INDY-7 - Tremeloo: Spotify Music Suggestions

CS 4850 Section 01 - Spring Semester 23

Team Members:

Vincent Green, Leiko Niwano,

Mark Walker, Khemrind Ung

Advisor: Sharon Perry

Website: https://tremeloo.com/

Table of Contents

Project Overview

Our project, Tremeloo, is a mobile application implemented in JavaScript and TypeScript utilizing React Native components and Spotify API endpoints. The app receives data from a user's Spotify account and displays their playlists. The app is then able to parse Spotify's database of music and suggest music tracks based off the content in each individual playlist.

Project Background

Spotify is a great utility as a music streaming app that enables many artists to display and monetize their created music. However, the way that the app works to suggest these songs to their userbase is somewhat wanting. This project works to resolve this issue by applying a different algorithm for recommendations based on the set of songs in a given user's playlist.

Project Planning and Management

General group interaction and file sharing were performed through Microsoft Teams, as well as deadline notification and schedule management. Team meetings were held on a weekly basis and recorded and transcribed via Teams functionality.

Version Control

For version control, we used GitHub. GitHub allowed us to track changes, collaborate with team members, and manage different versions of our code. GitHub's branching and merging features made it easy to work on specific features or fixes, while its collaboration tools helped streamline our development workflow and ensure high-quality code.

Project Requirements

1. Gain access to user's Spotify Account.
2. Load and display a list of the user's playlists.
   a. Send GET request for list of the user's playlists.
   b. Display list of playlists.
      i. Display playlist cover image.
      ii. Display playlist name.
3. Access each of the user's playlist
   a. Send GET playlist request.
4. Output a list of song recommendations for each unique playlist.
   a. Pass MD into recommendation algorithm.
   b. Display returned list of song names.
5. Add songs from the recommendation list to the user's playlist.
   a. Send POST request for chosen song by user using MD.
6. Recommendation Algorithm:
   a. Access MD from each song within a given playlist.
   b. Access Spotify song DB
   c. Compare MD of user's playlist songs to MD of Spotify database
   d. Generate list of songs matching MD criteria
7. Certain actions are restricted for Spotify free users. In these cases, Spotify *recommends* displaying the message depending on usage of either Android or iOS as follows:
   a. Android premium message: "Spotify Premium lets you play any track, ad-free and with better audio quality. Go to spotify.com/premium to try it for free."
   b. iOS premium message. "Spotify Premium lets you play any track, ad-free and with better audio quality." Link to quote.

Architecture

Design

Tools

Adobe XD and Adobe Illustrator were the main developer tools utilized in the production of the application. Adobe XD was used to produce mockups of each app screen. As well as demonstrating the overall flow of the app. Adobe Illustrator was utilized in the development of the various logos and images of the app. The app logo was created as scalable vector graphic to ensure image sharpness at any zoom magnification.

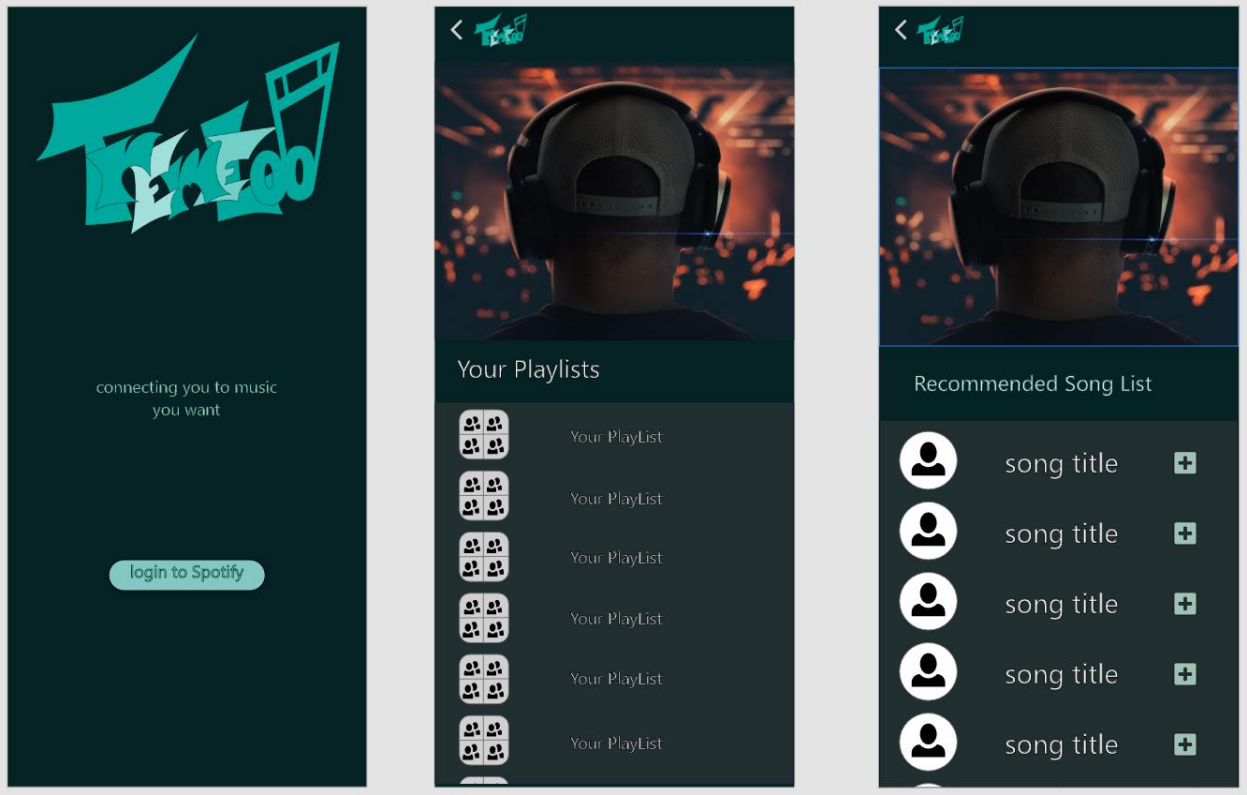

Figure 1. Early UI Mockup

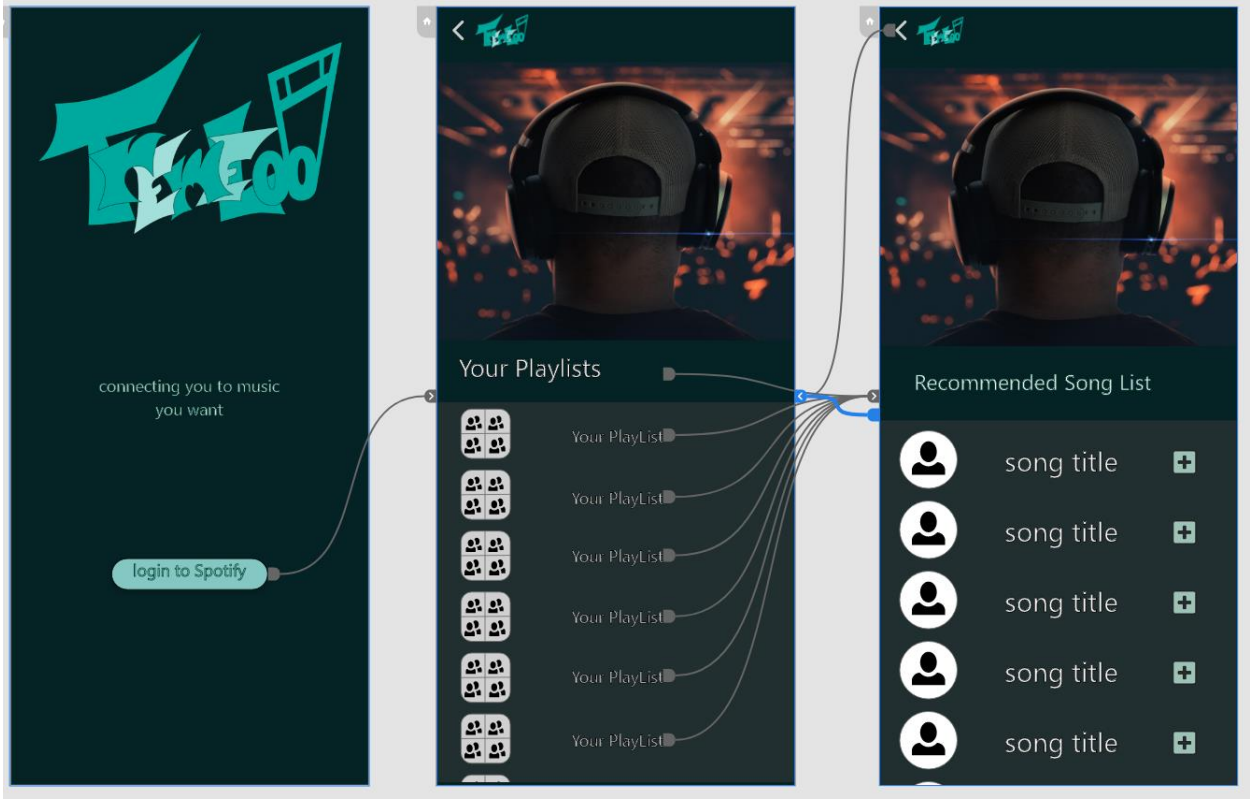

Figure 2: UI Design Flow

Analysis

The design mockup details the overall structure of the applications 3 main pages: home, playlists, and song recommendations. The home page consists of the app logo, a brief text slogan and a button for logging into the Spotify authentication process. The playlist page contains the navigation menu on top, a banner image, A playlist header for the list of playlists, and a scroll view element containing the user's playlists. The song recommendation page is designed in the same manner as the playlist page, with alteration to the scroll view header string and a displayed list of recommended songs.

Figure 2 is a UI mockup of the design flow for the app. The arrows are indicators for screen transitions. Clicking the login to Spotify button transitions the user from the home page screen to the playlist screen. On the playlist screen, clicking a playlist name transitions the user to the song recommendation screen. The user is so capable of navigating back to the previous screen by clicking either the back button in the upper left corner or swiping right on the screen.

Implementation

Tools

The implementation of the project is conducted using React-Native integrating with the Spotify API. React-Native uses the Metro bundler to compile various JavaScript files into a single consolidated file.

React-Navigation, an implementation of native navigation for mobile platforms that comes out of the box for React Native, is used for screen-to-screen navigation.

Spotify API is at the core of functionality, which is accessed through a wrapper that provides types and well-defined JavaScript functions for each relevant endpoint. For more information, see *JMPerez/spotify-web-api-js* on GitHub.

Node.js is used as a backend server to power our React application.

Analysis

The UI is implemented in JavaScript with React, React-Native, and React-Navigation libraries. The main app.js file imports the react-navigation library which is utilized to build a stack navigator for each of the various screens. Each screen is a separate JS file located in the pages folder within the component directory. Each page contains its own CSS styles sheet. All logos and images are located in the asset's directory. Additionally, the frontend employs the use of React Contexts, which allows any page in the code to access and update their UI upon changes in the underlying backend data. It serves as our connection to the backend and is responsible for UI updates after any backend function is used.

The backend is implemented in TypeScript with Expo, React, and Spotify API libraries. The backend file returns an object consisting of 3 main functions (Authenticate, RefreshRecommendations, and AddTracks), a boolean variable (IsAuthenticated) and an array (Playlists). The Authenticate function grants access to the user's Spotify account and grants the app permission to access and edit the user's playlists. The RefreshRecommendations functions performs an API call to Spotify to retrieve a newly updated list of songs displayed for recommendation to the user. The function selects songs within the user's currently selected playlist as parameters for Spotify's recommendation endpoint, adds a popularity constraint to ensure the track's quality, and records recommendation history so that each recommendation is only suggested once. The AddTracks function makes an API call to add the selected recommended song to user's currently selected playlist. The IsAuthenticated variable is a simple Boolean indicating whether a valid access token has been retrieved from the Spotify API. The Playlist array is a list of each individual playlist located within the user's Spotify library.

Testing

Tools

The project was executed using Expo CLI and Expo Go – open-source command line interfacing and framework that enable the development and testing of React Native applications in both iOS and Android environments.

Test Case ID	Test Objective	Precondition(s)	Steps	Expected Result	Actual Result	Status (Pass/Fail)	Comments
TC_PL_01	Successful user login to access app	1. User has a valid Spotify account	1. User hits login button 2. User enters login information 3. User is granted access to app	User is able to login and access app	User can login and access app	Pass	
TC_PL_02	User is able to view and select a playlist	1. User is able to login 2. User has created playlists on Spotify	1. Playlist data is present after login 2. User is able to select a playlist	User is able to view their created playlists and select one of them	Playlists are visible and buttons are adjacent that allow progression	Pass	Possible additional functionality in allowing creation of playlist when none have been created prior.
TC_PL_03	User is able to view recommended songs off of selected playlist	1. User has selected playlist to be analyzed	1. Song Metadata is accquired from playlist 2. Algorithm is run to determine best song matches to playlist 3. Song matches are displayed to user	Algorithm works and songs are displayed	Algorithm works and song data is displayed	Pass	Consider tweaking algorithm to prioritize different criteria
TC_PL_04	User is able to add reccommended songs to selected playlist	1. All of the above test cases have been met 2. User is able to view song reccomendations	1. User can view song matches 2. Song matches can be appended to playlist through use of an "Add to "Playlist" button	Reccomendations are viewed and can be added to playlist selected	Reccomendations are viewed and can be added selected playlist	Pass	Added songs are removed from the recommendation list

Figure 3: Software Requirements Test

Analysis

As depicted in Figure 3, the software test met our needs and the app performed as expected. As for what happens through a user's perspective, when the user opens the app, they are directed to the homepage component. Upon clicking the login to Spotify button, a call to the authenticate function is made and the user is directed to a web browser to authenticate the app. Upon authentication, the user is directed to the playlist screen component. A scroll view box containing a list of each of the user's playlists is displayed from the playlist array. Clicking a playlist with the in the scroll view navigates the user to the song recommendation screen component. The Song Recs contains a scroll view box displaying a unique list of recommended songs generated for each playlist. Clicking the add to playlist button on the right side of the screen calls the AddTrack function, adding the song the current playlist. A new song is then added to the recommended song list.

Challenges

The initial steps architecture design for Tremeloo focused on the development tools needed for completing the requirements. The use of React-Native for ease of development in a single environment for both iOS and Android created a learning curve for the team. Conducting research into the use of React-Native was a necessary step. Knowledge of the Expo CLI was also a necessary component moving into the implementation phase. Initial plans were to use NativeBase libraries for implementation of the front-end; however, issues were had integrating NativeBase with React-Native. The team decided to move forward with implementation of the UI without NativeBase.

One challenge we had was to integrate Expo into the backend workflow. Since Expo was already installed in the front end, we needed to find a way to make it work seamlessly with the backend. The backend underwent major refactoring in order to integrate into React's declarative execution paradigm. The use of globally observed data in the form of React Contexts was critical in connecting backend changes to frontend updates.

Displaying an image of each song for the recommended song list was also an issue. Displaying an associated album image of each recommended song would require an update of the backing function for retrieving the recommended playlist. It was decided to leave out the recommended song image for the preliminary release. An update for the later version could be explored.

The lack of a physical android device for testing created another challenge. As result, it was necessary to utilize an emulator during the testing phase.

Conclusion

In conclusion, our React-Native Spotify suggestion app has successfully achieved its goal of providing personalized music recommendations to users based on their listening history and preferences. Our senior project has been an enriching experience in both team collaboration and app development. Working in a team environment allowed us to leverage each other's strengths and overcome obstacles together, resulting in a more comprehensive and polished final product.

Using a Gantt chart to pace ourselves was crucial in ensuring that we met our project goals and deadlines. This project management tool helped us prioritize tasks, allocate resources, and stay on track with our project timeline.

In terms of app development, using React-Native, Expo CLI, and Expo Go for testing provided us with a robust and efficient framework for building and testing our app. React-Native allowed us to create a cross-platform app with a consistent user experience, while Expo CLI and Expo Go allowed us to easily test our app on both Android and iOS.

Overall, this senior project has been a valuable learning experience that has equipped us with the skills and knowledge to tackle future app development projects in a team environment.

Appendix

A – Gantt Chart

Project Name:	Spotify Music Suggestions																		
Report Date:	4/6/2023																		
			Milestone #1									*Milestone #2*							C-Day
Deliverable	**Tasks**	**Assigned To**	01/12	01/19	01/26	02/02	02/09	02/16	02/23	03/02	03/09	03/16	03/23	03/30	04/06	04/13	04/20	04/27	
Requirements	Meet with group	All	8	8															
	Assign roles	All				2													
	Define requirements	Mark, Vincent			2	2	2												
	Review requirements with group	All				3	1												
Research	React-Native documentation	All			4	4													
	Spotify API documentation	All			4	4													
Project Design	Define tech required*	All				2	2												
	Discussion of the flow of the app (ERD concept design)	Khemrind, Vincent				2	4												
	Create UI concept of the flow of the app	Leiko, Vincent				2	4												
	Access Spotify Database	Khemrind, Leiko					4	2											
	Integrate with user / user data	Khemrind, Leiko						6	8										
	Develop music recommendation algorithm	All						5	10	10									
	Update playlist following recommendations	Khemrind							10	10									
	Develop working prototype	Khemrind, Leiko							15	15	5								
	Test prototype	All								4	4	2							
Development	Review prototype design	All									2	2	2						
	Rework requirements	All											6	6	6	6			
	Document changes	Mark														3	4		
	QA Testing	All													5	5	5		
Final report	Presentation preparation	Mark, Vincent														4	4	3	
	Website preparation	Mark, Vincent												4			2		
	final report draft	Mark, Vincent													8				
	Final report submission to D2L	Mark, Vincent															1	1	
		Total work hours	249	8	8	10	21	17	13	43	39	11	4	8	10	19	18	16	4

Legend	
Planned	
Delayed	
Number	Work: man hours

B – Unimplemented Phase 2 Requirements

- Display a list of songs currently within the user's playlist (PHASE 2).
    - Important steps for legal market use:
        - All MD displayed from Spotify MUST link back to the Spotify app.
        - If the user does not have the Spotify app installed, links must divert them to the app store to download the app.
    - Display song name from GET playlist request.
    - Display artist image from GET playlist request.
    - All songs must link to Spotify app. (PHASE 2)
        - Display song name and album artwork of selected song at the top of the page.
        - Newly displayed song links to the Spotify app.
    - Play preview of selected song. (PHASE 2).
    - Get preview URL from returned JSON object.