Logo

Description automatically generated

PES UNIVERSITY, BANGALORE

Department of Computer Science and Engineering

B-TECH (CSE)

V Semester

UE20CS303 – SOFTWARE ENGINEERING

PROJECT REPORT ON

MUSIC RECOMMENDATION SYSTEM

SUBMITTED BY:

RENITA KURIAN – PES1UG20CS331

RICHA SHAHI – PES1UG20CS334

RIMJHIM SINGH – PES1UG20CS337

RIYA JHA – PES1UG20CS344

Logo

Description automatically generated

PES UNIVERSITY, BANGALORE

Department of Computer Science and Engineering

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| Sl.No | TOPIC | PAGE No. |
| 1. | Proposal of the project | 3 |
| 2. | Software Requirement Specification | 4 |
| 3. | Project Plan | 11 |
| 4. | Design Diagram | 14 |
| 5. | Source Control Management using Github | 18 |
| 6. | Test Cases | 19 |
| 7. | Screenshots of the output | 25 |

Logo

Description automatically generated

PES UNIVERSITY, BANGALORE

Department of Computer Science and Engineering

*PROPOSAL OF THE PROJECT*

The project aims to create a Machine Learning model that recommends songs to the user based on the user’s favorite tracks and genres. The interface is a user-friendly GUI that enables the user to enter her choices and get her personalized tracks. It also offers features such as selection on the basis of genres. The ML model is trained using Spotify dataset and also uses Spotify developer API to extract choicest tracks for the user.

Logo

Description automatically generated

PES UNIVERSITY, BANGALORE

Department of Computer Science and Engineering

*SOFTWARE REQUIREMENT SPECIFICATION*

# 1. INTRODUCTION

## **1.1** **DOCUMENT PURPOSE**

The product whose software requirements are specified in this document is MusicWreck.

The purpose of this document is to present a detailed description of the product, MusicWreck. This document is intended to

* Explain the purpose and features of the product, MusicWreck
* The constraints under which the product must operate
* How the product would respond to different users’ requests?

The document’s primary goal is to help the reader get a better understanding of the project. The document is intended for the developers of the software, the end users of the product who have been identified in the later sections, and to the professors who would review the project.

## **1.2** **PRODUCT SCOPE**

The software being developed is Machine Learning based Music Recommendation System. The product would recommend songs to the user based on her preferences of genres and songs she like. The product would –

* Recommend songs to the user based on genres.
* User can also select certain songs from the dropdown option to get a personalised playlist
* Would be provided with a Spotify playlist with all these songs

The points specified above would make a competent music recommendation system for the user according to her likelihood of musical tracks.

**1.3** **INTENTED AUDIENCE AND DOCUMENT OVERVIEW**

#### 1.3.1 Intended Audience:

This document is primarily intended for the:

* Developers of this software
* Software engineers who would work on further development of the project
* The professors who would review the document and finally,
* Clients that is novice or professional event managers, volunteers.

#### 1.3.2 Document Overview:

The first chapter, that is the Introduction section of the document is intended to introduce the reader to the product, MusicWreck.

The second chapter, Overall Description section of SRS document provides an overview of the overall functionality of the product. It describes the informal requirements.

The third chapter, Specific Requirements section, of SRS document is written primarily for the developers and describes in technical terms the details of the functionality of the product.

The second and the third chapter of the document describe the same software product, but are intended for different audiences and thus use different language.

# 1.4 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

|  |  |  |
| --- | --- | --- |
| 1 | Employer | Employer is an individual who has contacted the event organiser. |
| 2 | Event Manager | Event Manager is an individual who is responsible for the whole event and can view the entirety of the event being planned on the software. He/She is usually the lead event organizer. |
| 3 | HTTPS | HTTPS stands for Hypertext Transfer Protocol Secure. This protocol is a widely used communications protocol for secure communication over a computer network, with especially wide deployment on the Internet. |
| 4 | Spotify | Organisation that’s going to provide all the dataset and playlist interface for the user. |
| 5 | SRS | SRS stands for Software Requirement Specification. It is a document that completely describes all of the functions of a proposed system and the constraints under which it must operate. |
| 6 | Team Head | Team head is an individual who is responsible for all the actions undergoing under his/her team. |
| 7 | UI | UI stands for User Interface. It is defined as the space where interaction between humans and machines occurs. |
| 8 | View | View means to display and look at data on screen. |
| 9 | ML model | The Machine Learning Model being used for the recommendation system. |
| 10 | Dataset | The dataset being used for building the recommendation system taken [kaggle](https://www.kaggle.com/). |
| 11 | API | Application Programming Interface to use HTTP requests to use Spotify Developers Features. |

### 1.5 References and Acknowledgments

#### 1.5.1 References:

* [**https://docs.google.com/document/d/e/2PACX- 1vRIW\_TuZ6z0ASjAoxgJgmzjGYLCDx019tKvphaTwK\_Za7fnMKywUuXI0- s5wr0nQI\_gprm6J6y7L9rL/pub**](https://docs.google.com/document/d/e/2PACX-1vRIW_TuZ6z0ASjAoxgJgmzjGYLCDx019tKvphaTwK_Za7fnMKywUuXI0-s5wr0nQI_gprm6J6y7L9rL/pub)
* [**https://jinja.palletsprojects.com/en/3.0.x/templates/**](https://jinja.palletsprojects.com/en/3.0.x/templates/)
* [**www.python.com**](http://www.python.com/)
* [**https://developer.spotify.com/documentation/web-api/quick-start/**](https://developer.spotify.com/documentation/web-api/quick-start/)

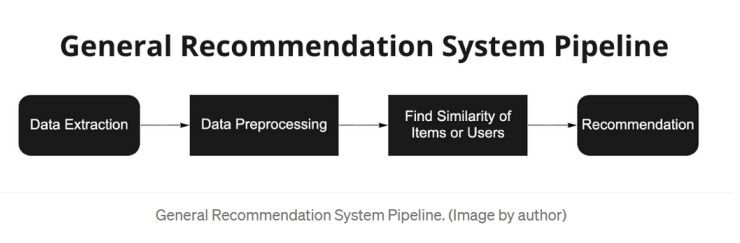
## **2. Overall Description**

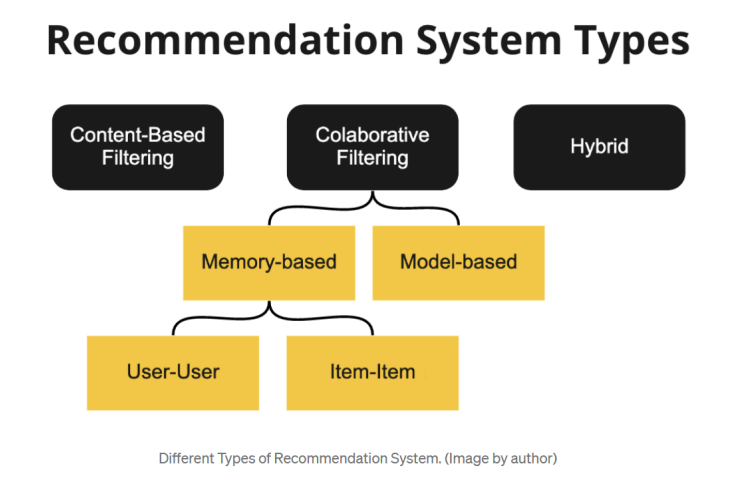
#### 2.1 Product Perspective

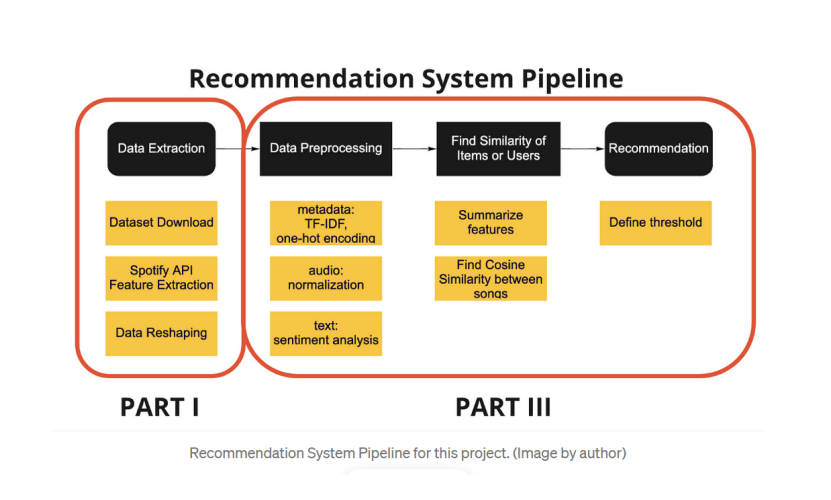
#### 

The product MusicWreck is a music recommendation system product. It mainly is trained on a Spotify dataset. The user is suggested with song recommendation after she picks her most liked songs. By using music recommender system, the music provider can predict and then offer the appropriate songs to their users based on the characteristics of the music that has been heard previously. In a nutshell, recommendation systems recommend things that the people might like based on your own watch history or you and friends watch history as a collective.

#### 2.2 Product Functionality





****

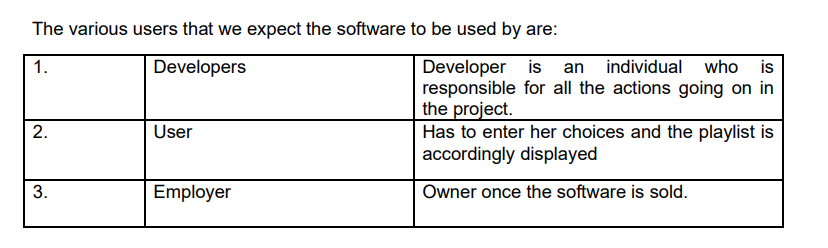
### 2.3 Users and Characteristics

The system will support four types of user privileges:

● Developer

● User

● Employer



### 2.4 Operating Environment

The software will be designed to work on any version of Windows, Linux (kernel 2.7 and above) and Mac platform. The software is completely web based and runs on popular web browsers namely firefox, chrome, internet explorer ( IE8 and above). These web browsers are preferred since they support HTML.

2.5 Design and Implementation Constraints

We have to design different pages for different types of users such as developers and target audience. The implementation part is yet to be done. But, we have a clear picture as to how our pages would look. The communication protocol will be http and smtp. There are a number of tools which can be used for its implementation. The maximum number of users at a time is yet to be decided we will try to deploy it at an online hosting website or make an installable python executable.

### 2.6 User Documentation

No tutorials have been developed as of now.

### 2.7 Assumptions and Dependencies Assumptions

The user is familiar with internet and web based software like social networking sites. The browsers which the user is using is either Google Chrome 10.0 and above or Mozilla Firefox 4.0 and above.

Dependencies

* Spotify Dataset
* Spotify developers API
* Streamlit or Flask
* Other modules and packages to be decided based on implementation

### 3. Specific Requirements

### 3.1. External Interface Requirements

#### 3.1.1. User Interfaces

The user interface design is simple and clear. Users can simply choose enter a song and obtain recommendations

#### 3.1.2. Hardware Interfaces

Not applicable.

#### 3.1.3. Software Interfaces

The software is operating system independent. It would run on Linux, Windows and Mac.

#### 3.1.4. Communications Interfaces

A web browser is a basic necessity for the software to be deployed. Playlist is shared via SMTP protocol or User has to enter Spotify credentials to get it

Logo

Description automatically generated

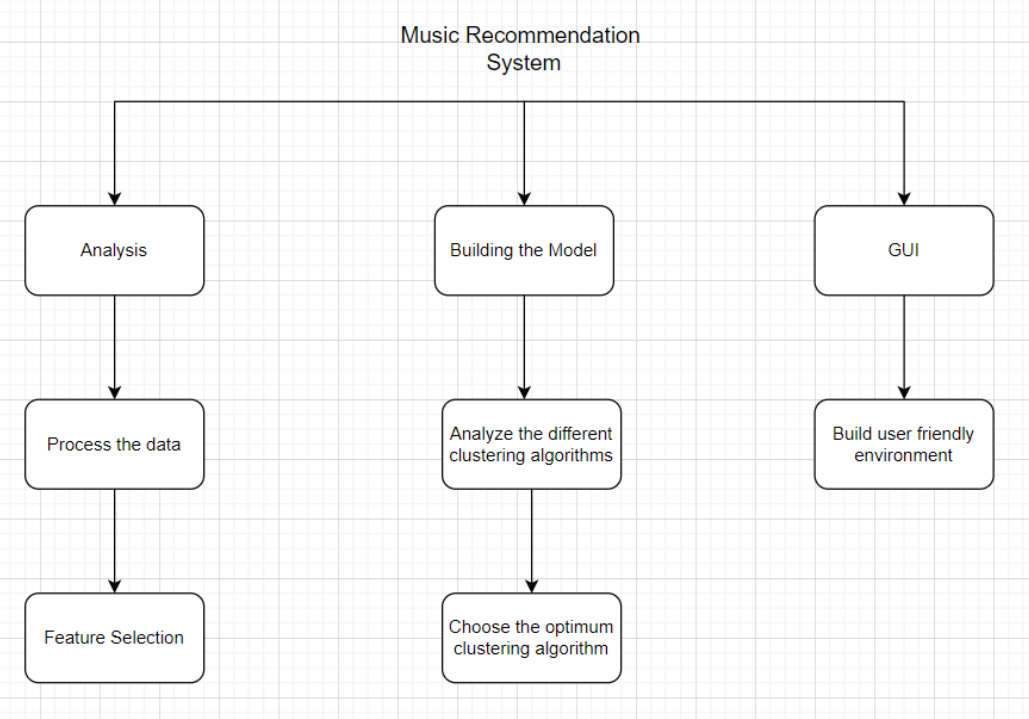
PES UNIVERSITY, BANGALORE

Department of Computer Science and Engineering

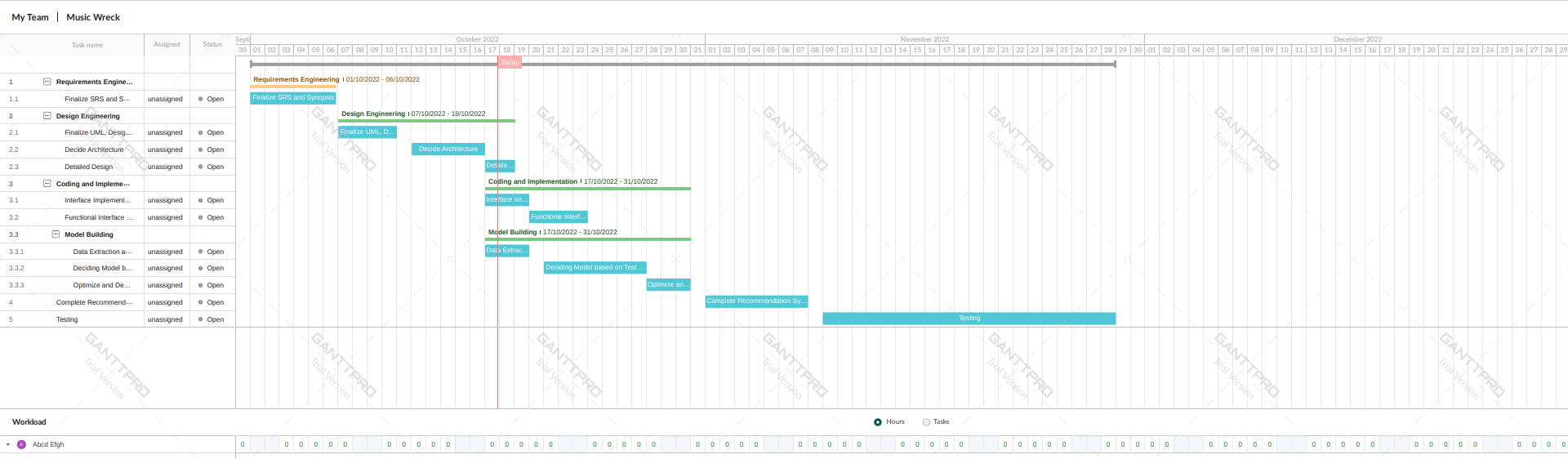
*PROJECT PLAN*

We have decided to go with Agile methodology with client Server Tiered architecture. We chose to use JIRA for Agile project management and tracking project development. The work was equally distributed using WBS and multiple sprints. Each sprint was focused on making the product better by adding features iteratively.

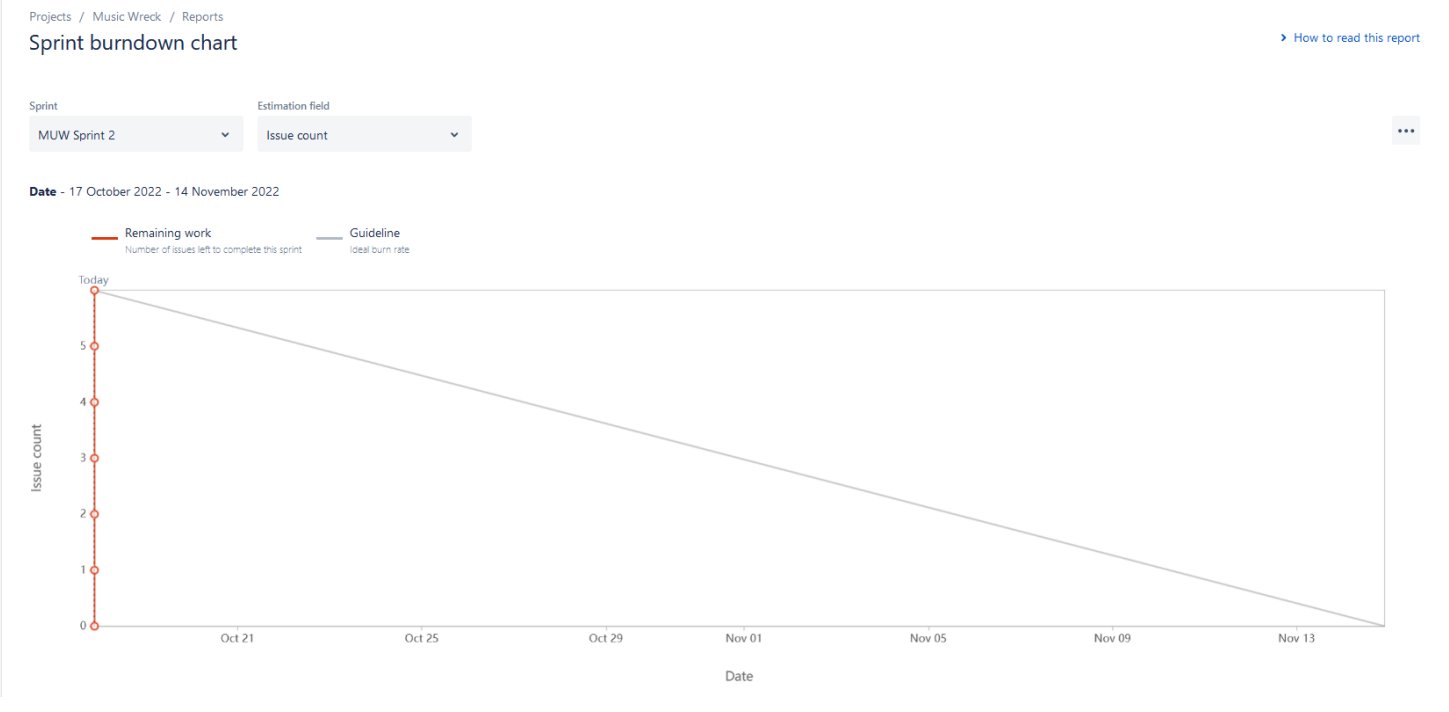
Work Breakdown Structure:



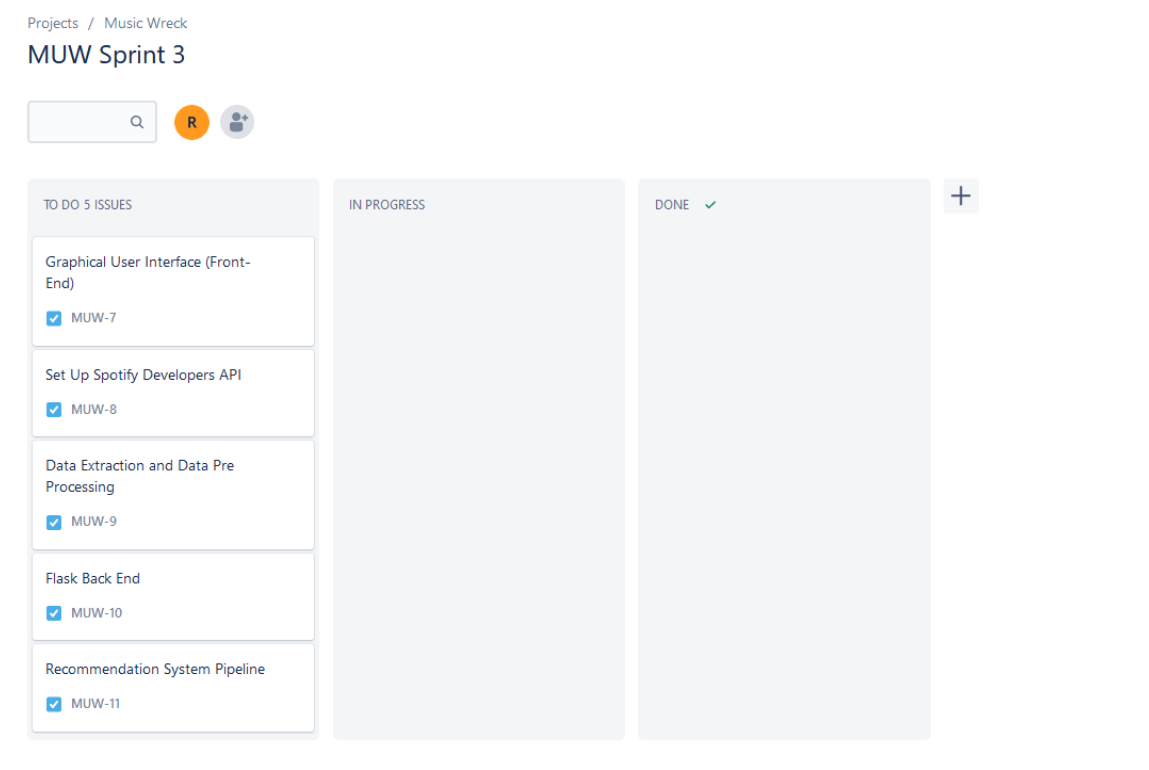
Gantt Chart:

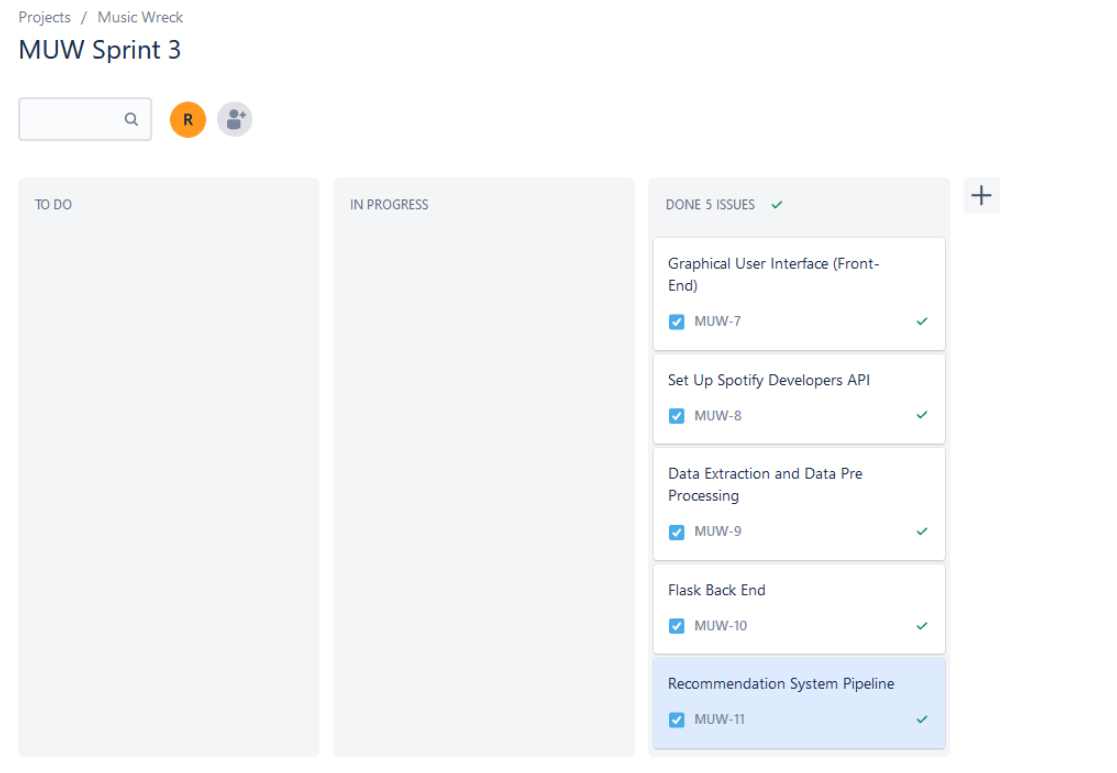


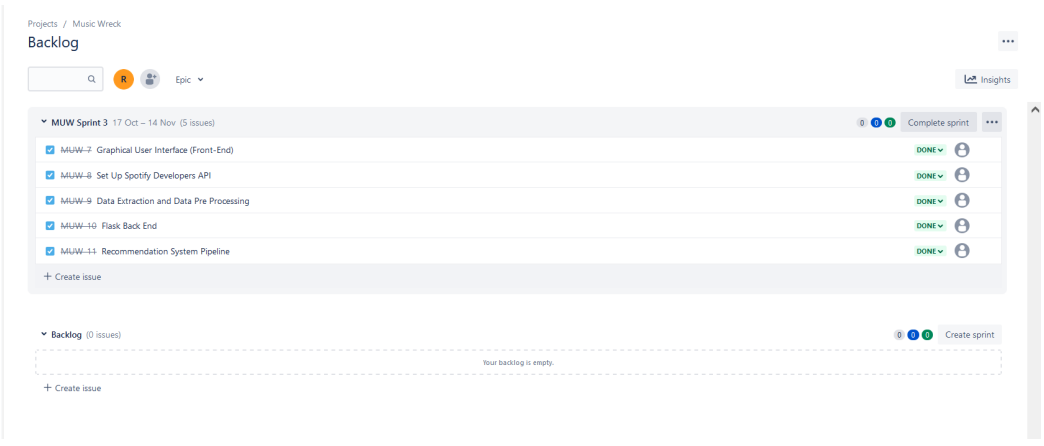
Sprint Burndown Chart:



Sample screenshots from JIRA for sprint Reports:







Logo

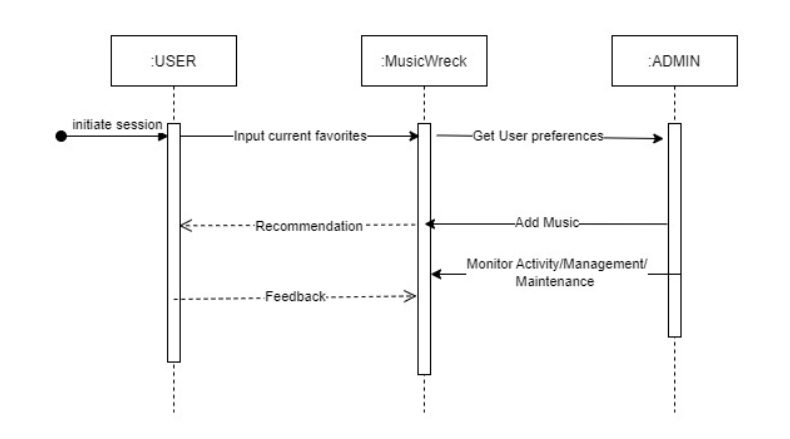
Description automatically generated

PES UNIVERSITY, BANGALORE

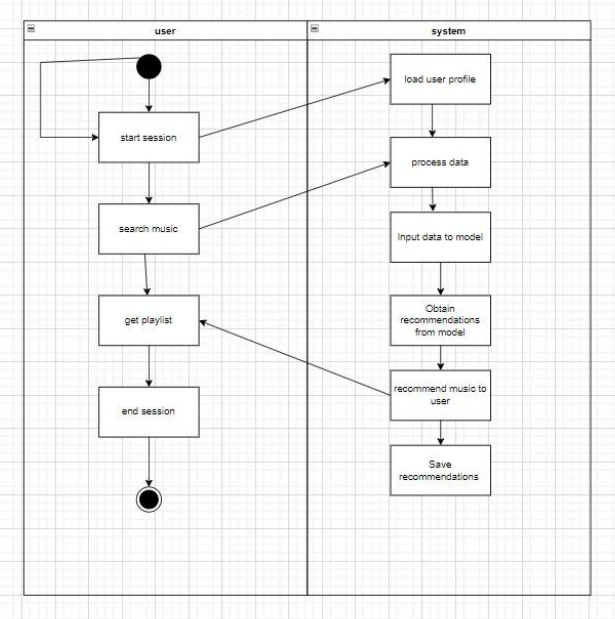
Department of Computer Science and Engineering

*DESIGN DIAGRAM*

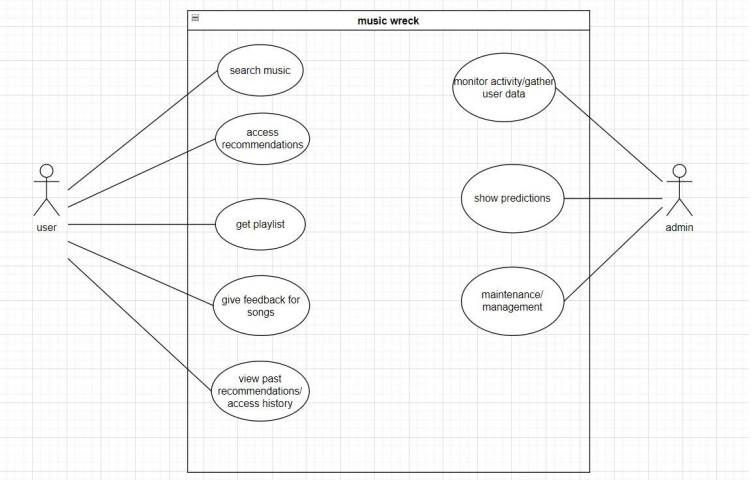
UML Diagram:



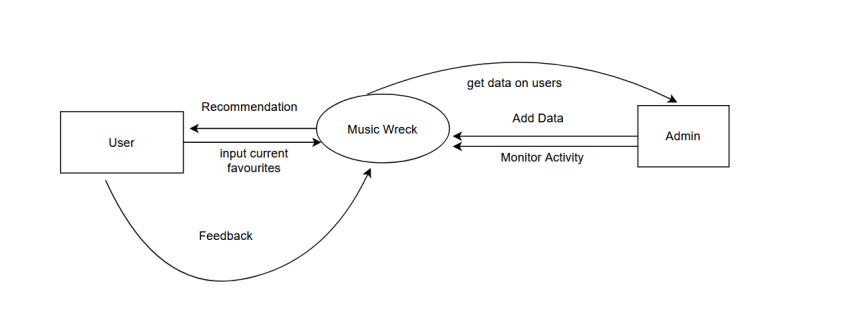
Activity Diagram:



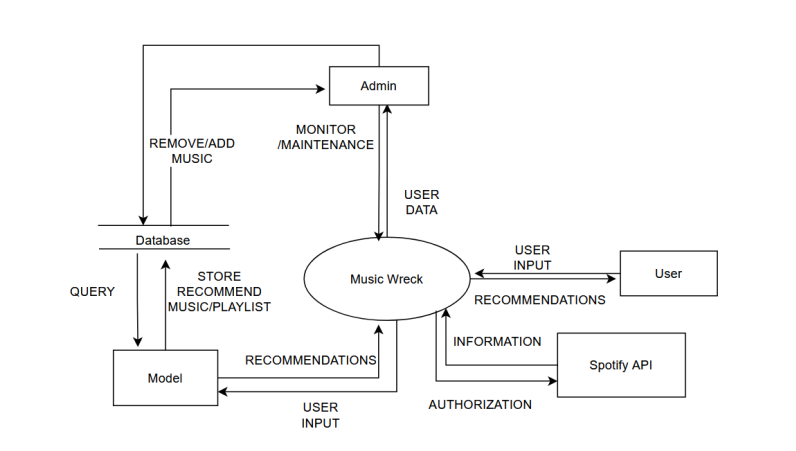
Use Case Diagram:



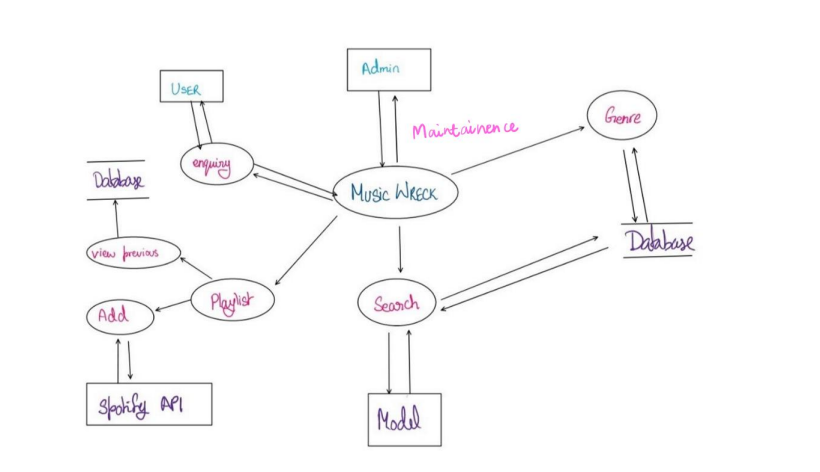
Top Level DFD:



First Level Decomposition:



Second Level Decomposition:



Logo

Description automatically generated

PES UNIVERSITY, BANGALORE

Department of Computer Science and Engineering

*SOURCE CONTROL MANAGEMENT USING GITHUB*

We chose to use Git for our source control management tool. All members were given access to the project repository in order to fix bugs or add new features. Members working on a new feature would open a new branch, make the relevant changes and then send in a pull request to merge once they were confident with the feature functionality.

Logo

Description automatically generated

PES UNIVERSITY, BANGALORE

Department of Computer Science and Engineering

*TEST CASES*

**Test Suite:**

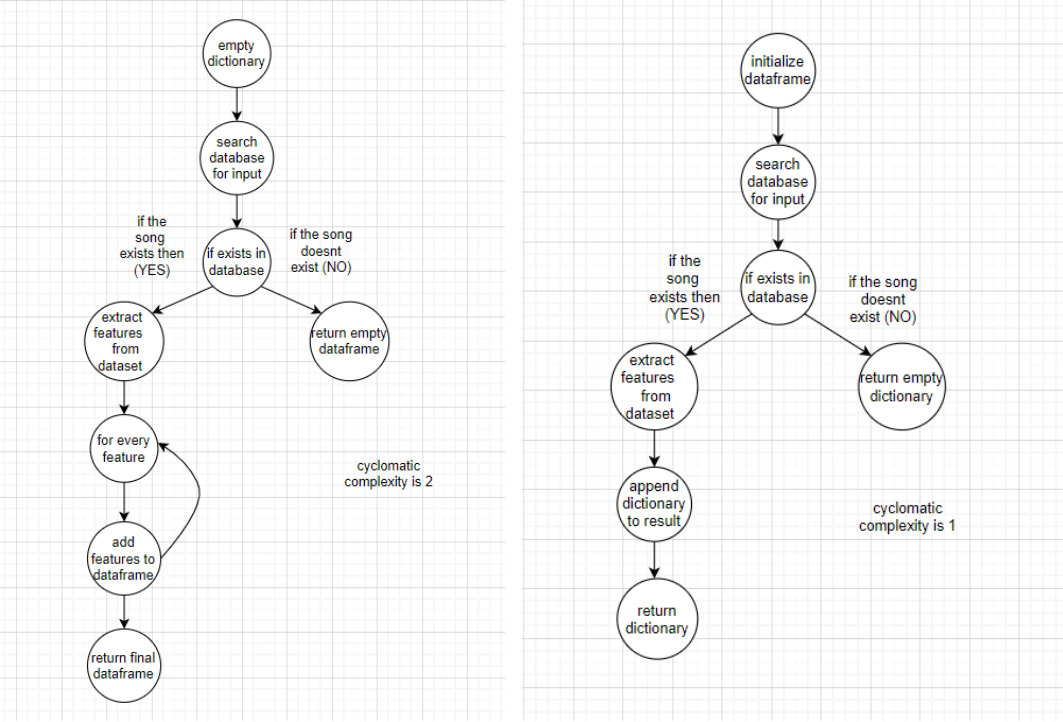
1. Take input from user
2. Find relevant song data from the input given by users using the Spotify API
3. Get recommendations from model
4. Output the recommendations obtained by the model to the frontend

**Test Strategy:**

We have used four testing strategies -

1. Static testing:

Initially we were outputting a data frame after extracting features from the API for which we were using a ‘for’ loop. In order to optimize the code and minimize complexity we are now outputting a dictionary instead of a ‘for’ loop. As we see from the screenshot below, the number of nodes are 7 & edges are 6. Hence the complexity is reduced.



Initial Design Optimized Design

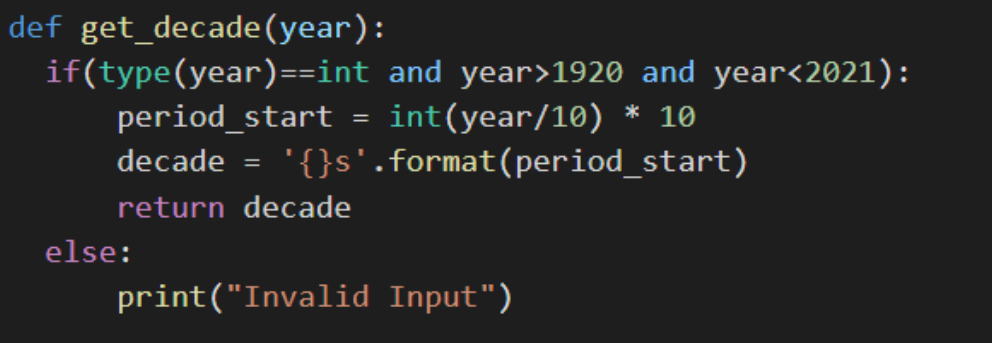
Complexity = E – N + 2

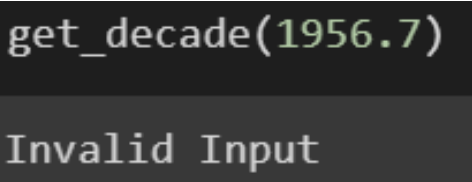
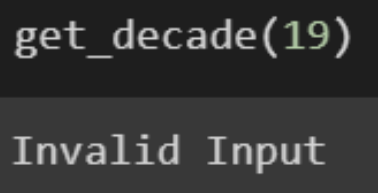
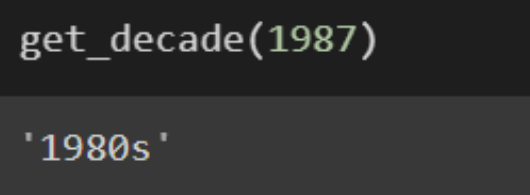
Where E = edges, N = nodes

For initial design, complexity = 2

For optimized design, complexity = 1

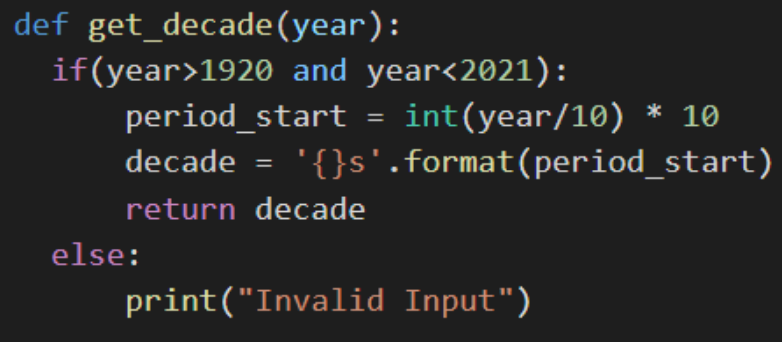
1. Mutation testing:

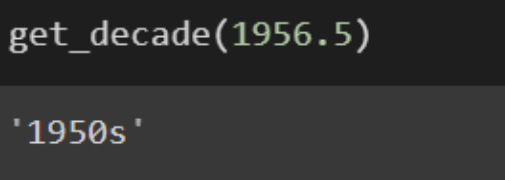




The get\_decade function takes in year as input and outputs the decade to which it belongs. The above screenshots show the code and output for various test cases of the function.

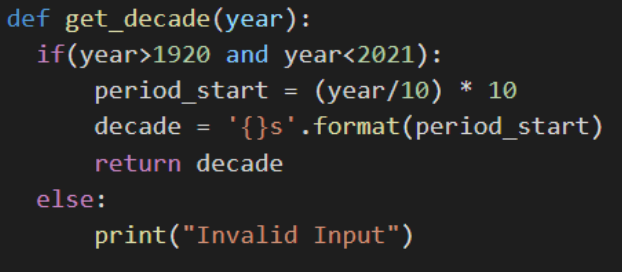
Mutation 1 -

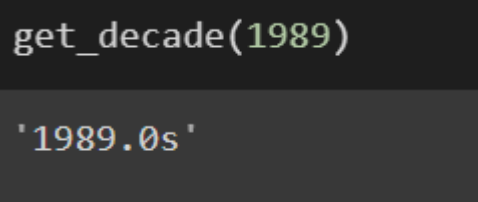




In the above mutated code the condition to check for data type of year is removed. From the above screenshot it is seen that the wrong output is obtained. Year cannot be of type float but the function inputs float value as a valid input for year.

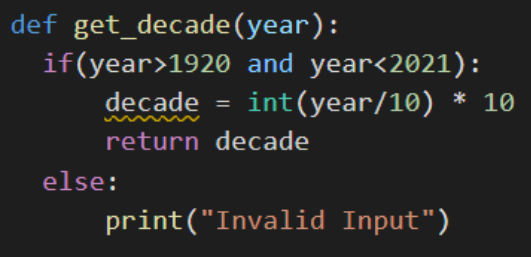
Mutation 2 -

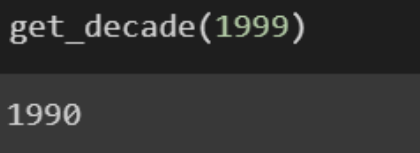




In the above mutated code the explicit type conversion to int is omitted. This causes the function to give invalid output. The function returns a float value as year but year is supposed to be of type int.

Mutation 3 -





In the above mutated code a line of code is removed from the original function being tested. It can be seen that the output gives out a year. However, it cannot be clearly determined that the result is the decade to which the year belongs. 1950s indicate a decade whereas 1950 is a year.

1. Unit testing:

The project can be divided into different units such as

➢ Pre-processing of data

➢ Building the model

➢ Finally obtaining recommendations using Spotify API

The Pre-processing of data can be further divided into smaller units such as

➢ Filtering out essential features

➢ Grouping data by decade as opposed to year

The Model building Unit can be further divided into smaller units such as

➢ Creating pipeline

➢ Training the model

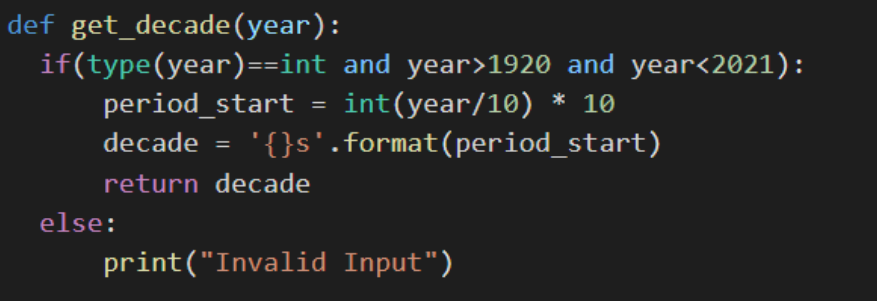
Obtaining recommendation Unit can be further divided into smaller units such as

➢ Getting input from user

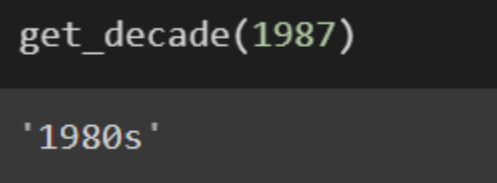
➢ Cross Checking input with Spotify API to obtain relevant features

➢ Recommend based on model output

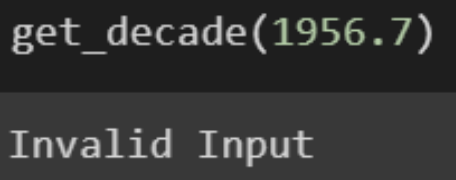
Test Case for pre-processing:



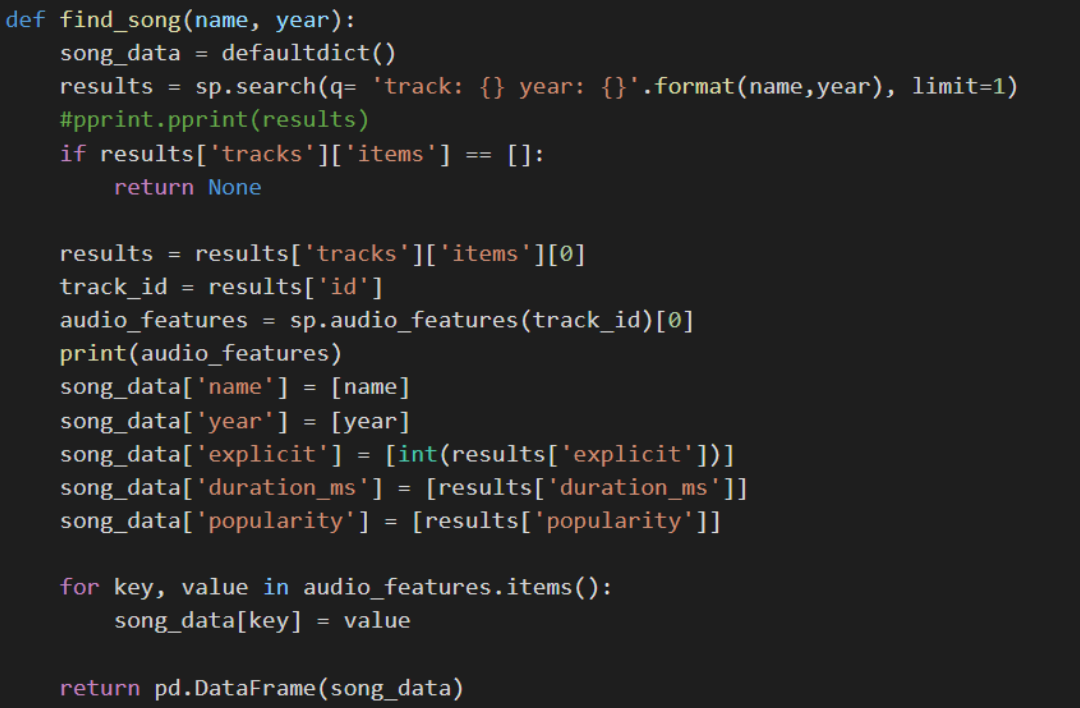
The above screenshot shows the function get\_decade used to preprocess the data. The function takes in year as input and outputs the decade to which it belongs.



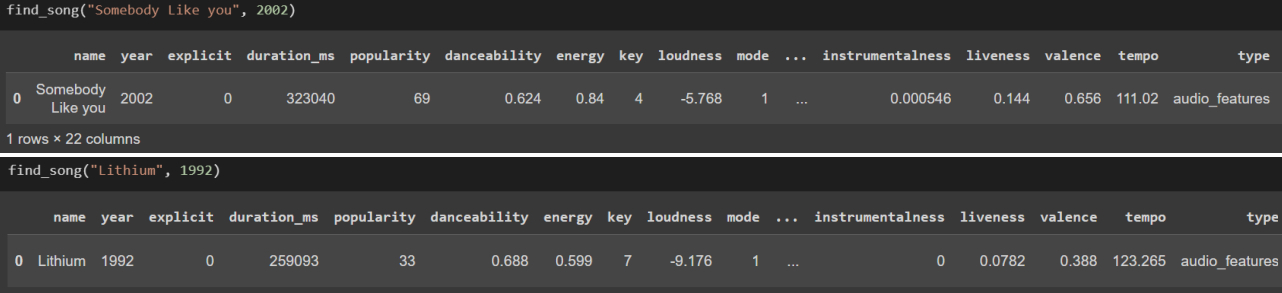
The above screenshot shows an example of a test case for valid data. In this case, the year 1987 is given as input and the function outputs 1980s.



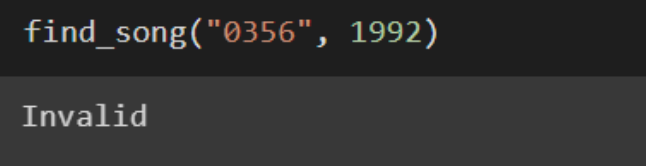
The above screenshot shows an example of a test case for invalid data. The year input is given as 1956.7 which is a float value and hence results in invalid input.



The above function find\_song is used in the recommendation unit to obtain song details from spotify API. The function inputs the song name and year and outputs a dataframe with track details.



The above screenshots are examples of valid test cases which result in data frames with track info such as popularity, liveness, energy etc.



The above screenshot shows an example of a test case with invalid data.

1. Integration testing:

We have also tested all the units together to make sure all of them are working properly in synergy.

**Test Plan:**

* This product is aimed to give users a user-friendly experience with no complicated technology knowledge required.
* In order to produce a bug free product that provides a user-friendly environment, we have chosen to use four testing types - unit, integration, static and mutation.
* Testing is performed by developers after each and every sprint.

Logo

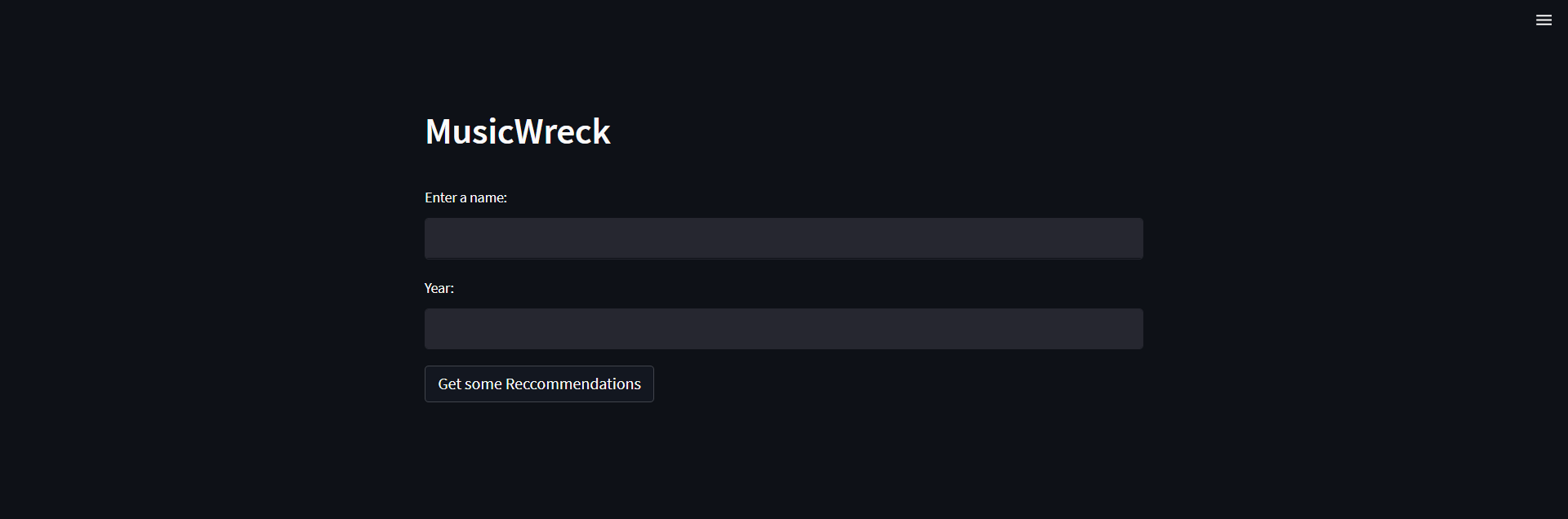
Description automatically generated

PES UNIVERSITY, BANGALORE

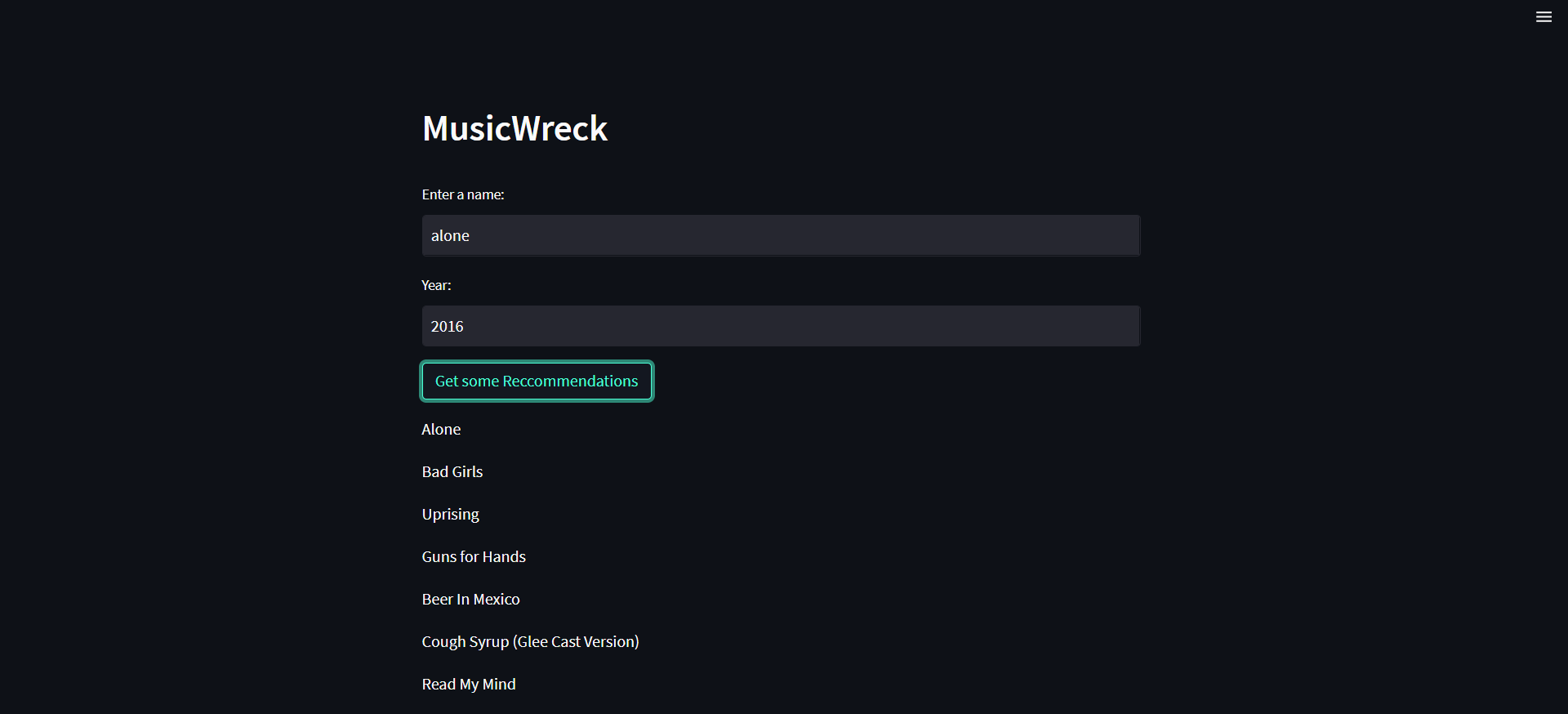
Department of Computer Science and Engineering

*SCREENSHOTS OF THE OUTPUT*

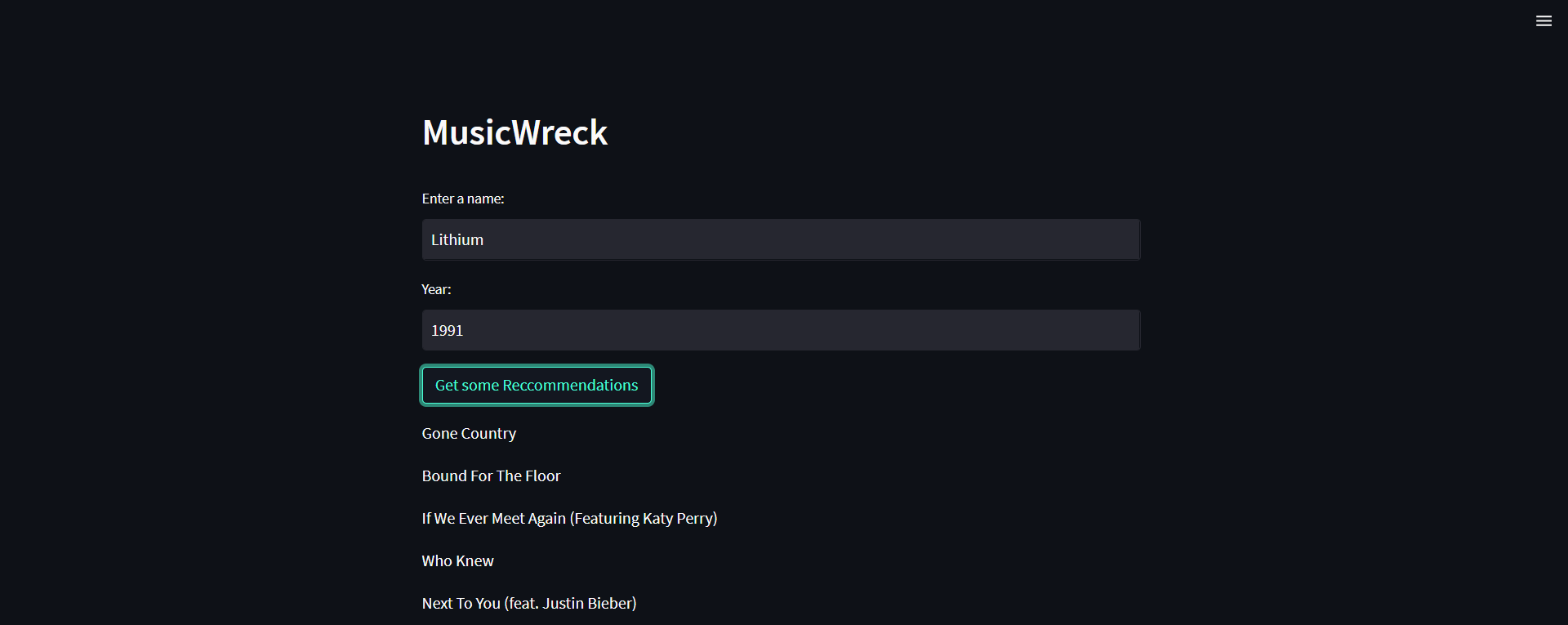
Sample screenshots of UI



Screenshot of home page



Screenshot of page with sample results 1



Screenshot of page with sample results 2