6 months or their all-time period.

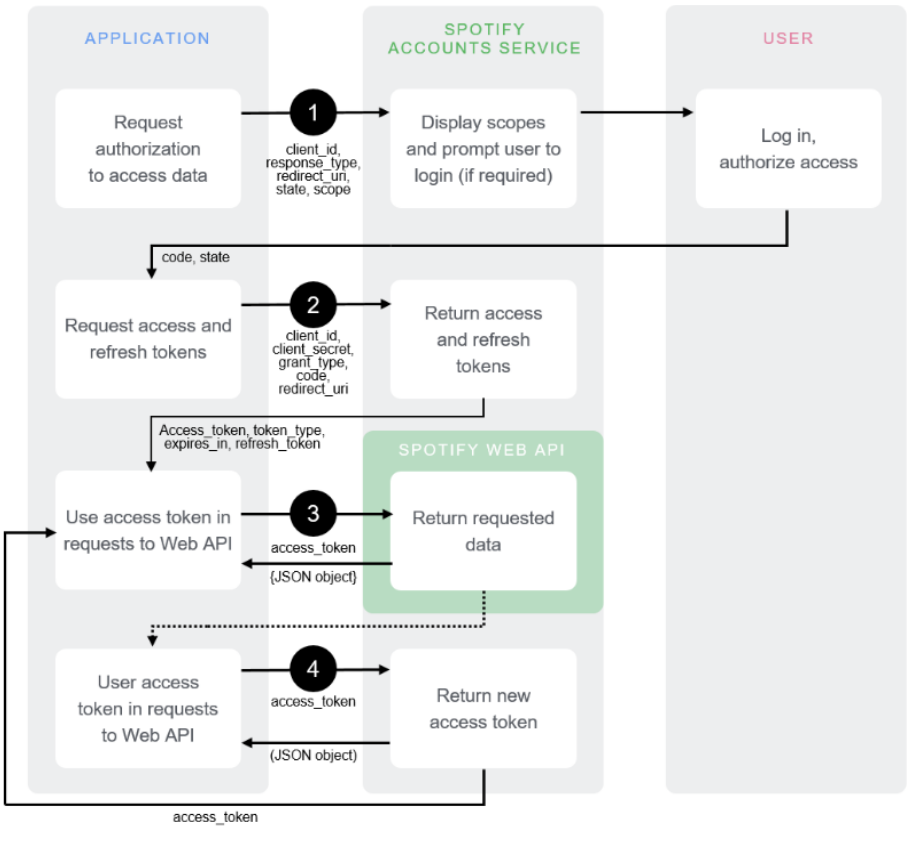
The app has 4 main components:

1. **Login/authorisation** - Once opened, the application displays a login button prompting the user to login via Spotify. Once the user is logged in and authorised they're redirected to the homepage and able to use the app.
2. **Top artists page** – Displays the users top 10 artists for the last 4 weeks, 6 months and all time. The page will display the rank, artist artwork and artist name. The user must click on each time frame to get the top 10 list for that specific period.
3. **Top tracks page** – Display the users top t10 racks for the last 4 weeks, 6 months and all time. The page will display the rank, track artwork and track name and Artist.The user must click on each time frame to get the top 10 list for that specific period.
4. **Quiz** – Upon clicking on the quiz page, the user will be presented with the first question of the quiz and the available options in a multiple choice style quiz. The user must select an answer, after which they're presented with the outcome of their response. The selected answer will have a 'green' background if correct and a 'red' one if incorrect. Text signifying whether the user's response was correct/incorrect, will also appear on the bottom of the page. Additionally, a 'next' button will also appear at the bottom of the screen so that the user can click it to move onto the next question. During the quiz, there's a counter that increments the user's score whenever they select the correct answer. After selecting an answer for the final question the user is presented with their score and prompted with a message to go to the top artists and top tracks pages to further explore that data.

3 Specification & Design

3.1 Technical requirements

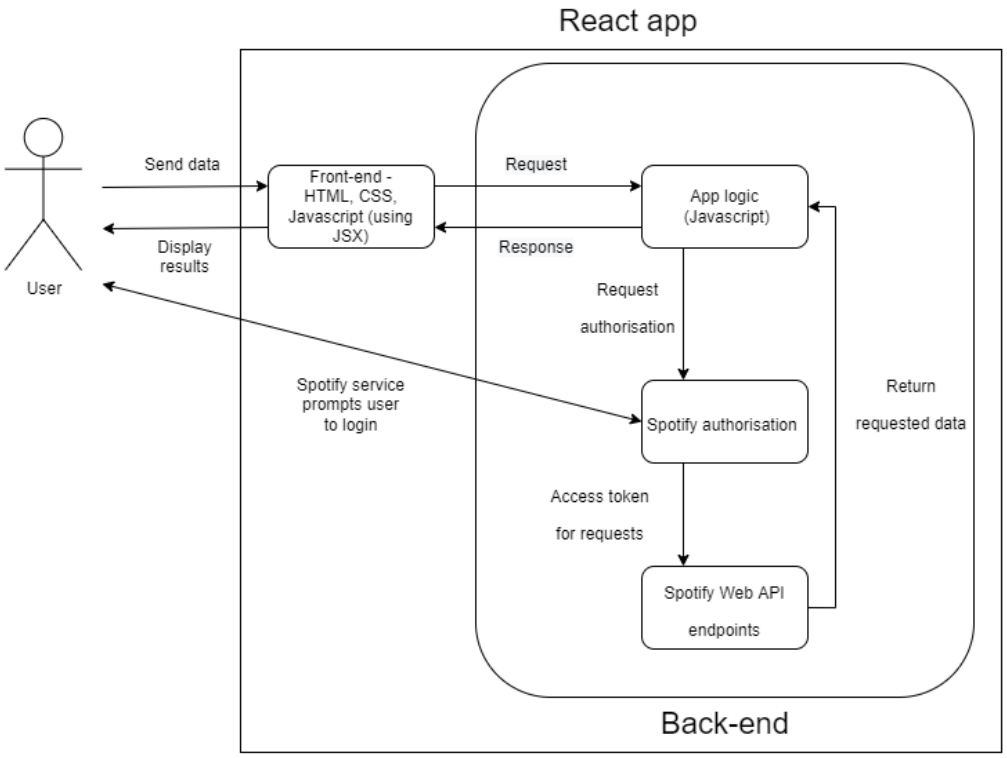
- Spotify login and authorisation code flow - to enable the use of GET requests for the user's data
- Access token from Spotify
- Authorisation code was created following Spotify's Authorisation Code Flow and Authorisation Guide.



3.2 Non-technical requirements

- The user must have a Spotify account in order to use the app.
- The user must have been using their Spotify account so that the data exists.

3.3 Design and Architecture:



The architecture above illustrates the design of the app. The user accesses the app via their browser and is immediately prompted to login using Spotify to facilitate authorisation. Following this, the user can request data through a button click, and the backend logic we implemented sends this request to the API's endpoints to retrieve the data and present it using the UI elements we built.

4 Implementation and Execution

4.1 Development Approach and Team Member Roles

As a team we split the workload by focusing on a particular feature in each sprint, this feature was then broken down into smaller sub-tasks that were divided amongst the team members, that way each member was able to gain experience in each aspect of the entire development process, ensuring that each member contributes to coding the frontend, backend, and testing.

The main deliverables in this project can be categorised as such:

• Requirements: the main requirements of the app were identified which included, but not limited to, items from the user requirements, components and file organisation to using third-party libraries and testing.

• Source code: this was accessible to all team members who each actively contributed to the source code of the application. Each member worked on their assigned pages of the app which included the code logic, styling and testing.

• Testing: each member was involved in the testing process of the app during the last sprint stage so that equal opportunity was given to all to gain some experience in testing. Each page was tested by one member of the team and in no particular order.

• Documentation and presentation: all team members contributed to the documentation via a shared google document so that all updates were live and each member has access to the most up-to-date version of the file at all times.

4.2 Tools and Libraries

Tools:

- **Spotify API & Spotify API documentation.**

  The Spotify API allowed us to fetch the data needed for the app to function and the documentation was useful in understanding how the API works and how to use it.

- **Figma for wireframing**

  Figma is a powerful design tool that helped us bring our design idea to life and create an interactive wireframe that we used to carry out user testing. The wireframes were also used as a guide throughout the development process to ensure the end product stayed true to our original design.

- **Google docs** for collaborating on writing the documentation and project plan.
- **GitHub** for version control and collaboration.
- **Visual Studio Code** as our IDE of choice
- **Zoom and Slack** for communication.

Libraries:

- **Axios** - to make http requests to Spotify web API endpoints.
- **React-router-dom** - for implementation of different pages and their routing.
- **Styled components** - enhanced CSS for component styling. Removes mapping between components and styles; instead, to define a style, a react component is created containing the styles. Useful (and easier) for dynamic styling the components we used e.g. buttons, navbar. This was combined with the normal CSS module for building our pages.
- **React-icons** - for implementing icons
- **Material UI** for building UI components
- **React testing library.**

4.3 Implementation process

It was essential that we started with implementing the login and authorisation processes so that not only could users gain access to the app but we as developers could request data from the API. Once those functionalities were implemented we could then proceed to fetching the required data and building the user interface to display it. Having gathered the necessary data, we then focused on building the quiz and its UI components. Due to time limitations, any bugs or issues encountered during the implementation

process were assessed in terms of severity and priority. Any bugs that severely affected the performance of the app were dealt with immediately. Other's were added to our backlog and resolved at a later time as we prioritised more important tasks.

4.4 Agile development

Time management and organisation were essential to the project to ensure we produced a minimum viable product given the time constraint and the challenges that came with working with new technologies. So having a robust project plan was imperative to ensure our goals were successfully managed and completed. The team used an Agile approach to the development process where we split the workload into 3, 1-week sprints. Each sprint focused on a particular phase of the development process from ideation and requirements gathering to testing and delivery. We maintained daily communication through slack and ensured we updated each other on our progress daily, as well as sharing issues we've come across and finding solutions together. After each sprint we held a review to reflect on how we performed in the last sprint, discussing what went well, what didn't and what improvements we could make for the next sprint. This ensured that we were learning from our past performance and enhancing our efficiency as a team.

User stories are used in agile methodology to describe the functionalities of the system from user's perspective, detailing exactly what the user wants to do and why. We used user stories to form our backlog of tasks to work through during scrum sprints. They were useful in helping us understand the system requirements from the user's point of view and ensured we kept the user in mind throughout the whole process.

The user stories in order of priority are as follows:

As a user I want to…

- Login to the app so that I can access my Spotify data.
- View my top artists so that I can know which artists I listened to the most.
- View my top tracks so that I can know which tracks I listened to the most.
- Take a quiz so that I can test myself on my listening activity.

4.5 Implementation challenges

- Authorisation flow was somewhat complicated and took a few attempts to succeed. We came across some errors such as CORS and authorisation errors which we eventually resolved. This was top priority and the first thing we implemented because our project was completely built around using the Spotify Web API.
- The quiz element proved to be more challenging to develop than planned. The first strategy required creating a quiz without using the Spotify API in order to write and lay out the basic backbone, which involved hardcoding the questions and answers and defining which of the answers was the correct value via a boolean. This worked as intended, however as the list of answers was to be unique to each user, we quickly went onto the next phase, which was to link this to the user's Spotify account through the Spotify API. The implementation of the Spotify API with the quiz answer options was more challenging than initially thought, even more so with the added time constraint. As a result, to ensure all other areas of the app were given some attention, the decision was made to proceed with the static method because the quiz was a critical element in our app and it was imperative that it be a part of the app and be displayed

during the demo. We plan to integrate the dynamic quiz (through the Spotify API) and increase the number of questions presented in the quiz as part of the next minimum viable product.

- Using new technologies that we had very little experience in was quite challenging and made for a very steep learning curve. As a result, we had to rely on external resources and tutorials for further learning.
- Overall, the time constraint was limiting in terms of how many features we could add to the app. Providing that we had more time, there are several potential features we could add to further engage the user with the data. These include:
    - Ranking game - the user is presented with a randomised list of their top 10 artists and tracks which they must rank in the correct order.
    - Guess the song - the user has to guess their top song from a 30 second preview.
    - Score history - The user's scores are saved and visualised in a diagram so they can review how they performed in the past.

5 Testing and Evaluation

5.1 Testing Strategy

We used the React testing library to carry out unit testing to test small, individual parts of the code. This ensured that the main parts of the app functioned as expected and produced the desired outcomes, as well as highlighting minor bugs.

5.2 Functional and User Testing

During the design phase we held several rounds of useability testing to ensure our app's design met our requirements. We met with a group of users from our target audience and provided an interactive prototype created using Figma to allow users to test the app and provide feedback. This was extremely beneficial as it allowed us to test our design with our target audience and highlight any potential issues with our design. Using the feedback from users we were able to make improvements on our design, making it more usable and accessible. This resulted in several iterations of our design, each iteration implementing new changes until our final design was more in line with user requirements.

Interactive-prototype:
https://www.figma.com/proto/2O1hyttvKguIDZ3b0XraSS/project-wireframe?node-id=70%3A51&scaling=scale-down&page-id=0%3A1&starting-point-node-id=70%3A51

5.3 System Limitations

- Although we would have liked to get more top items from the user's profile like top albums, genres and podcasts, the only data available were top tracks and artists.
- Manual refreshing of the page shows the login button again - Authorise.js needed to be run on all the pages so that when the access token expires after an hour, the user can login again and find their top artists/tracks. We used React-Router-DOM to navigate through the pages within the app (doesn't actually reload the browser). When you refresh the page, the real DOM updates and the login button appears again

- Deployment - our app can only be run locally. Future improvements include hosting on an external server such as Netlify or Heroku so the public can access our app without the need to install on their machine (pulling from Github, npm install, npm start).
- Spotify development restrictions - currently our app is in 'Development mode' according to Spotify's development guide meaning that only a maximum of 25 users can use our app and their emails must be explicitly added to the developer's list to authenticate with our app. With more time, a quota extension request could be made so that Spotify verifies our app and allows for public use with no restrictions to the number of users or needing their emails beforehand..

6 Conclusion

We set out to create a React web app application that uses the Spotify API to get the user's top artists and tracks and then allowed them to test themselves on their own music taste. We were able to achieve this using newly learned technologies such as HTML, CSS, Javascript and React and demonstrated a good understanding of them. Although we came across several challenges, they did not block us from moving forward and we were able to either resolve them or find an appropriate workaround. Using an Agile approach helped us work more efficiently and produce the app within the given timeframe.