SPOTIFY CLONE PROJECT

TEAM MEMBERS:

ROLL NUMBER	NAME
YOGANATH M	22ALR116
ROSHINI G	22ALR082
PRADEEPA S	22ALR065

INTRODUCTION:

A whole journey through full-stack web development using React, Node.js, and MongoDB is involved in creating a Spotify clone. The component-based architecture of React enables a dynamic and user-friendly UI, while the event-driven paradigm of Node.js empowers the backend by effectively managing API interactions and server-side logic. MongoDB's data storage scalability and flexibility enable it to meet the varied requirements of a music streaming service. In addition to being replicated, this project offers developers an in-depth education on intricate subjects like deployment tactics, data modeling, and application design. Every line of code advances one's knowledge and understanding of web development principles. Furthermore, the Spotify clone project highlights JavaScript's revolutionary power and demonstrates how well suited it is for creating complex, real-time applications. Developers work together to give their creation life, transforming it into a platform for imagination, ingenuity, and engaging user experiences.

Key features:

1. React is used for user authentication and registration.
2. Playlist and Music Library Administration
3. Online Music
4. Learn and Investigate
5. Data Retrieval and Backend Interface
6. User Profile Administration

REQUIREMENTS:

- **Front-End Programming**

1. **CSS (Cascading Style Sheets):** CSS is used to define the visual appearance and layout of web pages. It allows developers to apply styles, such as colors, fonts, spacing, and positioning, to HTML elements, creating a visually appealing and consistent user interface.

2. **JavaScript:** JavaScript is a powerful programming language that adds interactivity and dynamic behavior to web pages. It enables developers to manipulate and modify HTML and CSS, handle user interactions, perform calculations, validate forms, and make asynchronous requests to servers.

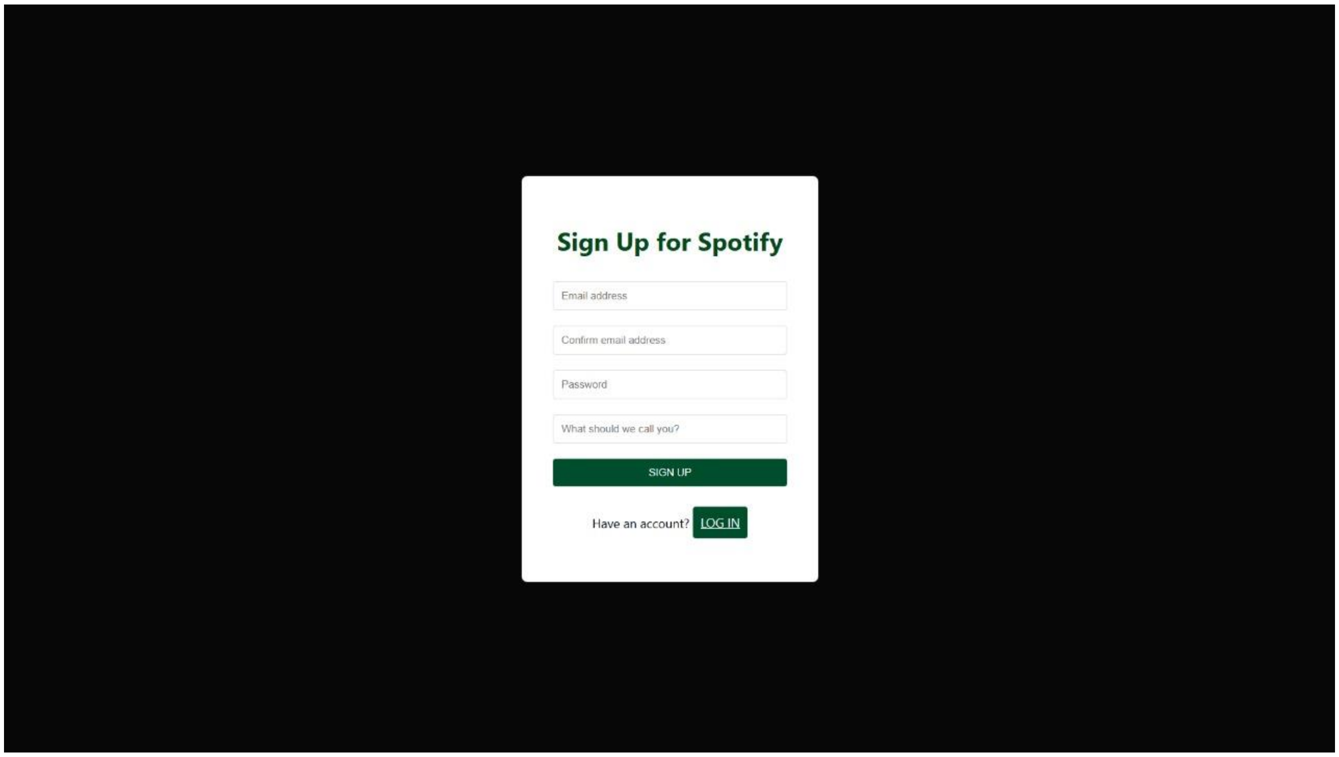
3. **React:** React is a popular JavaScript library for building user interfaces. It provides a component-based approach to web development, allowing developers to create reusable UI components that update efficiently based on changes in data. React uses a virtual DOM (Document Object Model) to optimize performance and facilitate the building of complex web applications.

- **Back-End Programming**

1. **Mongo DB:** A well-known NoSQL database, MongoDB stores data in adaptable documents that resemble JSON. It has excellent performance and scalability for managing massive data volumes. Because of its schema-less design, dynamic schema updates and data modeling are made simple. Among MongoDB's characteristics are replica sets for high availability, automated sharding for horizontal scaling. Its versatility, scalability, and ease of use make it popular in big data, real-time analytics, web applications, and mobile apps.

2. **Node JS:** A server-side JavaScript runtime environment called Node.js is used to create backend applications that are fast and scalable. It is lightweight and efficient because it makes use of an event-driven, non-blocking I/O paradigm. Development is streamlined when JavaScript code can be written for both the client and server sides thanks to Node.js Because Node.js is asynchronous, it's perfect for microservices design, real-time applications, and managing several connections.

UI DESIGN:


Sign Up for Spotify

Email address

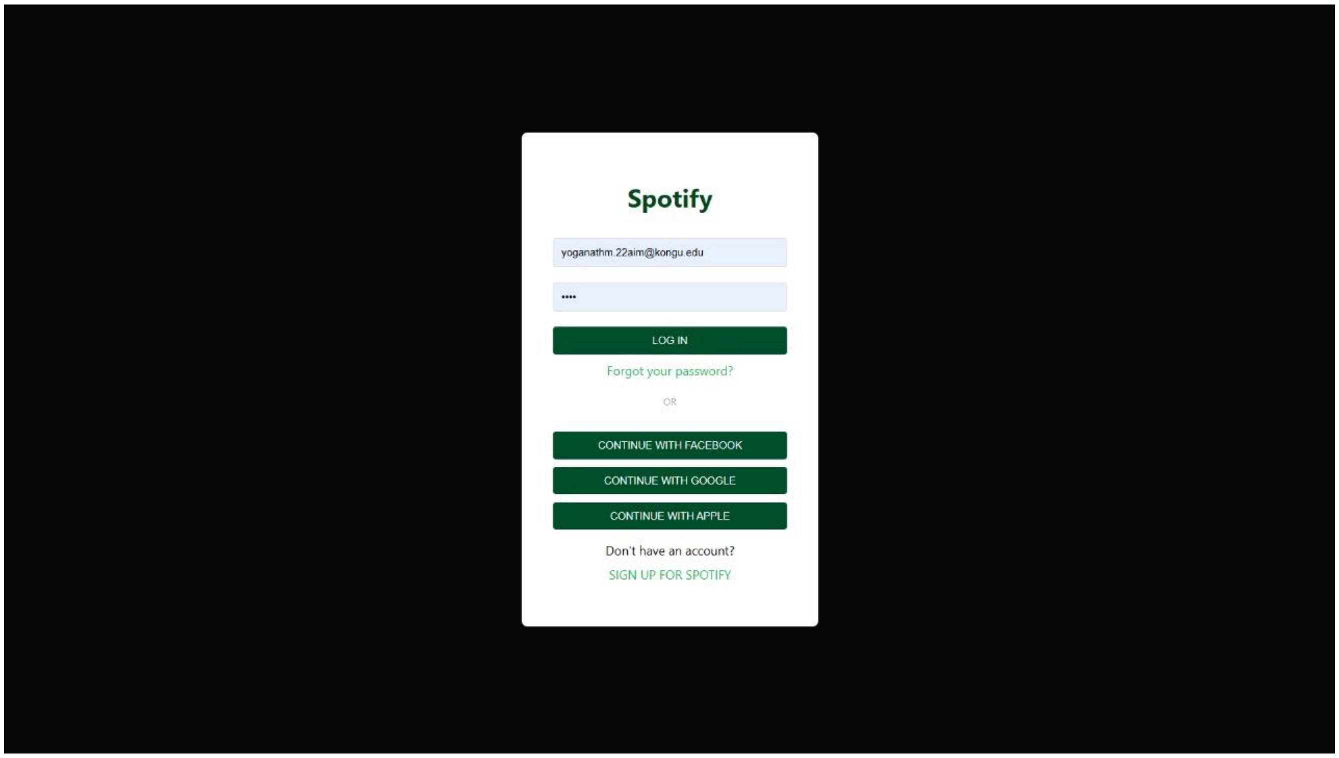
Confirm email address

Password

What should we call you?

SIGN UP

Have an account? LOG IN


Spotify

yoganathm.22aim@kongu.edu

••••

LOG IN

Forgot your password?

OR

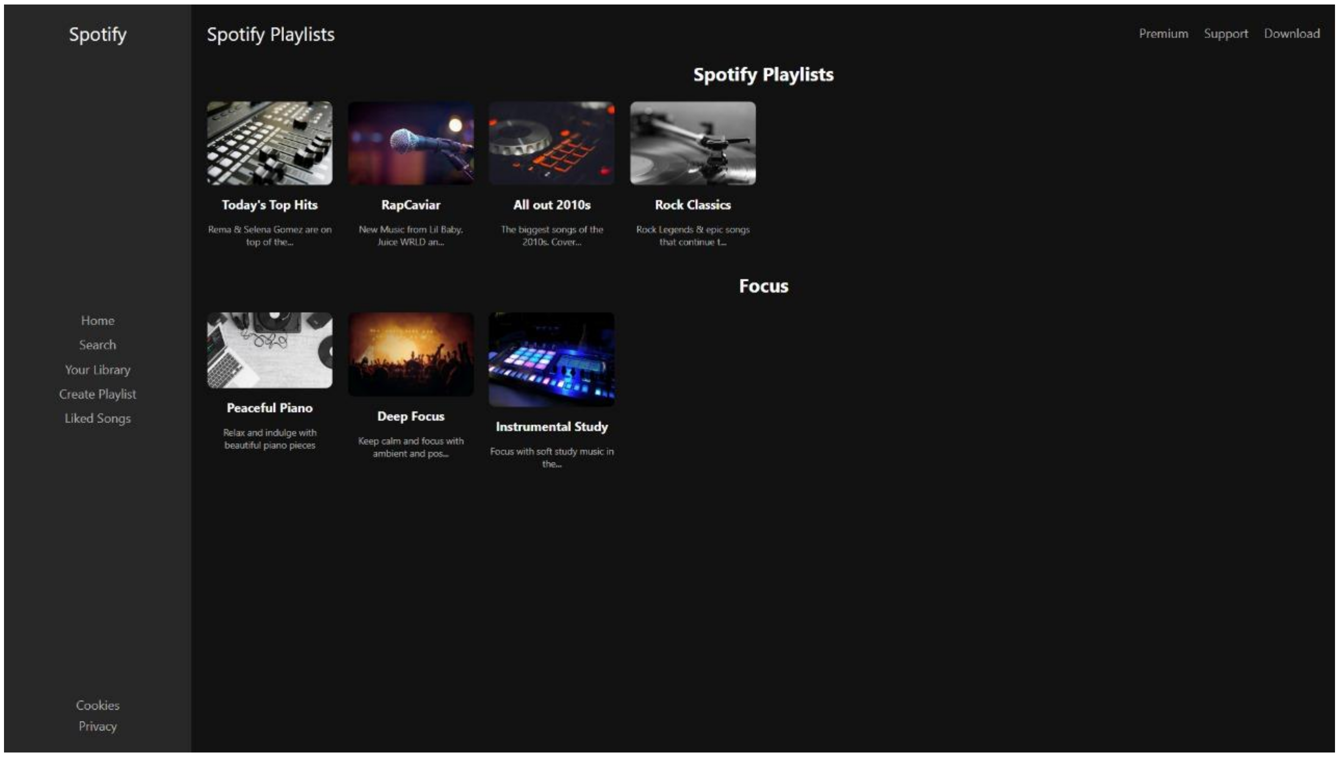
CONTINUE WITH FACEBOOK

CONTINUE WITH GOOGLE

CONTINUE WITH APPLE

Don't have an account?

SIGN UP FOR SPOTIFY



Spotify Playlists

Spotify Playlists

Today's Top Hits
Rema & Selena Gomez are on top of the...

RapCaviar
New Music from Lil Baby. Juice WRLD an...

All out 2010s
The biggest songs of the 2010s. Cover...

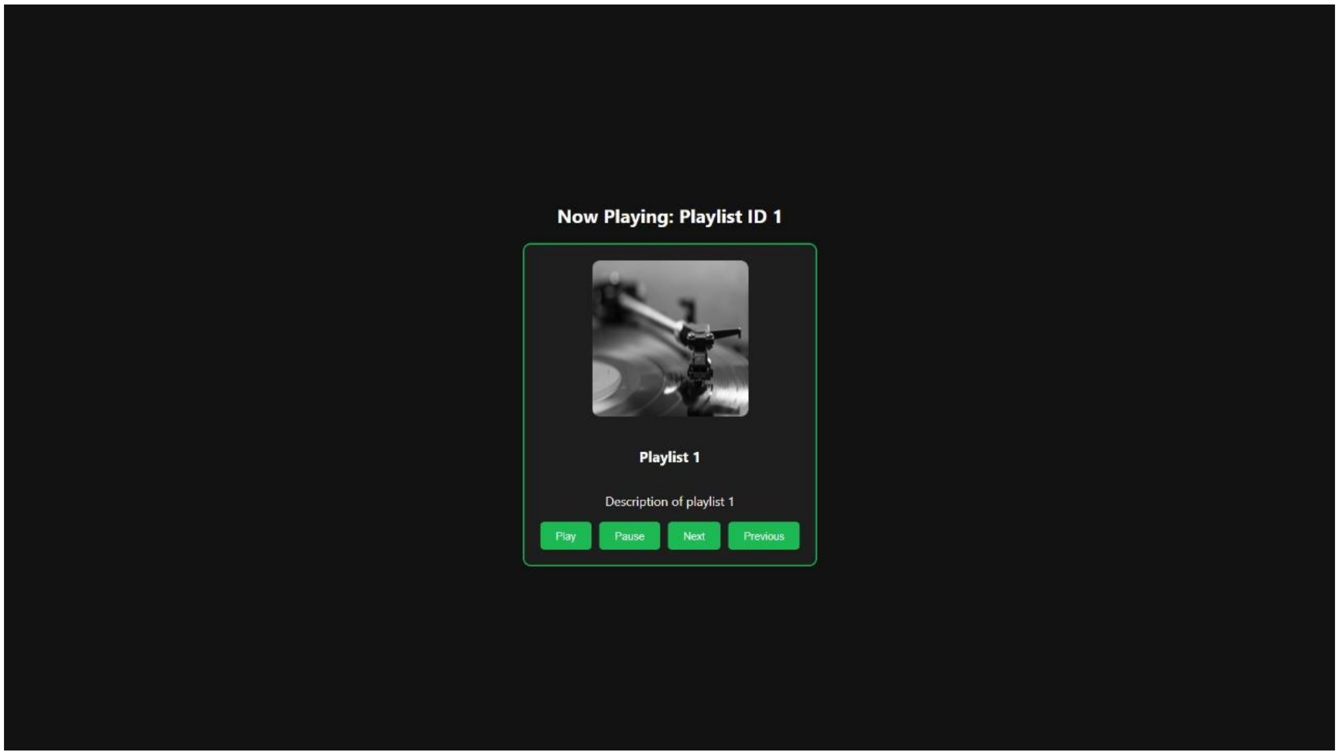
Rock Classics
Rock Legends & epic songs that continue t...

Focus

Peaceful Piano
Relax and indulge with beautiful piano pieces

Deep Focus
Keep calm and focus with ambient and pos...

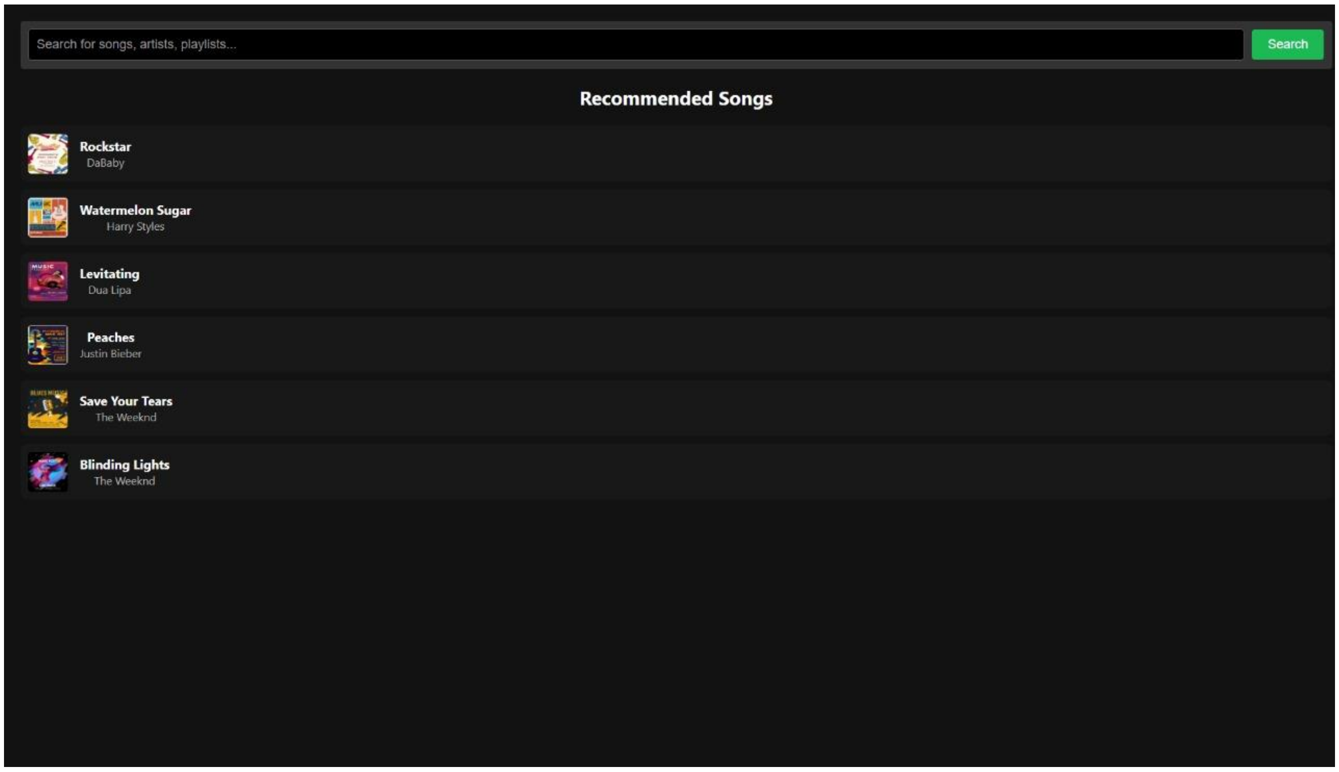
Instrumental Study
Focus with soft study music in the...



Now Playing: Playlist ID 1

Playlist 1

Description of playlist 1



SAMPLE CODING:

- Sign up:

```
import React from 'react';
import { useNavigate } from 'react-router-dom';
import './Login.css';

const SignUp = () => {
  const navigate = useNavigate();

  const handleLoginClick = () => {
    navigate('/');
  };

  return (
    <div className="login-container">
      <div className="login-box">
        <h1>Sign Up for Spotify</h1>
        <form>
          <input type="text" placeholder="Email address" required />
          <input type="text" placeholder="Confirm email address" required />
          <input type="text" placeholder="Password" required />
          <input type="text" placeholder="What should we call you?" required />
          <button type="submit">SIGN UP</button>
        </form>
        <p>Have an account? <button onClick={handleLoginClick} className="link-
      button">LOG IN</button></p>
```

</div>

```
    </div>
  );
};

      export default SignUp;
```

- Login.js

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import axios from 'axios';
import './Login.css';

const Login = () => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const navigate = useNavigate();

  const handleLoginSubmit = async (e) => {
    e.preventDefault();

    try {
      const response = await axios.post('http://localhost:5000/login', { email, password });
      if (response.data.msg === 'Login successful') {
        navigate('/playlist');
      } else {
        alert(response.data.msg);
      }
    } catch (error) {
      console.error('There was an error logging in:', error);
      alert('Login failed. Please check your credentials and try again.');
    }
  };

  return (
    <div className="login-container">
      <div className="login-box">
        <h1>Spotify</h1>
        <form onSubmit={handleLoginSubmit}>
          <input
            type="text"
            placeholder="Email address or username"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            required
          />
          <input
            type="password"
            placeholder="Password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            required
          />
          <button type="submit">LOG IN</button>
        </form>
        <a href="#">Forgot your password?</a>
        <div className="divider">OR</div>
        <button className="facebook-btn">CONTINUE WITH FACEBOOK</button>
        <button className="google-btn">CONTINUE WITH GOOGLE</button>
        <button className="apple-btn">CONTINUE WITH APPLE</button>
        <p>Don't have an account? <a href="/signup">SIGN UP FOR SPOTIFY</a></p>
      </div>
    </div>
  );
};

export default Login;
```

- Playlist.js

```
import React from 'react';
import { useNavigate } from 'react-router-dom';
import './Playlist.css';

const Playlist = () => {
  const navigate = useNavigate();

  const playlists = [
    { id: 1, name: "Today's Top Hits", description: "Rema & Selena Gomez are on top of the...",
      image: "image1.jpg" },
    { id: 2, name: 'RapCaviar', description: 'New Music from Lil Baby, Juice WRLD an...',
      image: "image2.jpg" },
    { id: 3, name: 'All out 2010s', description: 'The biggest songs of the 2010s. Cover...', image:
      "image3.jpg" },
    { id: 4, name: 'Rock Classics', description: 'Rock Legends & epic songs that continue t...',
      image: "image4.jpg" },
    { id: 5, name: 'Peaceful Piano', description: 'Relax and indulge with beautiful piano pieces',
      image: "image5.jpg" },
    { id: 6, name: 'Deep Focus', description: 'Keep calm and focus with ambient and pos...',
      image: "image6.jpg" },
    { id: 7, name: 'Instrumental Study', description: 'Focus with soft study music in the...', image:
      "image7.jpg" }
  ];

  const handleClick = (id) => {
    navigate(/music-player/${id});
  };

  return (
    <div className="playlist-page">
      <div className="sidebar">
        <div className="logo">Spotify</div>
        <nav>
          <a href="#" onClick={() => navigate('/')}>Home</a>
          <a href="#" onClick={() => navigate('/search')}>Search</a> {/* Add Search link */}
          <a href="#" onClick={() => navigate('/playlist')}>Your Library</a>
          <a href="#" onClick={() => navigate('/create-playlist')}>Create Playlist</a>
          <a href="#" onClick={() => navigate('/liked-songs')}>Liked Songs</a>
        </nav>
        <div className="footer-links">
          <a href="#" onClick={() => navigate('/cookies')}>Cookies</a>
          <a href="#" onClick={() => navigate('/privacy')}>Privacy</a>
        </div>
      </div>
      <div className="main-content">
        <div className="header">
          <div className="logo">Spotify Playlists</div>
          <div className="nav-links">
            <a href="#" onClick={() => navigate('/premium')}>Premium</a>
            <a href="#" onClick={() => navigate('/support')}>Support</a>
            <a href="#" onClick={() => navigate('/download')}>Download</a>
          </div>
        </div>
        <div className="playlist-container">
          <h2>Spotify Playlists</h2>
          <div className="playlists">
            {playlists.slice(0, 4).map((playlist) => (
              <div key={playlist.id} className="playlist-item" onClick={() =>
      handleClick(playlist.id)}>
                <img src={/images/${playlist.image}} alt={playlist.name} />
                <h3>{playlist.name}</h3>
                <p>{playlist.description}</p>
              </div>
            ))}
          </div>
          <h2>Focus</h2>
          <div className="playlists">
            {playlists.slice(4).map((playlist) => (
              <div key={playlist.id} className="playlist-item" onClick={() =>
      handleClick(playlist.id)}>
                <img src={/images/${playlist.image}} alt={playlist.name} />
                <h3>{playlist.name}</h3>
                <p>{playlist.description}</p>
              </div>
            ))}
          </div>
        </div>
      </div>
    </div>
  );
};

      export default Playlist;
```

- Search.js

```
import React, { useState } from 'react';
import './Search.css';

const Search = () => {
  const [query, setQuery] = useState('');
  const [results, setResults] = useState([]);

  const exampleData = [
    { id: 1, name: "Blinding Lights", artist: "The Weeknd", image: "image1.jpg" },
    { id: 2, name: 'Rockstar', artist: 'DaBaby', image: "image2.jpg" },
    { id: 3, name: 'Watermelon Sugar', artist: 'Harry Styles', image: "image3.jpg" },
    { id: 4, name: 'Levitating', artist: 'Dua Lipa', image: "image4.jpg" },
    { id: 5, name: 'Peaches', artist: 'Justin Bieber', image: "image5.jpg" },
    { id: 6, name: 'Save Your Tears', artist: 'The Weeknd', image: "image6.jpg" },
    { id: 7, name: 'Blinding Lights', artist: 'The Weeknd', image: "image7.jpg" }
  ];

  const handleSearch = (event) => {
    event.preventDefault();
    const filteredResults = exampleData.filter(item =>
    item.name.toLowerCase().includes(query.toLowerCase()) ||
    item.artist.toLowerCase().includes(query.toLowerCase()));
    setResults(filteredResults);
  };

  return (
    <div className="search-page">
      <div className="search-bar">
        <form onSubmit={handleSearch}>
          <input
            type="text"
            value={query}
            onChange={(e) => setQuery(e.target.value)}
            placeholder="Search for songs, artists, playlists..."
          />
          <button type="submit">Search</button>
        </form>
      </div>
      <div className="search-results">
        <h2>{query ? 'Search Results' : 'Recommended Songs'}</h2>
        {query ? (
          results.length > 0 ? (
            results.map(result => (
              <div key={result.id} className="search-item">
                <img src={/images/${result.image}} alt={result.name} />
                <div>
                  <h3>{result.name}</h3>
                  <p>{result.artist}</p>
                </div>
              </div>
          ))
        ) : (
          <p>No results found.</p>
        )
      ) : (
        exampleData.map(item => (
          <div key={item.id} className="search-item">
            <img src={/images/${item.image}} alt={item.name} />
            <div>
              <h3>{item.name}</h3>
              <p>{item.artist}</p>
            </div>
          </div>
        ))
      )}
    </div>
  </div>
 );
};

   export default Search;
```

- Music player.jsx

```
import React, { useState, useEffect } from 'react';
import { useParams } from 'react-router-dom';
import SpotifyWebApi from 'spotify-web-api-js';
import './MusicPlayer.css'; // Import custom CSS for MusicPlayer

const spotifyApi = new SpotifyWebApi();

const MusicPlayer = () => {
  const { id } = useParams();
  const [currentTrack, setCurrentTrack] = useState(null);
  const [isPlaying, setIsPlaying] = useState(false);
  const [progressMs, setProgressMs] = useState(0);
  const [durationMs, setDurationMs] = useState(0);
  const [error, setError] = useState(null); // State for error handling

  useEffect(() => {
    const fetchSongDetails = async () => {
      try {
        const response = await spotifyApi.getTrack(id);
        setCurrentTrack(response);
        setIsPlaying(true); // Auto-play the song when details are fetched
      } catch (error) {
        console.error('Error fetching song details:', error);
        setError(error); // Set error state if request fails
      }
    };

    if (id) {
```

```
    fetchSongDetails();
  }
}, [id]);

useEffect(() => {
  const interval = setInterval(() => {
    spotifyApi.getMyCurrentPlaybackState().then((response) => {
    if (response && response.is_playing) {
        setProgressMs(response.progress_ms);
        setDurationMs(response.item.duration_ms);
      }
    }).catch((error) => {
      console.error('Error fetching playback state:', error);
      setError(error); // Set error state if request fails
    });
  }, 1000); // Update every second

  return () => clearInterval(interval);
}, []);

const handlePlayPause = () => {
  if (isPlaying) {
    spotifyApi.pause().then(() => setIsPlaying(false)).catch((error) => {
      console.error('Error pausing playback:', error);
      setError(error); // Set error state if request fails
    });
  } else {
    spotifyApi.play().then(() => setIsPlaying(true)).catch((error) => {
      console.error('Error resuming playback:', error);
      setError(error); // Set error state if request fails
    });
  }
};

const formatDuration = (ms) => {
  const totalSeconds = Math.floor(ms / 1000);
  const minutes = Math.floor(totalSeconds / 60);
  const seconds = totalSeconds % 60;
  return ${minutes}:${seconds < 10 ? '0' : ''}${seconds};
};

if (error) {
  return <div className="error-message">Error: {error.message}</div>;
}

return (
  <div className="music-player-container">
    {currentTrack && (
      <div className="music-player">
        <div className="album-cover-container">
          <img src={currentTrack.album.images[0].url} alt={currentTrack.album.name}
      className="album-cover" />
          </div>
          <div className="track-info">
            <h2>{currentTrack.name}</h2>
            <p>{currentTrack.artists.map((artist) => artist.name).join(', ')}</p>
          </div>
          <div className="controls">
            <div className="progress">
              <div className="progress-bar" style={{ width: ${(progressMs / durationMs) *
      100}% }}></div>
            </div>
            <div className="playback-controls">
              <button className="control-button" onClick={handlePlayPause}>
                {isPlaying ? 'Pause' : 'Play'}
              </button>
            </div>
            <div className="duration">
              <span>{formatDuration(progressMs)}</span> /
      <span>{formatDuration(durationMs)}</span>
            </div>
          </div>
        </div>
      )}
    </div>
  );
};

      export default MusicPlayer;
```

CONCLUSION:

To sum up, creating a Spotify clone website with React, Node.js, and MongoDB is the pinnacle of modern web development techniques. The platform offers a smooth and captivating user experience, enabling music lovers to easily browse through their favorite songs by utilizing React's component-based architecture. Node.js, on the other hand, gives the backend efficiency and scalability, guaranteeing that the platform can meet the demands of an expanding user base without sacrificing performance. The project's agility is further increased by the integration of MongoDB as the database solution, which makes it possible to store and retrieve music data effectively while taking changing user preferences into account.

The Spotify clone not only mimics the features of the original but also establishes the groundwork for further advancements in the field of music streaming services through the convergence of these technologies. This project is proof of the ability of contemporary databases and web development frameworks to produce engaging digital experiences that appeal to people all around the world. In the end, the Spotify clone is a monument to the inventiveness and inventiveness of developers in using technology to improve our online experience.