"Ss. Cyril and Methodius" University in Skopje
FACULTY OF COMPUTER
SCIENCE AND ENGINEERING

TEAM PROJECT FOR COURSE
ADVANCED WEB DESIGN 2023/24

TITLE:
Spotify-Clone

Students:

Pance Prendjov

Jane Panov

Mentors:

Boban Joksimoski

Mila Dodevska

Spotify Clone Documentation

1. Project Overview

The Spotify Clone project is designed to replicate key functionalities of Spotify, providing users with an intuitive and personalized music streaming experience. The application integrates with the Spotify API to offer seamless authentication, playlist management, and track playback. By focusing on core features such as user login, playlist viewing, and track control, this project serves as both a practical tool for music enthusiasts and an educational resource for developers interested in API integration and web application development.

1.1. Introduction

Music streaming services like Spotify have become integral to modern entertainment, offering vast libraries of music and personalized recommendations. The Spotify Clone project aims to deliver a similar experience by recreating Spotify's core features in a simplified format. This application allows users to authenticate with Spotify, view their playlists, and enjoy track playback, all while reflecting changes in real-time on the original Spotify platform.

1.2. Goals and Objectives

The primary goals of the Spotify Clone project include:

- **Seamless Authentication**: Implementing secure user login using Spotify's OAuth 2.0.
- **Comprehensive Playlist Management**: Displaying all user playlists with detailed track information.
- **Real-time Track Playback**: Providing a responsive playback experience with real-time updates.
- **User-Friendly Interface**: Ensuring an intuitive and accessible user interface.
- **Educational Value**: Offering a learning platform for developers on API integration and frontend development.

1.3. Scope

The scope of this project includes:

- **User Authentication**: Secure login and token management.
- **Playlist Display**: Showing user playlists and tracks with detailed information.
- **Track Playback**: Enabling playback controls and real-time updates.
- **Responsive Design**: Adapting the application for various devices and screen sizes.

2. Key Features

2.1. User Authentication

Authentication is a fundamental feature that allows users to securely access their Spotify accounts through the application. This feature is implemented using Spotify's OAuth 2.0, which involves:

- **Authorization Flow**: Users are redirected to Spotify's login page to authenticate.
- **Token Exchange**: Upon successful login, authorization codes are exchanged for access tokens.
- **Token Management**: Access tokens are stored securely and used to interact with Spotify's API.

2.1.1. Security Considerations

To ensure security:

- **Secure Communication**: All interactions with Spotify's API are conducted over HTTPS.
- **Token Expiration**: Tokens are refreshed periodically to maintain secure sessions.
- **Error Handling**: Comprehensive error handling is implemented to manage authentication failures.

2.2. Playlist Display

Once authenticated, users can view their Spotify playlists in a user-friendly interface. This feature includes:

- **Sidebar Navigation**: A sidebar on the left side of the application displays all public playlists associated with the user's account.
- **Playlist Details**: Each playlist includes detailed information such as track names, album titles, and durations.

2.2.1. Implementation Details

- **API Integration**: The Spotify Web API is used to fetch user playlists and track details.
- **User Interface**: Playlists are displayed in a clean and organized manner, with options to view individual tracks and their details.

2.3. Track Playback

The track playback feature provides users with a seamless listening experience, reflecting changes in real-time on Spotify. Key components include:

- **Playback Controls**: Users can control playback with options to play, pause, skip, and adjust volume.
- **Track Information**: Details of the currently playing track, including the album cover, song title, and artist name, are displayed in the footer.

2.3.1. Functional Components

- **Playback Control**: The playback control interface includes buttons for previous/next track, play/pause, and volume adjustment.
- **Real-Time Updates**: Changes in playback status are reflected in real-time on the Spotify platform, ensuring synchronization between the application and Spotify.

2.4. User Interface

The user interface is designed to provide an intuitive and engaging experience:

- **Landing Page**: The main body of the landing page shows a default album, displaying all tracks with titles, album names, and durations.
- **Sidebar Navigation**: The sidebar provides quick access to all user playlists, allowing users to switch between different playlists and tracks.
- **Footer Display**: The footer shows the details of the currently playing song, including album art, track name, and artist name.

2.4.1. Design Considerations

- **Responsiveness**: The application layout adapts to various screen sizes using CSS Flexbox and Grid.
- **Accessibility**: Design practices ensure that the application is accessible to users with disabilities, including keyboard navigation and screen reader support.

3. Technologies Used

3.1. React

React is used for building the user interface, leveraging its component-based architecture to create reusable and efficient UI components.

3.1.1. Benefits

- **Component Reusability**: Enhances development efficiency by allowing reuse of UI components.
- **State Management**: Simplifies state management with hooks and context.

3.2. Spotify API

The Spotify Web API is crucial for accessing Spotify's music catalog and user data. It enables features such as:

- **User Authentication**: Handling OAuth 2.0 for secure login.
- **Playlist Management**: Fetching user playlists and track details.
- **Playback Control**: Integrating with Spotify's playback system for real-time updates.

3.2.1. Integration Challenges

- **API Rate Limits**: Managing rate limits to avoid disruptions in functionality.
- **Token Management**: Handling token expiration and renewal efficiently.

3.3. Styled Components

Styled Components is used for styling React components, offering:

- **Scoped Styles**: Encapsulating styles within components to avoid conflicts.
- **Dynamic Styling**: Applying styles based on component props.

3.3.1. Advantages

- **Modular Design**: Promotes modularity and maintainability of styles.
- **Component-Level Styling**: Simplifies style management by keeping styles close to the components they affect.

3.4. Axios

Axios is utilized for making HTTP requests to the Spotify API, providing:

- **Promise-Based API**: Simplifies handling asynchronous operations.
- **Error Handling**: Facilitates robust error handling for HTTP requests.

3.4.1. Benefits

- **Request Interceptors**: Allows modification of requests and handling of errors.
- **Response Transformation**: Transforms data before it is processed by the application.

3.5. Yarn

Yarn is employed for managing project dependencies, offering:

- **Faster Installations**: Uses caching to speed up dependency installation.
- **Deterministic Resolution**: Ensures consistent dependency versions.

4. Installation and Setup

4.1. Prerequisites

Ensure the following prerequisites are installed:

- **Node.js**: Version 14 or higher.
- **Yarn**: Version 1.22 or higher.
- **Spotify Developer Account**: Required for accessing the Spotify API.

4.2. Clone the Repository

Clone the project repository using Git:

git clone https://github.com/janepanov/funki-spotify-clone.git

4.3. Install Dependencies

Navigate to the project directory and install dependencies:

cd spotify-clone

yarn install

4.4. Configuration

```
export default function Login() {
  const handleClick = () => {
    const clientId = "YOUR_CLIENT_ID";
    const redirectUrl = "http://localhost:3000/";
    const apiUrl = "https://accounts.spotify.com/authorize";
    const scope = [
      "user-read-email",
      "user-read-private",
      "user-read-playback-state",
      "user-modify-playback-state",
      "user-read-currently-playing",
      "user-read-playback-position",
      "user-top-read",
      "user-read-recently-played",
    ];

    window.location.href = `${apiUrl}?client_id=${clientId}&redirect_uri=${redirectUrl}&scope=${scope.join(
      " "
    )}&response_type=token&show_dialog=true`;
  };
```

4.5. Start the Application

Run the application locally using Yarn:

yarn start

Navigate to http://localhost:3000 in your browser to access the application.

5. Usage

5.1. Authentication

To authenticate with Spotify:

1. Click on the “Login with Spotify” button.
2. Redirected to Spotify’s login page, enter your credentials.
3. Upon successful login, you will be redirected back to the application with access to your Spotify data.

5.2. Viewing Playlists

To view your playlists:

1. Navigate to the sidebar on the left side of the application.
2. Select a playlist to view its tracks.
3. Click on a track to start playback and view track details.

5.3. Playing Tracks

To control playback:

- **Play/Pause**: Use the play/pause button to start or stop the track.
- **Skip**: Use the skip buttons to move to the next or previous track.
- **Volume**: Adjust the volume using the volume slider.
- **Track Details**: The footer displays the album art, song title, and artist of the currently playing track.

5.4. Switching Playlists

To switch between playlists:

1. Click on a different playlist from the sidebar.
2. The application updates to show the tracks from the selected playlist.
3. Playback controls remain functional, allowing you to play, pause, skip, and adjust volume for the new selection.

6. Future Enhancements

6.1. Playlist Creation

Future updates will include features for creating and managing custom playlists. This will involve:

- **User Interface**: Designing an interface for creating and editing playlists.
- **API Integration**: Implementing API calls for creating and updating playlists on Spotify.

6.2. Advanced Search

An advanced search feature will allow users to search for tracks, albums, and artists within the application:

- **Search Bar**: Adding a search bar to the interface.
- **API Integration**: Implementing search functionality using Spotify’s search API.

6.3. Playlist Sharing

Users will be able to share their playlists with others:

- **Sharing Options**: Providing options to share playlists via social media or direct links.

- **API Integration**: Utilizing Spotify's sharing features to enable playlist sharing.

7. Conclusion

The Spotify Clone project demonstrates the potential of integrating external APIs into web applications to recreate popular services like Spotify. By focusing on core functionalities such as authentication, playlist management, and track playback, this project provides a valuable learning experience while offering a functional tool for music enthusiasts. Future enhancements will further enrich the user experience, highlighting the dynamic capabilities of modern web development.