



FINAL YEAR PROJECT REPORT

BS (SOFTWARE ENGINEERING)

RECOMMENDER SYSTEM FOR MUSIC

SUBMITTED BY

MUHAMMAD WALEED	50227
SYED HUMAL HASSAN	40828
SYED MUHAMMAD AUN	51986

SUPERVISOR

SIR. FAHAD NAJEEB

COORDINATOR

DR. AARIJ MAHMOOD HUSSAIN

FACULTY OF ENGINEERING, SCIENCE AND TECHNOLOGY

IQRA UNIVERSITY, KARACHI

MARCH 2024



FACULTY OF ENGINEERING, SCIENCE AND TECHNOLOGY

DEPARTMENT OF SOFTWARE ENGINEERING

FINAL YEAR PROJECT REPORT

BACHELOR OF SOFTWARE ENGINEERING

MUHAMMAD WALEED (50227)

SYED HUMAL HASSAN (40828)

SYED MUHAMMAD AUN (51986)

PROJECT:

RECOMMENDER SYSTEM FOR MUSIC

SUPERVISOR:

SIR. FAHAD NAJEEB

MARCH 2024

ABSTRACT

Recommending music based on the user's music preferences is a way to improve the user's listening experience. Finding the correlation between user data (such as music listening history, emotions, etc.) and music is a difficult task.

Nowadays, there are a ton of music streaming services available. It is difficult for consumers to select music they enjoy with so many options. Recommendation systems fill that need. This project aims to create a system that recommends music to users based on their listening history and preferences.

Two approaches are combined in our system: content-based filtering, which looks at music features, and collaborative filtering, which looks at other users' likes. These methods can be combined to provide more insightful recommendations.

We leverage extensive user interaction data as well as specifics about the music. Then, using machine learning, our system figures out what music to recommend to each user. We test how well our system works using measures like accuracy and how often it suggests music the user actually likes.

Our tests show that our system is pretty good at figuring out what music users will enjoy. We also learned that it's important to keep updating the system based on what users like and don't like.

Overall, our project helps make music streaming better by giving users personalized recommendations that match their tastes.

While multimedia technology is developing rapidly following the pace of the times, we have also entered the era of big data with massive amounts of data. Music is one of the contents of multimedia information. While there is a huge user demand, its own data volume is also very considerable and continues to grow. Nowadays, with abundant music resources, how to efficiently obtain the songs of interest in the vast and complicated sea of music, a targeted solution of a personalized music recommendation system is proposed. Music recommendation algorithms predict and push user behavioral preferences based on user behavioral information and music characteristics. The development of music recommendation algorithm has also shifted from the technical route recommended by users' personal preferences to the mutual recommendation among users. In recent years, the focus of attention has been on tapping potential preferences. In general, the technology of music recommendation algorithms is getting better and better.

We have approved this manuscript for submission and presentation as fulfillment of Bachelor of Software Engineering/ Computer Science.

Supervisor: Sir Fahad Najeeb

Date: 12-03-2024

Project Coordinator: Dr. Aarij Mahmmod Hussain

Date: 12-03-2024

DECLARATION

I hereby declare that the work has been done by myself to fulfill the requirement of the BS (Software Engineering) and no portion of the work contained in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

I hereby further declare that in the event of any infringement of the provision of the Act whether knowingly or unknowingly the university shall not be liable for the same in any manner whatsoever and undertake to indemnify and keep the university indemnified against all such claims and actions.

© MUHAMMAD WALEED [50227]

© SYED HUMAL HASSAN [40828]

©SYED MUHAMMAD AUN [51986]

ACKNOWLEDGEMENT

First, we thank Almighty Allah who praise us with the ability to think, work and deliver what we are assigned to do. Secondly, we must be grateful to our supervisor “**Sir. Fahad Najeeb**” who helps us in this project. We also acknowledge our teachers that throughout our studies helps us and guides us, departmental staff, university staff or other then this. We are also thankful to the FYP instructor “**Dr. Mansoor Ebrahim**” and “**Dr. Aarij Mahmood Hussain**” for his precious support throughout the tenure as he is the best instructor for FYP who makes every student to be updated with the project progress and lead to the completion with great success within the time period given. We are also grateful to our family and friends, for supporting and encouraging us to complete this project. Finally, we would like to thank all the colleagues of IQRA University who have been with us in all difficult times with suggestions and supportive words which carry us to make this project a reality.

LIST OF ACRONYMS

1. **ML:** Machine Learning
2. **CF:** Collaborative Filtering
3. **CB:** Content-Based
4. **GUI:** Graphical User Interface
5. **API:** Application Programming Interface
6. **UI:** User Interface
7. **UX:** User Experience
8. **SQL:** Structured Query Language

TABLE OF CONTENTS

CHAPTER – 1	9
1.0 Introduction.....	9
1.1 Problem Statement	9
1.2 Motivation.....	9
1.3 Objective	10
1.4 Challenges.....	10
1.5 Structure of Report.....	10
1.5.1 Chapter 2: Technology Background.....	11
1.5.2 Chapter 3: Requirements & Methodology	11
1.5.3 Chapter 4: Project Plan & Initial Design.....	11
1.5.4 Chapter 5: Project Design & Development	12
1.5.5 Chapter 6: Testing	12
1.5.6 Chapter 7: Conclusion	12
CHAPTER – 2	12
2. Technology Background	12
2.1 Background of the technology:	12
2.1.1 Android Studio	12
2.1.2 Java.....	13
2.3 Literature Review:.....	13
CHAPTER – 3	15
3.1 Introduction:.....	15
3.2 Project Plan:	15
3.3 Functional Requirements:	17
3.3.1 User Panel:	Error! Bookmark not defined.
3.4 Non-Functional Requirements:	18
3.5 Hardware Requirements:.....	19
3.6 Summary:	19
CHAPTER – 4	19
4.1 Introduction:.....	19
4.2 Data Flow Diagram:	19
4.2 Entity Relationship Diagram.....	22
4.3 Use Cases	23
4.5 Summary:	27
CHAPTER – 5	27

5.1 Introduction:.....	27
5.2 Prototype Design:.....	27
5.3 Database Queries:.....	Error! Bookmark not defined.
5.4 External Libraries:.....	35
5.5 Screenshots:	36
5.6 Summary:	44
CHAPTER – 6	44
6.1 Introduction:.....	44
6.2 Test Cases:	44
6.3 Summary:	45
CHAPTER – 7	1
7.1 Introduction:.....	1
7.2 System Limitations and Challenges:	1
7.3 Future Work:	2
7.4 Conclusion:	2
REFERENCES.....	Error! Bookmark not defined.
APPENDIX.....	Error! Bookmark not defined.
Bussiness Canvas:	Error! Bookmark not defined.
Detailed Gantt Chart:	Error! Bookmark not defined.
Software Manual:	Error! Bookmark not defined.

CHAPTER – 1

1.0 Introduction

The sophisticated music recommendation system developed for this project responds to the increased need for individualized music recommendations in the ever-changing ecosystem of digital music platforms. The system is able to deliver precise recommendations that are in line with each user's specific tastes by combining collaborative filtering with content-based filtering techniques. In order to find trends and commonalities across users, collaborative filtering examines user behavior, including listening preferences and habits. By using this strategy, the algorithm is able to suggest songs that people with similar tastes have appreciated. The intrinsic qualities of music tracks, such as genre, tempo, and instrumentation, are the focus of content-based filtering, which looks for commonalities and suggests appropriate matches.

The system makes use of data mining and machine learning algorithms to guarantee the efficacy and accuracy of recommendations. Large volumes of user data, including listening history, explicit feedback, and contextual data like location, time, and mood, are processed by these algorithms. These criteria enable the system to recognize the unique tastes of every user and provide recommendations that are suitable for their preferences. This suggestion system is also intended to change and improve over time. It integrates real-time updates and continuously monitors user interactions to reflect the most recent trends in user behavior. The recommendations are kept current and pertinent thanks to this iterative process, which also introduces users to intriguing new music.

The system includes mechanisms for evaluation and feedback as well. This enables users to comment on the suggested tracks, allowing the system to improve its algorithms and the precision of suggestions in the future. The system strives to improve user pleasure and involvement in the music discovery process by continuously assessing and improving its suggestion skills. To sum up, this research introduces a sophisticated music recommendation system that combines collaborative filtering with content-based filtering. The system may give customized music recommendations that take into account each user's individual tastes by utilizing machine learning and data mining algorithms. Real-time updates and evaluation methods are incorporated to keep the system dynamic and give users accurate and pertinent recommendations for improved music listening experiences.

1.1 Problem Statement

Our project aims to combat the information overload and lack of personalized music recommendations on streaming platforms. With the vast amount of music available, users often struggle to find songs that match their preferences, leading to frustration and reduced engagement. Current recommendation systems usually rely on generic algorithms that do not take into account the individual tastes and preferences of users, resulting in suboptimal recommendations. Therefore, our project aims to develop a music-adapted recommendation system that uses user data and advanced algorithms to provide highly personalized recommendations, improving the music discovery process and increasing user satisfaction.

1.2 Motivation

The most commonly used MRS methods are Collaborative Filtering (CF), which is a process in which a system analyzes user preferences based on its historical usage data, and content-based (CB) techniques, which analyze object descriptions to identify objects of interest. To the user. CF recommends songs to people based on other users who have the same music preferences as them. In contrast, CB uses metadata and content such as album, artist, and genre to recommend songs. It is found that CF generally gives better recommendations than CB. However, this only applies if usage data (such as ratings for previous tracks) is available. Otherwise, it will not produce accurate results

and will therefore suffer from the cold start problem, which involves two types of problems - new products and new users. The first problem concerns new products that are intended to be recommended but with insufficient information such as ratings. The second occurs when a new user joins the system and not much is known about him. Therefore, the recommendation system cannot provide personalized recommendations to users until they start rating different items. CB is generally less susceptible to the cold start problem because it can still recommend products even if it doesn't have enough ratings. However, CB has its own problems and is not without problems. The biggest one is that it recommends songs that the user already knows. The purpose of this thesis is to present a solution to solve the MRS cold start problem and provide users with new music recommendations that they have never heard. Therefore, the motivation of this thesis is to increase the efficiency of a music recommendation system with versatility and novelty, reducing the effects of the Cold-Start problem by combining three recommendation techniques into hybrid approaches. It provides people with a solution that allows them to get more accurate and better recommendations based on their own music preferences.

1.3 Objective

The main goal of this research is to provide both new and existing users with an easy way to discover new songs, while providing personalized recommendations that accurately identify users' musical preferences. By analyzing user interactions in the system, we try to identify which bands or artists users are ready to listen to at any given moment. We understand that users are looking for versatility and novelty in their music consumption, our goal is twofold. First, we try to develop context-aware recommendations using previously collected user feedback data, including ratings and listening history. This facilitates the creation of contextual music recommendation systems that adapt to users' changing preferences and circumstances. Second, we aim to support recommender system algorithms with a multi-strategy approach, combining the results of different core recommenders and general recommenders to generate a comprehensive final recommendation. With a multifaceted approach, we strive to improve the accuracy and efficiency of recommendations delivered to users and ensure a more enriching and personalized music discovery experience.

1.4 Challenges

However, many aspects are very specific to music recommendation, and some others are at least more relevant to music than to other application areas of recommendation systems. These aspects relate to both technical and non-technical issues and include but are not limited to:

Catalog Considerations: While major online stores can easily have hundreds of thousands of catalog products, as of 2018 Spotify has run out of recommended products. 35 million pieces. This can make the implementation of academic approaches particularly difficult. In addition, new songs are constantly being published and added to the list. And for at least some genres of music, the freshness or regency of recommendations can be an important quality factor to consider. In addition, the metadata for the items in the list can be very noisy and incomplete and can require significant effort to achieve. Clean it up and derive the missing information, especially since a huge number of new stories appear every year.

Preferences: Users of the Ringo system were asked to indicate their preferences for 125 artists (popular and random). Although some current systems (such as Microsoft Groove) also ask users to provide initial preferences for artists and genres, music recommenders often have to rely on implicit preference signals in listening logs, sometimes with explicit statements such as "skip" features. Apart from the problem of correctly interpreting the large amount of implicit feedback, an additional challenge in this context is that preferences can change over time.

Iterative recommendations: Many recommendation algorithms, especially those based on an abstraction of the matrix completion problem, aim to predict. Importance untraced however, listening to the same songs over and over again is

common in the music industry. If such repetitive consumption is to be supported, algorithmic approaches must be able to decide which songs to recommend repeatedly and when to recommend those songs. Instant consumption and feedback: Unlike many e-commerce domains, recommendations made on a music streaming service can be instantly "consumed" by listeners using, for example, a personal radio station. The biggest challenge in this context is that the system should be able to give the user an opportunity to "correct" recommendations or give feedback (e.g. with a like or dislike button).

The Music we want to listen to can be highly context dependent. Relevant contextual factors can be, for example, the mood of the user, the time of day, or whether the user is listening to music alone or in a group. Capturing and considering these contextual factors can be critical to the perception and acceptance of quality referrals. Purposes of listening to music: One of the characteristics of related music is that you often listen to music with a very specific purpose: to create a certain atmosphere. For a party, get motivated to wash the dishes, improve the experience of reading a good book, relax before bed, etc. This means that recommended items must fit not only the current context, but also the user's goal. Musical taste and expressed preferences can have social effects: the type of music we like and listen to depends not only on our own mood, but can also be significantly influenced by our social environment ("social inclusion") and/or community trends such as. A collection in some scenarios, therefore, it can be particularly useful to consider the user's social environment and corresponding past behavior in the recommendation process. While users share their tastes and preferences on social networks, it is not always clear whether people are really listening to what they publicly "like" or are using their public profiles to create a desired image of themselves.

1.5 Structure of Report

We have reached a major milestone with the completion of Chapter One of our Music Recommendation System project. The core idea of our music suggestion software has been explained in this chapter. The first section goes into great detail about how we came up with ideas and investigated several methods for recommending music. The problem statement highlights the possible social advantages of this application while succinctly articulating the challenges we hope to solve. We give a thorough explanation of the inspiration for the creation of our music recommendation algorithm under the Motivation category. If one looks further into Chapter One, the project goals are presented in detail. This part includes managerial targets, academic goals, and research objectives that are crucial to determining the course of our project. To ensure a complete grasp of potential obstacles, we also explore the potential difficulties that might emerge during the application's development and usage phases.

The project report's following chapters will proceed in the following order:

1.5.1 Chapter 2: Technology Background

Using knowledge from the body of research and literature already available in the subject, this chapter will explore a number of different music recommendation algorithms and techniques in detail.

1.5.2 Chapter 3: Requirements & Methodology

We will describe here how we collected music data, the sources we used, and the preparation techniques we used. This chapter will also cover the specifics of the functional and non-functional requirements that direct the development process.

1.5.3 Chapter 4: Project Plan & Initial Design

This chapter will give an overview of the system's architecture and functional layout, as well as a rough concept for our music recommendation system.

1.5.4 Chapter 5: Project Design & Development

This chapter will give an overview of the system's architecture and functional layout, as well as a rough concept for our music recommendation system.

1.5.5 Chapter 6: Testing

We will construct test cases in this chapter.

- i. Perform front end (design testing), which may include user control testing, spelling checks, and alignment, among other things.
- ii. Carry out backend testing (source code)
- iii. Use a tool to conduct testing and incorporate the results in report.

1.5.6 Chapter 7: Conclusion

We will wrap up our report by reviewing the main conclusions, going over some of the drawbacks, and suggesting future directions for improvement and study in the area of music recommendation systems.

CHAPTER – 2

2. Technology Background

Technology Background offers a thorough rundown of the essential ideas and methods applied in music recommendation systems. The importance of music recommendation in the digital age is first discussed, and then the two main strategies—collaborative filtering and content-based filtering—are examined. The chapter explores the workings of collaborative filtering, focusing on similarities between items and the use of similarity measures like Pearson correlation coefficient and cosine similarity. The topic of content-based filtering is next covered,

With an emphasis on applying TF-IDF factorization to extract useful features from music data. Hybrid techniques that integrate content-based and collaborative filtering are also discussed, along with related issues and concerns including privacy and scalability. A summary of cutting-edge methods and new directions in music recommendation system research is provided at the end of the chapter.

2.1 Background of the technology:

The online application provides a smooth user experience because it is based on a strong recommender system designed specifically for music lovers. Users are met with a secure login page as soon as they access the site, guaranteeing customized interactions. A large search bar makes it easy for customers to comb through a vast

music library by using keywords to find their chosen songs. The application's user-friendly playlist management features, which let users effectively select and arrange their music libraries, further increase user engagement.

The platform enhances user pleasure and music discovery by providing personalized song recommendations through the use of sophisticated algorithms. Transitions between features are made easier with streamlined navigation elements, such as a handy home button. Furthermore, safe session management is ensured via a separate logout button. To receive a unique identifier, each user must complete a brief registration process. This allows for a personalized music experience that is catered to the tastes and preferences of each individual.

2.1.1 Visual Studio

For a number of strong reasons, we have chosen Visual Studio as our development environment. First off, Visual Studio provides strong support for Python development, including an extensive feature set and set of tools designed to maximize efficiency and optimize the development process. Its easy interaction with the Flask framework and Python guarantees a productive coding environment and a smooth workflow.

Furthermore, developers can tailor Visual Studio's vast plugin ecosystem to their own project needs, which makes it easier

to integrate Flask and Jinja 2 for template rendering. Furthermore, Visual Studio's user-friendly interface and strong debugging features facilitate quick testing and iteration, guaranteeing the dependability and stability of our web application. Last but not least, its interoperability with SQLite3, the local database management system of our choice, makes database operations simpler and enables smooth data administration within the program. All things considered, Visual Studio seems to be the best option for our project, providing a stable and

feature-rich development environment that is customized to meet our demands for Python-based web applications.

2.1.2 Python

Python is a popular high-level interpreted programming language that is easy to learn and understand. Guido van Rossum was the creator, and it was originally published in 1991. Python is a great option for both novice and seasoned developers since it places a strong emphasis on readable code and simple grammar. It is compatible with procedural, object-oriented, and functional programming paradigms, among others. These are Python's five main characteristics:

Identifying Features:

- i. **Readability:** Python places a strong emphasis on legible and well-organized code syntax, which improves teamwork and lowers maintenance costs.
- ii. **Dynamically Typed:** Python's dynamic typing enables variable declarations that are flexible and don't require explicit type annotations, which encourages experimentation and quick development.
- iii. **Large Standard Library:** Python has an enormous standard library that covers a wide range of functions, reducing the need for outside dependencies and speeding up development.
- iv. **Interpreted and Interactive:** Python's interpretive design allows interactive mode to be supported and instantaneous code execution, which speeds up debugging and prototyping.
- v. **Cross-platform:** Python's compatibility with several platforms guarantees a smooth implementation on

various operating systems, augmenting accessibility and adaptability for both developers and users.

2.3 Literature Review:

In today's music streaming services, recommender systems have become essential tools that have completely changed the way users find new music that suits their tastes. Collaborative filtering (CF) and content-based filtering (CBF) are two of the most widely used strategies in these systems. With its reliance on user behavior data, CF has been the subject of much research in the last few years. Research has investigated sophisticated matrix factorization methods to improve recommendation accuracy, including temporal dynamics. Notwithstanding its efficacy, CF is not without problems. These include the data sparsity problems in user-item interaction matrices and the cold-start problem for new users. Researchers are using hybrid techniques that integrate CBF and CF to overcome these issues.

Conversely, CBF uses intrinsic properties of musical pieces, like pace, timbre, and genre, to produce suggestions. The effectiveness of CBF in capturing user preferences based solely on content elements was shown in early tests. Nevertheless, CBF may face challenges because to its narrow diversity of recommendations and its incapacity to adjust to changing user preferences over time. One potential way to get around these restrictions is to use hybrid recommender systems, which combine collaborative strategies with content-based features. Hybrid systems provide more reliable and precise recommendations by utilizing the advantages of both CF and CBF, hence accommodating a broader spectrum of customer preferences.

Optimizing music recommender systems still presents issues, notwithstanding the progress made in CF and CBF approaches. These include incorporating contextual data, such user mood and location, improving algorithms to manage large-scale data effectively, and addressing privacy issues with user data. Furthermore, scalability and real-time performance are important factors to take into account, particularly for platforms that serve millions of users at once. Prospective avenues for research could center around utilizing cutting-edge technology such as deep learning and natural language processing to augment the precision and customization of recommendations. All things considered, content-based and collaborative filtering will remain essential in influencing the direction of music recommendation systems, spurring innovation, and raising user pleasure in the world of digital music.

CHAPTER – 3

3.1 Introduction:

In this chapter we will discuss about how much work is done on the development of our project according to the project plan. This chapter will cover the in-detail process and objective of the project. As our app is built on windows that are being reused within the system itself the developers had to take a systematic approach for the app to work smoothly. Our project plan is strategically planned with the Gantt chart and other organizational tools. Each activity has specific time period allotted according to the complexity of the task, which is why the days in work may vary.

We will also discuss in detail about the Functional, Non-Functional and Hardware requirements of our project. The functional requirement are taken in to full consideration as they are the necessary part in order to get the basic requirement by the project such as, user login, playlist management, recommendation for music, while on the other hand, the non-functional requirement such as, performance, user experience and usability are also thoroughly planned.

3.2 Project Plan:

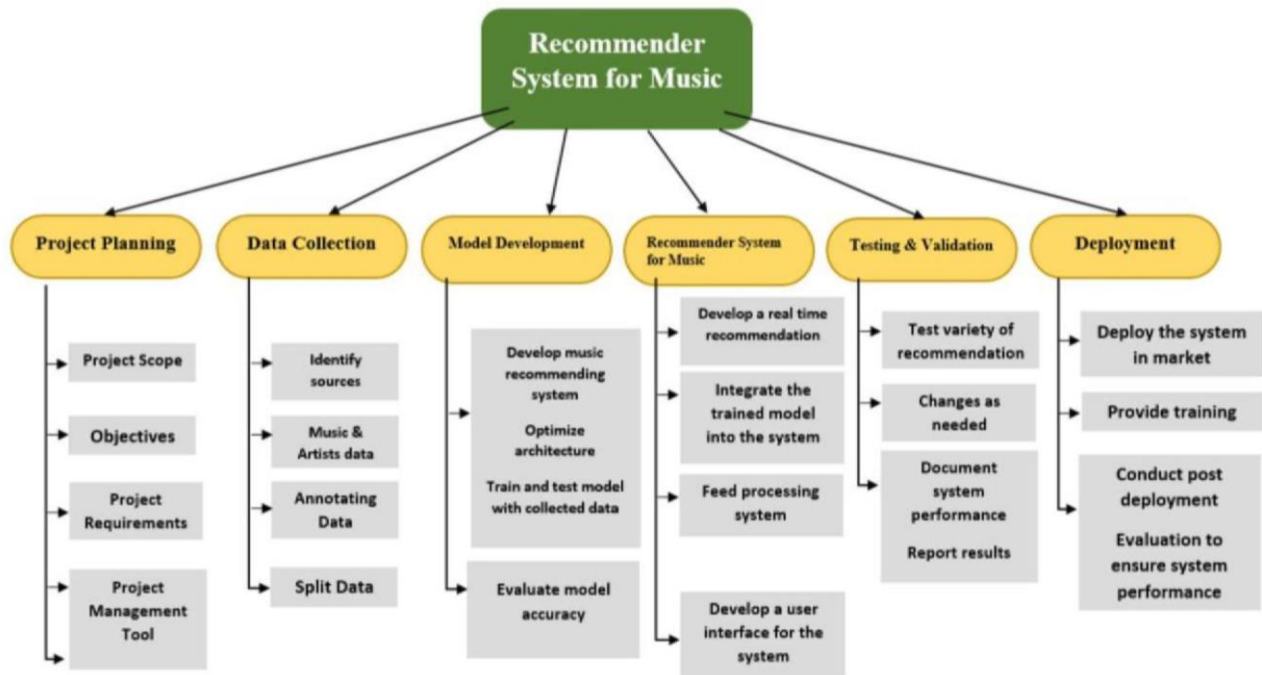


Figure 1: Project Plan

1	3/7/2023	4/1/2023	Data Collection
2	4/2/2023	5/1/2023	Algorithm Research
3	5/2/2023	6/1/2023	Algorithm Development
4	6/2/2023	7/3/2023	Training
5	7/4/2023	8/5/2023	Validation
6	8/6/2023	9/5/2023	Real-Time Implementation
7	9/6/2023	10/7/2023	User Interface Design
8	10/8/2023	11/9/2023	Performance Optimization
9	11/10/2023	12/7/2023	Deployment
10	12/8/2023	1/7/2024	Testing and Evaluation
11	1/8/2024	2/9/2024	Documentation
12	2/10/2024	2/28/2024	Maintenance and Updates

Figure 2: Gantt Chart

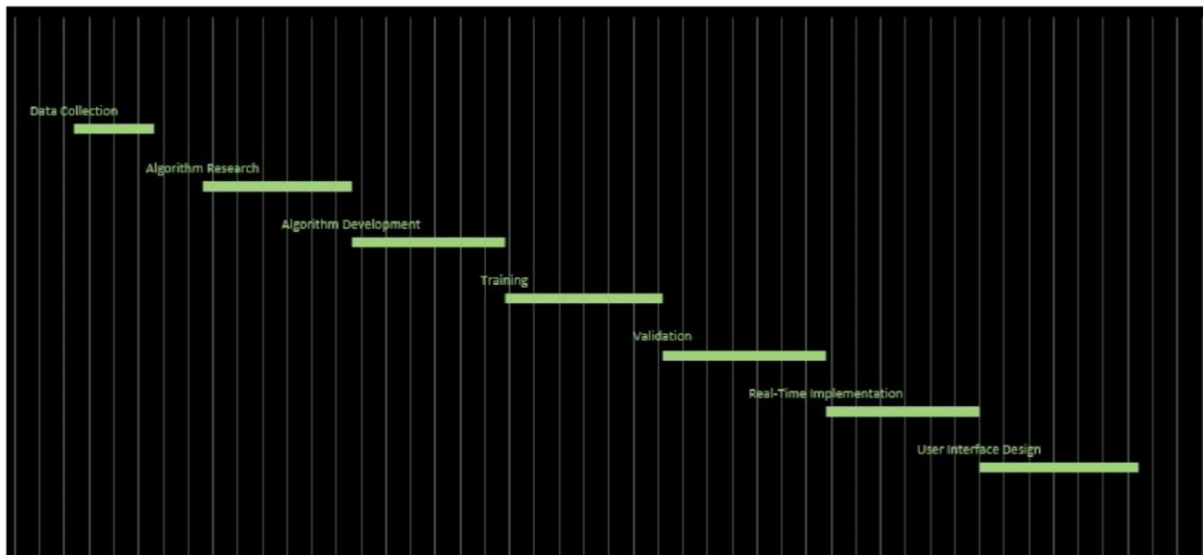
3.3 Functional Requirements:

1. User Registration and Authentication:

- Allow users to create accounts and authenticate themselves to access personalized recommendations.
- To authenticate, utilize safe methods like username/password, social network login, or email confirmation.

2. User Profile Management:

- Enable users to create and manage their profiles, including preferences, favorite genres, and artists.
- As consumers' musical choices change, let them update their profile information.



3. Music Recommendation Generation:

- To provide individualized music suggestions, analyse user data, including listening history, preferences, and contextual information.
- To provide precise and varied recommendations, combine content-based filtering and collaborative filtering strategies.
- To make choices that are appropriate, take into account elements like genre, artist, mood, tempo, and popularity.

4. Recommendation Display and Exploration:

- Make recommendations to consumers in a way that is both aesthetically pleasing and user-friendly.
- Display suggested tracks, albums, or playlists together with any supporting metadata (such as the artist name, album cover, and the date of release).

- By giving users the opportunity to play, store, or add suggested tracks to their playlists, you may encourage users to explore recommended music.

5. Recommendation Filtering and Customization:

- Ensure that customers have the option to filter recommendations based on details such the genre, artist, release date, or user popularity.
- Give consumers the ability to customize the combination of tailored recommendations and chance finds.
- Enable users to specify their preferences or restrictions, such as excluding certain genres or artists from recommendations.

6. Playlist Management:

- Providing users with the ability to create, change, and manage their playlists.
- Permit users to directly add suggested music to their playlists or make new playlists based on suggestions.
- Whenever necessary, offer alternatives to rename playlists, rearrange music, or eliminate playlists.

3.4 Non-Functional Requirements:

1. Performance:

- a. The system should react fast to user input, offering suggestions immediately or as soon as possible.
- b. The process of creating suggestions should be effective, processing vast volumes of user data, and providing recommendations in a reasonable amount of time.
- c. The system must be scalable, able to handle rising user traffic and data volumes without noticeably degrading performance.

2. Accuracy and Relevance:

- a. The system's recommendations must be both accurate and pertinent, closely matching the user's Preferences and interests.
- b. In order to maintain a high level of suggestion quality, the system should reduce instances of irrelevant Or mismatched recommendations.
- c. The efficiency of the recommendation algorithms should be evaluated using evaluation criteria like Precision, recall, and accuracy.

3. User Experience and Usability:

- a. Users should be able to access and engage with the suggestion features with ease thanks to the system's straightforward and user-friendly design.
- b. The information presented in the suggestion display should be concise and wellorganized, and it should be visually appealing.
- c. A seamless and enjoyable user experience should be supported by the system, encouraging user pleasure and enjoyment throughout the music discovery process.

4. Adaptability and Learning:

- a. The quality and accuracy of recommendations should be continuously improved by the system as it adapts and learns from user interactions and feedback.
- b. The algorithms for making recommendations should be adaptable, able to take into account changes in user tastes and adapt to changing musical trends.

- c. To ensure that recommendations are current and pertinent, the system should be able to accept new information and updates.

3.5 Hardware Requirements:

1. **Device:** Any modern device with web browsing capability (laptop, desktop, tablet, smartphone).
2. **Operating System:** Compatible with Windows, macOS, Linux, iOS, and Android.
3. **Processor:** Adequate processing power for web browsing and media playback.
4. **Memory (RAM):** Minimum 2 GB RAM for smooth operation and content loading.
5. **Internet Connection:** Stable connection with sufficient bandwidth for streaming and interaction.
6. **Screen Resolution:** Support for common resolutions across devices and screen sizes.
7. **Audio Output:** Compatibility with speakers or headphones for listening to music.
8. **Optional:** Additional hardware support for advanced features like high-resolution audio or smart device integration.

3.6 Summary:

In this chapter, a detailed Project Plan, Functional, Non-Functional requirements and other planning mechanisms are discussed in detail that will be required in our project. We have also mentioned an introduction regarding our web application how we can perform our task so we make a milestone chart in this first we describe our task week wise in summary activity and then we make a Gantt chart according to summary activity. In Gantt chart we were describing task name or duration for implementation of our “Recommender system for music” After Gantt chart we describe Non-Functional requirements of our web application the requirements are performance and user experience.

CHAPTER – 4

4.1 Introduction:

In this chapter, we will talk about the design and specification of our project. Using diagrams, we will elaborate on our project in great detail by gathering all the information necessary for our application and then setting up a framework that will make the application's flow easy to understand. To help the user grasp the entire flow of our application, we have employed a variety of diagrams. In this stage, we also go into great detail about entity relationship diagrams (ERDs), data flow diagrams (DFDs), and UML in respect to our application. The system work flow and specifications for our user-friendly web application are depicted in these diagrams. It also uses a flowchart to demonstrate how each screen work flow functions. Development has begun in order to ensure that the user can use it when all the information has been obtained and the design has been established.

The goal of creating the entity and data flow diagrams is to illustrate the precise implementation flow of our program and to direct the course of our system, i.e., how we carry out each task. For users, this will give them a comprehensive understanding of the system's overall coding in depth. Every diagram includes every functional input and output of the system in detail, ensuring that everything works as it should.

4.2 Data Flow Diagram:

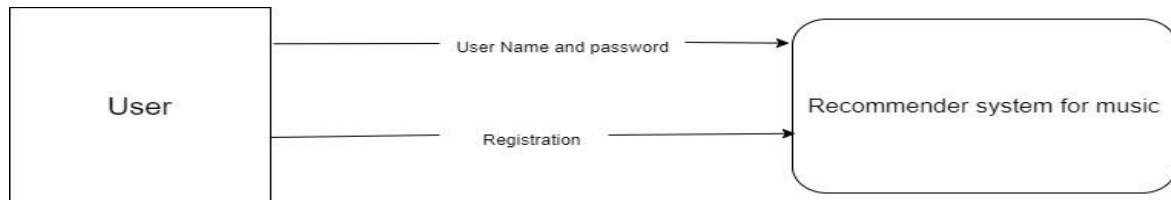


Figure 3 : DFD

Figure 3: DFD

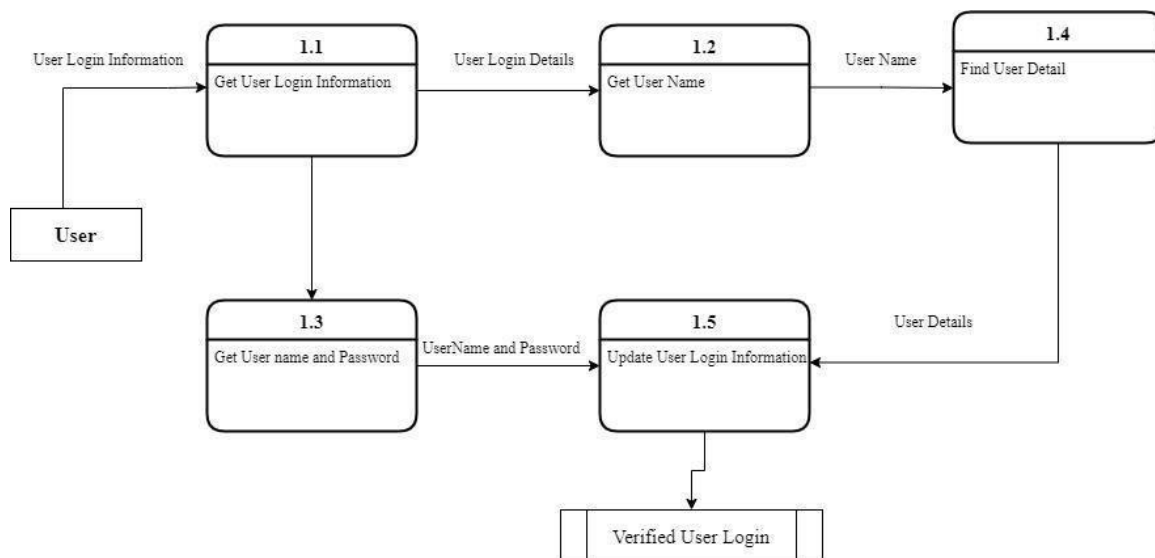


Figure 4: Data Flow Diagram Level 1

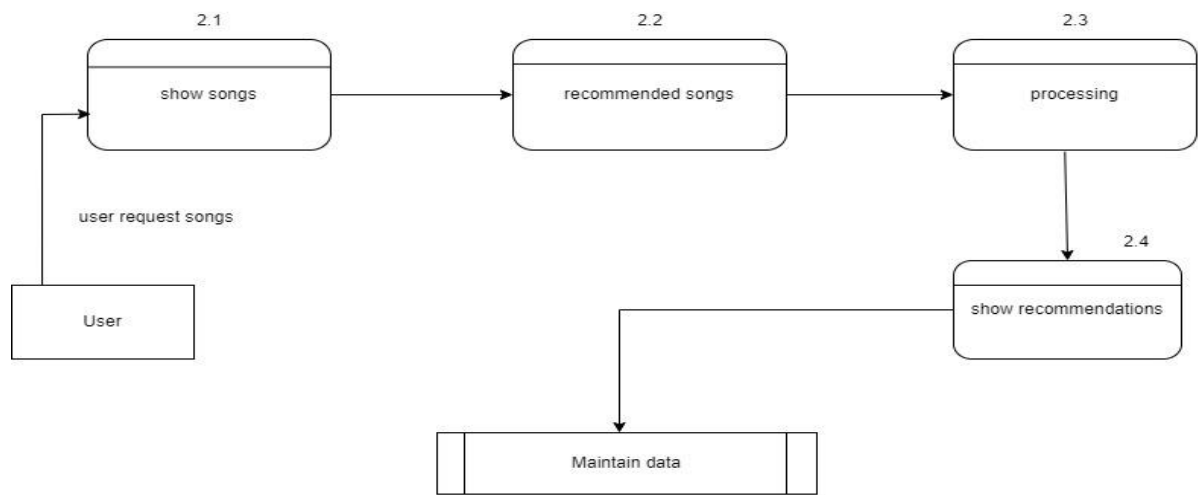


figure 5 : Data Flow Diagram Level 2

Figure 5: Data Flow Diagram Level 2

4.2 Entity Relationship Diagram

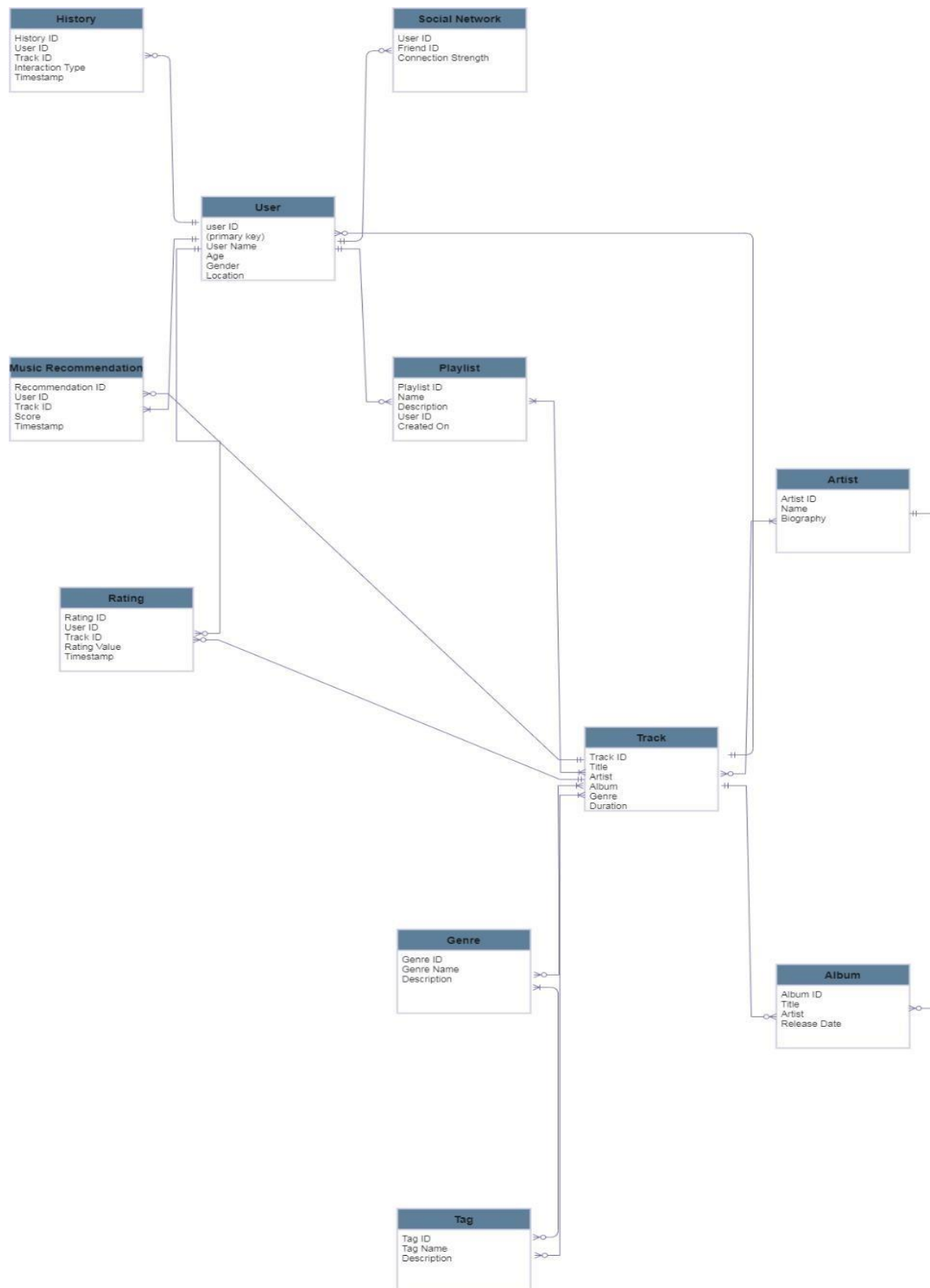
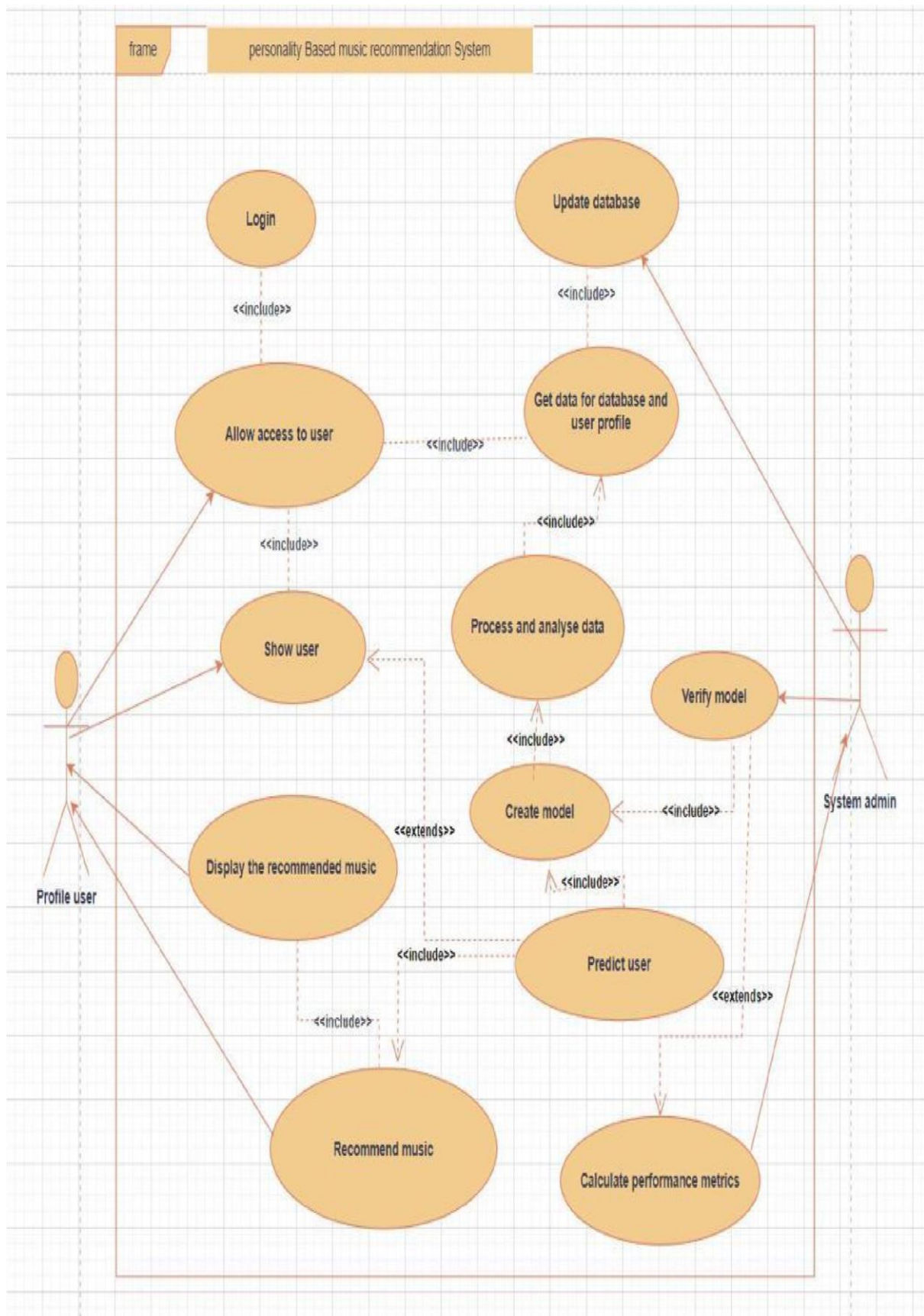


Figure 6: Entity Relationship Diagram

4.3 Use Cases



Use Case Narratives

User Registration and Profile Creation:

Use Case	User Registration and Profile Creation:
Use Case ID	UC001
Actor	User
Description	<ul style="list-style-type: none"> ➤ As a new user, I want to register an account on the music recommender system. ➤ After registration, I want to create my user profile by providing information about my music preferences, favorite genres, and artists. ➤ The system should guide me through the profile creation process and allow me to save my preferences for personalized recommendations.

Discovering New Music:

Use Case	Discovering New Music
Use Case ID	UC002
Actor	User
Description	<ul style="list-style-type: none"> ➤ As a user, I want to explore and discover new music that aligns with my tastes. ➤ I navigate to the "Discover" section of the system and am presented with a selection of recommended tracks or playlists. ➤ I can filter recommendations by genre, mood, or other criteria to narrow down my options. ➤ I can listen to preview snippets of recommended tracks, read descriptions, and access additional information about the recommended music.

Personalized Recommendations:

Use Case	Personalized Recommendations
Use Case ID	UC003
Actor	User
Description	<ul style="list-style-type: none">➤ As a user, I want to receive personalized music recommendations based on my preferences.➤ I visit the recommendation section, and the system displays a list of personalized recommendations tailored to my music taste.➤ The recommendations are based on my listening history, favorite genres, and the preferences of users similar to me.➤ I can interact with the recommendations, such as saving tracks to my library or adding them to playlists.

Creating and Managing Playlists:

Use Case	Creating and Managing Playlists
Use Case ID	UC004
Actor	User
Description	<ul style="list-style-type: none">➤ As a user, I want to create and manage playlists to organize my favorite tracks.➤ I navigate to the playlist management section and create a new playlist, providing a name and optional description.➤ I can search for tracks by title, artist, or genre and add them to my playlist.➤ I have the ability to reorder tracks, remove tracks, and edit playlist details as needed.

Social Sharing and Interaction:

Use Case	Social Sharing and Interaction:
Use Case ID	UC005
Actor	User
Description	<ul style="list-style-type: none">➤ As a user, I want to share my favorite tracks or playlists with my friends or social media followers.➤ I select a track or playlist I want to share and use the system's social sharing feature to post it on my preferred social media platform.
	<ul style="list-style-type: none">➤ I can also follow other users with similar music tastes and discover new tracks or playlists through their recommendations.

Providing Feedback and Rating:

Use Case	Providing Feedback and Rating
Use Case ID	UC006
Actor	User
Description	<ul style="list-style-type: none">➤ As a user, I want to provide feedback and ratings on recommended tracks or playlists.➤ I can rate tracks or playlists using a star rating system or provide written reviews.➤ The system uses my feedback to refine future recommendations and improve the accuracy of the recommendation algorithms.

4.5 Summary:

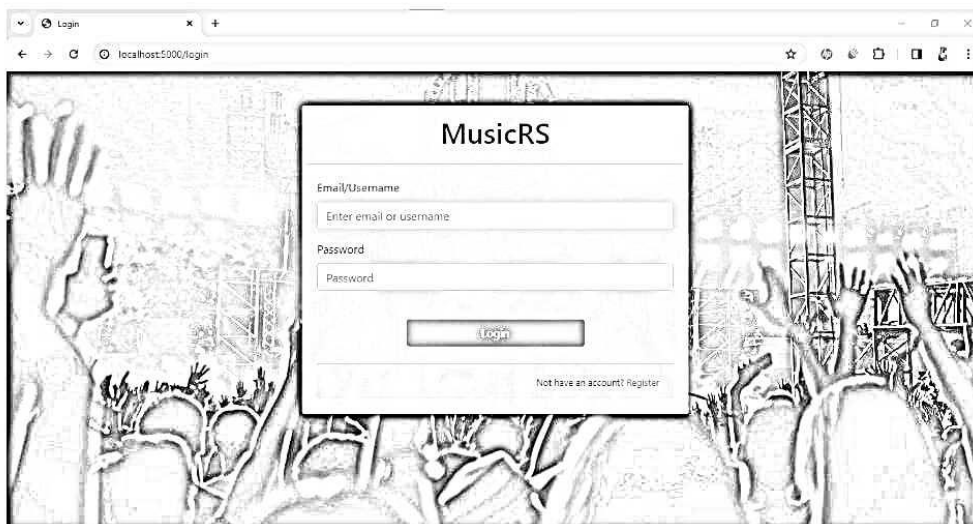
We explore our project's design and specification in this chapter in an effort to provide readers a thorough grasp of its architecture and functionality. We carefully lay out the flow of our application using a variety of diagrams, such as entity relationship diagrams (ERDs), data flow diagrams (DFDs), and Unified Modeling Language (UML). The purpose of these diagrams is to provide users with a clear and simple understanding of the system's specs and process. We also use flowcharts to demonstrate how each screen in the application operates in sequence. Our priority continues to be making sure the application is effective and user-friendly as we move forward with development, with the eventual objective of giving users a simple and intuitive experience. The entity and data flow diagrams play a crucial role in directing the implementation process and guaranteeing that all of the system's functional inputs and outputs are taken into account, which guarantees smooth operation and peak performance.

CHAPTER – 5

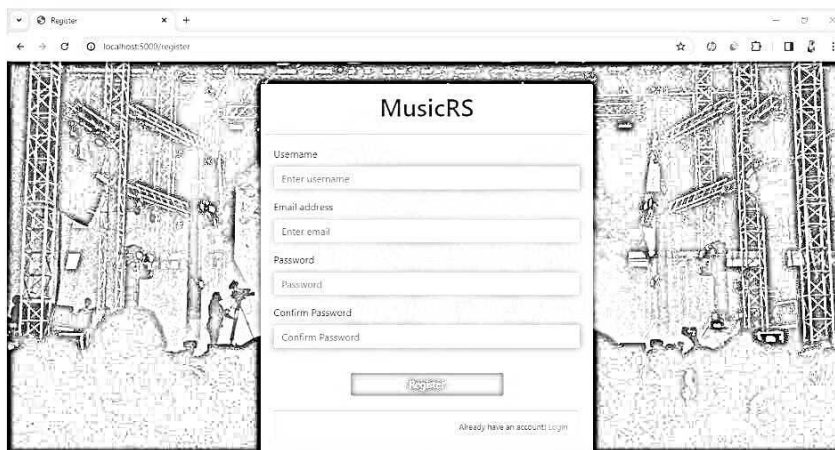
5.1 Introduction:

In this chapter we are discussing aspects which are used in our project, and prototype design which is generally used to evaluate a new design to enhance precision by system analysts and users and frontend and backend design of our project. We are also discussing about the database queries which are used in Sql Lite 3 and some external libraries and we are showing screenshots of our application as it fulfills user requirements. We have briefly provided a few clarifications about the sort of functionalities available on the system, source code of validation and etc.

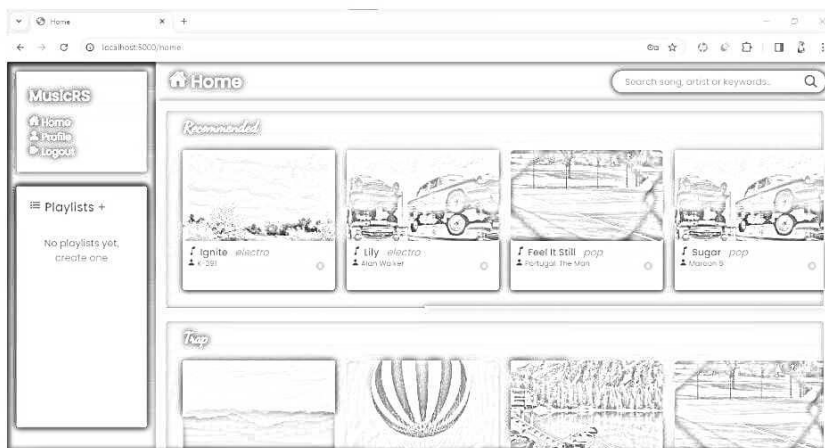
5.2 Prototype Design:



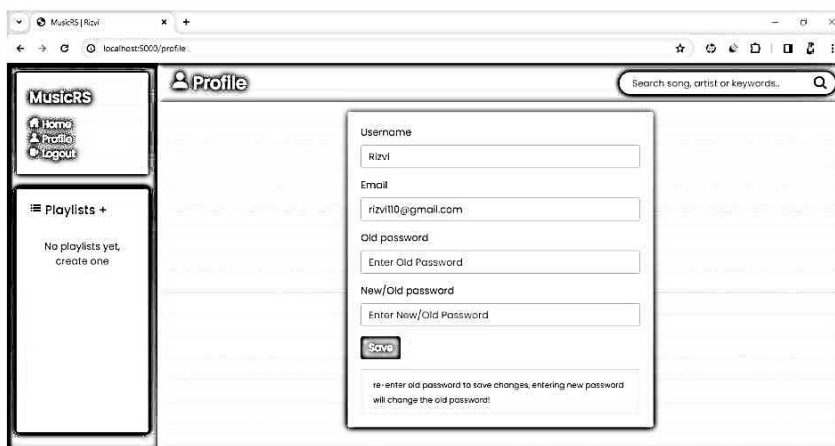
Prototype 1



Prototype 2



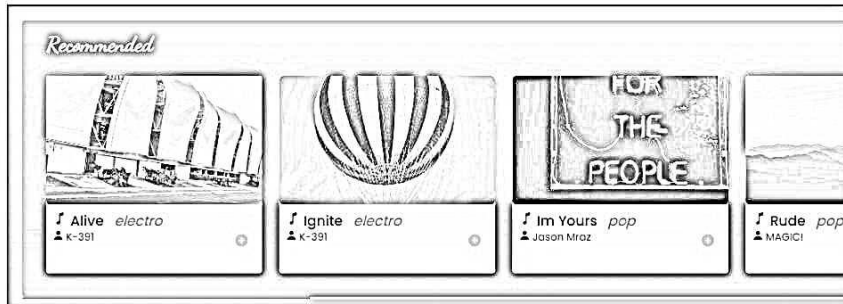
Prototype 3



Prototype 4



Prototype 5

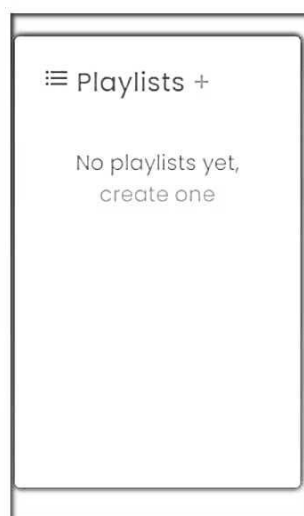


Prototype 6



Prototype 7

Playlist Creation



Prototype 8



Admin Panel User ID Showcasing

Prototype 9

5.3 Database Queries:

```

conn = sqlite3.connect('database.db')
c = conn.cursor()
c.execute("CREATE TABLE IF NOT EXISTS users
(id INTEGER PRIMARY KEY AUTOINCREMENT,
username TEXT NOT NULL UNIQUE,
email TEXT NOT NULL UNIQUE,
password TEXT NOT NULL,
ratings TEXT,
playlists TEXT)")
c.execute("CREATE TRIGGER IF NOT EXISTS set_initial_id
AFTER INSERT ON users
BEGIN
    UPDATE users SET id = (SELECT MAX(id) FROM users) - 1 WHERE id = (SELECT MAX(id)
FROM users);
END;")
try:
    c.execute("INSERT INTO users (username, email, password, ratings, playlists) VALUES (?, ?, ?, ?, ?)",
("admin", "admin@musicrs"
,"scrypt:32768:8:1$HN8OL1ZaEVTH6JBN$1f9c8f3e9ce80e143b9cb38f038d7b7472988a8d41d5ec1e023b6c6d
172be32c381fa3aea234116002f37764563cc096909eaba342b4e0101c35cb3121f2e175",json.dumps([]),json.dum
ps([])))
except:
    pass
conn.commit()

app = Flask(__name__)
app.config['SECRET_KEY'] = 'MusicRS'

def get_admin(username):
    conn = sqlite3.connect('database.db')
    c = conn.cursor()
    c.execute("SELECT * FROM users WHERE username=?", (username,))

```

```

admin = c.fetchone()
conn.commit()
return admin

def delete_user(id):
    conn = sqlite3.connect('database.db')
    c = conn.cursor()
    c.execute("DELETE FROM users WHERE id = ?",(id,))
    conn.commit()

def edit_user(id,username,email,password):
    conn = sqlite3.connect('database.db')
    c = conn.cursor()
    if username != "":
        try:
            c.execute("UPDATE users SET username = ? WHERE id = ?",(username,id,))
        except:
            conn.commit()
            return "Cannot edit because username is already taken!"
    if email != "":
        try:
            c.execute("UPDATE users SET email = ? WHERE id = ?",(email,id,))
        except:
            conn.commit()
            return "Cannot edit because email is already taken!"
    if password != "":
        c.execute("UPDATE users SET password = ? WHERE id = ?",(generate_password_hash(password),id,))
    conn.commit()
    return "ok"

def get_users():
    conn = sqlite3.connect('database.db')
    c = conn.cursor()
    c.execute("SELECT * FROM users")
    users = c.fetchall()
    users = [i for i in users if 'admin' not in i]
    conn.commit()
    return users

def get_user(username):
    conn = sqlite3.connect('database.db')
    c = conn.cursor()
    c.execute("SELECT * FROM users WHERE username=?", (username,))
    user = c.fetchone()
    conn.commit()
    return user

def rate_song(userid,songid,rating):

```

```

conn = sqlite3.connect('database.db')
c = conn.cursor()
c.execute("SELECT * FROM users WHERE id=?", (str(userid)))
user = c.fetchone()

if user:
    ratings = json.loads(user[4])
    co = True
    for i in ratings:
        if str(songid) in i.keys():
            i[str(songid)] = int(rating)
            co = False
            break
    if co:
        ratings.append({str(songid):int(rating)})
    c.execute("UPDATE users SET ratings = ? WHERE id=?", (json.dumps(ratings),str(userid)))

c.execute("SELECT * FROM users WHERE id=?", (str(userid)))
user = c.fetchone()
if user:
    ratings = json.loads(user[4])
    print(ratings)
    df = pd.read_csv('ratings.csv')
    df['userid'] = df['userid'].astype(str)
    df['songid'] = df['songid'].astype(str)
    mask = (df['userid'] == str(userid)) & (df['songid'] == str(songid))
    if mask.any():
        df.loc[mask, 'ratings'] = int(rating)
        df.to_csv('ratings.csv', index=False)
        cf.refresh()
    else:
        print(f"User {userid} has not rated song {songid} before.")
conn.commit()

def get_rating(userid,songid):
    conn = sqlite3.connect('database.db')
    c = conn.cursor()
    c.execute("SELECT * FROM users WHERE id=?", (str(userid)))
    user = c.fetchone()
    rating = 0
    if user:
        ratings = json.loads(user[4])
        for i in ratings:
            if str(songid) in i.keys():
                rating = int(i[songid])
                break
    conn.commit()
    return rating

```



```

def check_playlist(userid,name):
    conn = sqlite3.connect('database.db')
    c = conn.cursor()
    c.execute("SELECT * FROM users WHERE id=?", (str(userid)))
    user = c.fetchone()
    exists = False
    #[{n:[{ ]}]}]
    if user:
        playlists = json.loads(user[5])
        for i in playlists:
            if name in i.keys():
                exists = True
                break
    conn.commit()
    return exists

def create_playlist(userid,name):
    conn = sqlite3.connect('database.db')
    c = conn.cursor()
    c.execute("SELECT * FROM users WHERE id=?", (str(userid)))
    user = c.fetchone()
    exists = 0
    #[{n:[{ ]}]}]
    if user:
        playlists = json.loads(user[5])
        playlists.append({ name:[] })
        playlists = json.dumps(playlists)
        c.execute("UPDATE users SET playlists = ? WHERE id=?", (playlists,str(userid)))
    conn.commit()
    return exists

def get_playlists(userid):
    conn = sqlite3.connect('database.db')
    c = conn.cursor()
    c.execute("SELECT * FROM users WHERE id=?", (str(userid)))
    user = c.fetchone()
    #[{n:[{ ]}]}]
    conn.commit()
    if user:
        playlists = json.loads(user[5])
        if len(playlists) > 0:
            _playlists = []
            for i in playlists:
                _playlists.append(list(i.keys())[0])
            return _playlists
    return []

```

```

def add_to_playlist(userid,name,songid):
    conn = sqlite3.connect('database.db')
    c = conn.cursor()
    c.execute("SELECT * FROM users WHERE id=?", (str(userid)))
    user = c.fetchone()
    #[{n:[{ ]}]
    if user:
        playlists = json.loads(user[5])
        for i in playlists:
            if name in i:
                for j in i[name]:
                    if str(j["songid"]) == str(songid):
                        conn.commit()
                        return True
                    i[name].append(songs[songid])
                break
        playlists = json.dumps(playlists)
        c.execute("UPDATE users SET playlists = ? WHERE id=?", (playlists,str(userid)))
    conn.commit()
    return True

```

```

def remove_from_playlist(userid,name,songid):
    conn = sqlite3.connect('database.db')
    c = conn.cursor()
    c.execute("SELECT * FROM users WHERE id=?", (str(userid)))
    user = c.fetchone()
    #[{n:[{ ]}]
    if user:
        playlists = json.loads(user[5])
        for i in playlists:
            if name in i:
                for j in i[name]:
                    if str(j["songid"]) == str(songid):
                        i[name].remove(j)
                        break
        playlists = json.dumps(playlists)
        c.execute("UPDATE users SET playlists = ? WHERE id=?", (playlists,str(userid)))
    conn.commit()
    return True

```

```

def remove_playlist(userid,name):
    conn = sqlite3.connect('database.db')
    c = conn.cursor()
    c.execute("SELECT * FROM users WHERE id=?", (str(userid)))
    user = c.fetchone()
    #[{n:[{ ]}]
    if user:
        playlists = json.loads(user[5])

```

```

    for i in playlists:
        if name in i:
            playlists.remove(i)
    playlists = json.dumps(playlists)
    c.execute("UPDATE users SET playlists = ? WHERE id=?", (playlists, str(userid)))
conn.commit()
return True

def get_playlist(userid, name):
    conn = sqlite3.connect('database.db')
    c = conn.cursor()
    c.execute("SELECT * FROM users WHERE id=?", (str(userid)))
    user = c.fetchone()
    #[{n:[{ ]}]}]
    conn.commit()
    if user:
        playlists = json.loads(user[5])
        for i in playlists:
            if name in i:
                return i[name]
    return []

```

5.4 External Libraries:

1. For Main.py:

```

from flask import Flask, render_template, request, jsonify, flash, redirect, url_for,
make_response

import json
import jwt
import datetime

from werkzeug.security import generate_password_hash, check_password_hash
from functools import wraps
import matplotlib.pyplot as plt
import numpy as np
import time
import io
import sqlite3

from CBFfiltering import *
from CFiltering import *

```

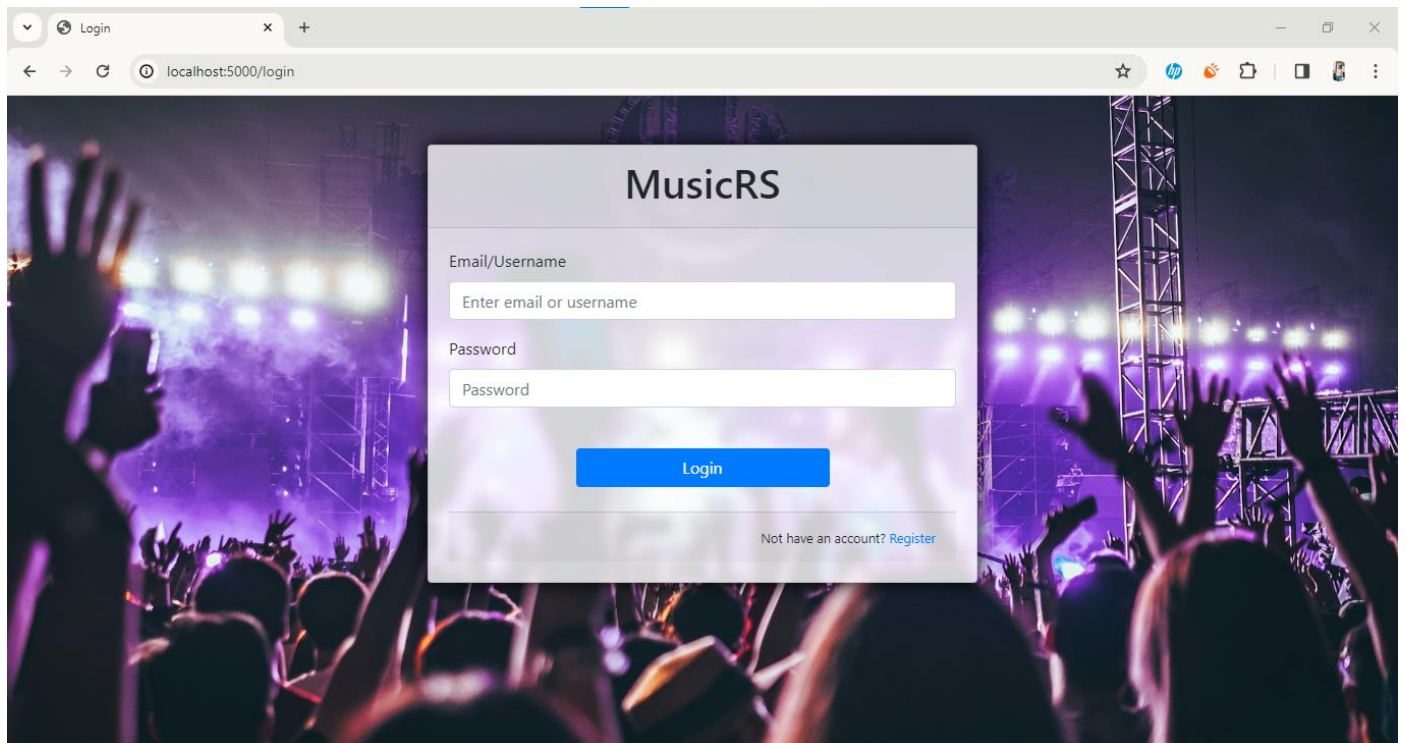
4. For Collaborative Filterings:

```
import pandas as pd  
from nltk.stem import PorterStemmer  
from sklearn.feature_extraction.text import TfidfVectorizer  
from sklearn.metrics.pairwise import cosine_similarity
```

5. For Content Based Filtering:

```
import pandas as pd  
from sklearn.metrics.pairwise import cosine_similarity
```

5.5 Screenshots:



Login Windows

Register

localhost:5000/register

MusicRS

Username

Email address

Password

Confirm Password

Register

Already have an account? [Login](#)

Registration Windows

Home

localhost:5000/home

MusicRS

- Home
- Profile
- Logout


Playlists +

No playlists yet, [create one](#)


Home

Search song, artist or keywords..


Recommended




Ignite *electro*
K-391



Lily *electro*
Alan Walker





Feel It Still *pop*
Portugal. The Man





Sugar *pop*
Maroon 5

Trap









Home Windows

MusicRS | Rizvi

localhost:5000/profile

Profile

Search song, artist or keywords..

MusicRS

- Home
- Profile
- Logout

Playlists +

No playlists yet,
[create one](#)

Username

Rizvi

Email

rizvil10@gmail.com

Old password

Enter Old Password

New/Old password

Enter New/Old Password

Save

re-enter old password to save changes, entering new password will change the old password!


Profile Windows

Home


Search song, artist or keywords..

Search Bar


Recommended




Alive *electro*
K-391



Ignite *electro*
K-391



Im Yours *pop*
Jason Mraz



Rude *pop*
MAGIC!

Recommended Songs

Rap



Still Wiz
Wiz Khalifa



See You Again
Wiz Khalifa



Millions
Wiz Khalifa



This Time Around
Wiz Khalifa

Rap Songs

Pop



Perfect
Ed Sheeran



Shape of You
Ed Sheeran

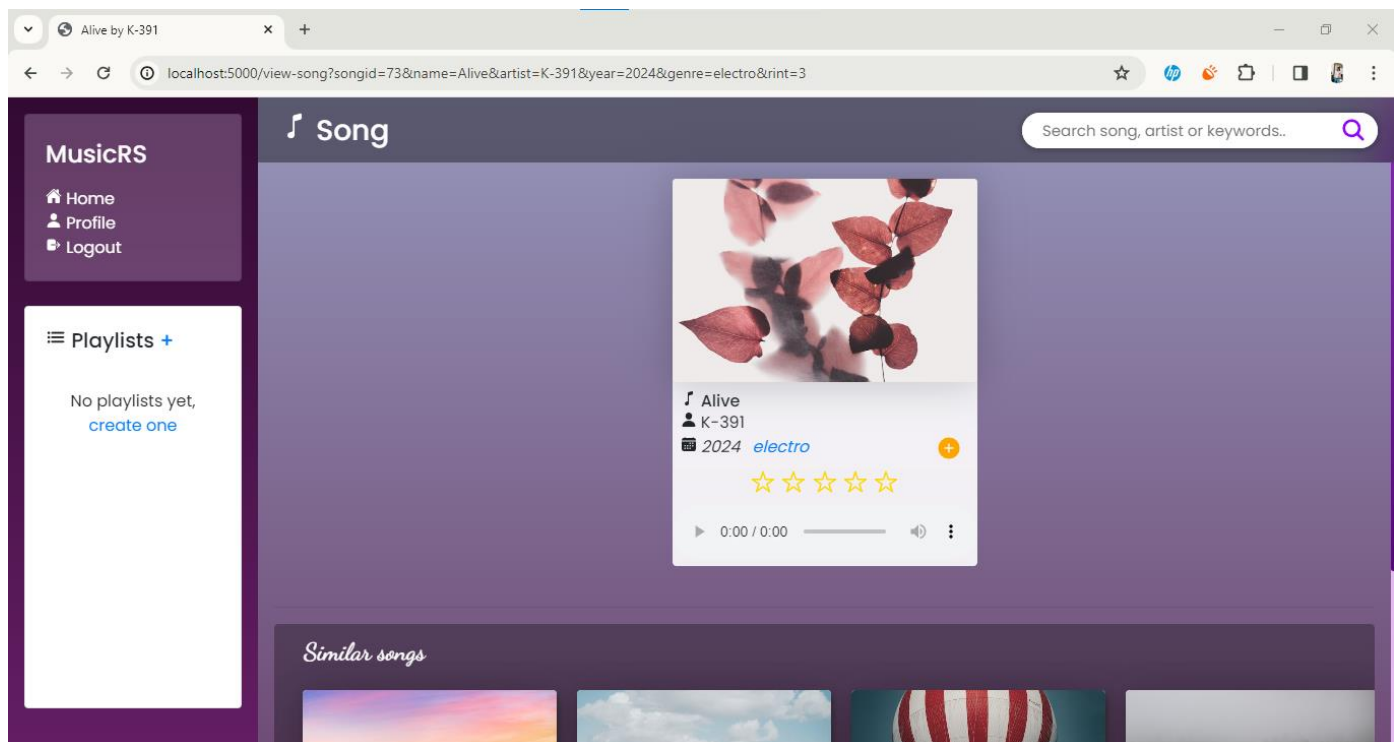


What Do I Know
Ed Sheeran

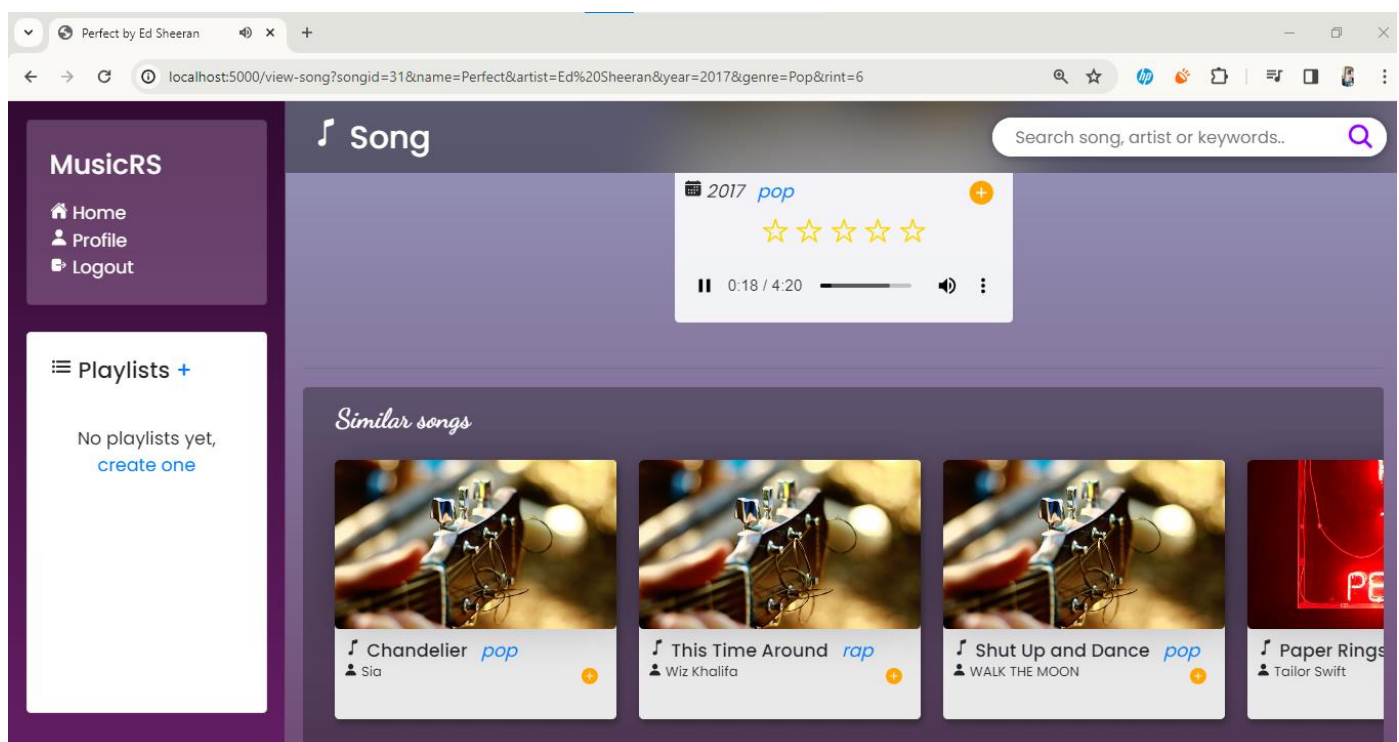


Dive
Ed Sheeran

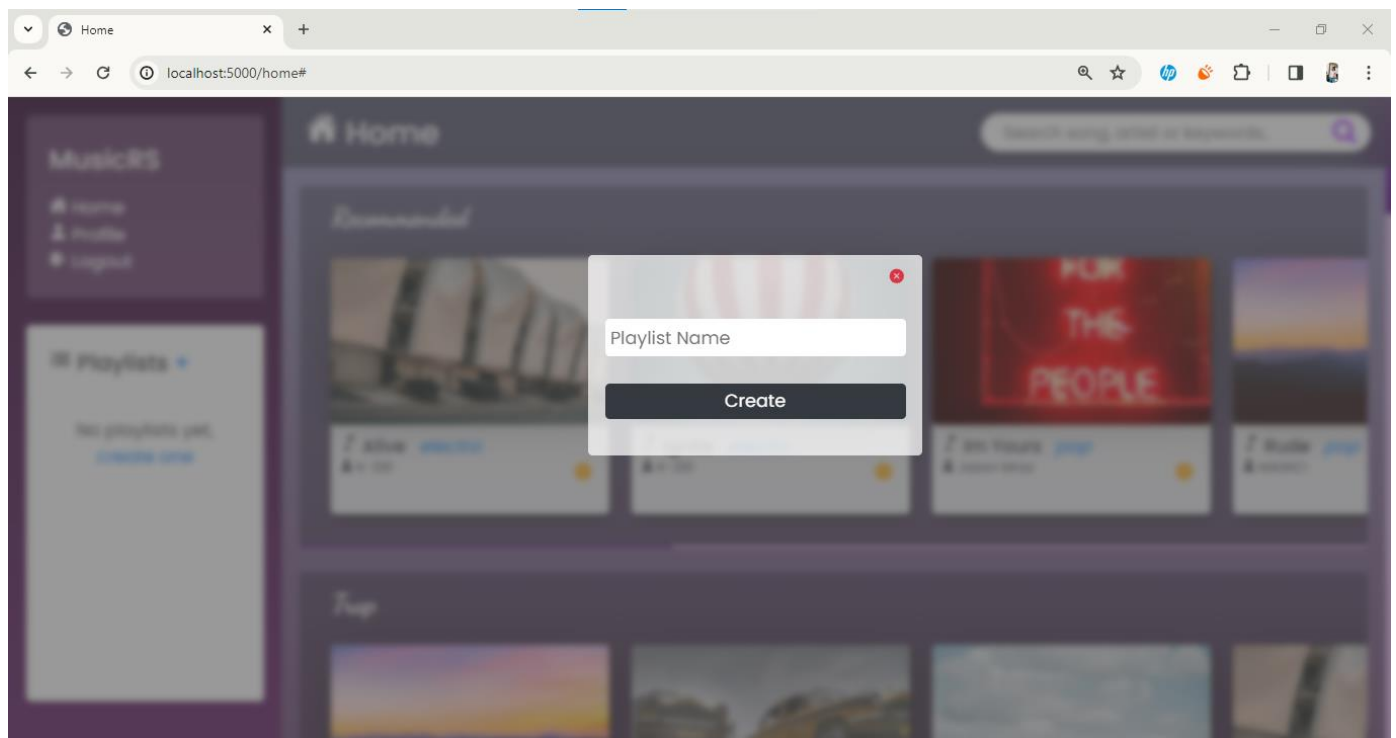
Pop Songs



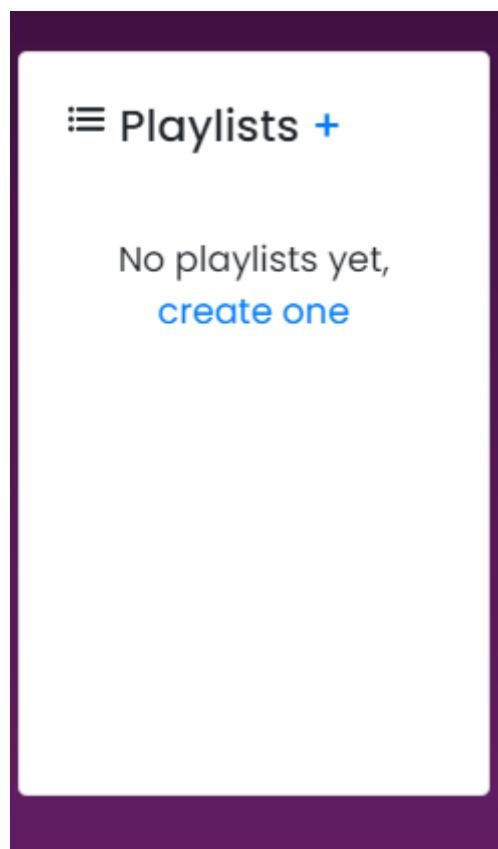
Playing Songs



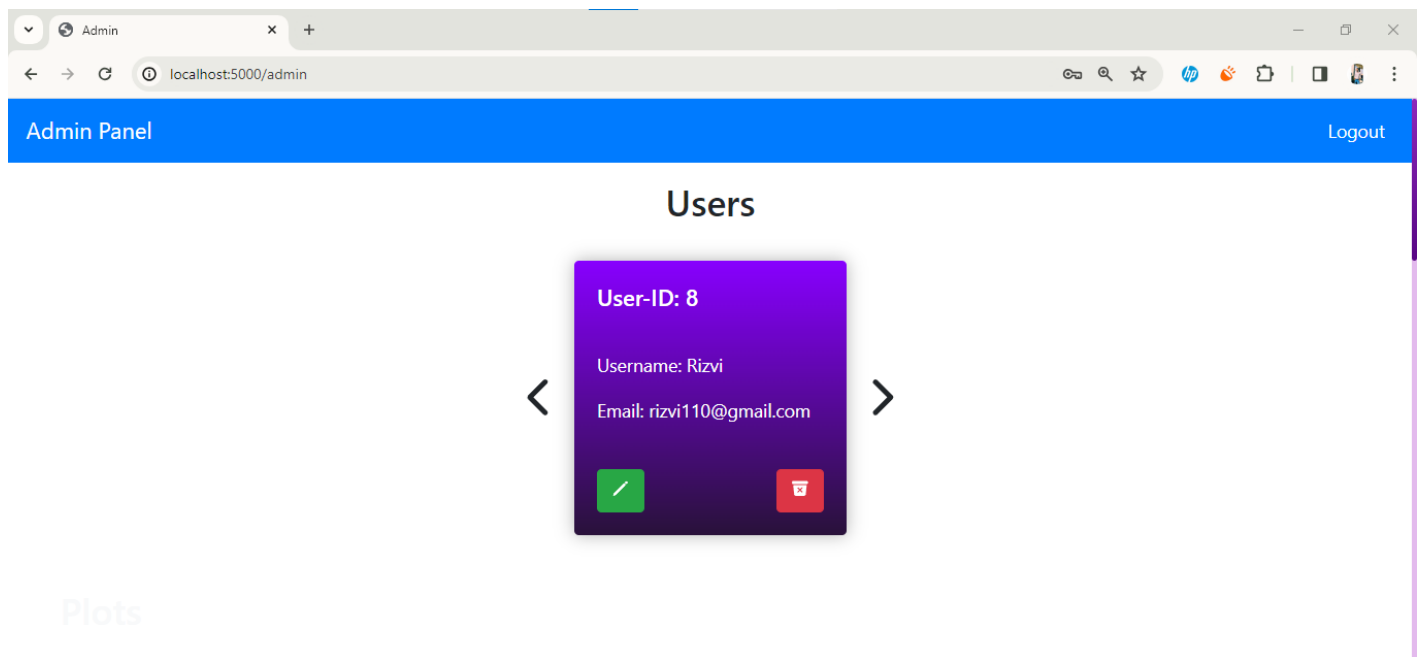
Similar Songs



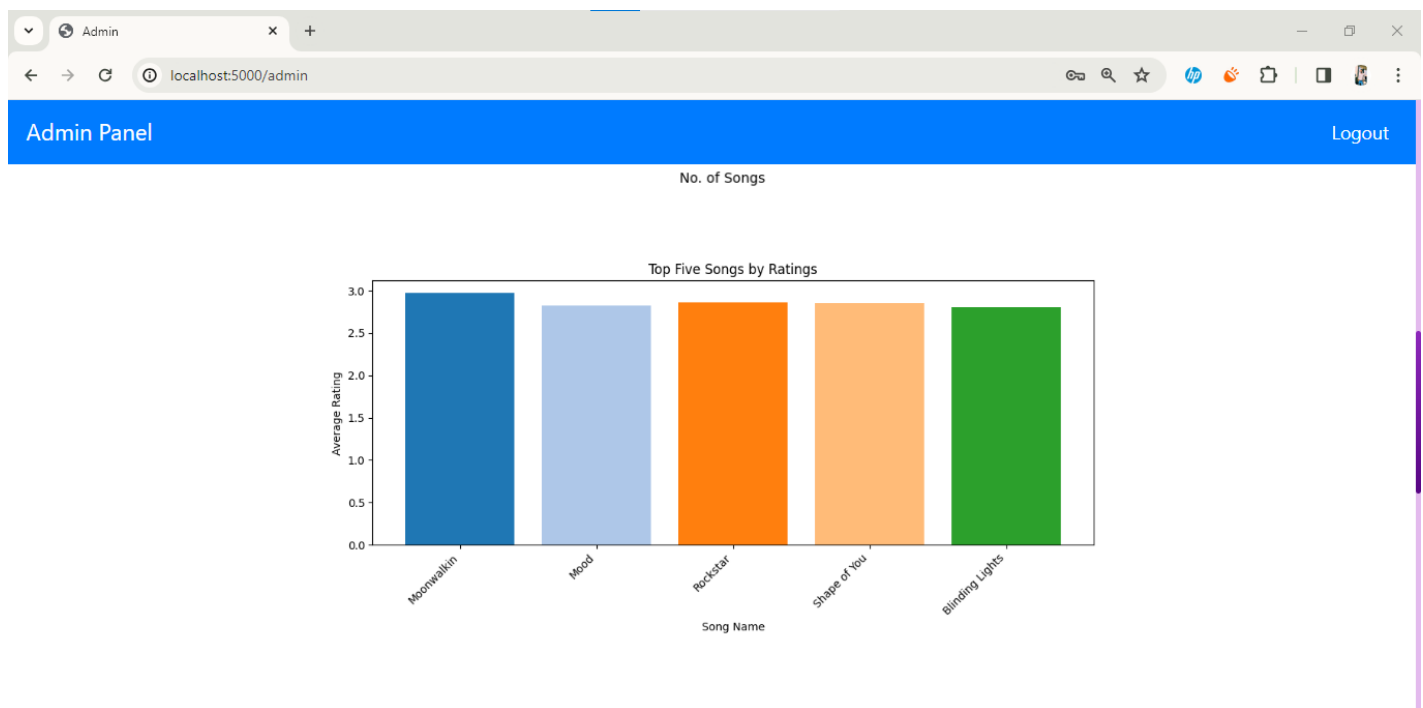
Playlist Creation



Viewing Playlist

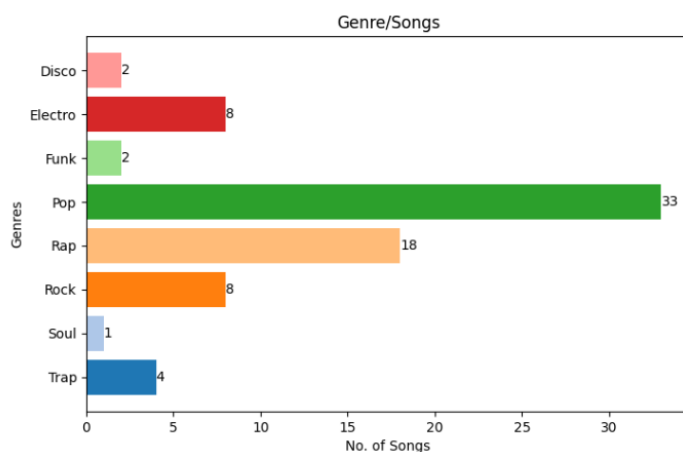


Admin Panel User ID Showcasing

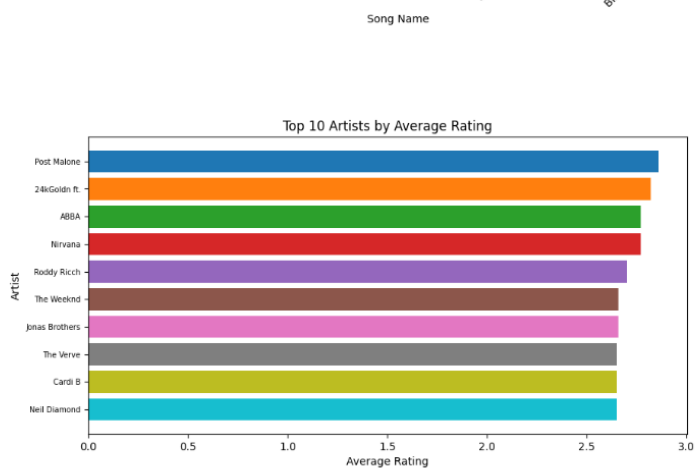


Rating Graphs

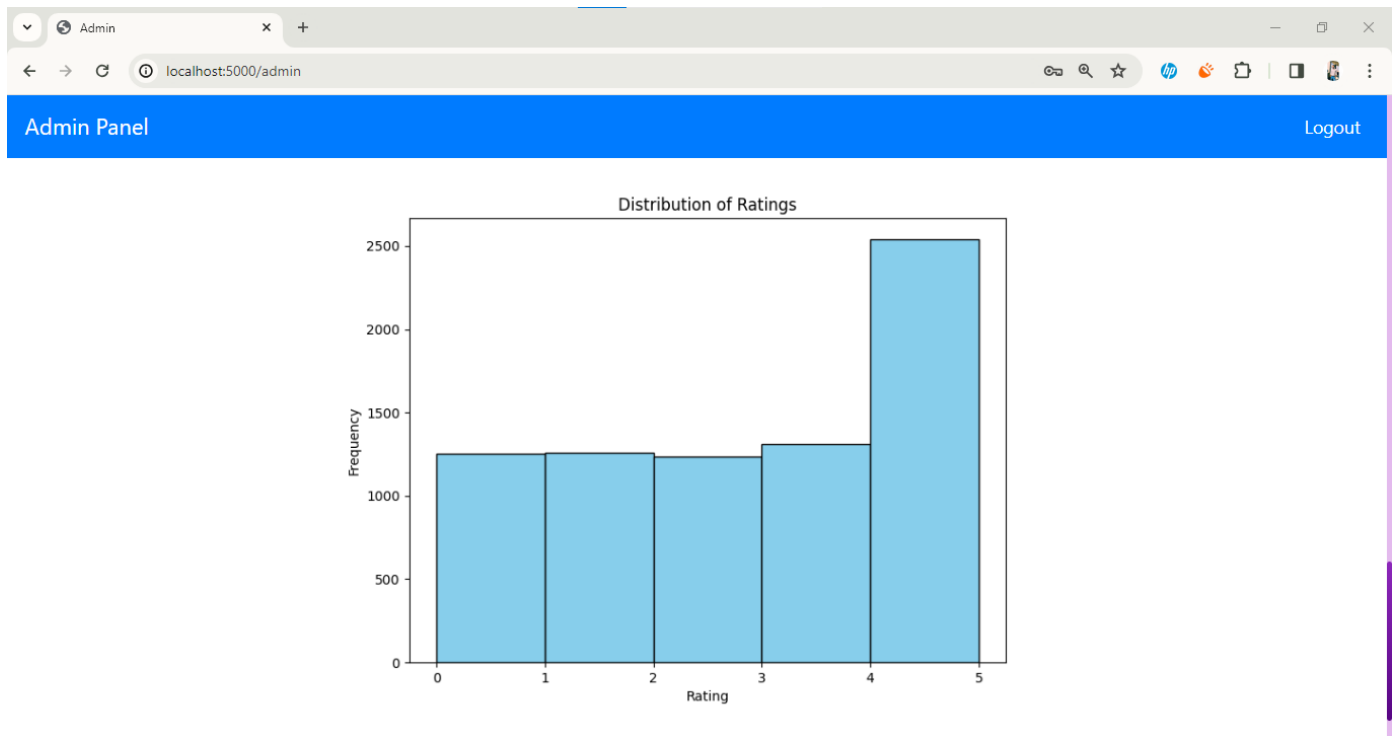
Plots



Songs Rating Graph



Artist Rating Graph



Distribution of Rating Graph

5.6 Summary:

This chapter delves into a number of crucial elements that are essential to our project, such as the prototype design that aims to improve precision as assessed by system analysts and end users. We also look at our project's frontend and backend design components and analyze the database queries used by Sql Lite 3 and third-party frameworks. We illustrate our application's compliance with user requirements using screenshots, along with succinct explanations of its features and the source code that was used for validation procedures.

CHAPTER – 6

6.1 Introduction:

In this chapter, we have discussed the test cases employed to ascertain the functionality and efficacy of the music recommender system. These test cases encompass both usability and functionality tests, ensuring that readers are acquainted with every aspect of the system's operation, from minor nuances to major functionalities. Following the completion of the implementation phase, testing assumes a crucial role in validating the system's proper functioning. Each screen and feature, including buttons, underwent rigorous testing in alignment with the specified requirements and functionalities. Despite the complexity of the application, the implementation of reusable code components significantly streamlined testing processes. For instance, certain functionalities, such as data utilization for both music recommendations and playlist creation, were consolidated within a single interface, maximizing code efficiency. As a result of code reusability, the total test cases for the application amounted to 16, each meticulously documented with details including requirement references, project and application names, test case ID, description, steps, expected results, pass or fail status, preparation, running, and completion dates.

6.2 Test Cases:

Requirement Reference	1	Project Name	Recommender System For Music
Test Case Id	1.1	Test Type	Functionality
Test Case Description	Verify the functionality of user authentication, including logging in with valid credentials and handling invalid login attempts.		
Test Steps	<ol style="list-style-type: none"> 1. Open the music recommender system's login page. 2. Enter valid credentials (username and password). 3. Click on the "Login" button. 		
Expected Result	The user should be logged in successfully and redirected to the home page..		
Actual Result	User is logged in and directed to the home page..		
Pass/Fail	pass		
Date Prepared	5 th feb2024		
Date Run	2nd March 2024		
Prepared By	Muhammad AUN and Muhammad Waleed		
Tested By	Syed Humal Hasan Rizvi		

Table 23: Test case 1

Requirement Reference	1	Project Name	Recommender System For Music
Test Case Id	1.2	Test Type	Functionality
Test Case Description	Verify the functionality of managing user profiles, including updating profile information and preferences.		
Test Steps	<ol style="list-style-type: none"> 1. Navigate to the profile settings page. 2. Update profile information (e.g., change preferred genres, add favorite artists). 3. Save changes. 		
Expected Result	Profile information should be updated successfully, and changes should be reflected in the user's profile.		
Actual Result	Profile information is updated, and changes are reflected in the user's profile.		
Pass/Fail	Pass		
Date Prepared	5 th feb 2024		
Date Run	2nd March 2024		
Prepared By	Muhammad Aun and Muhammad Waleed		
Tested By	Syed Humal Hasan		

Table 24: Test case 2

Requirement Reference	1	Project Name	Recommender System For Music
Test Case Id	1.3	Test Type	Functionality
Test Case Description	Verify the accuracy and relevance of the recommendations generated by the system..		
Test Steps	<ol style="list-style-type: none"> 1. Navigate to the recommendations page. 2. Verify the recommended songs displayed. 		
Expected Result	Recommended songs should align with the user's preferences and listening history, covering a diverse range of genres and artists..		
Actual Result	Recommended songs align with the user's preferences and listening history, covering a diverse range of genres and artists.		
Pass/Fail	Pass		
Date Prepared	5 th feb 2024		
Date Run	2nd March 2024		

Prepared By	Muhammad Aun and Muhammad waleed
Tested By	Syed Humal Hassan

Table 25: Test case 3

Requirement Reference	1	Project Name	Recommender System for Music
Test Case Id	1.4	Test Type	Functionality
Test Description	Verify the functionality of the feedback mechanism, including providing feedback on recommended songs and its impact on future recommendations.		
Test Steps	<ol style="list-style-type: none"> 1. Like or dislike a recommended song. 2. Provide a rating for the song (if available). 3. Save feedback. 		
Expected Result	The system should record the user's feedback and adjust future recommendations accordingly.		
Actual Result	User feedback is recorded, and future recommendations are adjusted based on the provided feedback.		
Pass/Fail	Pass		
Date Prepared	5 TH FEB 2024		
Date Run	2nd March 2024		
Prepared By	Muhammad Aun and Muhammad Waleed		
Tested By	Syed Humal Hassan		

Table 26: Test case 4

Requirement Reference	1	Project Name	Recommender System For Music
Test Case Id	1.5	Test Type	Functionality
Test Description	verify the functionality of the search feature, including searching for specific songs, albums, or artists.		
Test Steps	<ol style="list-style-type: none"> 1. Enter a search query (e.g., song title, artist name) in the search bar. 2. Press Enter or click on the search button. 		
Expected Result	Relevant search results should be displayed based on the user's query..		
Actual Result	Relevant search results are displayed based on the user's query.		
Pass/Fail	Pass		
Date Prepared	5 TH FEB 2024		
Date Run	2nd March 2024		
Prepared By	Muhammad Aun and Muhammad Waleed		

Tested By	Syed Humal Hassan
------------------	-------------------

Table 27: Test case 5

Requirement Reference	1	Project Name	Recommender System For Music
Test Case Id	1.6	Test Type	Functionality
Test Case Description	Verify the functionality of managing playlists, including creating new playlists, adding/removing songs, and renaming/deleting playlists.		
Test Steps	<ol style="list-style-type: none"> 1. Navigate to the playlists section. 2. Create a new playlist. 3. Add songs to the playlist and remove songs if necessary. 4. Rename the playlist. 5. Delete the playlist. 		
Expected Result	Playlist management actions should be performed successfully, and changes should be reflected in the user's playlist collection.		
Actual Result	Playlist management actions are performed successfully, and changes are reflected in the user's playlist collection		
Pass/Fail	Pass		
Date Prepared	5 th feb 2024		
Date Run	2nd March 2024		
Prepared By	Muhammad Aun and Muhammad Waleed		
Tested By	Syed Humaal Hassan		

Table 28: Test case 6

Requirement Reference	1	Project Name	Recommender System For Music
Test Case Id	1.7	Test Type	Functionality
Test Case Description	Verify the performance and scalability of the system under various loads.		
Test Steps	<ol style="list-style-type: none"> 1. Simulate a large number of concurrent users accessing the system simultaneously. 2. Measure the system's response time and resource utilization. 		
Expected Result	The system should maintain acceptable response times and handle the load without crashing or experiencing significant performance degradation.		
Actual Result	The system maintains acceptable response times and handles the load without crashing or experiencing significant performance degradation.		
Pass/Fail	Pass		

Date Prepared	5 th feb 2024
Date Run	2nd March 2024
Prepared By	Muhammad Aun and Muhammad Waleed
Tested By	Syed Humal Hassan

Table 29: Test case 7

Requirement Reference	1	Project Name	Recommender System For Music
Test Case Id	1.8	Test Type	Functionality
Test Case Description	Verify that the system handles errors and exceptions gracefully.		
Test Steps	<ol style="list-style-type: none"> 1. Trigger various error conditions (e.g., invalid input, network failure, unexpected server error). 2. Observe how the system responds to each error condition 		
Expected Result	The system should display appropriate error messages and handle errors without crashing or compromising data integrity		
Actual Result	The system displays appropriate error messages and handles errors without crashing or compromising data integrity.		
Pass/Fail	Pass		
Date Prepared	5 th feb 2024		
Date Run	2nd March 2024		
Prepared By	Muhammad Aun and Muhammad Waleed		
Tested By	Syed Humal Hassan		

Table 30: Test case 8

6.3 Summary:

We test our software to get our expected results of our software or whether a system under test satisfies requirements or works correctly. After test cases, we get satisfied results the usability test case.

CHAPTER – 7

7.1 Introduction:

In this final chapter, a comprehensive summary of the music recommender system's work throughout the final year of the project is provided, along with discussions on encountered challenges, limitations, and potential future directions. Both major and minor aspects of the project are addressed, with a focus on elucidating the system's limitations to enhance user understanding. The future work section outlines potential enhancements for the software, aiming to make it more effective and beneficial. The music recommender system, designed to cater to diverse musical preferences, offers personalized recommendations to users. It currently features functionalities for recommending music based on user preferences, but there are numerous opportunities for further development. Future work will concentrate on refining the system's usability and expanding its feature set. This includes the addition of new recommendation algorithms, enhancement of user interfaces, and incorporation of interactive features to engage users actively. The goal is to create a more personalized and interactive experience for each user, potentially including advanced recommendation techniques and customization options based on individual musical interests and preferences.

7.2 System Limitations and Challenges:

The music recommender system represents a groundbreaking approach to personalized music discovery. Despite its effectiveness and interactive design, the system is not without its limitations and challenges. Here, we discuss these factors:

- i. **Dependency on Internet Connection:** Similar to other technology-driven platforms, the music recommender system relies heavily on internet connectivity. Without a stable connection, users may experience difficulties accessing the system, hindering their ability to explore music recommendations.
- ii. **Accessibility Concerns:** The primary limitation of the music recommender system lies in its accessibility. Users who lack familiarity with basic technology, particularly in rural areas, may face challenges in navigating the system. This can result in a significant learning curve for parents or guardians attempting to utilize the platform.
- iii. **Lack of Parental Controls:** Unlike some educational apps, the music recommender system does not incorporate built-in parental controls. This absence of restrictions means that parents cannot monitor or regulate the content their children engage with, potentially raising concerns about exposure to inappropriate content.
- iv. **Limited Object Recognition:** The system's camera functionality is constrained by its reliance on a predefined database of objects. As a result, the camera may only recognize objects that are already cataloged within this database, limiting its scope and effectiveness.

Throughout the development process, several challenges were encountered:

- i. **Integration of Object Recognition:** The integration of the object recognition feature posed a significant challenge during development. Ensuring seamless functionality and accurate recognition required meticulous attention to detail and extensive testing.
- ii. **Compatibility Issues:** One major hurdle faced by the music recommender system is its compatibility limitations. Unsupported devices or outdated browsers may pose compatibility issues, potentially restricting access for certain users.

Addressing these limitations and challenges will be crucial for enhancing the accessibility, functionality, and overall user experience of the music recommender system. Efforts to improve internet independence, enhance accessibility, implement parental controls, and refine object recognition capabilities will be key areas of focus for future development endeavors.

7.3 Conclusion:

The music recommender system offers a personalized and interactive platform for music discovery, yet faces notable limitations and challenges. It heavily depends on internet connectivity, potentially hindering access in areas with poor network coverage, while accessibility concerns may arise for users with limited tech proficiency, particularly in rural regions. The absence of built-in parental controls raises worries about

content appropriateness for children, and the system's object recognition feature is constrained by its predefined database. Development hurdles include seamlessly integrating object recognition and addressing compatibility issues. To enhance the system, future endeavors should prioritize improving internet independence, enhancing accessibility, implementing parental controls, and refining object recognition capabilities, aiming for a more comprehensive and user-friendly music recommendation experience.