Project Report on

SoundBox – A Web Music Player

at

U. V. Patel College of Engineering

Internal Guide :

Prof. Rachana Modi

Prepared By :

Mr. Akshat Sharma
(19012011077)
Mr. Kaushal Vibhakar
(19012011091)

B.Tech Semester VII
(Computer Engineering)
Nov-Dec, 2022

Submitted to,
Department of Computer Engineering
U.V. Patel College of Engineering
Ganpat University, Kherva - 384 012

U.V. PATEL COLLEGE OF ENGINEERING

26/11/22

C E R T I F I C A T E

TO WHOM SO EVER IT MAY CONCERN

This is to certify that **Mr. Akshat Sharma** student of **B.Tech. Semester VII (Computer Engineering)** has completed his full semester on site project work titled "**SoundBox**" satisfactorily in partial fulfillment of the requirement of Bachelor of Technology degree of Computer Engineering of Ganpat University, Kherva, Mehsana in the year 2022-2023.

Prof. Rachana Modi	**Dr. Paresh M. Solanki**
College Project Guide	**Head, Computer Engineering**

U.V. PATEL COLLEGE OF ENGINEERING

26/11/22

C E R T I F I C A T E

TO WHOM SO EVER IT MAY CONCERN

This is to certify that **Mr. Kaushal Vibhakar** student of **B.Tech. Semester VII (Computer Engineering)** has completed his full semester on site project work titled "**SoundBox**" satisfactorily in partial fulfillment of the requirement of Bachelor of Technology degree of Computer Engineering of Ganpat University, Kherva, Mehsana in the year 2022-2023.

Prof. Rachana Modi	**Dr. Paresh M. Solanki**
College Project Guide	**Head, Computer Engineering**

ACKNOWLEDGEMENT

This satisfaction that successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation it made it possible, whose constant guidance and encouragement crown all efforts with success. We are grateful to our guide **Prof. Rachana Modi** for the guidance, inspiration and constructive suggestions that helpful us in the preparation of this project. We also thank our colleagues who have helped in successful completion of the project.

INDEX

LIST OF FIGURES

LIST OF TABLES

1. Introduction

1.1 Problem Statement :

Most of the Applications right now are bloated with unnecessary features which is a bad experience for the user and they are full of ads. So an Application was needed which overcomes these problems.

1.2 Objective :

To Implement Online Music Web Application with a proper user interface. The motivation of this project comes from my desire to learn the increasingly growing field of React with Firebase server database designing, website designing and their growing popularity by taking up this project.

1.3 Basic Workflow :

Register > Login > Home Page > Logout

1.4 Benefits :

This application will have no ads and no subscription requirements and so it’s completely free to use.

1.5 Project Duration :

6 Months.

2. Feasibility Study

2.1 Study of Current System :

Due to the fierce competition between music player applications, many developers tried to add many features, advertise and content to their respective music player in order to retain their users and attract new users. This trend has made it harder for users to get content from their music player, which also means it's harder to filter the content that they want. With the continuous iteration of application and a growing number of features, the music player will become even more bloated and the user's experience will become less smooth.

2.2 Requirements of New System :

All the existing systems are non-responsive and full of ads.

Whenever a new system (a hardware or software) is to be introduced, there is a need to study the new system in every aspect or manner before working on it. It gives the idea whether the project is adequate or not. This is where Feasibility study comes in. The three key considerations are involved in the feasibility analysis are as under :

2.3 Technical Feasibility :

It refers to technical know-how.

Existing technology supports the system completely.

2.4 Operational Feasibility :

It means that it’s possible to practically implement the project. While installing this software, the hardware and software requirements should be specified.

Our system will be easy to install and use. Hence our system is operationally feasible.

2.5 Economical Feasibility :

It refers to financial support required. It refers to finance incurred during the development of the project.

Economical Feasibility is mainly concerned with the cost incurred in the implementation of the software. Since this project is developed using React with JavaScript and Firebase Server which is most commonly available and even the cost involved in the installation process is not high. Hence this project has good economical feasibility.

2.6

Hardware Requirements : Processor : Intel Pentium 4 or More
: RAM : 1 GB or More
: Internet Connectivity is a must for this purpose.
: Basic Peripherals

Software Requirements : JavaScript
: React.js
: Tailwind
: Firebase Database Servers
: Microsoft Visual Studio 2008

3. System Requirements Study

3.1 Functional Requirements :

These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Examples : Register & Login.
: Verification Email is sent when the user registers for the first time.
: OTP is sent when user registers with mobile number.
: Login via Google Account.

3.2 Non-functional Requirements :

These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioural requirements.
They basically deal with issues like : Portability
: Security
: Maintainability
: Reliability
: Scalability
: Performance
: Reusability

Examples : Each user must use different Emails.
: Password should contain Alphabets as well as Numbers.

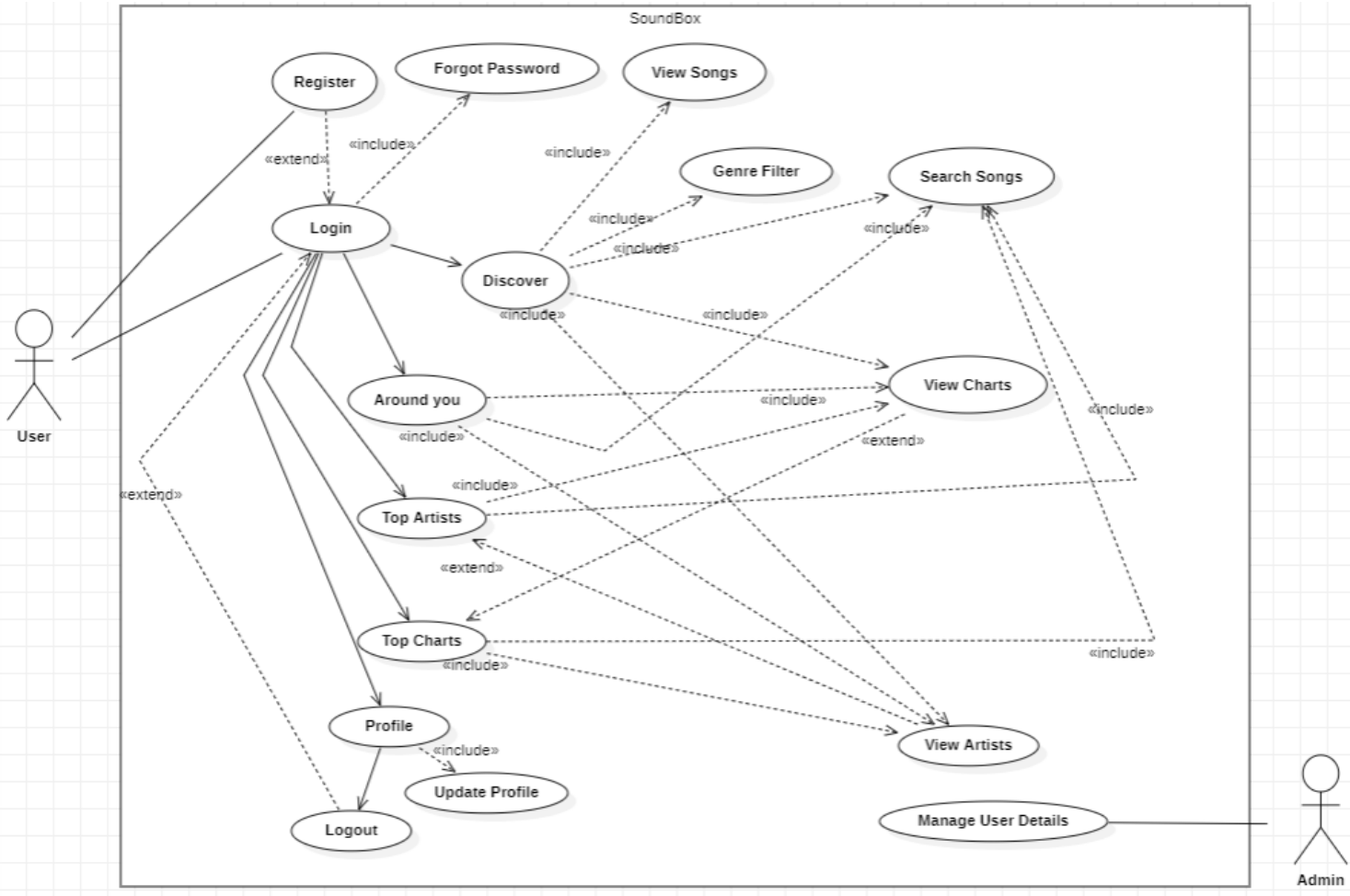
4. Project Planning

4.1 Gantt Chart :

(Figure 4.1)

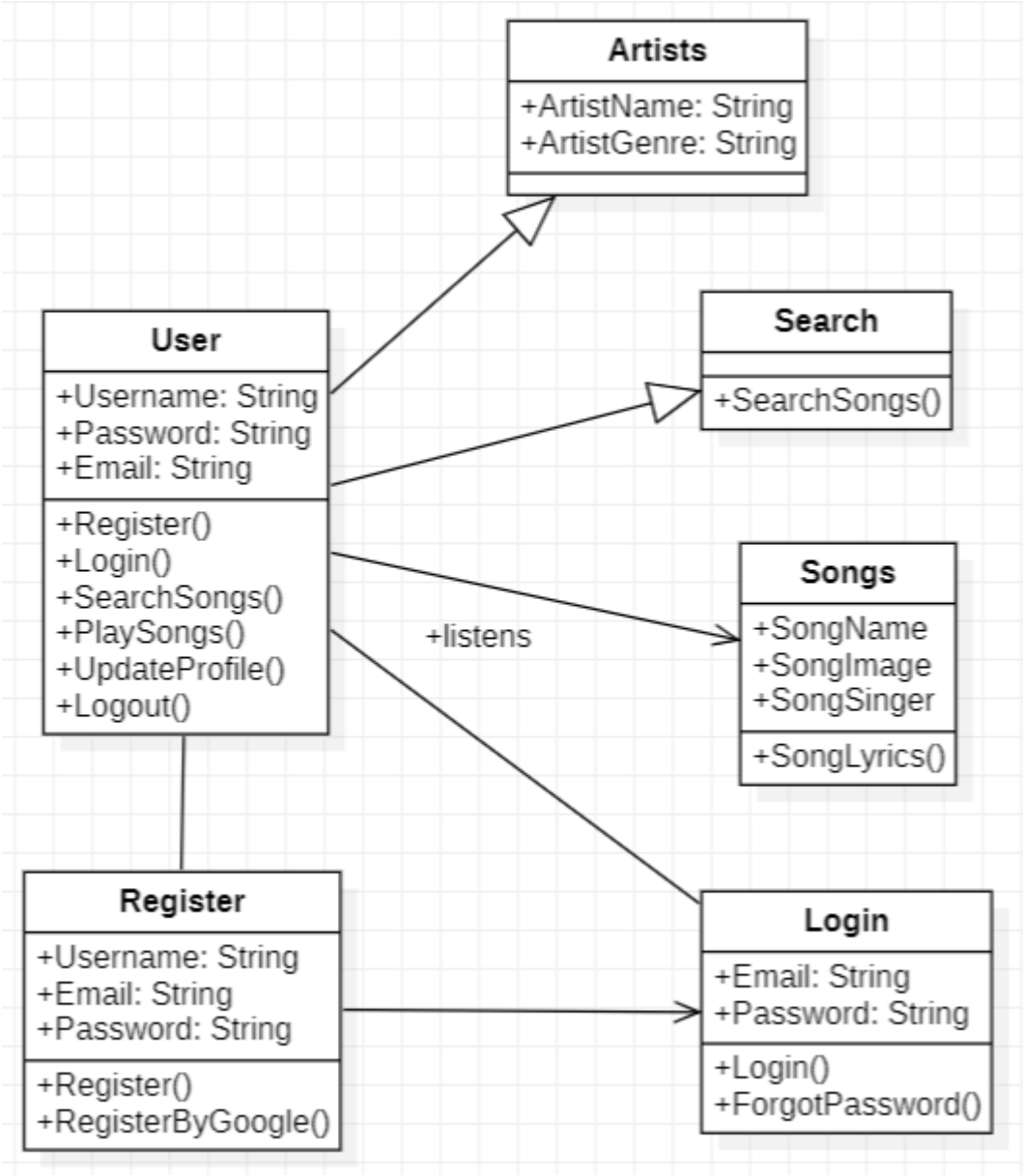
5. System Design

5.1 Use Case Diagram :



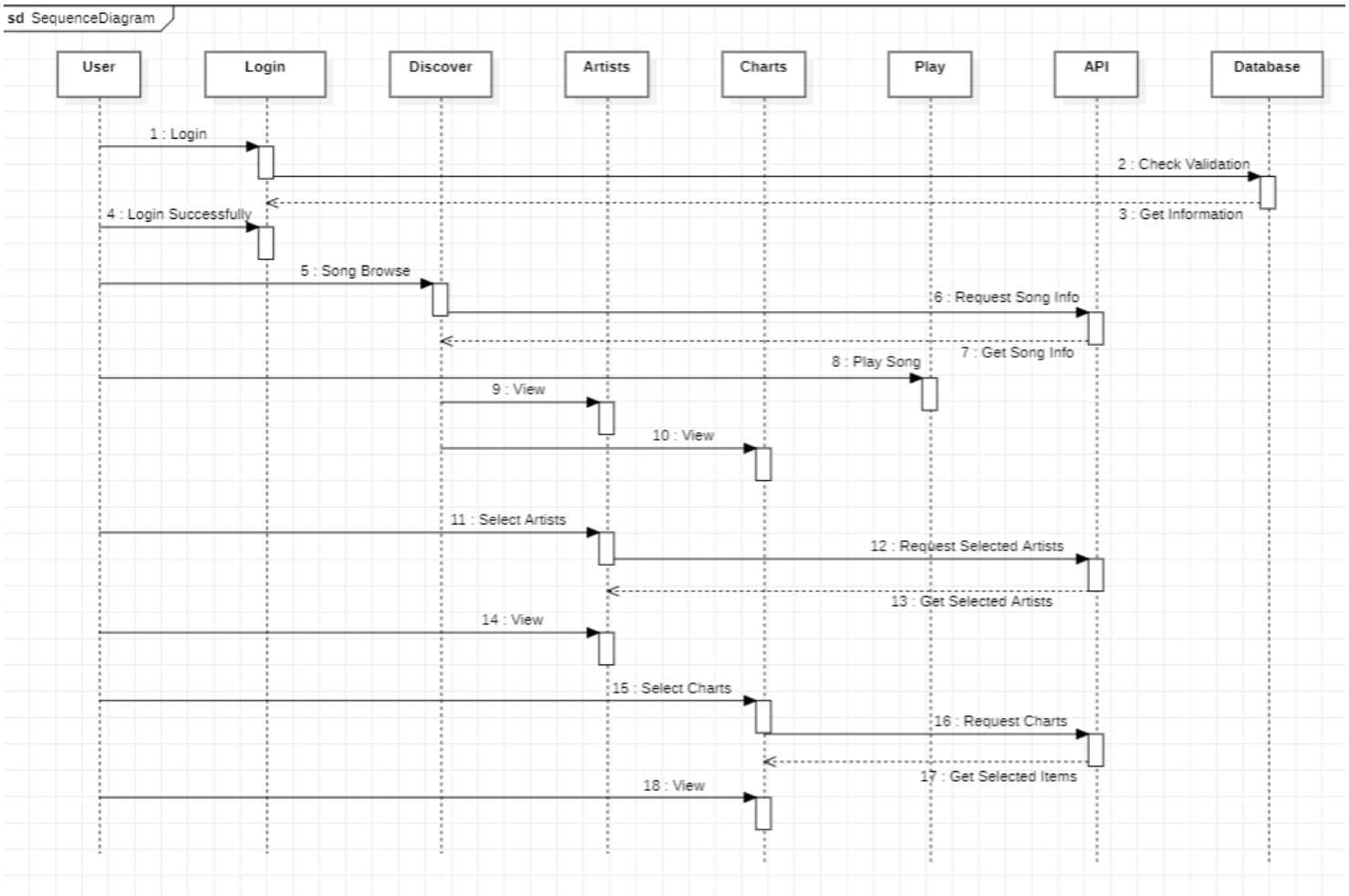
(Figure 5.1)

5.2 Class Diagram :



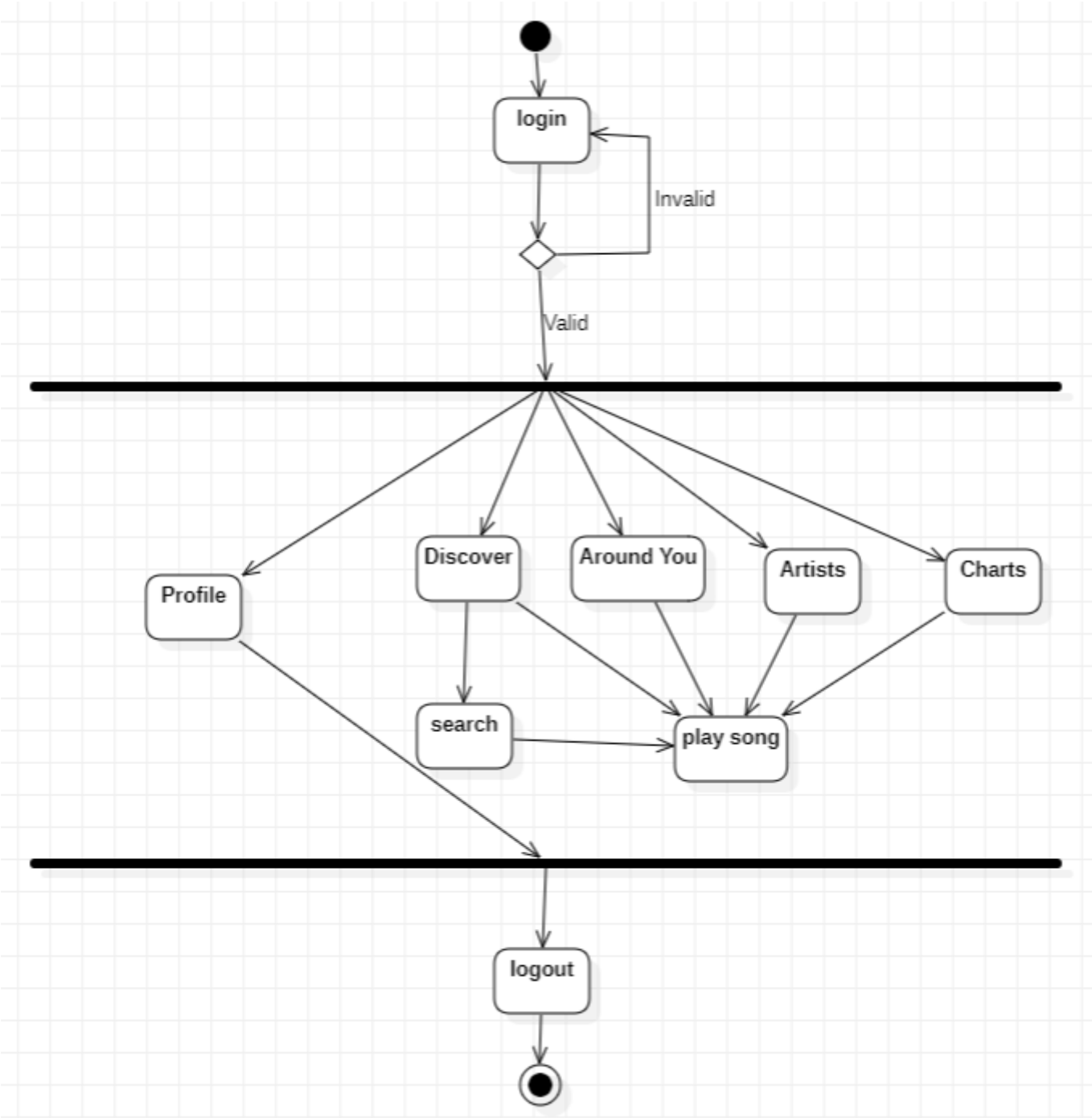
(Figure 5.2)

5.3 Sequence Diagram :



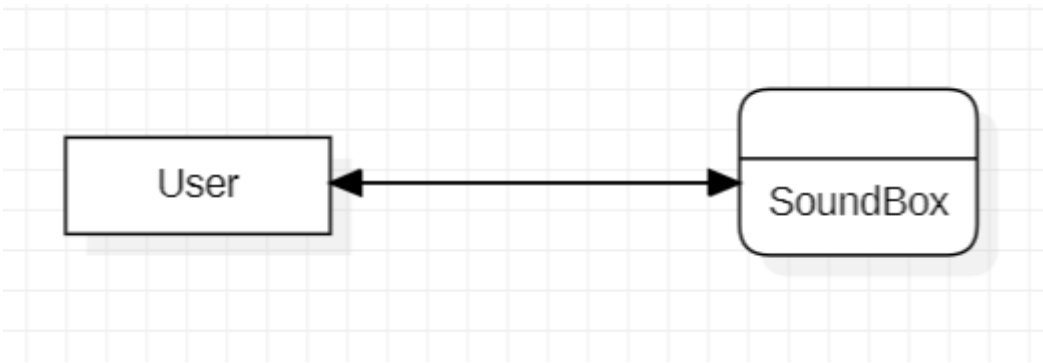
(Figure 5.3)

5.4 Activity Diagram :

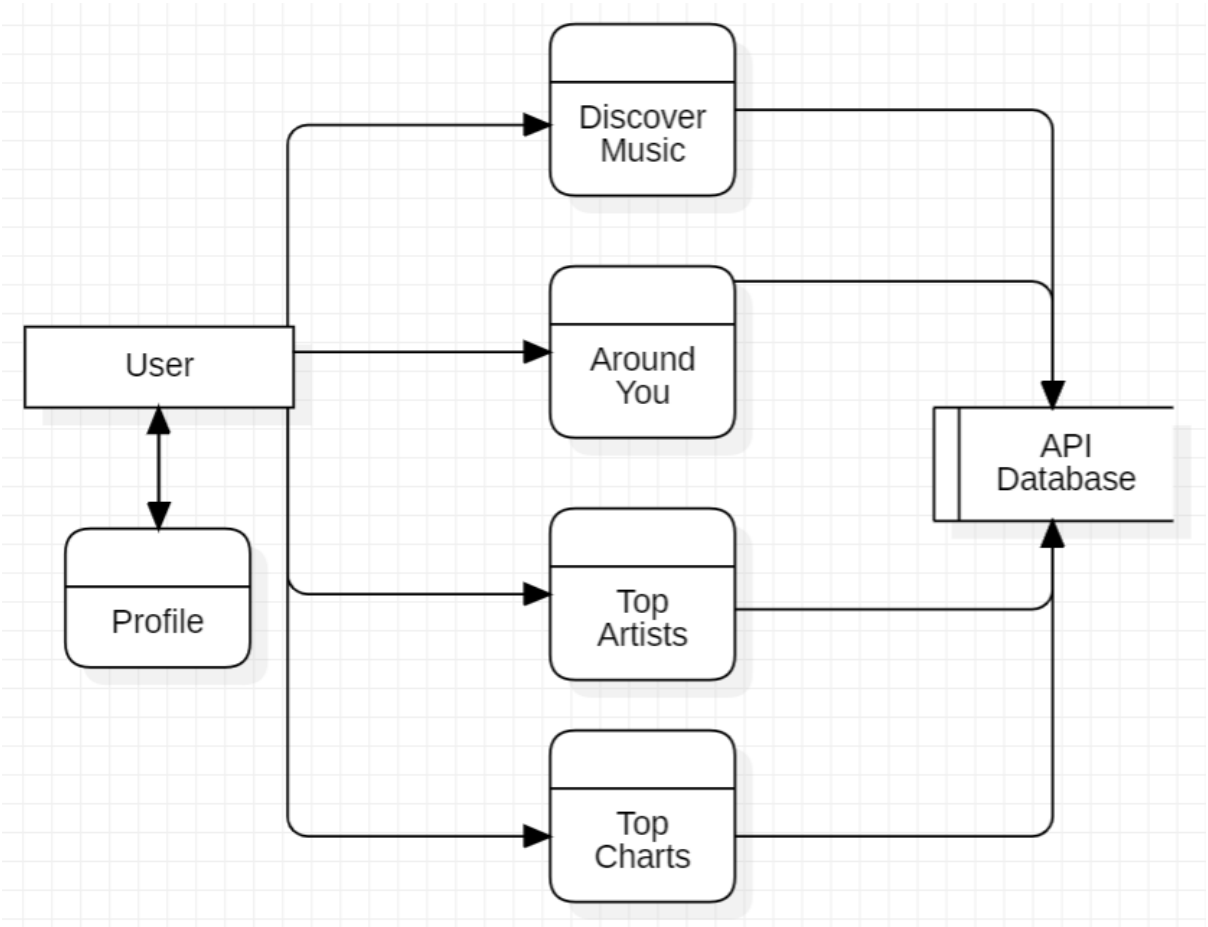


(Figure 5.4)

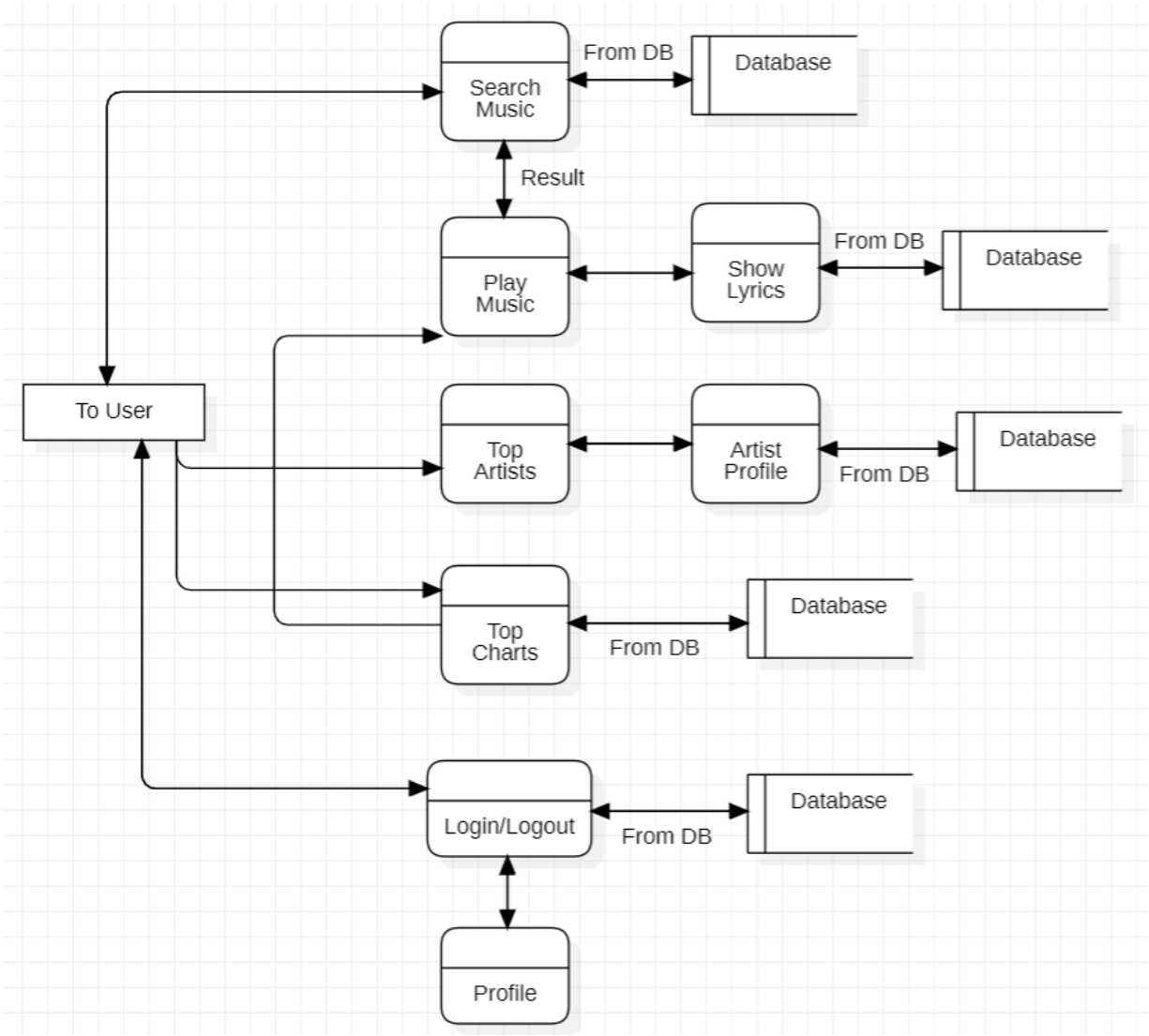
5.5 Data Flow Diagrams :



Zero Level DFD (Figure 5.5)



First Level DFD (Figure 5.6)



Second Level DFD (Figure 5.7)

6. Data Dictionary :

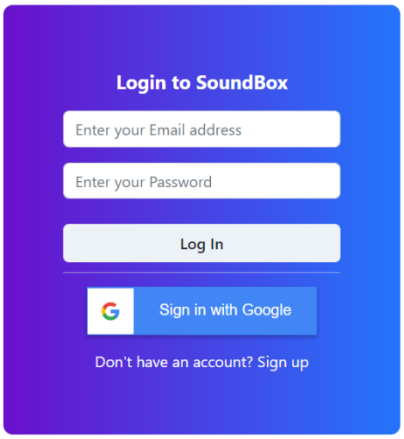
6.1 User Table :

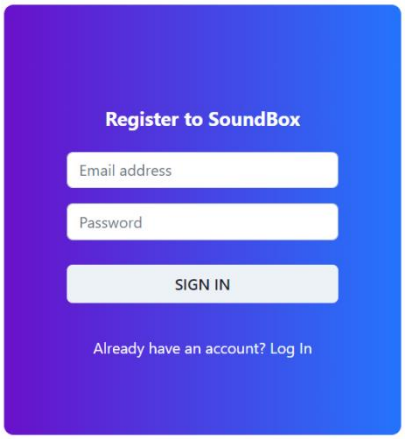
Field Name	Data Type	Constraint	Description
User Name (Primary Key)	Varchar	Not Null	Profile Name
Password	Varchar	Not Null	Password for Login Procedure
Email	Varchar	Not Null	Email for Login Procedure

(Table 6.1)

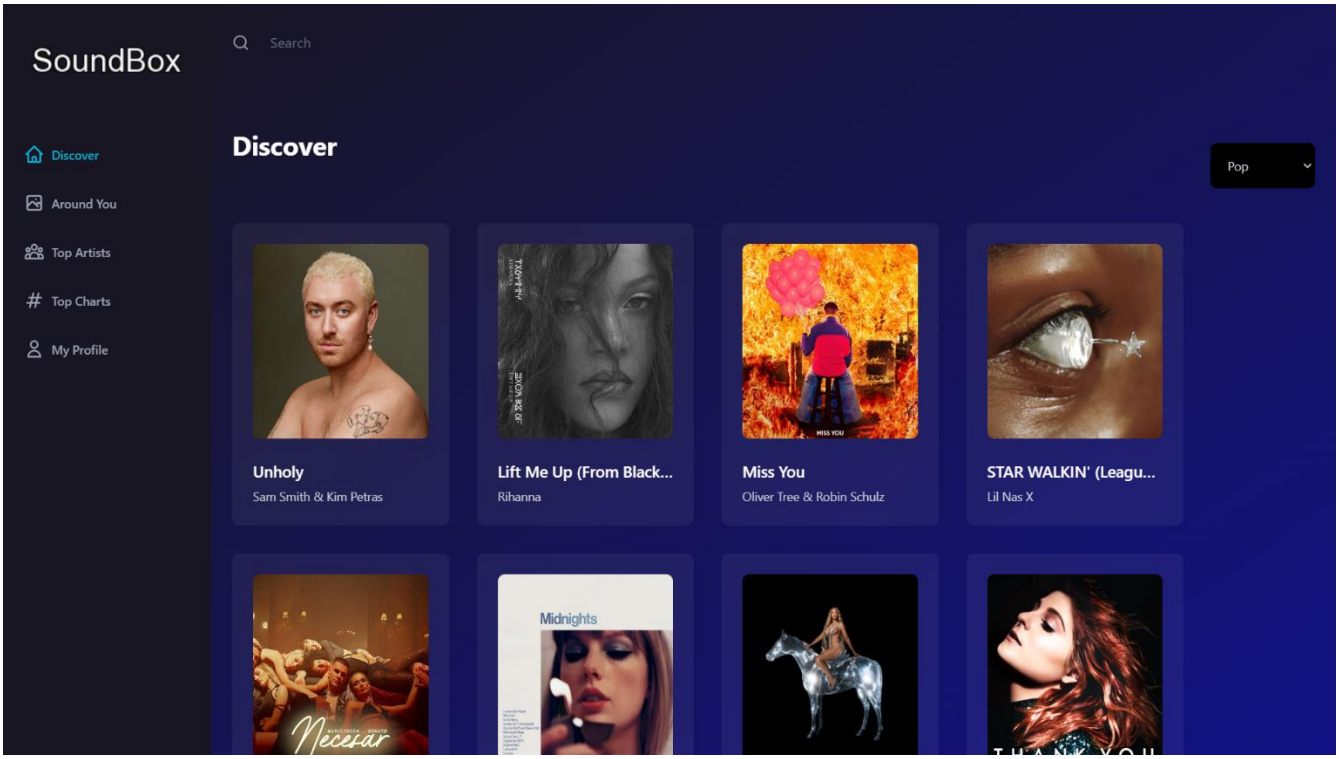
7. Prototype

Login & Signup UI :





Homepage (Discover Page) :



8. Implementation

8.1 Coding :

Login.jsx :

```
import React, { useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import { Form, Alert } from "react-bootstrap";
import { Button } from "react-bootstrap";
import GoogleButton from "react-google-button";
import { useUserAuth } from "../context/UserAuthContext";
const Login = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState("");
  const { logIn, googleSignIn } = useUserAuth();
  const navigate = useNavigate();
  const handleSubmit = async (e) => {
    e.preventDefault();
    setError("");
    try {
      await logIn(email, password);
      navigate("/discover");
    } catch (err) {
      setError(err.message);
    }
  };
  const handleGoogleSignIn = async (e) => {
    e.preventDefault();
    try {
      await googleSignIn();
      navigate("/discover");
    } catch (error) {
      console.log(error.message);
    }
  };
  return (
    <>
      <div className="p-4 box">
        <h2 className="mb-3">Login to SoundBox</h2>
        {error && <Alert variant="danger">{error}</Alert>}
        <Form onSubmit={handleSubmit}>
          <Form.Group className="mb-3" controlId="formBasicEmail">
            <Form.Control
              type="email"
```

```
              placeholder="Enter your Email address"
              onChange={(e) => setEmail(e.target.value)}
            />
          </Form.Group>
          <Form.Group className="mb-3" controlId="formBasicPassword">
            <Form.Control
              type="password"
              placeholder="Enter your Password"
              onChange={(e) => setPassword(e.target.value)}
            />
          </Form.Group>
          <div className="d-grid gap-2">
            <Button variant="primary" type="Submit">
              Log In
            </Button>
          </div>
        </Form>
        <hr />
        <div>
          <GoogleButton
            className="g-btn"
            type="dark"
            onClick={handleGoogleSignIn}
          />
        </div>
      </div>
      <div className="p-4 box mt-3 text-center">
        Don't have an account? <Link to="/signup">Sign up</Link>
      </div>
    </>
  );
};
export default Login;
```

Signup.jsx :

```
import React, { useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import { Form, Alert } from "react-bootstrap";
import { Button } from "react-bootstrap";
import { useUserAuth } from "../context/UserAuthContext";
const Signup = () => {
  const [email, setEmail] = useState("");
  const [error, setError] = useState("");
  const [password, setPassword] = useState("");
```

```
  const { signUp } = useUserAuth();
  let navigate = useNavigate();
  const handleSubmit = async (e) => {
    e.preventDefault();
    setError("");
    try {
      await signUp(email, password);
      navigate("/");
    } catch (err) {
      setError(err.message);
    }
  };
  return (
    <>
      <div className="p-4 box">
        <h2 className="mb-3">Register to SoundBox</h2>
        {error && <Alert variant="danger">{error}</Alert>}
        <Form onSubmit={handleSubmit}>
          <Form.Group className="mb-3" controlId="formBasicEmail">
            <Form.Control
              type="email"
              placeholder="Email address"
              onChange={(e) => setEmail(e.target.value)}
            />
          </Form.Group>
          <Form.Group className="mb-3" controlId="formBasicPassword">
            <Form.Control
              type="password"
              placeholder="Password"
              onChange={(e) => setPassword(e.target.value)}
            />
          </Form.Group>
          <div className="d-grid gap-2">
            <Button variant="primary" type="Submit">
              Sign up
            </Button>
          </div>
        </Form>
      </div>
      <div className="p-4 box mt-3 text-center">
        Already have an account? <Link to="/">Log In</Link>
      </div>
    </>
  );
};
export default Signup;
```

Discover.jsx :

```
import React from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { Error, Loader, SongCard } from '../components';
import { selectGenreListId } from '../redux/features/playerSlice';
import { useGetSongsByGenreQuery } from '../redux/services/shazamCore';
import { genres } from '../assets/constants';
const Discover = () => {
  const dispatch = useDispatch();
  const { genreListId } = useSelector((state) => state.player);
  const { activeSong, isPlaying } = useSelector((state) => state.player);
  const { data, isFetching, error } = useGetSongsByGenreQuery(genreListId || 'POP');
  if (isFetching) return <Loader title="Loading songs..." />;
  if (error) return <Error />;
  const genreTitle = genres.find(({ value }) => value === genreListId)?.title;
  return (
    <div className="flex flex-col">
      <div className="w-full flex justify-between items-center sm:flex-row flex-col mt-4 mb-
10">
        <h2 className="font-bold text-3xl text-white text-left">Discover {genreTitle}</h2>
        <select
          onChange={(e) => dispatch(selectGenreListId(e.target.value))}
          value={genreListId || 'pop'}
          className="bg-black text-gray-300 p-3 text-sm rounded-lg outline-none sm:mt-0 mt-5">
          {genres.map((genre) => <option key={genre.value}
value={genre.value}>{genre.title}</option>)}
        </select>
      </div>
      <div className="flex flex-wrap sm:justify-start justify-center gap-8">
        {data?.map((song, i) => (
          <SongCard
            key={song.key}
            song={song}
            isPlaying={isPlaying}
            activeSong={activeSong}
            data={data}
            i={i}
          />
        ))}
      </div>
    </div>
  );
};
export default Discover;
```

AroundYou.jsx :

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { useSelector } from 'react-redux';
import { Error, Loader, SongCard } from '../components';
import { useGetSongsByCountryQuery } from '../redux/services/shazamCore';
const CountryTracks = () => {
  const [country, setCountry] = useState('');
  const [loading, setLoading] = useState(true);
  const { activeSong, isPlaying } = useSelector((state) => state.player);
  const { data, isFetching, error } = useGetSongsByCountryQuery(country);
  useEffect(() => {
    axios
      .get(`https://geo.ipify.org/api/v2/country?apiKey=at_UHbgHg8tZGEfzkzwvc1HcODVFkVMM`)
      .then((res) => setCountry(res?.data?.location.country))
      .catch((err) => console.log(err))
      .finally(() => setLoading(false));
  }, [country]);
  if (isFetching && loading) return <Loader title="Loading Songs around you..." />;
  if (error && country !== '') return <Error />;
  return (
    <div className="flex flex-col">
      <h2 className="font-bold text-3xl text-white text-left mt-4 mb-10">Around you <span
className="font-black">{country}</span></h2>
      <div className="flex flex-wrap sm:justify-start justify-center gap-8">
        {data?.map((song, i) => (
          <SongCard
            key={song.key}
            song={song}
            isPlaying={isPlaying}
            activeSong={activeSong}
            data={data}
            i={i}
          />
        ))}
      </div>
    </div>
  );
};
export default CountryTracks;
```

TopArtists.jsx :

```
import React from 'react';
import { ArtistCard, Error, Loader } from '../components';
import { useGetTopChartsQuery } from '../redux/services/shazamCore';
const TopArtists = () => {
  const { data, isFetching, error } = useGetTopChartsQuery();
  if (isFetching) return <Loader title="Loading artists..." />;
  if (error) return <Error />;
  return (
    <div className="flex flex-col">
      <h2 className="font-bold text-3xl text-white text-left mt-4 mb-10">Discover Top
Artists</h2>
      <div className="flex flex-wrap sm:justify-start justify-center gap-8">
        {data?.map((track) => <ArtistCard key={track.key} track={track} />)}
      </div>
    </div>
  );
};
export default TopArtists;
```

TopCharts :

```
import React from 'react';
import { useSelector } from 'react-redux';
import { Error, Loader, SongCard } from '../components';
import { useGetTopChartsQuery } from '../redux/services/shazamCore';
const TopCharts = () => {
  const { data, isFetching, error } = useGetTopChartsQuery();
  const { activeSong, isPlaying } = useSelector((state) => state.player);
  if (isFetching) return <Loader title="Loading Top Charts" />;
  if (error) return <Error />;
  return (
    <div className="flex flex-col">
      <h2 className="font-bold text-3xl text-white text-left mt-4 mb-10">Discover Top
Charts</h2>
      <div className="flex flex-wrap sm:justify-start justify-center gap-8">
        {data.map((song, i) => (
          <SongCard
            key={song.key}
            song={song}
            isPlaying={isPlaying}
            activeSong={activeSong}
            data={data}
            i={i}
```

```
          />
        ))}
      </div>
    </div>
  );
};
export default TopCharts;
```

MyProfile.jsx :

```
import React from 'react';
import { Button } from "react-bootstrap";
import { useNavigate } from "react-router";
import { useUserAuth } from "../context/UserAuthContext";
import { img } from '../assets';
const MyProfile = () => {
  const { logOut, user } = useUserAuth();
  const navigate = useNavigate();
  const handleLogout = async () => {
    try {
      await logOut();
      navigate("/");
    } catch (error) {
      console.log(error.message);
    }
  };
  return (
    <>
      <div className="flex flex-col">
        <div className="relative w-full flex flex-col">
          <div className="w-full bg-gradient-to-l from-transparent to-black sm:h-48 h-28" />
          <div className="absolute inset-0 flex items-center">
            <img
              alt="Profile Image"
              src={img}
              className="sm:w-48 w-28 sm:h-48 h-28 rounded-full object-cover border-2 shadow-xl
shadow-black"
            />
            <div className="ml-5">
              <p className="font-bold sm:text-3xl text-xl text-white">
                {user && user.displayName}
              </p>
              <p className="text-base text-gray-400 mt-2">
                Email ID : {user && user.email}
              </p>
```

```
            </div>
          </div>
          <div className="w-full sm:h-44 h-24" />
        </div>
      </div>
      <div className="d-grid gap-2">
        <Button variant="primary" onClick={handleLogout}>
          Log out
        </Button>
      </div>
    </>
  );
};
export default MyProfile;
```

ArtistsDetails.jsx :

```
import React from 'react';
import { useParams } from 'react-router-dom';
import { useSelector } from 'react-redux';
import { DetailsHeader, Error, Loader, RelatedSongs } from '../components';
import { useGetArtistDetailsQuery } from '../redux/services/shazamCore';
const ArtistDetails = () => {
  const { id: artistId } = useParams();
  const { activeSong, isPlaying } = useSelector((state) => state.player);
  const { data: artistData, isFetching: isFetchingArtistDetails, error } =
useGetArtistDetailsQuery(artistId);
  if (isFetchingArtistDetails) return <Loader title="Loading artist details..." />;
  if (error) return <Error />;
  return (
    <div className="flex flex-col">
      <DetailsHeader
        artistId={artistId}
        artistData={artistData}
      />
      <RelatedSongs
        data={Object.values(artistData?.songs)}
        artistId={artistId}
        isPlaying={isPlaying}
        activeSong={activeSong}
      />
    </div>
  );
};
export default ArtistDetails;
```

SongDetails.jsx :

```
import React from 'react';
import { useParams } from 'react-router-dom';
import { useSelector, useDispatch } from 'react-redux';
import { DetailsHeader, Error, Loader, RelatedSongs } from '../components';
import { setActiveSong, playPause } from '../redux/features/playerSlice';
import { useGetSongDetailsQuery, useGetSongRelatedQuery } from '../redux/services/shazamCore';
const SongDetails = () => {
  const dispatch = useDispatch();
  const { songid, id: artistId } = useParams();
  const { activeSong, isPlaying } = useSelector((state) => state.player);
  const { data, isFetching: isFetchinRelatedSongs, error } = useGetSongRelatedQuery({ songid
});
  const { data: songData, isFetching: isFetchingSongDetails } = useGetSongDetailsQuery({ songid
});
  if (isFetchingSongDetails && isFetchinRelatedSongs) return <Loader title="Searching song
details" />;
  console.log(songData);
  if (error) return <Error />;
  const handlePauseClick = () => {
    dispatch(playPause(false));
  };
  const handlePlayClick = (song, i) => {
    dispatch(setActiveSong({ song, data, i }));
    dispatch(playPause(true));
  };
  return (
    <div className="flex flex-col">
      <DetailsHeader
        artistId={artistId}
        songData={songData}
      />
      <div className="mb-10">
        <h2 className="text-white text-3xl font-bold">Lyrics :</h2>
        <div className="mt-5">
          {songData?.sections[1].type === 'LYRICS'
            ? songData?.sections[1]?.text.map((line, i) => (
              <p key={`lyrics-${line}-${i}`} className="text-gray-400 text-base my-
1">{line}</p>
            ))
            : (
              <p className="text-gray-400 text-base my-1">Sorry, No lyrics found!</p>
            )}
        </div>
      </div>
```

```
      <RelatedSongs
        data={data}
        artistId={artistId}
        isPlaying={isPlaying}
        activeSong={activeSong}
        handlePauseClick={handlePauseClick}
        handlePlayClick={handlePlayClick}
      />
    </div>
  );
};
export default SongDetails;
```

Search.jsx :

```
import React from 'react';
import { useSelector } from 'react-redux';
import { useParams } from 'react-router-dom';
import { Error, Loader, SongCard } from '../components';
import { useGetSongsBySearchQuery } from '../redux/services/shazamCore';
const Search = () => {
  const { searchTerm } = useParams();
  const { activeSong, isPlaying } = useSelector((state) => state.player);
  const { data, isFetching, error } = useGetSongsBySearchQuery(searchTerm);
  const songs = data?.tracks?.hits.map((song) => song.track);
  if (isFetching) return <Loader title={`Searching ${searchTerm}...`} />;
  if (error) return <Error />;
  return (
    <div className="flex flex-col">
      <h2 className="font-bold text-3xl text-white text-left mt-4 mb-10">Showing results for
<span className="font-black">{searchTerm}</span></h2>
      <div className="flex flex-wrap sm:justify-start justify-center gap-8">
        {songs.map((song, i) => (
          <SongCard
            key={song.key}
            song={song}
            isPlaying={isPlaying}
            activeSong={activeSong}
            data={data}
            i={i}
          />
        ))}
      </div>
    </div>
  );
```

```
};
export default Search;
```

ProtectedRoute.jsx :

```
import React from "react";
import { Navigate } from "react-router-dom";
import { useUserAuth } from "../context/UserAuthContext";
const ProtectedRoute = ({ children }) => {
  const { user } = useUserAuth();
  console.log("Check user in Private : ", user);
  if (!user) {
    return <Navigate to="/" />;  }
  return children;
};
export default ProtectedRoute;
```

UserAuthContext.jsx :

```
import { createContext, useContext, useEffect, useState } from "react";
import {
  createUserWithEmailAndPassword,
  signInWithEmailAndPassword,
  onAuthStateChanged,
  signOut,
  GoogleAuthProvider,
  signInWithPopup,
} from "firebase/auth";
import { auth } from "../firebase";
const userAuthContext = createContext();
export function UserAuthContextProvider({ children }) {
  const [user, setUser] = useState({});
  function logIn(email, password) {
    return signInWithEmailAndPassword(auth, email, password);
  }
  function signUp(email, password) {
    return createUserWithEmailAndPassword(auth, email, password);
  }
  function logOut() {
    return signOut(auth);
  }
  function googleSignIn() {
    const googleAuthProvider = new GoogleAuthProvider();
    return signInWithPopup(auth, googleAuthProvider);
```

```
  }
  useEffect(() => {
    const unsubscribe = onAuthStateChanged(auth, (currentuser) => {
      console.log("Auth", currentuser);
      setUser(currentuser);
    });
    return () => {
      unsubscribe();
    };
  }, []);
  return (
    <userAuthContext.Provider
      value={{ user, logIn, signUp, logOut, googleSignIn }}
    >
      {children}
    </userAuthContext.Provider>
  );
}
export function useUserAuth() {
  return useContext(userAuthContext);
}
```

Index.jsx :

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import { Provider } from 'react-redux';
import './index.css';
import { store } from './redux/store';
import { ArtistDetails, TopArtists, AroundYou, Discover, Search, SongDetails, TopCharts, Login,
Signup, MyProfile } from './pages';
import ProtectedRoute from './components/ProtectedRoute';
import { UserAuthContextProvider } from "./context/UserAuthContext";
import App from './App';
ReactDOM.createRoot(document.getElementById('root')).render(
  <div>
    <BrowserRouter>
      <Provider store={store}>
        <UserAuthContextProvider>
          <Routes>
            <Route
              path="/discover"
              element={
                <ProtectedRoute>
```

```
            <App><Discover /></App>
          </ProtectedRoute>
        }
      />
      <Route
        path="/around-you"
        element={
          <ProtectedRoute>
            <App><AroundYou /></App>
          </ProtectedRoute>
        }
      />
      <Route
        path="/top-artists"
        element={
          <ProtectedRoute>
            <App><TopArtists /></App>
          </ProtectedRoute>
        }
      />
      <Route
        path="/top-charts"
        element={
          <ProtectedRoute>
            <App><TopCharts /></App>
          </ProtectedRoute>
        }
      />
      <Route
        path="/my-profile"
        element={
          <ProtectedRoute>
            <App><MyProfile /></App>
          </ProtectedRoute>
        }
      />
      <Route
        path="/artists/:id"
        element={
          <ProtectedRoute>
            <App><ArtistDetails /></App>
          </ProtectedRoute>
        }
      />
      <Route
        path="/songs/:songid"
```

```
              element={
                <ProtectedRoute>
                  <App><SongDetails /></App>
                </ProtectedRoute>
              }
            />
            <Route
              path="/search/:searchTerm"
              element={
                <ProtectedRoute>
                  <App><Search /></App>
                </ProtectedRoute>
              }
            />
            <Route path="/signup" exact element={<Signup />} />
            <Route path="/" exact element={<Login />} />
          </Routes>
        </UserAuthContextProvider>
      </Provider>
    </BrowserRouter>
  </div>
);
```

App.jsx :

```
import { useSelector } from 'react-redux';
import { Route, Routes } from 'react-router-dom';
import { Searchbar, Sidebar, MusicPlayer, TopPlay } from './components';
import { ArtistDetails, TopArtists, AroundYou, Discover, Search, SongDetails, TopCharts, Login,
Signup, MyProfile } from './pages';
const App = ({ children }) => {
  const { activeSong } = useSelector((state) => state.player);
  const MainLayout = ({ children }) => {
    return (
      <div className="relative flex">
        <Sidebar />
        <div className="flex-1 flex flex-col bg-gradient-to-br from-black to-[#121286]">
          <Searchbar />
          <div className="px-6 h-[calc(100vh-72px)] overflow-y-scroll hide-scrollbar flex
xl:flex-row flex-col-reverse">
            <div className="flex-1 h-fit pb-40">
              {children}
            </div>
          </div>
        </div>
```

```
      </div>
    )
  }
  const BaseLayout = ({ children }) => {
    return (
      <div className="relative flex">
        <Sidebar />
        <div className="flex-1 flex flex-col bg-gradient-to-br from-black to-[#121286]">
          <Searchbar />
          <div className="px-6 h-[calc(100vh-72px)] overflow-y-scroll hide-scrollbar flex
xl:flex-row flex-col-reverse">
            <div className="flex-1 h-fit pb-40">
              {children}
            </div>
            {activeSong?.title && (
              <div className="absolute h-28 bottom-0 left-0 right-0 flex animate-slideup bg-
gradient-to-br from-white/10 to-[#2a2a80] backdrop-blur-lg rounded-t-3xl z-10">
                <MusicPlayer />
              </div>
            )}
          </div> </div> </div>
    )
  }
  return (
    <>
      <MainLayout>{children}</MainLayout>
      {activeSong?.title && (
        <div className="absolute h-28 bottom-0 left-0 right-0 flex animate-slideup bg-gradient-
to-br from-white/10 to-[#2a2a80] backdrop-blur-lg rounded-t-3xl z-10">
          <MusicPlayer />
        </div>
      )}
    </>
  );
};
export default App;
```

Firebase.js :

```
import { initializeApp } from "firebase/app";
import { getAuth } from "firebase/auth";
const firebaseConfig = {
    apiKey: "AIzaSyBxHSC64Y_xKANIJkc6YBbpVJlYMgqsgVI",
    authDomain: "soundbox-619.firebaseapp.com",
    projectId: "soundbox-619",
```

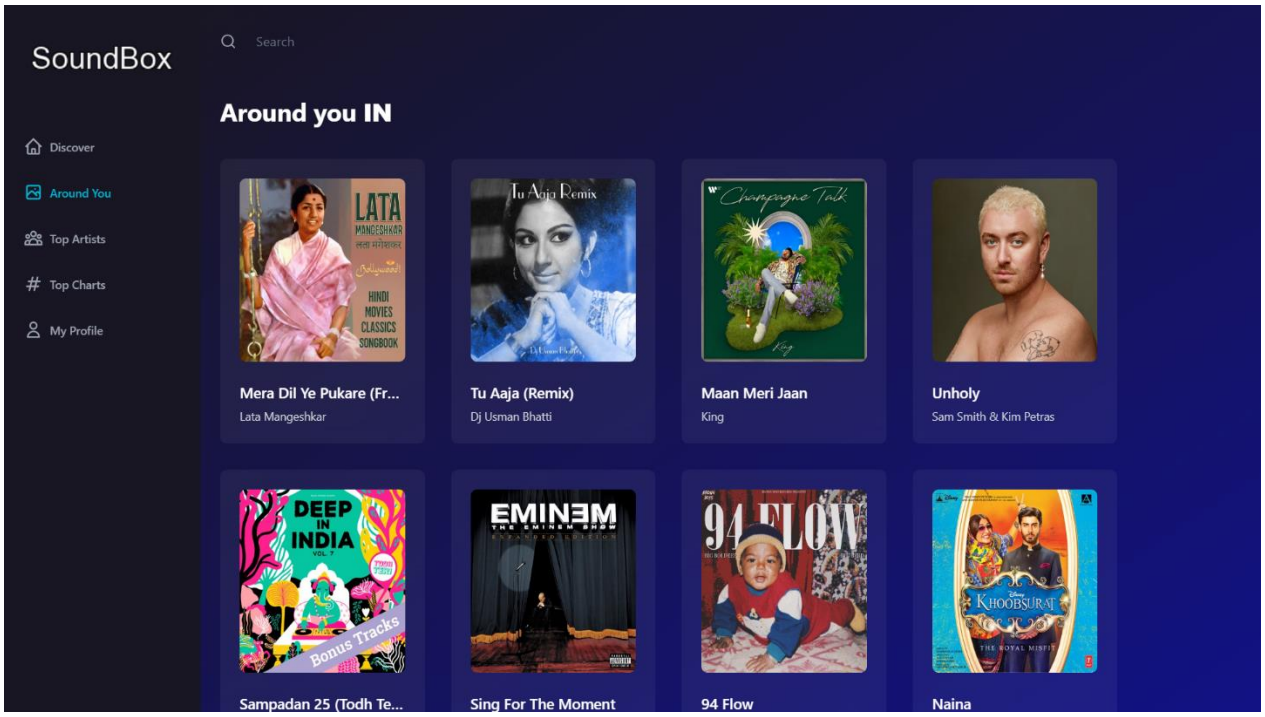
```
    storageBucket: "soundbox-619.appspot.com",
    messagingSenderId: "971462227342",
    appId: "1:971462227342:web:2ae5c99c0784f4ddf3a0df",
    measurementId: "G-K2Y6TCG9TS"
};
const app = initializeApp(firebaseConfig);
export const auth = getAuth(app);
export default app;
```

ShazamCore.jsx :

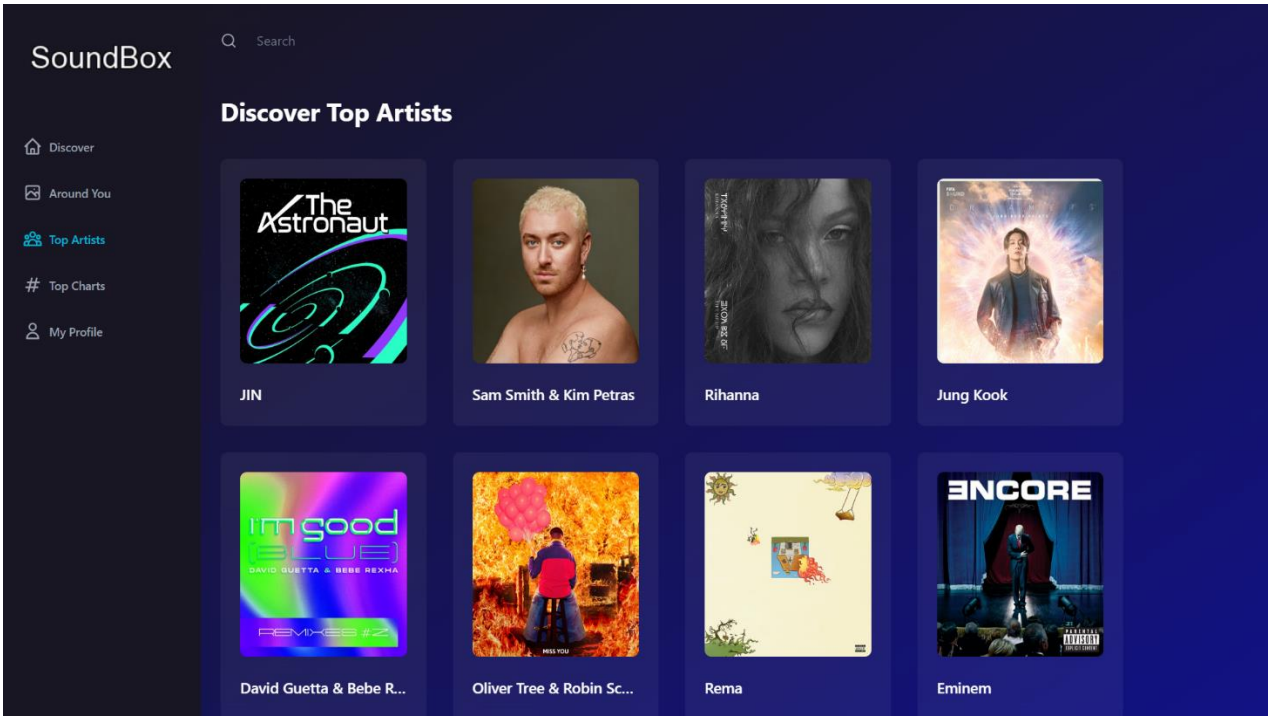
```
import { createApi, fetchBaseQuery } from '@reduxjs/toolkit/query/react';
export const shazamCoreApi = createApi({
  reducerPath: 'shazamCoreApi',
  baseQuery: fetchBaseQuery({
    baseUrl: 'https://shazam-core.p.rapidapi.com/v1',
    prepareHeaders: (headers) => {
      headers.set('X-RapidAPI-Key', '45cb4b580fmshb8e3206982904cbp19581ajsn4668bff45f11');
      return headers;
    },
  }),
  endpoints: (builder) => ({
    getTopCharts: builder.query({ query: () => '/charts/world' }),
    getSongsByGenre: builder.query({ query: (genre) => `/charts/genre-
world?genre_code=${genre}` }),
    getSongsByCountry: builder.query({ query: (countryCode) =>
`/charts/country?country_code=${countryCode}` }),
    getSongsBySearch: builder.query({ query: (searchTerm) =>
`/search/multi?search_type=SONGS_ARTISTS&query=${searchTerm}` }),
    getArtistDetails: builder.query({ query: (artistId) =>
`/artists/details?artist_id=${artistId}` }),
    getSongDetails: builder.query({ query: ({ songid }) => `/tracks/details?track_id=${songid}`
}),
    getSongRelated: builder.query({ query: ({ songid }) => `/tracks/related?track_id=${songid}`
}),
  }),
});
export const {
  useGetTopChartsQuery,
  useGetSongsByGenreQuery,
  useGetSongsByCountryQuery,
  useGetSongsBySearchQuery,
  useGetArtistDetailsQuery,
  useGetSongDetailsQuery,
  useGetSongRelatedQuery,
} = shazamCoreApi;
```

8.2 Screenshots :

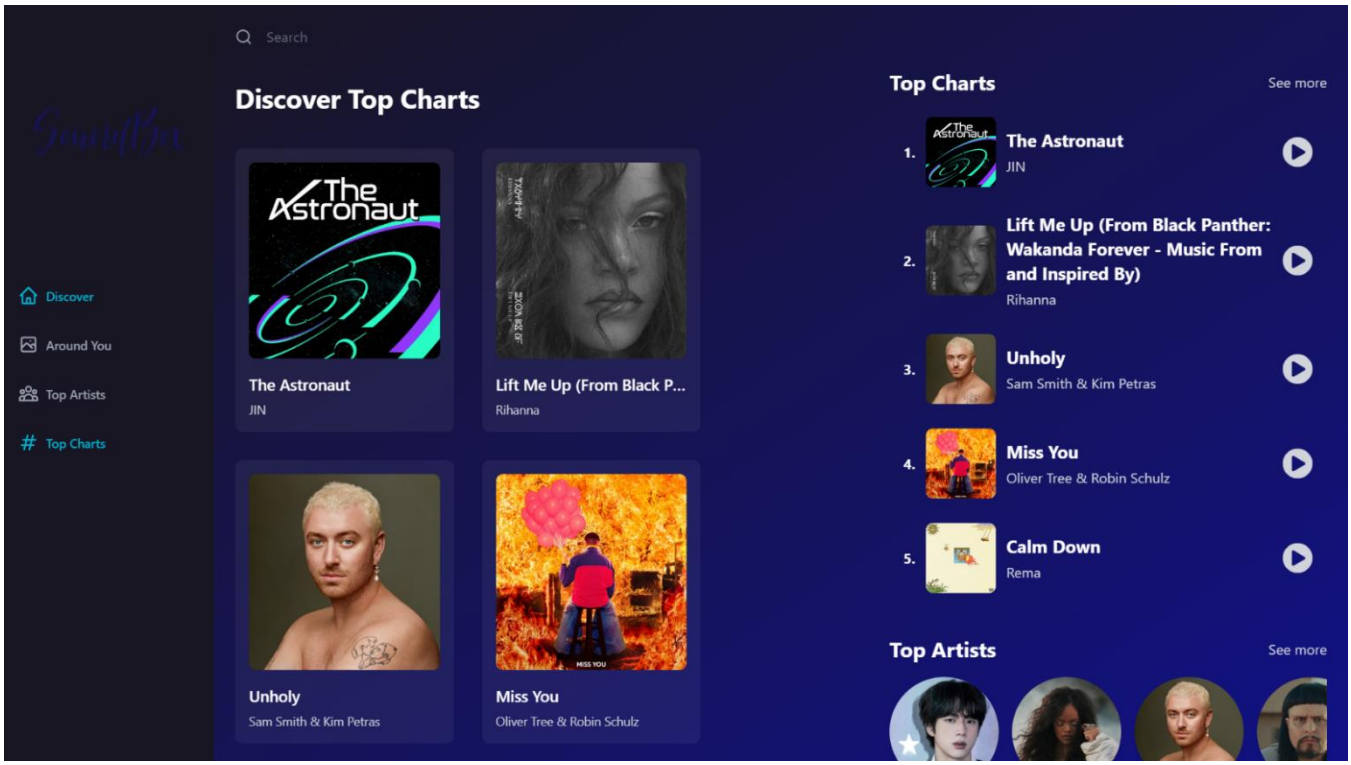
Around You Page :



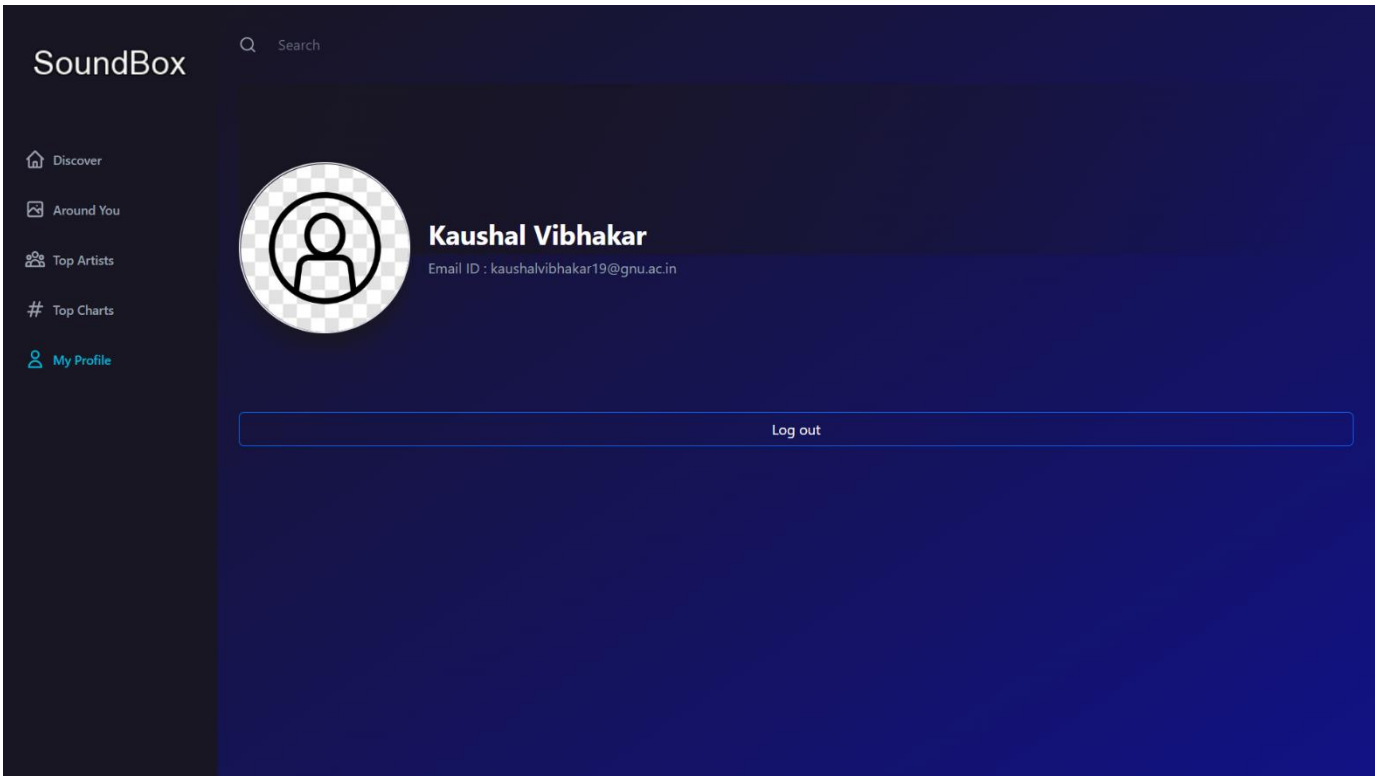
Top Artists Page :



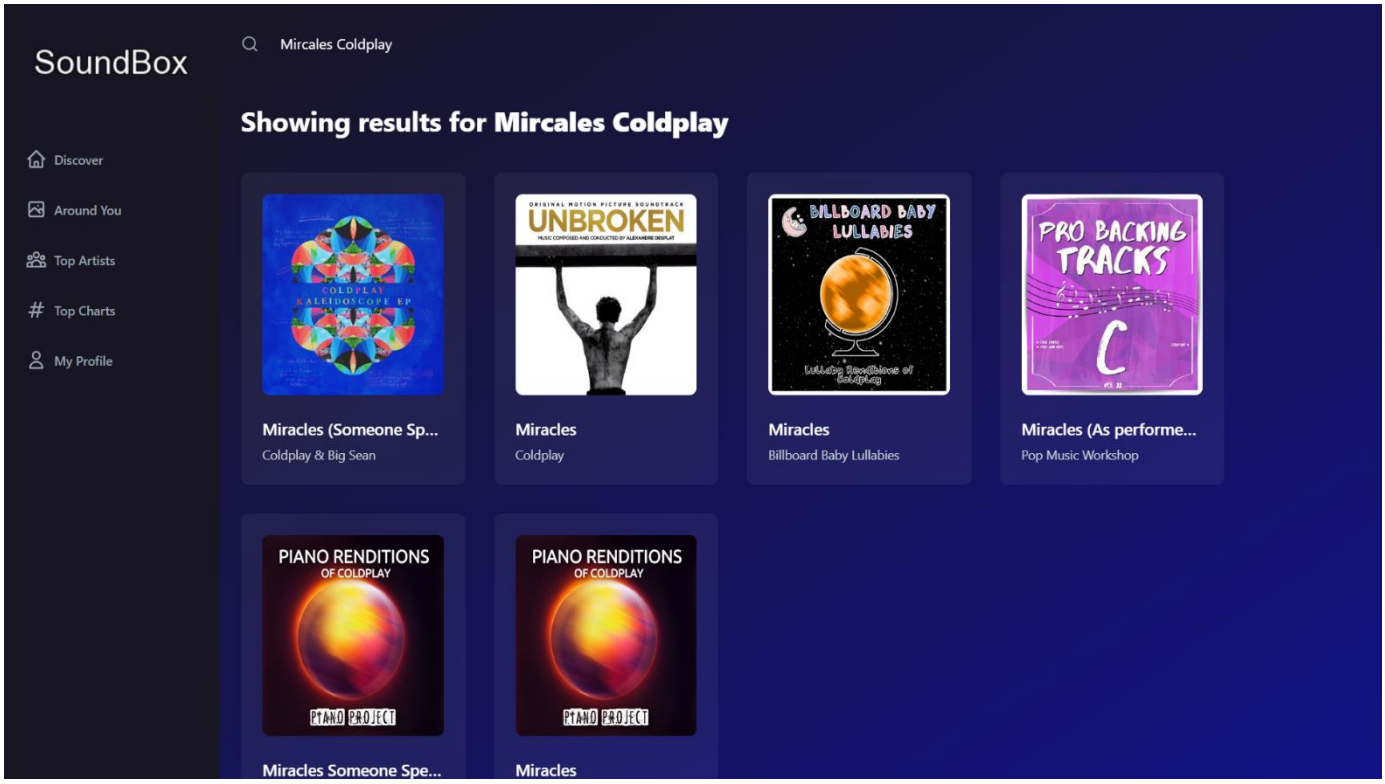
Top Charts Page :



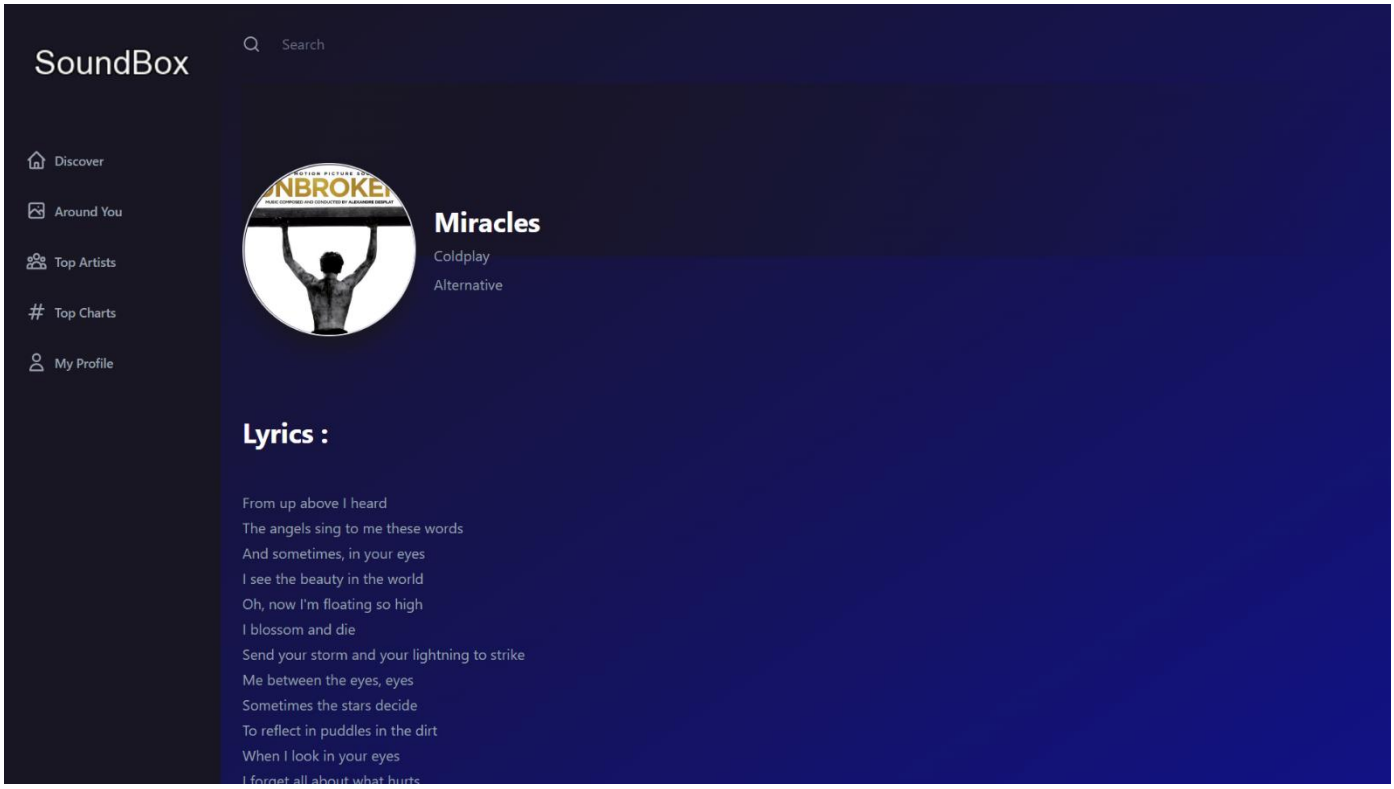
My Profile Page :



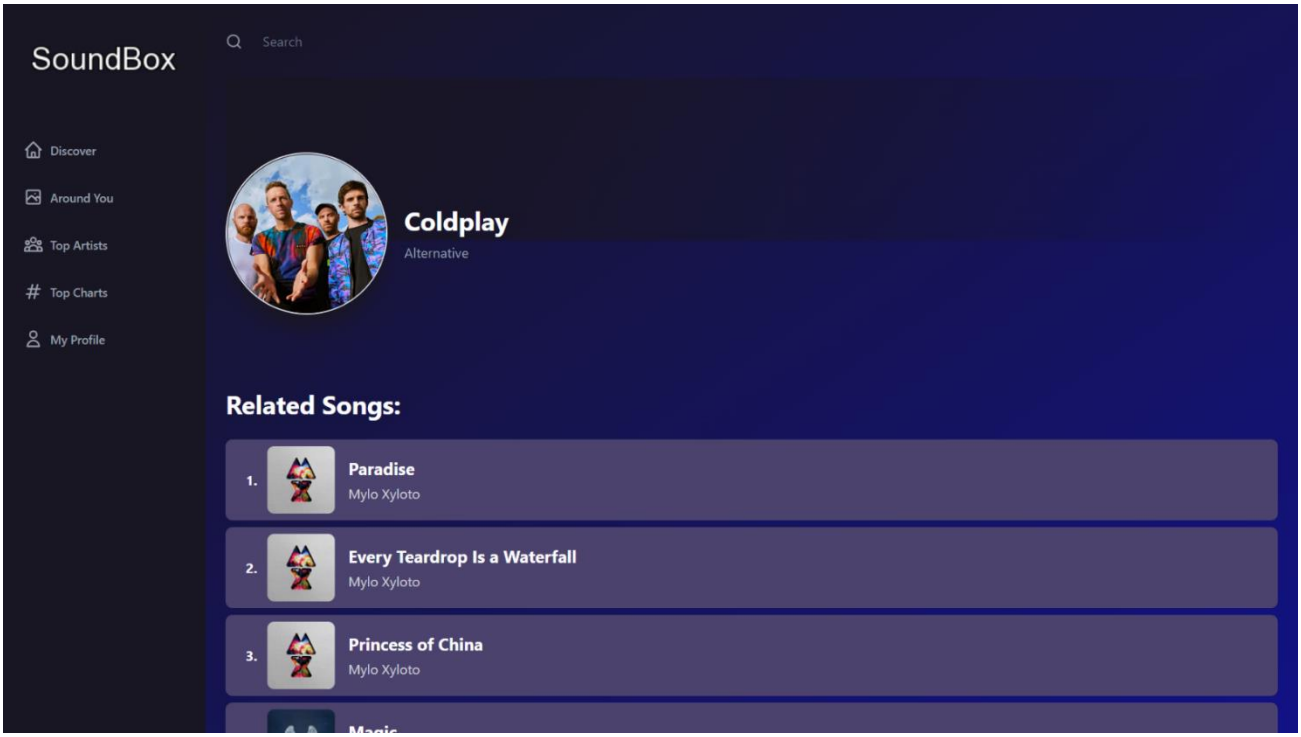
Search Result :



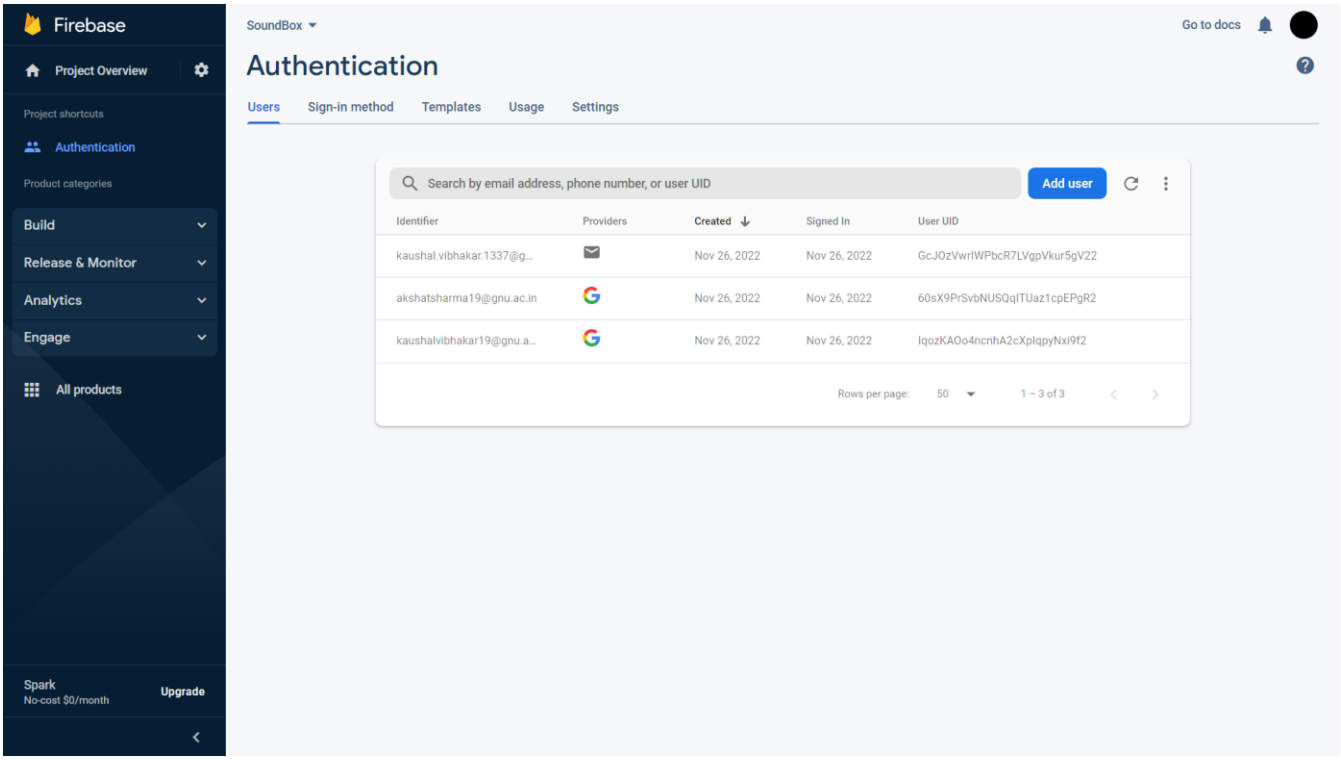
Song Details and Lyrics Page :



Artist's Profile Page :



Firebase Database :



9. Testing

9.1 Testing Plan :

The objective is to test the functionality of the 'SoundBox' Web application. Testing is done on both front end and back end of the application on the Windows environments. The features, which are to be tested, are Login Page, Signup Page and Home Page. All the major functionality of the application should work as intended and the pass percentage of test cases should be more than 95% and there should not be any critical bugs.

9.2 Test Cases :

The following are the test cases for all modules :

Login Module :

Id	Test_Senario	Test_Steps	Test_Data	Expected_Results	Actual_Results	Pass /Fail
1	Login with Valid Data	• Enter Email • Enter Password • Click On Login Button	• Valid Email Id • Valid Password	User Should Login into the Application	As Expected	Pass
2	Login with Valid Data	• Enter Email • Enter Password • Click On Login Button	• Valid Email Id • Invalid Password	User Should Login into the Application	Incorrect Password	Fail
3	Login with Valid Data	• Enter Email • Enter Password • Click On Login Button	• Invalid Email Id • Valid Password	User Should Login into the Application	Incorrect Email Id	Fail
4	Login with Valid Data	• Enter Email • Enter Password • Click On Login Button	• Invalid Email Id • Invalid Password	User Should Login into the Application	Incorrect Email id and Password	Fail

(Table 9.2.1)

Signup Module :

Id	Test_Senario	Test_Steps	Test_Data	Expected_Results	Actual_Results	Pass /Fail
1	Signup with Valid Data	• Enter Email id • Enter Password • Enter Confirm Password • Click On Sign up Button	• Filled Email • Filled Password • Filled Confirm Password	User Should Login into the Application	As Expected	Pass
2	Signup with Valid Data	• Enter Email id • Enter Password • Enter Confirm Password • Click On Signup Button	• Not filled Email • Filled Password • Filled Confirm Password	User Should Login into the Application	Empty field Email Id	Fail
3	Signup with Valid Data	• Enter Email id • Enter Password • Enter Confirm Password • Click On Signup Button	• Filled Email • Not filled Password • Filled Confirm Password	User Should Login into the Application	Empty field Password	Fail
4	Signup with Valid Data	• Enter Name • Enter Email • Enter Password • Click On Submit Button	• Filled Email • Filled Password • Not Filled Confirm Password	User Should Login into the Application	Empty field Confirm Password	Fail

(Table 9.2.2)

10. Conclusion & Future Scope

10.1 Conclusion :

Through the development of music player on Web platform, a clear understanding of overall process of the system is obtained.

The core part of the music player is mainly composed of main interface, file browsing and song listing, Grasping the development of the music player has had the preliminary scale small features.

Music player system realized the basic function of player : play, pause, rewind and fastforward a, volume adjustment is performed through the web player Itself, play mode, song search, seekbar, This development implicated the popular web development technology.

This is the combination management of react language in the open source web platform based on windows system configuration file. The system realized the music player programming.

10.2 Future Scope :

- Users can make playlists and add their desired songs.
- Users can like the songs.
- Users can search other users and view their public playlists.

Annexure

References :

- **React –** https://reactjs.org/docs/getting-started.html
- **Chakra UI** - https://chakra-ui.com/
- **React JS Tutorials** - https://www.javatpoint.com/reactjs-tutorial
- **Firebase** - https://firebase.google.com/
- **Spotify's Web Player** - https://open.spotify.com/

About Tools and Technology :

Information	Image
Visual Studio Code	Visual Studio Code
React JS	React JS
Tailwind CSS	Tailwind CSS
Firebase	Firebase

(Table 11.2)

About College :

U. V. Patel College of engineering (UVPCE) situated in Ganpat Vidyanagar campus was established in septmber-1997 under the aegis of Mehsana District Education Foundation with a view of educating and training young talented students of Gujarat at the field of engineering and technology to meet the needs of industries in Gujarat and beyond for the growth of the industries.

The College is named after Shri Ugarchandbhai Varanasibhai Patel, a leading industrialist of Gujarat, for his generous support. It is a self-financed institute approved by All India Council for Technical Education (AICTE), New Delhi, the Government of Gujarat and now it became the constituent college of Ganpat University.

The College is spread over 25 acres of land and is a part of Ganpat Vidyanagar Campus. It has two ultra-modern buildings of architectural splendor measuring 6100 sqm. and 2700 sqm., for housing class rooms, tutorial rooms, seminar hall, offices, drawing hall, workshop, library, well equipped different departmental laboratories, several computer labs with internet connectivity through 1 Gbps Fiber link, satellite link education center with two-way audio and one-way video link with Gandhinagar etc.

Placement plays a key role in shaping the future of the students, and keeping this in mind; the institute has forged healthy relations with the prominent industries. These tie-ups are mutually beneficial. The industries get a chance to employ the resources of the institute for their R & D. In turn they extend every possible help to the institute especially with regard to providing hands on training to the students. As part of this initiative, Incubation Centre/Startup activities have also been developed.