Intelligent Radio Jockey

(I.R.J)

By

Muhammad Fahad Akbar

Usama Bin Sabir

Rai Atif Ali

Bilal Nazir

Supervised by:

Dr. Naima Iltaf

Submitted to the faculty of Department of Computer Software Engineering,
Military College of Signals, National University of Sciences and Technology, Islamabad,
in partial fulfillment for the requirements of B.E Degree in Software Engineering.

June 2022

In the name of ALLAH, the Most benevolent, the Most Courteous

CERTIFICATE OF CORRECTNESS AND APPROVAL

This is to officially state that the thesis work contained in this report

"Intelligent Radio Jockey"

is carried out by

Muhammad Fahad Akbar

Usama Bin Sabir

Rai Atif Ali

Bilal Nazir

under my supervision and that in my judgement, it is fully ample, in scope and excellence, for the degree of Bachelor of Software Engineering in Military College of Signals, National University of Sciences and Technology (NUST), Islamabad.

Approved by

Supervisor

Dr. Naima Altaf

Date: _____

DECLARATION OF ORIGINALITY

We hereby declare that no portion of work presented in this thesis has been submitted in support of another award or qualification in either this institute or anywhere else.

ACKNOWLEDGEMENTS

Allah Subhan'Wa'Tala is the sole guidance in all domains.

Our parents, colleagues, and most of all supervisor, Dr. Naima Altaf without your guidance.

The group members, who through all adversities worked steadfastly.

Plagiarism Certificate (Turnitin Report)

This thesis has 10% similarity index. Turnitin report endorsed by Supervisor is attached.

RAI ATIF ALI

NUST Serial no

00000281036

MUHAMMAD FAHAD AKBAR

NUST Serial no

00000278776

USAMA BIN SABIR

NUST Serial no

00000278781

BILAL NAZIR

NUST Serial no

00000278790

_____

Signature of Supervisor

Dedication

Dedicated to our cherished parents and loved siblings whose never-ending support and cooperation drove us to this magnificent achievement.

And to our supervisor, Dr. Naima Iltaf who has given us great support and valuable suggestions throughout the journey of this project.

ABSTRACT

In today's world we all are extremely busy with our lives and have a lot of work to do. Since we don't have time, we are unable to multitask. So, we have provided a platform or an application through which users can listen to the songs of their interests as well as they can be informed about their interest after every two or three songs. In this project, we built an Intelligent Radio Jockey which recommends songs and news based on user's choice and likings. Users can search a song and select choices for news such as tech, gaming, and entertainment. Based on entered data, the user will be displayed a playlist which contains recommended songs as well as information in such a way that information will be conveyed to the user after 2 more songs. We have used collaborative filtering technique using subset of million songs dataset for songs recommendation. Frontend of our application will be built in React. We have used google news api to fetch news related to user interest.

Table of Contents

List of Figures

Chapter 1: Introduction

We live in a period of time, where the popularity of digital content streaming services is constantly on the rise. This trend is even more evident at the field of music streaming. Major providers like Spotify, Pandora or Apple Music claim to have tens of millions of active users and those numbers are constantly increasing. Most people prefer music streaming services over radio stations because they can conveniently choose what songs they want to listen to and this process is much more comfortable than having a music collection stored on CDs. Another advantage is that the user can listen to the information of his/her interest. All significant streaming service providers have collections containing tens of millions of songs. On the other hand the user can not listen to the news/information related to his/her interest because tv channels are broadcasting news of all niches. It is, however, very difficult for users to find relevant content in such an inexhaustible quantity of items. Because of that, providers try to offer convenient and personalized ways to discover music. This issue is mainly solved by recommender systems that suggest only such content that is likely to be considered useful by users. A predominant technique used in modern recommender systems is collaborative filtering. The basic assumption of this approach is that there are groups of users with similar taste and listening behavior. Subsequently, there is high a probability that a certain user will like a content that is popular amongst a group to where he belongs. Recommender systems which use collaborative filtering usually provide a high accuracy of suggestions and they can be easily deployed no matter what type of content is offered. That is the reason why they are used in most systems that generate music recommendations. An important feature of collaborative filtering (in its basic form) is that it needs no other information than users' ratings of items. However, streaming services usually record and store many additional data about musical works and users. These pieces of information can be used to

improve the quality of suggestions. Most companies try to utilize them but this is not a straightforward task and a lot of research is needed concerning this topic.

1.1 Overview

The system consists of a news recommender system and a music recommender system. First of all, we will get the user interests about songs and news and then when a user searches a song we will give song recommendations and news recommendation will be according to the choices which he/she select at the beginning. Our application is going to be the web application in which a user will be listening to songs and after 2 or 3 songs an information will be displayed and played.
It is the need of the hour to fashion some policy or mechanism to avoid such huge traffic jams. Hence in our proposed system of traffic control, we not just focus on the signal timings but also encounter the traffic density to get the best results.

1.2 Problem Statement

Music is a fundamental piece of human existence. Music is the wonderful sound that drives us to have peace and higher satisfaction while data keeps us informed pretty much every one of the occasions out there. With the headway in innovation, music has significantly advanced and expanded regarding quality and volume. The kind of music individuals makes and listen varies as per spot and culture. The flavor of music even contrasts from one individual to another and, surprisingly, in states of mind of same individual. Thus, it is exceptionally helpful if we would decide a workable technique to find what sort of music an individual may be keen on tuning in and utilize this finding to prescribe music to him/her. Cooperative filtering is one the most famous filtering methods today and with this task we expect to upgrade it. For this, we will gather information of clients which stand by listening to same sort of music and will suggest them those

tunes which other with a similar taste are tuning in. Then again, we in this bustling world don't have a lot of chance to pay attention to what's going on out there on the planet and skim that data as per our advantage. Subsequently, by means of this undertaking, we will actually want to keep our clients engaged and informed simultaneously.

1.3 Proposed Solution

The major goal of our proposed solution is to gather the data of those users who have the same taste in music and listen to same kind of songs / genre / artists and then recommend songs on the bases of this collected data and in order to keep our users informed we will be displaying and playing news between two or three songs. So, anyone using our app will be entertained and be informed at the same time.

1.4 Working Principle

The project mainly works on K-nearest-neighbor (KNN) machine learning algorithms. The project is divided into different modulus and every module is inter-woven with the next module. The list of modules is as under:

- Datasets and annotations
- Dataset training and processing
- Playlist extraction
- Displaying Playlist
- Getting User Choices for information
- Displaying Recommended Information

- Playing Information (Audio)

1.4.1 Datasets (Million Song Dataset):

The million-tune informational index is an open source assortment of sound highlights and metadata for 1,000,000 present-day well-known music tracks.

Its Purposes are:

To support research on algorithms that scale to business sizes.

To provide a reference dataset for assessing research.

As an easy option in contrast to creating a large dataset with APIs (e.g., The Echo Nest's).

To assist new analysts with beginning in the MIR field.

The center of the dataset is the element investigation and metadata for 1,000,000 melodies, given by The Echo Nest. The dataset incorporates no sound, just the determined elements. Note, notwithstanding, that example sound can be gotten from administrations like 7digital, utilizing code we give.

The Million Song Dataset is also a likewise a group of integral datasets contributed by the community.

Secondhand Songs dataset -> cover songs

Music match dataset -> lyrics

Last.fm dataset -> song-level tags and similarity

Taste Profile subset -> user data

This is my jam-to-MSD mapping -> more user data

Top MAGD dataset -> more genre labels

1.5 Objectives

1.5.1 General Objectives:

The goals of the project can be summed up with the points underneath:

1. Finding the users with common interest for songs recommendation system.

2. To find out if collaborative recommendation system can be upgraded using other techniques.

3. To provide users with the information of their interests.

1.5.2 Academic Objectives:

- Development of a smart and intelligent radio jockey.
- To implement Machine Learning techniques and simulate the results
- To increase productivity by working in a team
- To design a project that contributes to the welfare of society

1.6 Scope

- This project is aimed at recommending songs.
- IRJ will also recommend information/conversation of user's choices.
- Recommendations will be made based on interests and needs.
- Providing a user-friendly interface.

1.7 System Features

1.7.1 Song Recommender

Description

System will recommend song to user according to his/her interests and choices.

Action/ response Sequences

Action: User starts the application.

Response: User is redirected to Home Page.

Action: User will give his interest and choice for song recommendation.

Response: System applies recommendation algorithm and create playlist of recommended songs.

Functional Requirements

R1: Users should provide his interest and choice.

R2: System applies recommendation algorithm and create playlist of recommended songs.

R3: For new users, recommendations are made on the basis of his/her interest and choice.

1.7.2 Information Recommender

Description

System will recommend information to user according to his/her interests and choices.

Action/ response Sequences

Action: User starts the application.

Response: User is redirected to Home Page.

Action: User will give his interest and choice for information recommendation.

Response: System applies recommendation algorithm and play information of user interest between the gap of two to three songs.

Functional Requirements

R1: Users should provide his interest and choice.

R2: After two or more songs, the information of user's interest will be played. The playlist for the conversation will be generated on the basis of user's fields of interest like gaming, health and politics etc. User will be asked to select his/her interests when he/she registers on the application.

1.8. Nonfunctional Requirements

1.8.1 Performance Requirements

The information/conversation will be played after one or two songs and information will be 40 to 60 seconds.

1.8.2 Safety Requirements

The system should give accurate recommendations based on users' choices and interests. The user session is protected, and credentials are preserved without being changed in the database.

1.8.3 Security Requirements

Access permissions for the database system information may only be changed by the system's data administrator.

1.8.4 Software Quality Attributes

- **Availability:** The Intelligent Radio Jockey must be available any time users want to access it.
- **Correctness:** Both the songs and the conversation played must be according to the interest of the user.
- **Flexibility:** Since the innovation generally changes in the field of technology, new features will be added to the project and existing elements will be worked on because of the progressions on the lookout.
- **Reliability:** Project should be errors and bugs free. Since we live in reality the failures in the project ought to be uncommon.
- **User Friendly Interface:** Although the target audience is equipped for understanding and utilizing technology productively, the user interface must be easy to use. A user-friendly

interface will prevent user loss caused by a non-accommodating UI which will cause not becoming accustomed to utilizing the Intelligent Radio Jockey.

1.9 Deliverables

1.9.1 Music Recommender

It serves as music recommendation engine for Intelligent Radio Jockey using machine learning techniques like K-nearest-neighbor and Million Song Dataset.

1.9.2 Information Recommender:

It will be getting choices from user as input and based on those choices it will be recommending news to the user.

1.10 Relevant Sustainable Development Goals (innovation)

The relevant sustainable development goals covered by our project are: **Industry, innovation and infrastructure.**

Our Intelligent Radio Jockey (IRJ) will be innovation in the field of an entertainment and songs listening experience.

1.11 Structure of Thesis

Chapter 2 contains the literature review and the background and analysis study this thesis is based upon.

Chapter 3 contains the design and development of the project.

Chapter 4 introduces detailed evaluation and analysis of the code.

Chapter 5 contains the conclusion of the project.

Chapter 6 highlights the future work needed to be done for the commercialization of this project.

Chapter 2: Literature Review

A new product is launched by modifying and enhancing the features of previously launched similar products. Literature review is an important step for development of an idea to a new product. Likewise, for the development of a product, a detailed study regarding all similar projects is compulsory. Our research is divided into the following points.

- Industrial Background
- Existing solutions and their drawbacks
- Research Papers

2.1 Industrial background

Music industry is booming as Music is the charming sound that drives us to have peace and higher joy while information keeps us informed about all the events out there. Different kind of people listen to different type of music based on their taste culture and liking. Thus, it is extremely helpful if we would decide a good strategy to find what sort of music an individual may be keen on tuning in and utilize this finding to prescribe music to him/her. Cooperative filtering is one the most famous filtering methods today and with this venture we intend to improve it. For this, we are going to collect data of users which listen to same kind of music and will recommend them those songs which other with the same taste are listening. On the other hand, we in this busy world don't have much time to listen to what is happening out there in the world and skim that information according to our interest. Hence, via this project, we will be able to keep our users entertained and informed at the same time.

And now industries are inclining towards smart industries, automation, based on new technologies (Machine Learning (ML) techniques, Artificial Intelligence (AI)). Hence, Intelligent Radio Jokey provides good market growth and impacts economy directly as it is a fully automated.

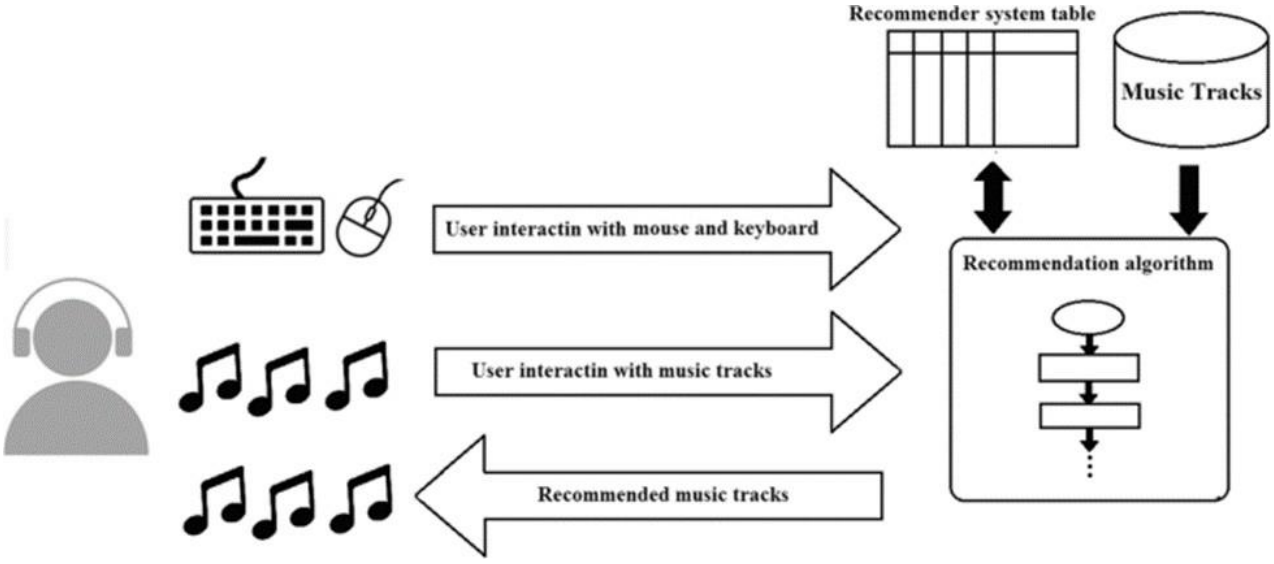
2.2 Existing solutions and their drawbacks

Different solutions are previously being provided for the traffic congestion problem, but every product has some pros and cons. Following are some solutions which are already being prepared and being implemented.

- **Spotify** (It provides user song listening experience, but the user can not listen to some interesting news of their interest along with song listening)
- **YouTube** (Users can watch videos and listen to songs on this platform but it also lacks the Simultaneous song and news listening experience)

2.2.2 MUSIC RECOMMENDATION TEQUNIQE

Song will be recommended to users using collaborative filtering techniques which will be achieved through K nearest neighbor machine learning algorithm and using the subset of million song dataset



Chapter 3: Interfacing and Detection

3.1 TASTE DETECTION

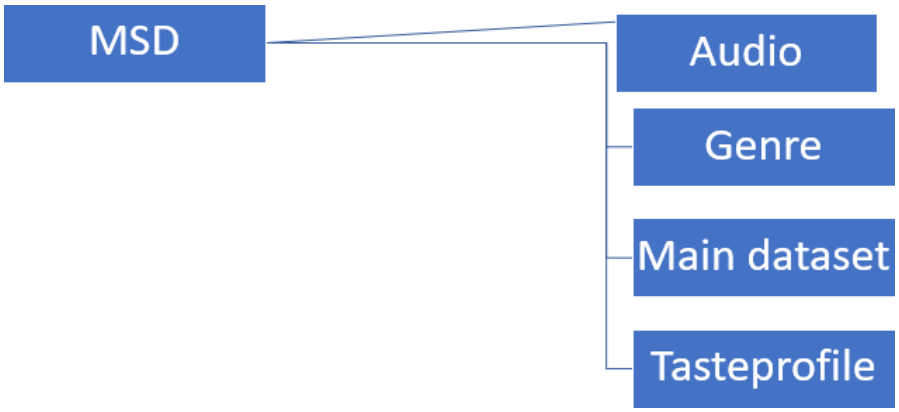
To Detect the taste of the user our major goal is to gather the data of those users which has the same taste in music and listen to same kind of songs/genre/artist and then recommend songs on the base of this collected data.

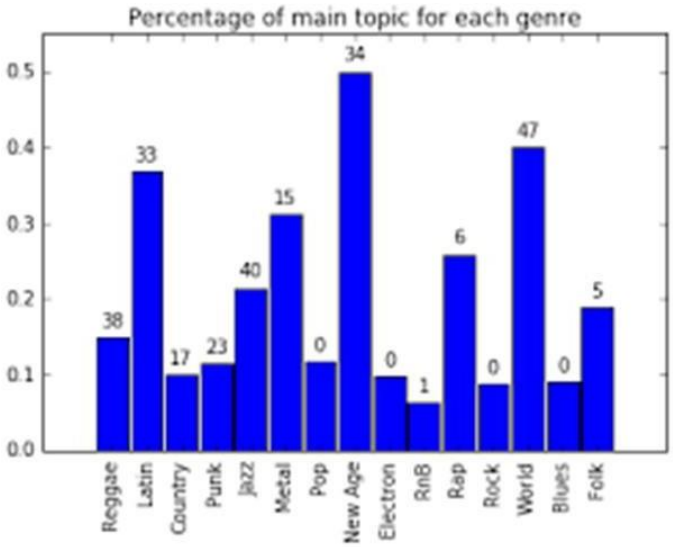
3.1.1 Preparing Dataset

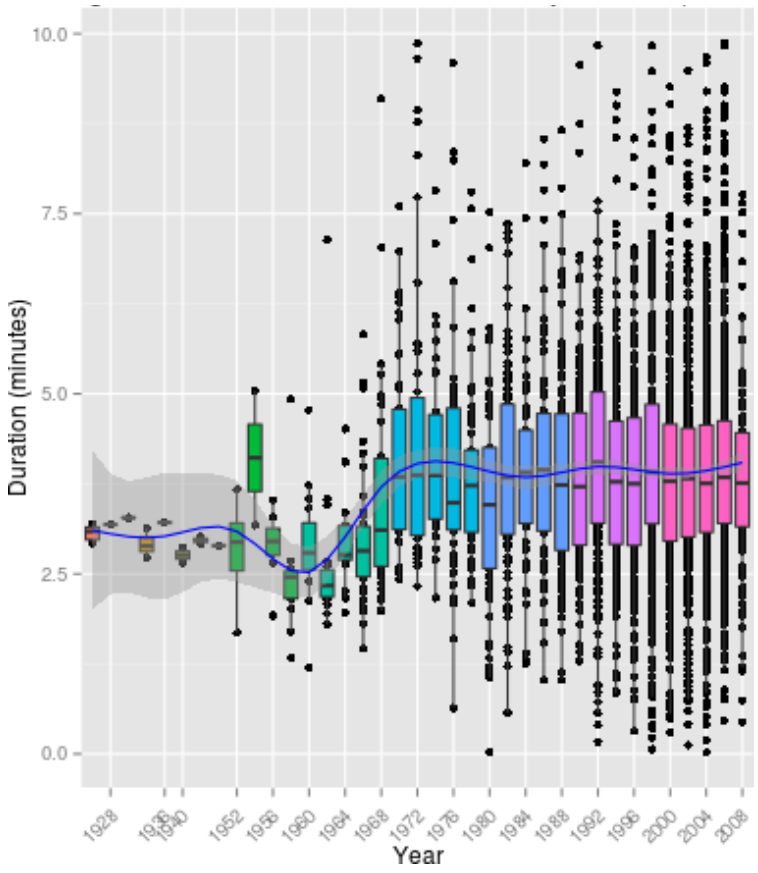
Dataset refers to a compilation of instances that share a mutual attribute. Dataset is used to sculpt a machine learning algorithm going forward. The more data is provided, the better is the efficiency. In this model we used about 10,000 songs of different genera like pop, rock, jazz, hip-hop etc.

Figure 6-a: Proposed Dataset

To recommend songs, million song Dataset is used.

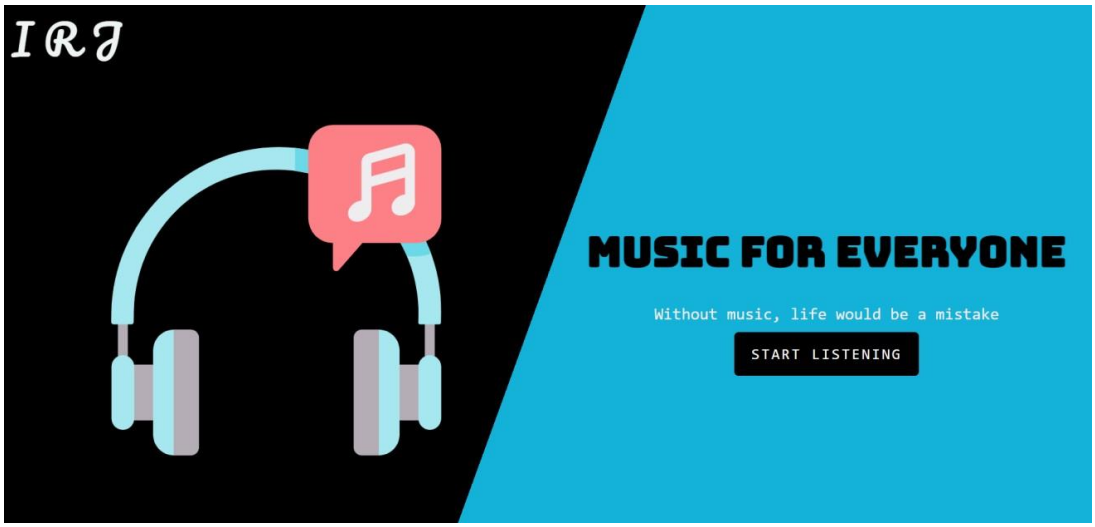




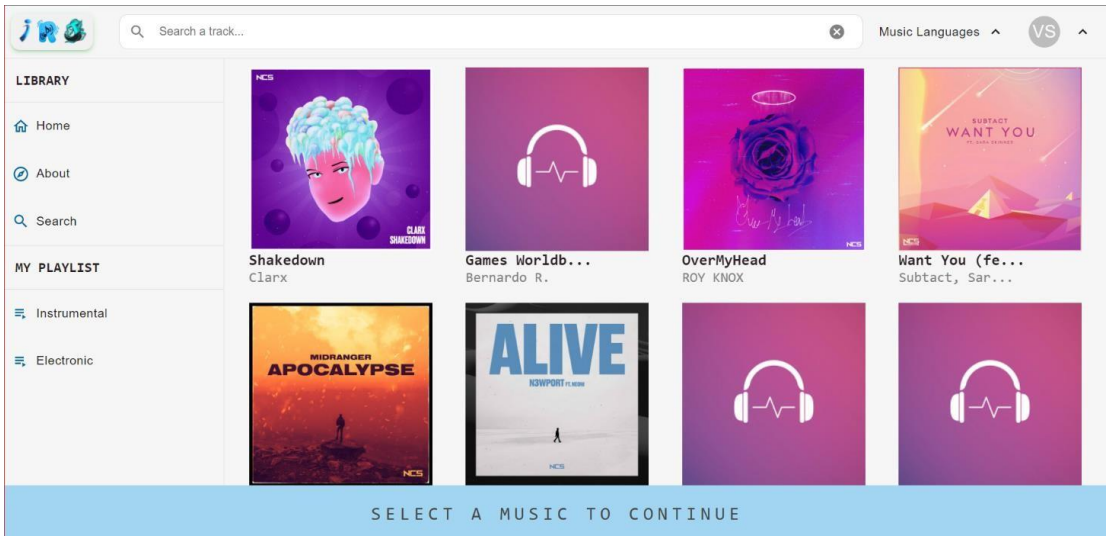

10.0
7.5
5.0
2.5
0.0
Duration (minutes)
1928
1936
1940
1952
1956
1960
1964
1968
1972
1976
1980
1984
1988
1992
1996
2000
2004
2008
Year

3.1OUTPUT SHOWN ON GUI:

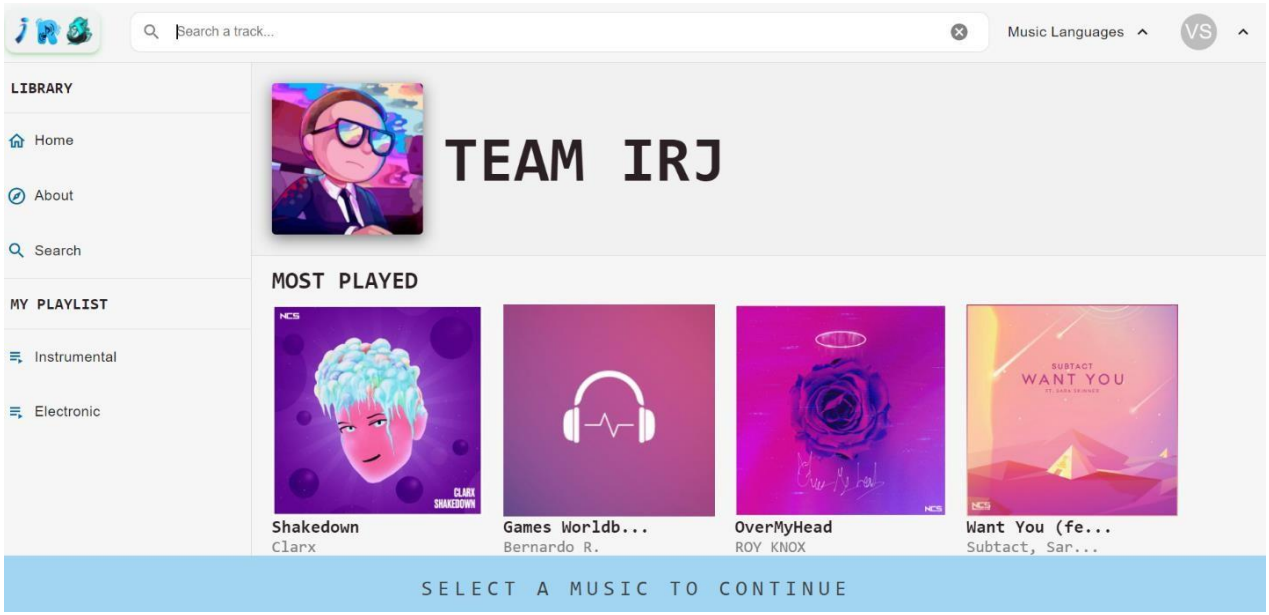
3.1.1 Start:



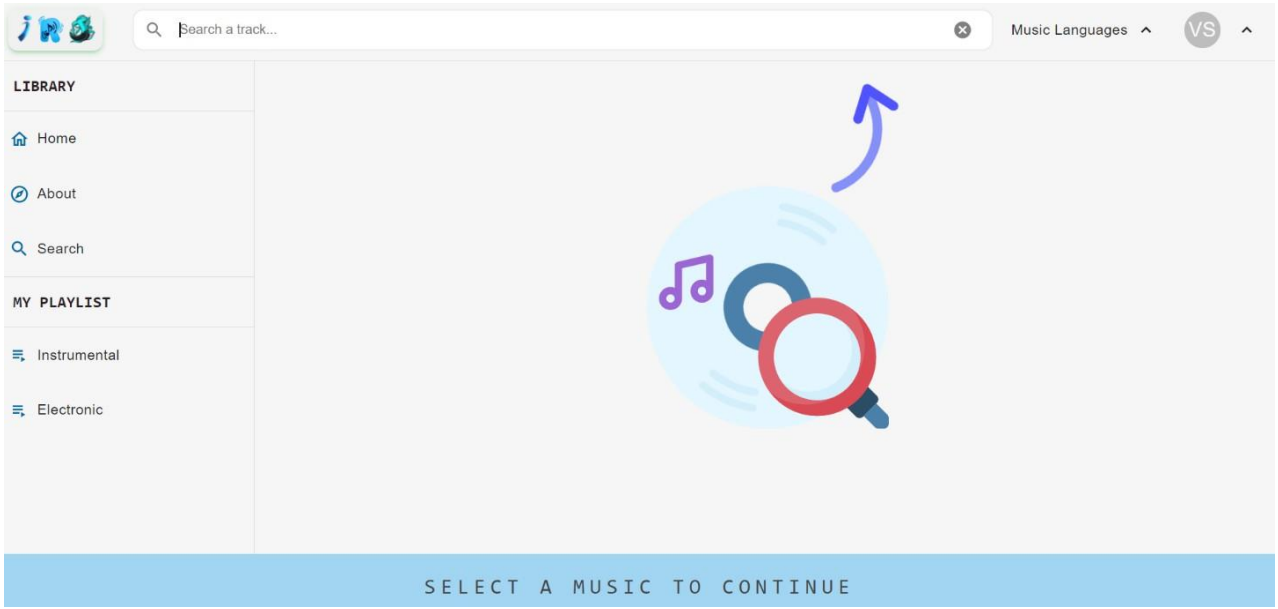
3.1.2 Home Page



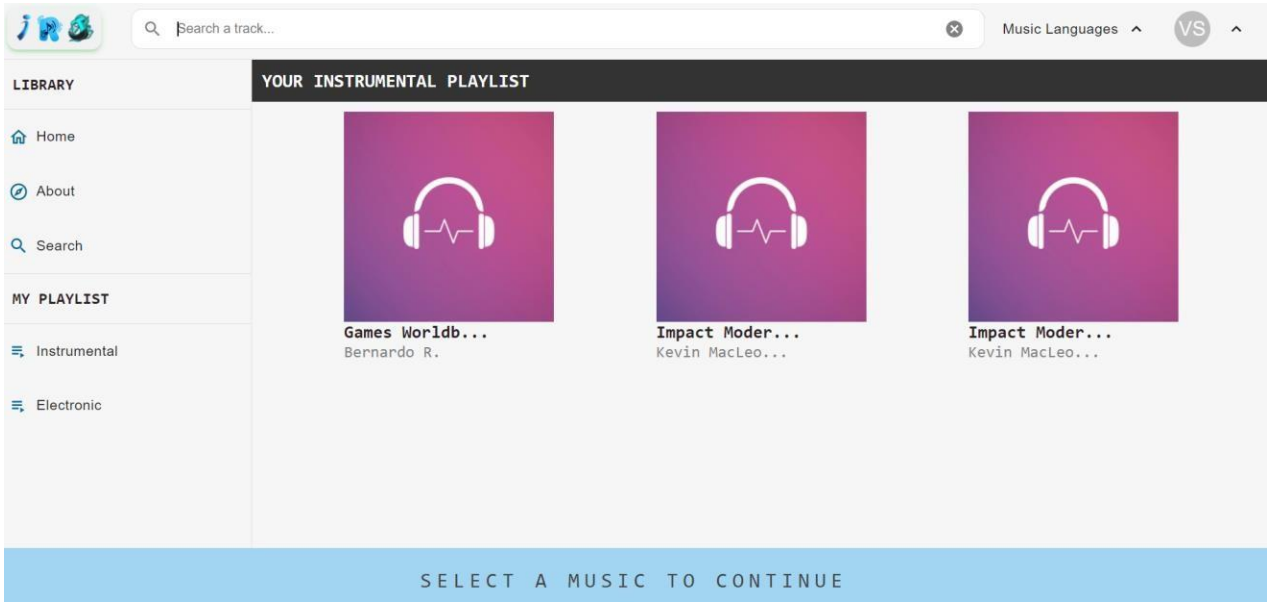
3.1.3 Profile

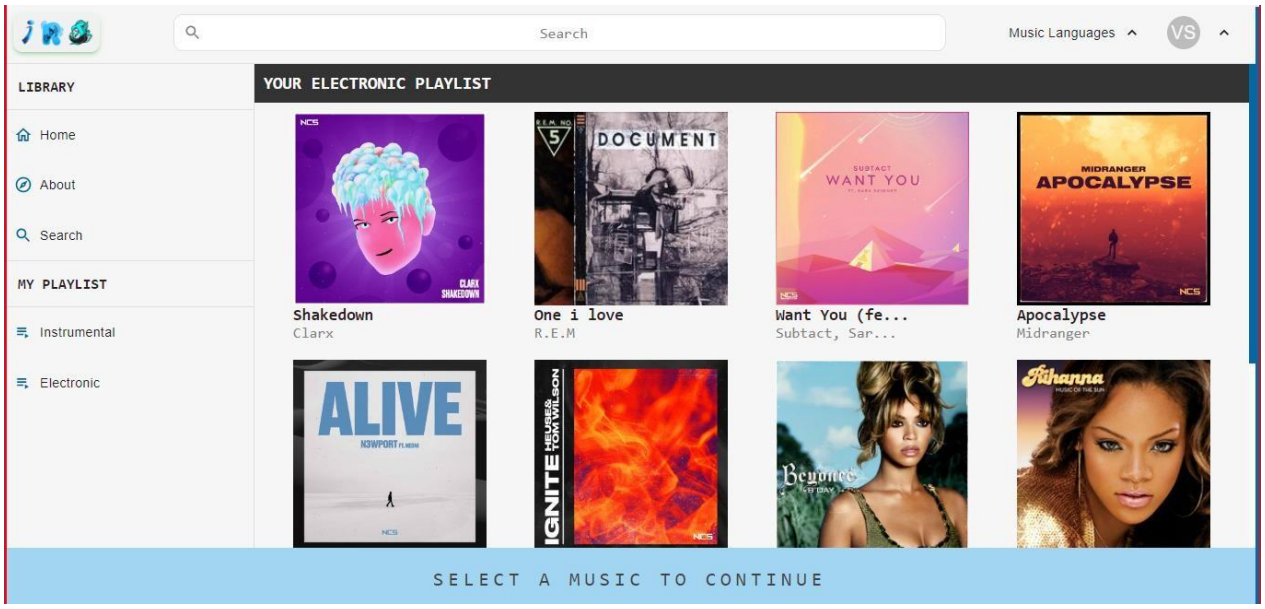


3.1.4 Search Menu

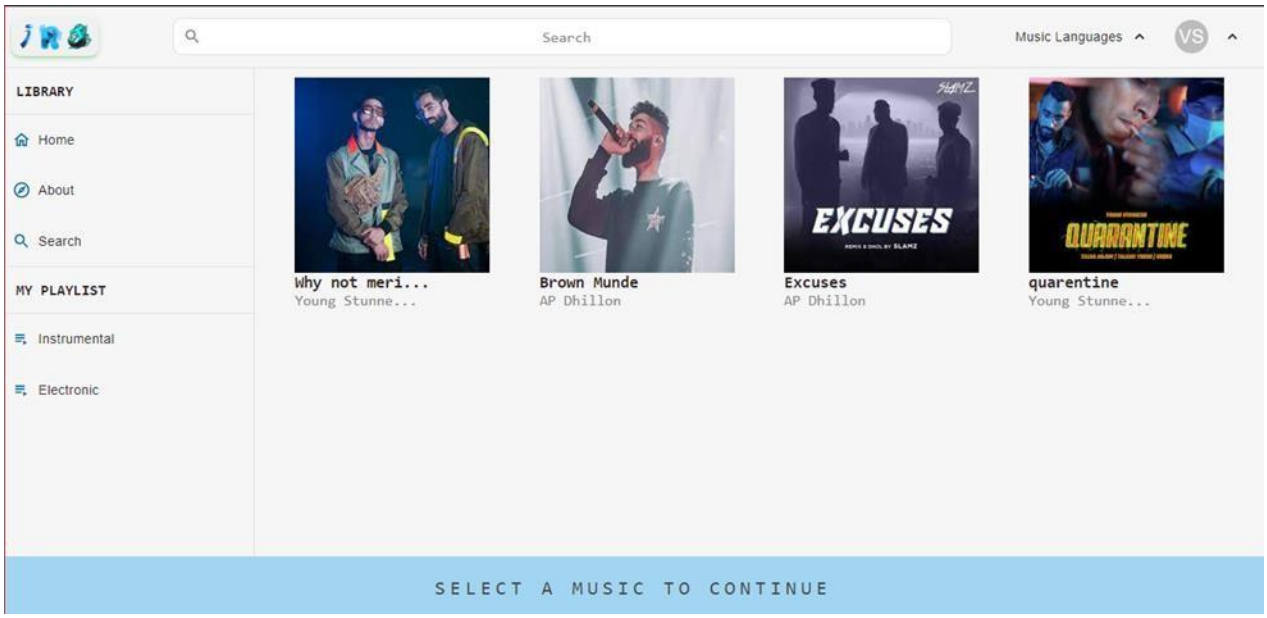


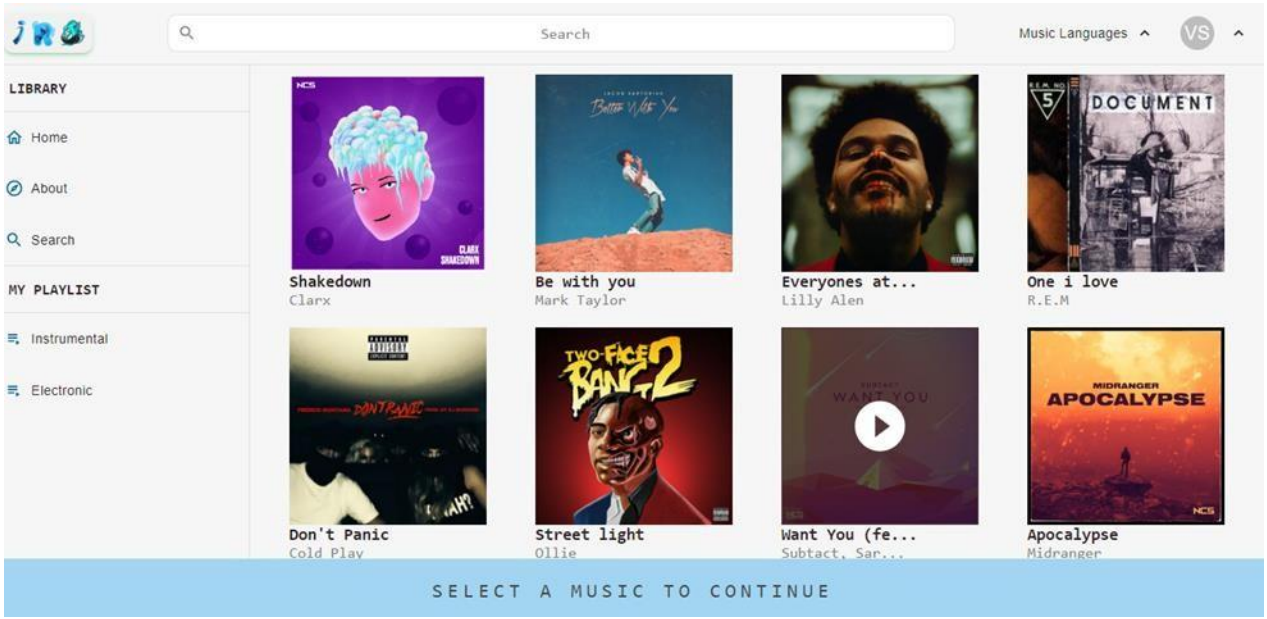
3.1.5 Playlists





3.1.6 Songs by Languages





3.2.1 Ways to Create GUI:

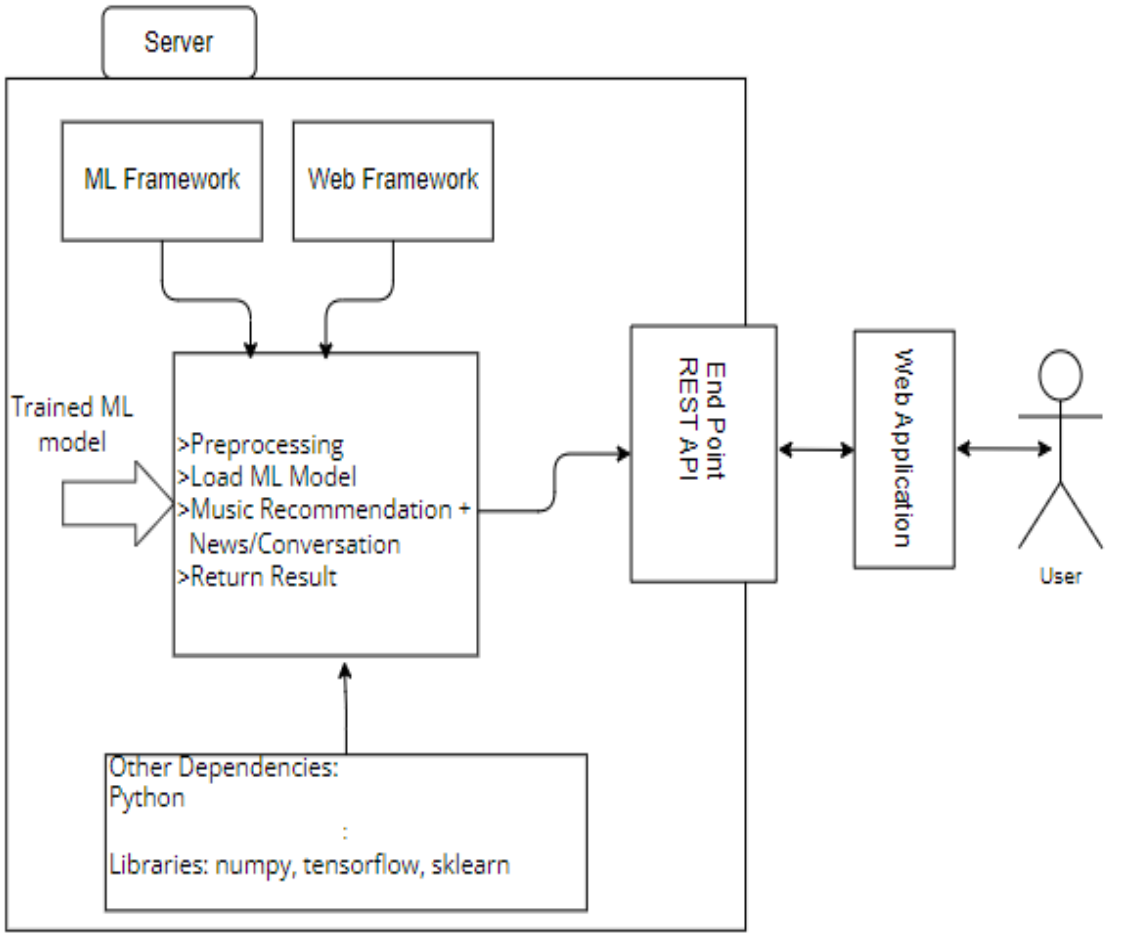
There are different ways to create the GUI of our app. We could have made using Python. But as we have learned a web development course, so we collectively decided to use and practice those technologies which we learned in the past. Why did we make this decision? because we need to get an on-hand experience with those technologies. So, we made our GUI using HTML, CSS, Bootstrap, JavaScript, and ReactJS.

3.2 DECISION MAKING:

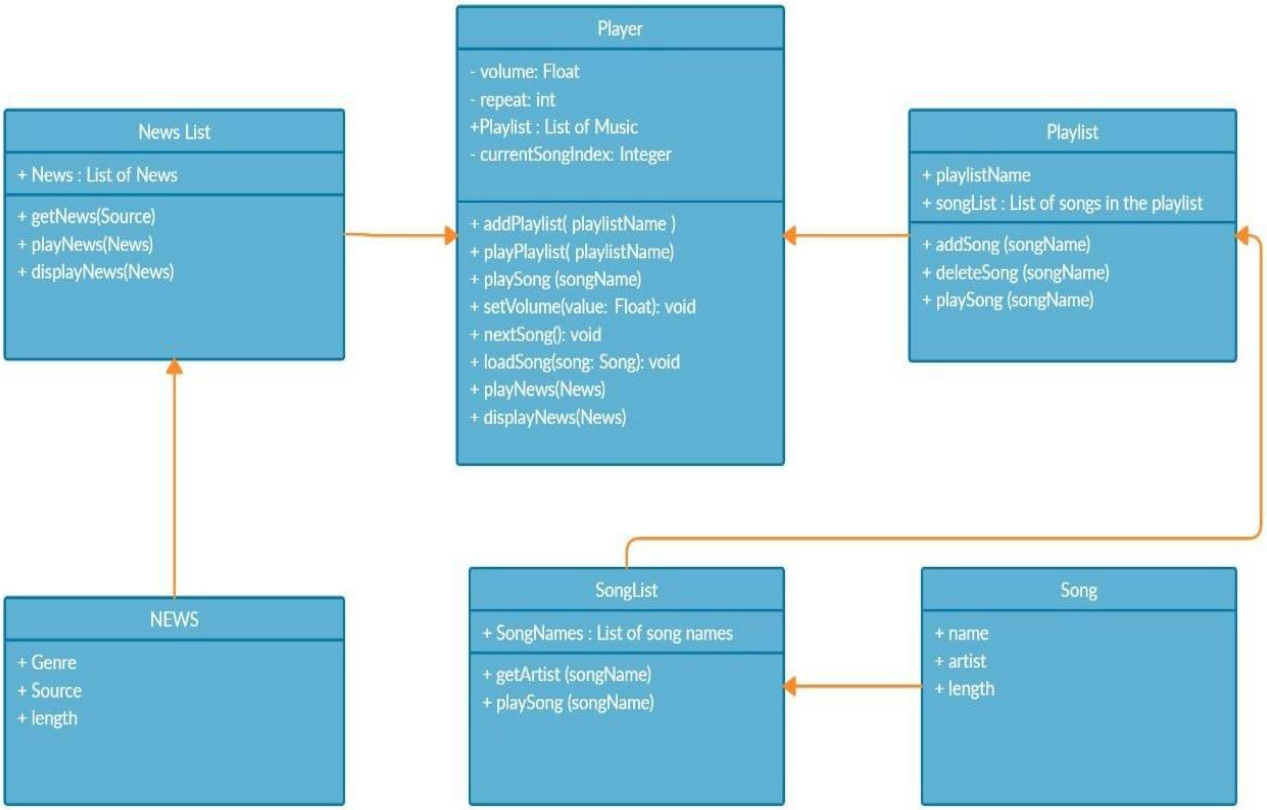
3.2.1 Working of GUI:

After launching the application, the user will be asked for songs preferences like artists or genre and information preferences like sports, politics, etc., and then a user will be redirected to the page where he/she can listen to the songs, and once a user search for any song our model will run at the backend and will make a playlist for him/her, and that playlist will be shown to the user. After two or three songs information will be played in the form of audio and will be shown to the user.

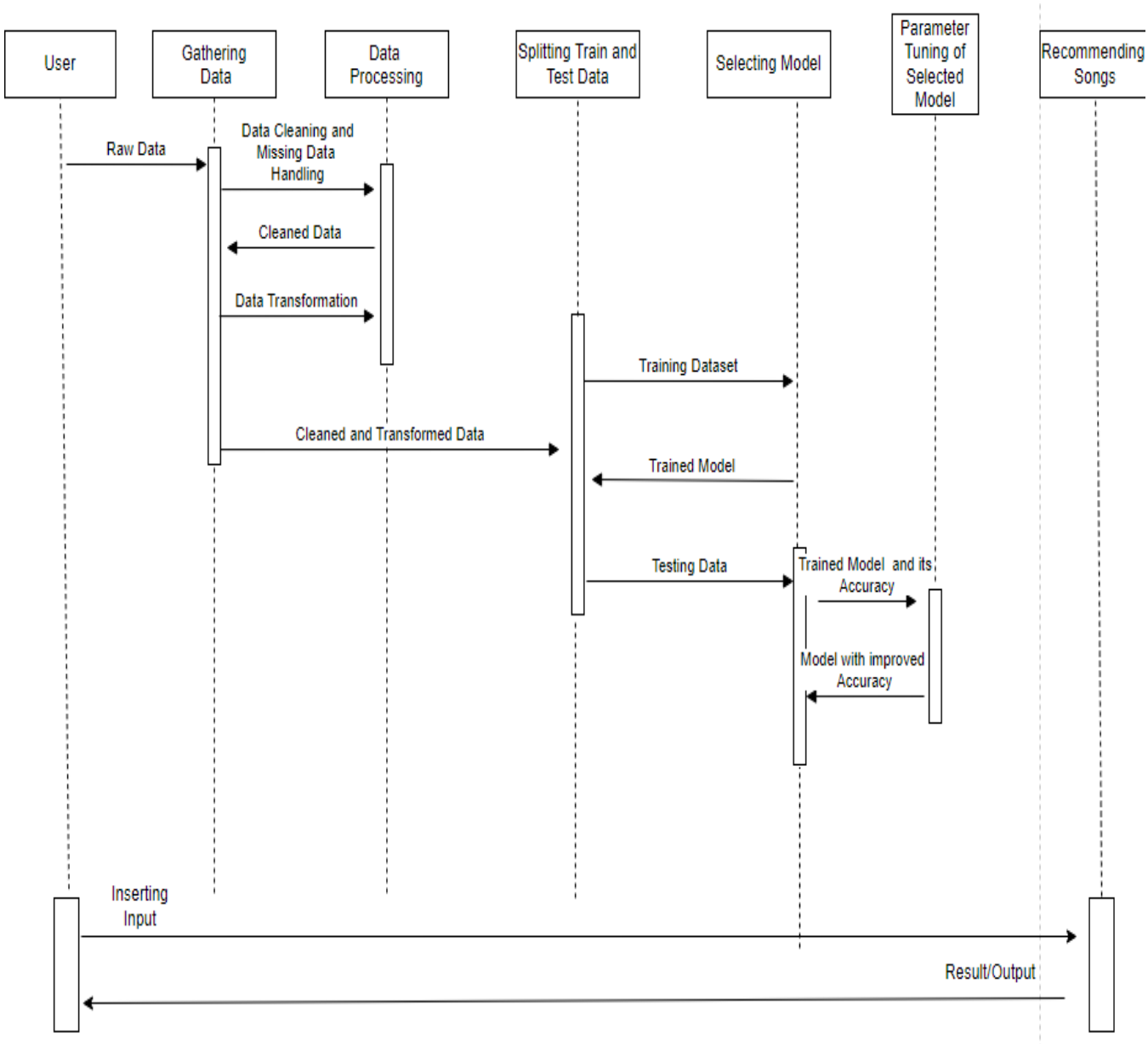
3.3 PROPOSED BLOCK DIAGRAM:



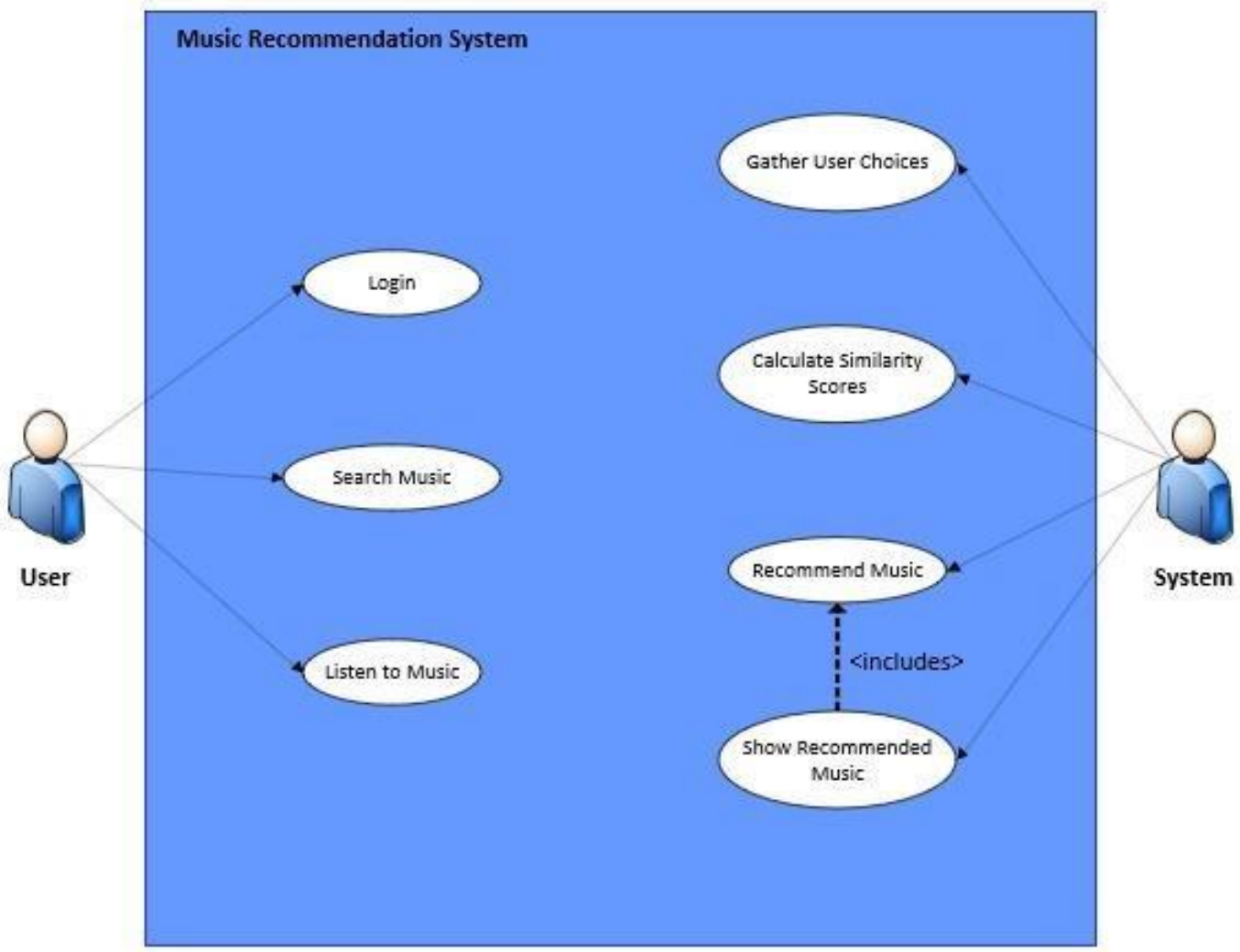
3.4 PROPOSED CLASS DIAGRAM:

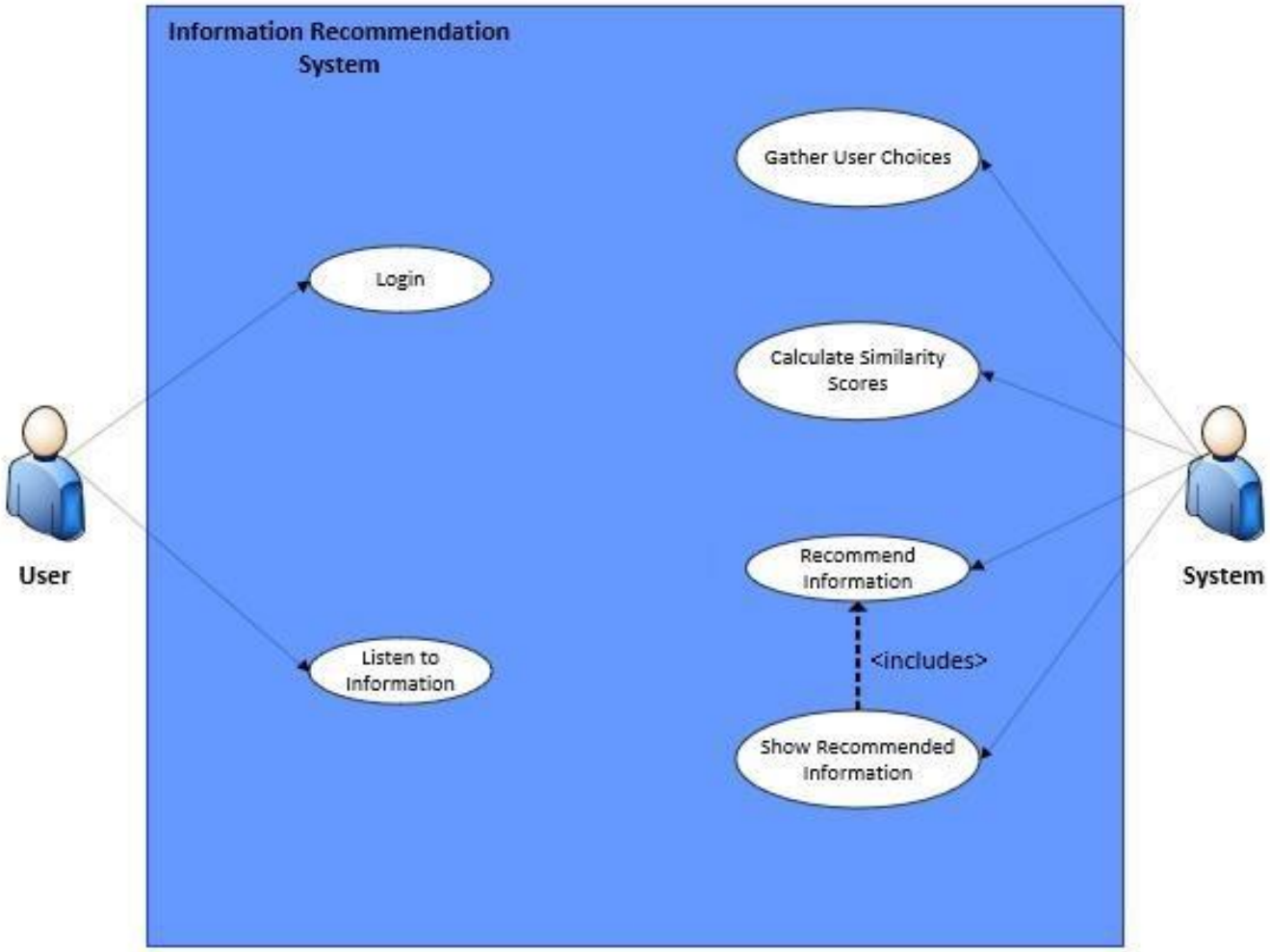


3.5 PROPOSED SEQUENCE DIAGRAM:



3.6 USE CASE DIAGRAMS:



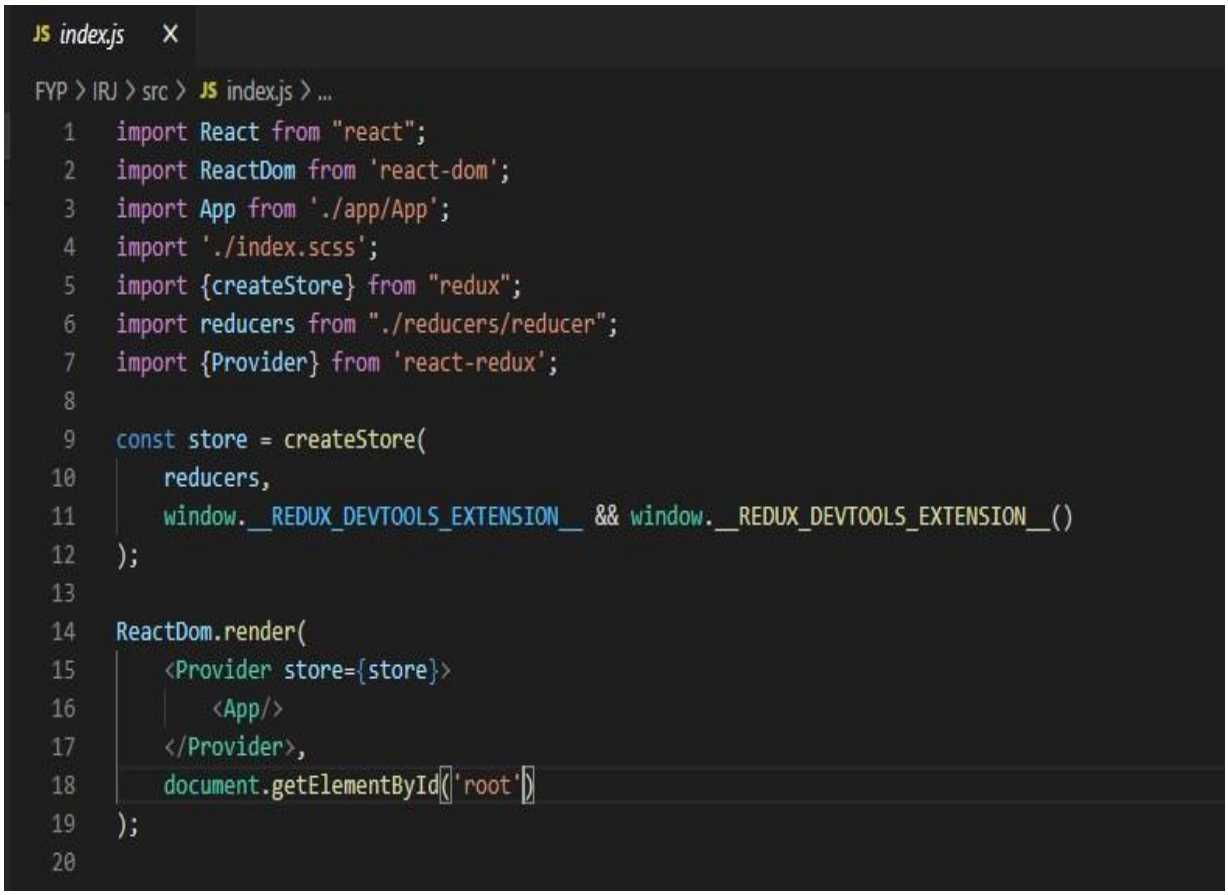

Information Recommendation System
Gather User Choices
Login
Calculate Similarity Scores
User
System
Recommend Information
<includes>
Listen to Information
Show Recommended Information

4 CODE ANALASIS

4.1 Frontend

Index.js is the main page of the react app where all the components are rendered. Index.js typically handles application startup, routing and other functions of application and does require other modules to add functionality.

4.1.1 Index:



4.1.2 Search:

JS Search.jsx X

src > components > Pages > JS Search.jsx > Search

```jsx
import React, {useEffect, useState} from 'react';
import './css/Search.scss';
import Container from "../fragment/Container";
import {useSelector} from "react-redux";
import MusicCard from "../fragment/MusicCard";
import SearchMusic from "../assets/img/searchMusic.svg";
import SearchMusicMp3 from "../assets/img/searchMusicMp3.svg";
import SearchMusicDisc from "../assets/img/searchMusicDisc.svg";
import ArrowUp from '../assets/img/left.svg';
import axios from "axios";
import musicDB2 from '../../db/music2';
import musicDB3 from '../../db/music3';
import musicDB4 from '../../db/music4';

const Search = () => {
    const {playlists, search} = useSelector(state => state.musicReducer);
    const [searchResult, setSearchResult] = useState([]);
    const [Recommendations, setRecommendations] = useState([]);
    useEffect(() => {
        setRecommendations([]);

        setSearchResult(playlists.filter((i) => (
            (i.name.toLowerCase().startsWith(search))
            ||
            (i.author_name.toLowerCase().startsWith(search))
            ||
            (i.musicName.toLowerCase().startsWith(search))
            ||
            (i.lang && i.lang.toLowerCase().startsWith(search))
        )));
    }, [search, playlists]);
    //useEffect ()

```

4.1.3 About:

JS About.jsx

src > components > Pages > JS About.jsx > About

```jsx
import React from 'react';
import './css/About.scss';
import Container from "../fragment/Container";
import Developer from "../fragment/Developer";
import Attribution from "../fragment/Attribution";

const About = () => {
    return (
        <Container>
            <div className={"About"}>

                <Developer/>
                <Attribution/>
            </div>
        </Container>
    );
}

export default About;

```

4.1.4 Music Card:

JS MusicCard.jsx ×

src > components > fragment > JS MusicCard.jsx > MusicCard

```
1   import React, {useEffect, useState} from 'react';
2   import '../assets/scss/MusicCard.scss';
3   import PlayCircleFilledWhiteIcon from "@material-ui/icons/PlayCircleFilledWhite";
4   import {useDispatch} from "react-redux";
5   import {increaseTimesPlayed, setCurrentPlaying} from "../../actions/actions";
6   import Name from "./Name";
7   import {Skeleton} from "@material-ui/lab";
8   import Box from "@material-ui/core/Box";
9
10  function MusicCard(props) {
11      const {name, img, author_name} = props.music;
12
13      const [isHovered, setHovered] = useState(false);
14
15      function handleResponse() {
16          setHovered(!isHovered);
17      }
18
19      const dispatch = useDispatch();
20
21      function handlePlay() {
22          dispatch(setCurrentPlaying(props.music))
23          dispatch(increaseTimesPlayed(props.music.id));
24      }
25
26      const [loaded,setLoaded] = useState(false);
27
28      useEffect(()=>{
```

4.1.5 Profile:

JS Profile.jsx

src > components > Pages > JS Profile.jsx > ...

```jsx
import React, {useEffect, useState} from 'react';
import './css/Profile.scss';
import {Avatar} from "@material-ui/core";
import {useSelector} from "react-redux";
import MusicCard from "../fragment/MusicCard";
import Container from "../fragment/Container";
import Grade from 'grade-js';
import SideBarOptions from "../fragment/SideBarOptions";
import {PlaylistPlay} from "@material-ui/icons";


function Profile() {

    const {playlists} = useSelector(state => state.musicReducer);
    const [mostPlayed, setMostPlayed] = useState([]);

    function sortByProperty(property) {
        return function (a, b) {
            if (a[property] > b[property])
                return 1;
            else if (a[property] < b[property])
                return -1;

            return 0;
        }
    }

```

4.2 Backend

The function retrcmds () takes song name as parameter and returns 10 recommendations for the input song in the form of an array. If the recommendation engine could not find the recommendation for a given song, it will return message “No recommendations Found”.

```
In [51]: print(f"The recommendations for {song} are:")
         print(f"{new_recommendations}")

         The recommendations for stay with me are:
         ['Monsoon', 'Fortunate Fool', 'One More Night', 'Mama Kin', 'Swing Life Away', 'Angie 1993 Digital Remaster', 'You And Your Hea
         rt', 'Arco Arena', 'Just Say Yes', 'Stay With Me']
```

```
In [85]: def retrcmds(song):
             try:
               new_recommendations = model.make_recommendation(new_song=song, n_recommendations=10)
               print(f"The recommendations for {song} are:")
               print(f"{new_recommendations}")
               return new_recommendations
             except:
               new_recommendations = "No Recommendations Found"
               return new_recommendations
```

```
In [86]: retrcmds("st")

         The recommendation system could not find a match for st
         Starting the recommendation process for st ...

Out[86]: 'No Recommendations Found'
```

4.3 Fetching the News:

```
app.get('/api/fortune', cors(), function(req, res) {
  pipeSourceByTop(req, res, 'fortune');
});

app.get('/api/reuters', cors(), function(req, res) {
  pipeSourceByTop(req, res, 'reuters');
});

app.get('/api/techcrunch', cors(), function(req, res) {
  pipeSourceByTop(req, res, 'techcrunch');
});

app.get('/api/wallstreetjournal', cors(), function(req, res) {
  pipeSourceByTop(req, res, 'the-wall-street-journal');
});

app.get('/api/time', cors(), function(req, res) {
  pipeSourceByTop(req, res, 'time');
});
```

```
app.get('/api/top', cors(), function(req, res) {
  pipeSourceByTop(req, res, 'google-news');
});

app.get('/api/arstechnica', cors(), function(req, res) {s
  pipeSourceByTop(req, res, 'ars-technica');
});

app.get('/api/associatedpress', cors(), function(req, res) {
  pipeSourceByTop(req, res, 'associated-press');
});

app.get('/api/cnn', cors(), function(req, res) {
  pipeSourceByTop(req, res, 'cnn');
});

app.get('/api/espn', cors(), function(req, res) {
  pipeSourceByTop(req, res, 'espn');
});
```

```
<h4>Google News</h4>
<Nav vertical>
  <SidebarNavLink to="/top">Top Stories</SidebarNavLink>
  <SidebarNavLink to="/arstechnica">Ars Technica</SidebarNavLink>
  <SidebarNavLink to="/associatedpress">Associated Press</
  SidebarNavLink>
  <SidebarNavLink to="/cnn">CNN</SidebarNavLink>
  <SidebarNavLink to="/espn">ESPN</SidebarNavLink>
  <SidebarNavLink to="/fortune">Fortune</SidebarNavLink>
  <SidebarNavLink to="/reuters">Reuters</SidebarNavLink>
  <SidebarNavLink to="/techcrunch">TechCrunch</SidebarNavLink>
  <SidebarNavLink to="/wallstreetjournal">Wall Street Journal</
  SidebarNavLink>
  <SidebarNavLink to="/time">Time</SidebarNavLink>
  <SidebarNavLink to="/usatoday">USA Today</SidebarNavLink>
</Nav>
```

```
export class MainContainer extends React.Component {
  render() {
    return (
      <div>
        <Switch>
          <Route path='/' component={ArticlesContainer} />
          <Route path='/arstechnica' component={ArticlesContainer} />
          <Route path='/associatedpress' component={ArticlesContainer} />
          <Route path='/cnn' component={ArticlesContainer} />
          <Route path='/espn' component={ArticlesContainer} />
          <Route path='/fortune' component={ArticlesContainer} />
          <Route path='/reuters' component={ArticlesContainer} />
          <Route path='/techcrunch' component={ArticlesContainer} />
          <Route path='/wallstreetjournal' component={ArticlesContainer} />
          <Route path='/time' component={ArticlesContainer} />
          <Route path='/usatoday' component={ArticlesContainer} />
        </Switch>
        <Footer />
      </div>
```

lib > JS newsQuery.js > [e] <unknown>

```
1  module.exports = {
2    sourceByTop(source) {
3      return 'https://newsapi.org/v2/top-headlines?sources='
4        + source
5        + '&apiKey='
6        + process.env.API_KEY; //set API key via environment or dotenv file
7    }
8  }
9
```

```
getArticles(route) {
  this.setState({ loading: true }); //necessary for method reuse

  fetch(route) //requests API from back end (server.js)
    .then(response => {
      return response.json();
    }).then(json => {
      const articles = json.articles; //array of article objects/hashes
      this.setState({ articles: articles, loading: false });
      console.log('parsed json', this.state.articles);
    }).catch(ex => {
      console.log('parsing failed', ex);
    });
}
```

```
export class Article extends React.Component {
  render() {
    return (
      <FadeIn height={500}>
        {onload => (
          <Card className="article">
            <a href={this.props.url} target="_blank"><CardImg top width="100%"
            src={this.props.urlToImage} onLoad={onload} alt="Article image" /
            ></a>
            <CardBody>
              <CardTitle><a href={this.props.url} target="_blank">{this.props.
              title}</a></CardTitle>
              <CardText>{this.props.description}</CardText>
            </CardBody>
          </Card>
        )}
      </FadeIn>
    );
  }
}
```

4.3 Flask Server

Flask server connects python backend to React frontend. It listens for POST request for song name by frontend and returns the recommendations for that particular song.

```
from flask import Flask,render_template, url_for, redirect
from flask import jsonify
from flask import request
from flask_cors import CORS
app = Flask(__name__)
CORS(app)

@app.route("/login", methods=["GET","POST"])
def login():
    if request.method == "POST":
        user = request.data
        print(user)
        return redirect(url_for("user", usr=user))
    #else:
     # return render_template("login.html")


@app.route("/<usr>")
def user(usr):
    from ipynb.fs.full.CF_knn_music_recommend import retrcmds
    check = retrcmds(usr)
    return jsonify(check)



if __name__ == "__main__":
    app.run(debug=True)
```

Chapter 5: Conclusion

In this thesis, we discussed about Intelligent Radio Jockey that can provide song listing more efficiently than the typical song applications by providing user a song listening and a news listening experience simultaneously. Our proposed system has an advantage over other traditional systems due to the latest algorithms used for the recommendations of the song. Techniques used in our proposed system; Songs will be recommended to users using collaborative filtering techniques which will be achieved through K nearest neighbor machine learning algorithm and using the subset of million song dataset; are also briefly explained included their working and importance. The purpose of increasing productivity and overcoming problems in existing solutions is being achieved by using the modern techniques. Additionally, the objectives of recommending song and news to the user simultaneously keeps the users engaged and increases the productivity.

Our song and news recommendation model are run through Flask Server using Anaconda prompt and the Flask server connects the Recommendation models with React

Our proposed system, IRJ, is multi-functional and it is purely made for the core purpose of serving Pakistan. Moreover, IRJ provides an ease to access as no installation is require and it can be assessed through any browser.

Chapter 6: Future Work

Future milestones that need to be achieved to commercialize this project are the following.

6.1 Share Playlist to users of same interests:

The main objective of this application is given user an experience of listening music of their interest along with news. It will recommend song to user based on his previous search and user also get the taste of news in which he was interested.

In future we are looking forward to adding features where users can share the playlist with their friends so that it is more convenient and easier for them to listen to music of their interest.

6.2 Mobile Application:

In future, mobile application of Intelligent Radio Jockey can also be built which will allow users to use application on android/IOS thus making it convenient for users to use Intelligent Radio Jockey anywhere on their smart phones.

Bibliography

- https://link.springer.com/article/10.1007/s10462-021-10043-x
    - https://www.sciencedirect.com/science/article/pii/S1877050919310646
    - http://millionsongdataset.com/
    - https://newsapi.org/docs

Appendix: Glossary

Flask: Python based micro web framework.

React: An open source JavaScript library for creating frontend applications.

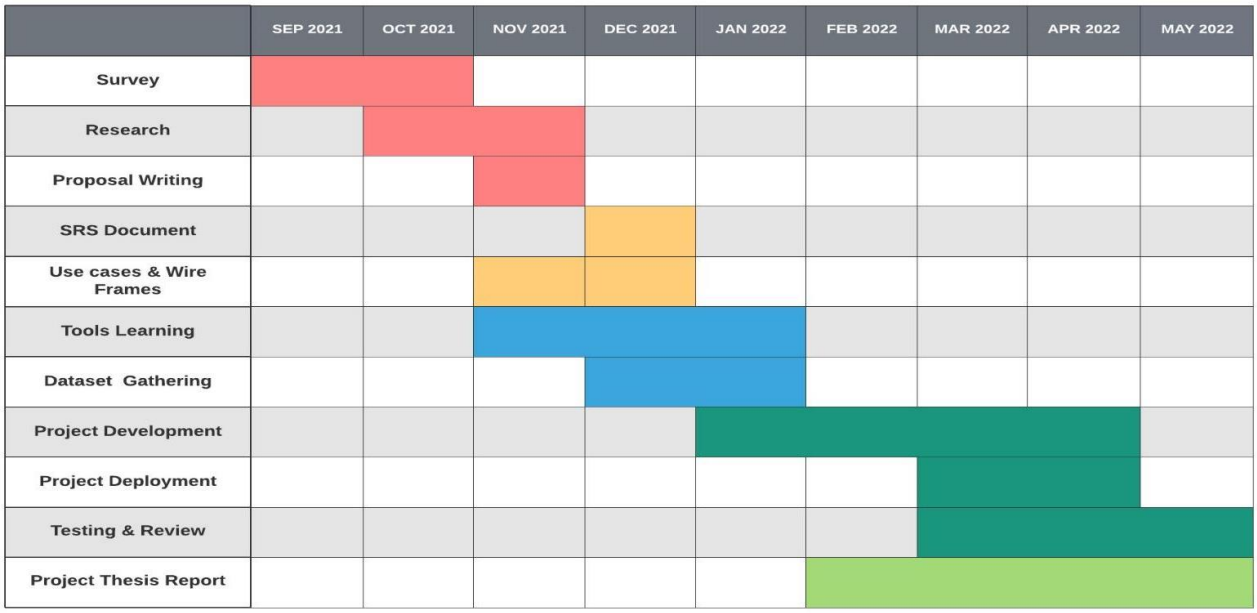
KNN: The k-nearest neighbor algorithm is supervised learning method used for classification and regression. The input consists of the k closest training examples in a data set. The output is determined by whether the algorithm is used for classification or regression.

API: Application programming interface.

OS: Operating System.

App: Application

Appendix: Work Plan



IRJ

By

Usama Sabir
Fahad Akbar
Rai Atif Ali
Bilal Nazir

Submission date: 27-Jun-2022 01:02PM (UTC+0500)
Submission ID: 1863586905
File name: FYP_Thesis_IRJ_Final.pdf (1.77M)
Word count: 4050
Character count: 26174

IRJ

ORIGINALITY REPORT

10%	8%	1%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Higher Education Commission Pakistan Student Paper	2%
2	jacksonlumber.cld.bz Internet Source	2%
3	bilkentprojectapollo.azurewebsites.net Internet Source	1%
4	gist.github.com Internet Source	1%
5	bickson.blogspot.com Internet Source	1%
6	en.wikipedia.org Internet Source	1%
7	etd.aau.edu.et Internet Source	1%
8	Submitted to American Public University System Student Paper	1%

Exclude quotes	On	Exclude matches	< 20 words
Exclude bibliography	On		