CS322 Music Programming
Project Documentation

Introduction

The project began in one of the CS322 labs, where our team (Jevgenij, Fabian, Fiachra) sat down at the table and discussed all the potential ideas for a project that we are going to work on. We opened the project ideas list and went through each idea carefully, finding what interests us the most. A couple of project ideas caught our attention straight away, for example “Mobile Sound App”, “Spotify Visualizer”, “Spotify Audio Analysis” and many others. After browsing the internet and researching about our potential projects, we stumbled upon the official Spotify website made for Software Developers at developer.spotify.com. This page instantly caught our eyes, as the front page design was extremely appealing visually, with a message “Build with Spotify’s 100 million songs”. On the page there was a button “See it in action”, which would slide the page down and show you a lot of interesting back-end code for functionality such as getting you top 5 played tracks or get 5 recommended songs based on your top 5 tracks that you’ve listened to.

That was it, our team was immediately certain about our project direction. We aimed to develop a project utilizing the Spotify Web API, harnessing its capabilities to perform exciting tasks such as generating playlists directly in a user's personal Spotify account, as well as managing them within our application. Since every member of our team was a regular Spotify user, we found ourselves intrinsically motivated to kickstart the development process immediately. After studying the possibilities of what Spotify API can do for us, our team spirit went up and interest skyrocketed. During the next team meeting, our team discussed our strengths and weaknesses, who wanted to do back-end or front-end work, full stack and database work, or just overall design and wireframes. We then assigned roles and tasks for everyone to begin our development journey.

After a couple of days of meetings and setting up the project, our team came up with a brilliant idea to create a Google Chrome extension that would serve as the main hub of our application. This idea was aligning perfectly with what we wanted to create, and Chrome extensions are easy to install and convenient to use. So the way to access all the features of the application, was simply installing the extension and clicking on the icon when it’s added to the browser. It was possible to use the extension to open new browser tabs, therefore our team could work on different pages and then simply add them to the extension, for example, through buttons inside the extension UI.

It was time to allocate the workload equally between everybody in the team. Fiachra expressed interest in front-end work, so we assigned him to start off with some wireframe sketches and then use these sketches to create a React.js page that had a bunch of buttons and boxes in it. Fabian took the initiative and assigned himself Database work, where he would connect our application with “Firebase”, and store various data like user information or song and playlist names. Jevgenij thought that the idea of having an extension at all times in Chrome browser was fascinating, so he took the job of creating the extension, its UI, and back-end functionality.

Our team has set up a GitHub repository for the project, wrote a “README.txt” file which initially contained a bunch of ideas for the project that we previously brainstormed. After everyone accepted the invitation to the GitHub repository, every member of the team created a separate branch and called it something along the lines of “name/beginning”, so we could all work simultaneously on different parts of the project, and then merge them together over time. We could now track each

other’s progress, and see how we are getting on in the process. Not everyone in the team was familiar with GitHub, so we had a couple of sessions together where we go through the basics of Git, and perform actions like “git commit” or “git push”. These sessions were extremely valuable, as it is almost impossible to build a successful application without a proper version control system.

After some weeks have passed, our team started to feel like a group of developers, and we started to get new thoughts and ideas for our project, for example, what would be the target audience of our application. We had a couple of discussions and came to a conclusion that our application will be made for avid Spotify users who seek a more personalized and enhanced music experience. This includes music enthusiasts who regularly curate playlists and are always on the lookout for new recommendations and tracks that align with their tastes. Additionally, the project caters to tech-savvy individuals interested in the practical application of APIs and those who enjoy integrating their digital experiences. By offering features like creation and generation of playlists and providing refined music suggestions, our project was going to appeal to a broad spectrum of Spotify's user base, ranging from casual listeners to hardcore music fans.

At this point of development, we still did not have a clear objective of the project or the scope. What should our application do for our end users and why will it be useful. We started to discuss various features of Spotify, which features were good, which ones were bad, which ones were missing and so on. We realized that Spotify is missing a feature. Currently, Spotify has no way of crafting the playlists the way a user would like to. You are not able to move the songs or podcasts around the UI, when a Spotify playlist is created, all of the tracks in the playlist have static positions, and you can’t move them. This was a problem because sometimes a user wants to order 5 of their favourite songs in a specific order, and listen to them in a row. This was a great opportunity for our team to create a solution, and we came up with an idea to create a separate React.js page where a user uploads their playlist, and dynamically drags the tracks from the playlist into a new playlist, in the order that is perfect for them. The user would then save this new playlist that they created and save it inside their Spotify account. This was our main objective, but of course, an equally important objective was of course to learn as much as possible and improve our skills in software development, agile principles, communication and team work.

Methodology

Before our team wrote even a single line of code, we all agreed that we should work with React.js as it is one of the most popular libraries in the world and a lot of employers looking for graduates with React skills. React was also an ideal candidate to create a web page where you can easily drag and drop content from one container to another. Everybody in our team wanted to improve our coding ability with React.js, however, in the end not everybody got to work with it. For the Google Chrome extension, React.js was not ideal because of how the Chrome extension communicates with the back-end code and Spotify API, so Jevgenij stuck to vanilla JavaScript, HTML and CSS. Since Fabian started to work on Firebase database, he also didn’t get the chance to work with React. Fiachra was the one who got the opportunity to create a dynamic React page with an appealing design and fluid functionality.

Our team used Balsamiq in the beginning of the project to create some low fidelity wireframes, sketches and just to design the overall visuals for the extension, react page and other potential UI elements of the project. Our team was already familiar with Balsamiq as we all successfully completed the CS280 User Experience & User Interface module, so we were confident in our ability to create high quality sketches and designs.

In order to implement Firebase into React.js we began by setting clear objectives for integrating Firebase, defining the specific features and functionalities we aimed to leverage from Firebase services. Next, we established a systematic approach to seamlessly integrate Firebase into our React.js application. We utilised the Firebase JavaScript SDK, incorporating components like Firebase Authentication, Firestore for real-time database functionality, and Firebase Hosting for deploying and hosting our application. The implementation involved creating appropriate React components to interact with Firebase services, ensuring a smooth flow of data between the frontend and the Firebase backend. Continuous communication within the group facilitated the resolution of challenges and ensured that each team member had a comprehensive understanding of the integrated Firebase features. Regular testing and debugging sessions were conducted to refine the implementation and address any issues that arose. Through this collaborative and methodical methodology, we successfully integrated Firebase into our React.js application, enhancing its functionality and data management capabilities.

A significant portion of our project's development was dedicated to the integration of the Spotify Web API, a complex yet fascinating process. This integration was pivotal for accessing Spotify's vast music database and leveraging its functionalities like fetching user-specific top tracks, generating personalized recommendations, and creating new playlists directly in user accounts. One of the initial challenges we encountered was handling user authentication seamlessly. We implemented OAuth 2.0 protocols to ensure secure and efficient user login, allowing access to their Spotify data while maintaining privacy and security. Another critical aspect was parsing and handling the JSON responses from Spotify, ensuring that the data was accurately captured and appropriately displayed within our extension.

One of the key successes in this integration was the ability to create playlists. We developed functionality that not only created these playlists but also populated them with tracks based on user preferences, a feature that truly personalized the user experience. This feature worked even better than expected, because the suggested songs were of top quality, as they are based on top 5 songs that you've listened to the most. It is almost guaranteed that the songs generated will be in line with the users' music taste. Throughout this process, we relied heavily on Chrome's developer console (DevTools) and print statements for real-time testing and debugging, which proved invaluable in understanding and refining our API interactions and the generated content.

Developing the user interface for our Spotify Playlist Enhancer was a major part of our project, where we put a lot of effort into making it user-friendly and pleasant to look at. We spent a lot of time fine-tuning the ".css" files, which helped us get the look and feel of the extension just right. We chose text sizes and fonts that were easy to read and looked great, making sure the interface was clear and inviting. Furthermore, we also paid close attention to the buttons. Their size, position, and colour were carefully chosen to make them attractive and fit the Spotify aesthetics. We made sure the buttons responded quickly when clicked, to give users immediate feedback. The background was another area we focused on. We tried different images and positions to make sure it looked good without distracting from the main content. The night time sky full of stars just fits so well into the Chrome extension behind the buttons. We also fixed the background so it wouldn't shift or stretch when new content was added, keeping the extension looking neat and professional. Additionally, we made sure that our interface worked well on different screen sizes and resolutions. Whether on a big monitor or a small laptop screen, everything needed to be visible and work properly. By focusing on these details and constantly improving our design, we made an interface that was not only good-looking, but also easy and enjoyable to use.

In the development of our Spotify Playlist Enhancer, thorough testing and debugging played a crucial role. To manage and track our testing process, we created an Excel sheet filled with 29 test cases. This sheet was essential for keeping track of what needed to be tested, the results of each test, and any additional notes or observations. It helped us stay organized and ensured that no aspect of the application was overlooked.

For more technical testing, we used a unit test script written in JavaScript. This kind of test was particularly useful for checking specific functions like getTopTracks(). By using these scripts, we could quickly verify if our functions were returning the correct data and behaving as expected. However, later our team learned that unit tests were not necessary because the size of the project and the functions was just not big enough. Instead, one of our most valuable tools for testing and debugging was the Chrome developer console. This tool allowed us to test our application in real time. We could see what was happening as we interacted with the extension, which made it easier to spot and fix issues. The console was especially helpful for checking how our API calls were working and for catching any errors that occurred. We believe our approach to testing and debugging was comprehensive. By combining organized tracking with Excel, and real-time observation with Chrome's developer tools, we were able to thoroughly test our extension and ensure its functionality and reliability.

Our team's collaboration on the Spotify Playlist Enhancer project was guided by Agile principles, which really helped us work well together. We had regular scrum meetings inside the 4th year project room where we all caught up on what we were doing, discussed any problems, showed new features or functionality, and planned what to do next. These meetings were super important for keeping everyone informed and making sure we were all heading in the right direction. Additionally, we also reached out to other students around us to get their help with testing the extension. We asked them to use it and see if they could find any bugs or make it crash. This turned out to be a great idea because their feedback helped us find and fix issues we hadn’t noticed before and greatly improve our application.

Conclusion

The Spotify Playlist Enhancer project was an ambitious endeavour to create a Chrome extension that seamlessly integrates with the Spotify Web API. The primary goal was to enrich the Spotify experience by enabling users to generate personalized playlists, discover music recommendations, and play these playlists directly through the extension. Throughout the project, we successfully developed key functionalities such as user authentication, efficient handling of API requests, and a dynamic user interface that was both intuitive and visually appealing. The essence of our project was to bridge the gap between Spotify's extensive music library and the individual user, tailoring the listening experience to each person's unique preferences. We also wanted to enable the user to craft their own playlist with drag and drop features on a dynamic React based page, but we were not able to finish this task in time, and instead, it was added to our future development plans document.

Looking ahead, our team has exciting plans to expand the capabilities of the Spotify Playlist Enhancer. One of our immediate goals is to complete the development of our React.js page, which is currently in progress. This page will introduce a new level of interactivity by allowing users to import their Spotify playlists and edit them extensively. Users will be able to rearrange songs, rename tracks, and make other modifications, providing a more tailored and personal playlist experience. Eventually, the end users will be able to craft their own perfect playlist and add it to their Spotify account effortlessly. Additionally, we're planning to integrate Firebase database, which we've already set up

but not used. While Firebase wasn't crucial for our initial project scope, it will become essential as our application grows in size, so we will need to store more user data and enhance the functionality of our application. This integration will enable us to manage user data more efficiently, paving the way for more advanced features and a richer user experience. These future developments are not only geared towards enhancing the functionality of our extension but also aim to make it more versatile and user-centric.

As we wrap up this project, it's clear that the Spotify Playlist Enhancer has been more than just a task; it's been a journey of growth for our team. This project has deepened our understanding of web development and API integration, showing a seemingly endless number of Spotify's developer tools, revealing the complexities and intricacies involved. Working together, we've learned the importance of clear communication, collaboration, and flexibility, which are key in any team environment, not necessarily software development.

The educational journey through various university modules such as "CS280 Introduction to UI/UX and Interaction Design", "Web Information Processing", "Computer Networks" and others played an indispensable role in the success of this project. The knowledge gained from these courses provided a strong foundation, enabling us to effectively design user-friendly interfaces, understand the intricacies of network communications, and process web data efficiently. These modules equipped us with the necessary skills and theoretical background to navigate through complex development tasks and make informed decisions.

From a learning perspective, this project has been invaluable. It pushed us to apply theoretical knowledge in a practical setting, enhancing our skills in programming, problem-solving, and creative thinking. The experience of turning a concept into a working, functional product has been incredibly satisfying. It's one thing to learn about coding and APIs in a classroom, but it's another to see your code come to life and actually work as intended and even exceed everyone's expectations.

This project, in essence, has been a significant milestone in our educational journey. It not only reinforced our technical abilities but also gave us a taste of real-world application development. The satisfaction of seeing our ideas materialize into a product that can enhance the Spotify experience is immeasurable and motivates us to continue exploring and innovating in the field of web development.