Tribhuvan University

Faculty of Humanities and Social Sciences

Online Music Library Platform

A PROJECT REPORT

Submitted To:

Department of Computer Application

Ratna Rajyalaxmi Campus

In partial fulfillment of the requirement for the Bachelor in Computer Application

Submitted By:

BISHAM RAJ PANDEY (6-2-40-15-2021)

BISHAL REGMI (6-2-40-14-2021)

Under the Supervision of

Bipin Timilsina

Tribhuvan University

Faculty of Humanities and Social Science

Ratan Rajya Laxmi Campus

SUPERVISOR'S RECOMMENDATION

I hereby recommend that this project prepared under my supervision by BISHAM RAJ PANDEY and BISHAL REGMI entitled "Online Music Library Platform" in partial fulfillment of the requirements for the degree of Bachelor of Computer Application is recommended for the final evaluation.

_____

Bipin Timilsina

SUPERVISOR

Department of Computer Application

Pradarshani Marg, Kathmandu

Tribhuvan University

Faculty of Humanities and Social Sciences

Ratan Rajya Laxmi Campus

LETTER OF APPROVAL

This is to certify that the project prepared by BISHAM RAJ PANDEY and BISHAL REGMI entitled "Music Library Platform", In partial fulfillment of the requirements for the degree of Bachelor in Computer Application has been evaluated in our opinion it id satisfactory in the scope and quality as a project for the required degree.

SIGNATURE of Supervisor	**SIGNATURE of HOD/Coordinator**
----------	----------
Bipin Timilsina Lecturer, Project Supervisor Ranta Rajyalaxmi Campus	Mr. Bhupendra Ram Luhar Coordinator Department of BCA Ratna Rajyalaxmi Campus
SIGNATURE of Internal Examiner	**SIGNATURE of External Examiner**
----------	----------

ACKNOWLEDGEMENT

The project detailed in the report was carried out and presented at Ratna Rajya Laxmi Campus, under the Faculty of Humanities and Social Sciences at Tribhuvan University of Technology, as a part of the Bachelor of Arts in Computer Application program. This project stands as evidence not only of technical skill but also of teamwork and performance in the face of various challenges. The successful completion of this endeavor is owed in large part to the invaluable assistance provided by experts.

Furthermore, we express our appreciation to the instructors of the Department of Computer Application for generously sharing their knowledge, which greatly contributed to the development of this project. We are also grateful to our friends for their steadfast support, and to the participants whose feedback helped improve our project.

We are deeply thankful to our supervisor, Mr. Bipin Timilsina, a lecturer whose guidance was instrumental in the success of this project. We also appreciate the support of our department coordinator, Mr. Bhupendra Ram Luhar.

Lastly, our heartfelt gratitude goes to our families, friends, and mentors. Without their unconditional love, care, and support, this achievement would not have been possible

ABSTRACT

The purpose of the "Music Library Platform" is to make it easy for users to find music they like. This system aims to facilitate the discovery of music, along with providing a platform for artists to post music. The required software and hardware are readily available and easy to use

Key Words :

User-

Artist-

Moderators-

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1. Introduction

Background

The Music Streaming Platform is a web-based application that allows users to listen to songs, like or dislike them, and create playlists. Designed specifically for younger audiences with shorter attention spans, it introduces a unique preview-based listening system. Songs are played in short segments, encouraging quick engagement. The platform uses real-time data and recommendation algorithms to deliver personalized music suggestions. Both users and artists benefit from the interactive and user-friendly design.

1.1 Problem Statement

The problems in current music streaming platforms are:

- Overwhelming content with poor personalization for new users
- Long tracks not engaging enough for younger audiences
- Ineffective or generic music recommendations
- Limited control for artists over preview and exposure

1.2 Objectives

The objectives of the Music Streaming Platform are:

- To implement a short-preview based music recommendation system
- To offer a personalized experience using user behavior and preferences
- To allow users to build playlists from liked songs easily

1.3 Scope and Limitation

1.3.1 Scope

The scopes of Music Streaming Platform are :-

- Stream songs with smart preview-based auto-swiping
- Recommend music using hybrid collaborative and content-based filtering
- Allow users to like, playlist, and interact with songs
- Provide artists with tools to upload and set previews

1.3.2 Limitation

The limitations of the Music Library Platform are:-

- Limited song selection in early versions
- Recommendations improve over time with more user data
- Preview system may not suit all genres or artists

1.4 Report Organization

The report can be organized into 5 chapters which are given below:

Chapter 1: Includes introduction includes the brief introduction of the system, statement of problem, objectives, scope and limitation.

Chapter 2: Includes background study and literature review includes the previous work related to the systems and similar works were studied and are summarized.

Chapter 3: Includes system analysis and design includes different feasibility analysis and designed system architecture, system flow diagram, dataflow diagram.

Chapter 4: Includes implementation and testing includes various implementation method and tools and also contains description of testing.

Chapter 5: Includes conclusion and future recommendations includes outcomes of the system, conclusion to the system and description about what features can be added in the future.

CHAPTER 2. BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Background Study

Music streaming platforms have revolutionized the way people access and enjoy music, shifting from physical media and downloads to on-demand, cloud-based listening experiences. Beginning with early services like Pandora and Last.fm, and later evolving through major players such as Spotify, Apple Music, and YouTube Music, these platforms have introduced features like personalized playlists, curated recommendations, and artist interaction. As technology and user behavior have advanced, streaming apps have adapted to shorter attention spans, mobile-first access, and real-time engagement. In response to these trends, modern platforms are experimenting with new formats, such as song previews, swipe-based interactions, and AI-driven recommendations to enhance user retention. This evolution reflects a growing demand for more intuitive, engaging, and adaptive music experiences that cater to the preferences of younger, tech-savvy audiences.

In recent years, a noticeable shift has occurred in how younger audiences consume music, with a growing preference for shorter, more engaging formats driven by social media trends. Platforms like TikTok and Instagram Reels have played a significant role in popularizing brief, catchy music clips that go viral and influence listening habits. As a result, many artists now release songs with standout snippets designed to trend on these platforms, often prioritizing hook-heavy segments over traditional full-length compositions. This shift has challenged traditional music streaming platforms to adapt by incorporating features that support bite-sized content discovery and interactive user experiences. Our music streaming app is built with this transformation in mind, offering short previews, swipe-based liking mechanisms, and intelligent recommendation algorithms to cater to these emerging patterns of music consumption.

2.2 Literature Review

The music streaming industry has undergone a significant evolution since the early 2000s, with platforms such as Pandora, Spotify, and Apple Music redefining how users discover and listen to music. These services leveraged algorithms to offer personalized recommendations, revolutionizing traditional radio-style listening. Over time, features like curated playlists, mood-based suggestions, and real-time activity tracking became integral to enhancing user engagement and satisfaction on streaming platforms.

Recommendation systems have remained at the core of user retention and satisfaction in music apps. Collaborative filtering, content-based filtering, and hybrid methods are commonly used to tailor music recommendations based on listening history, user preferences, and behavior patterns. However, younger audiences—especially those influenced by short-form content—demand quicker engagement, prompting platforms to experiment with features like song previews, story-style snippets, and interactive discovery interfaces.

Furthermore, user interactivity and seamless content navigation have become increasingly important in modern platforms. Swipe-based UI, short music segments, and real-time feedback mechanisms are being implemented to align with shorter attention spans and evolving content consumption habits. These trends illustrate the need for adaptable systems that prioritize personalized, fast-paced, and visually appealing experiences. By reviewing the progression of music streaming technologies and user expectations, developers can create platforms that effectively address modern user behavior and technological possibilities.

Machine learning and artificial intelligence (AI) have become crucial in advancing recommendation algorithms within music streaming platforms. Algorithms like matrix factorization, deep neural networks, and reinforcement learning are now utilized to predict user preferences more accurately. These intelligent systems analyze vast amounts of user data, including listening time, skip rate, likes, and playback frequency, to continuously refine recommendations. The application of such techniques allows platforms to deliver more personalized and engaging music discovery experiences.

Another major focus in recent developments is the inclusion of user-centric features, such as social sharing, collaborative playlists, and mood-based filtering. Studies indicate that users are more likely to engage with a platform that provides interactive and customizable experiences. Music applications are increasingly adopting intuitive interfaces and features like drag-and-drop playlist creation, song previews, and real-time collaboration to appeal to diverse user groups. These elements also foster a sense of community and help drive organic content discovery among users.

Moreover, the emergence of hybrid recommendation systems, which combine collaborative and content-based filtering techniques, has addressed some of the key limitations found in traditional models. By leveraging both user behavior and song metadata—such as genre, tempo, and mood—hybrid systems improve recommendation accuracy and cater to a broader audience, including new users with limited listening history. These systems have proven especially effective in reducing the cold-start problem and ensuring consistent user satisfaction across different user profiles.

CHAPTER 3. SYSTEM ANALYSIS AND DESIGN

3.1 System Analysis

System analysis involves collecting and understanding information about the Music Library Platform to identify problems and suggest improvements. It's a problem-solving process that requires communication between users and developers. The analyst closely examines the current system, identifying issues and proposing solutions. These proposals are compared with the existing system, and the best one is chosen after user approval. Preliminary study gathers facts and conducts feasibility studies to guide further analysis and decision-making.

3.2 Requirement Analysis

Requirement analysis was performed by examining existing systems. Systems like spotify, were examined and studied to gather requirements for the system.

3.2.1 Functional Requirement

Function requirements define the fundamental actions that system must perform. The functional requirements for the system are divided into three categories, moderators, atist and users as well as further details, referred to as use cases. The Use case Diagram of the system is given below:

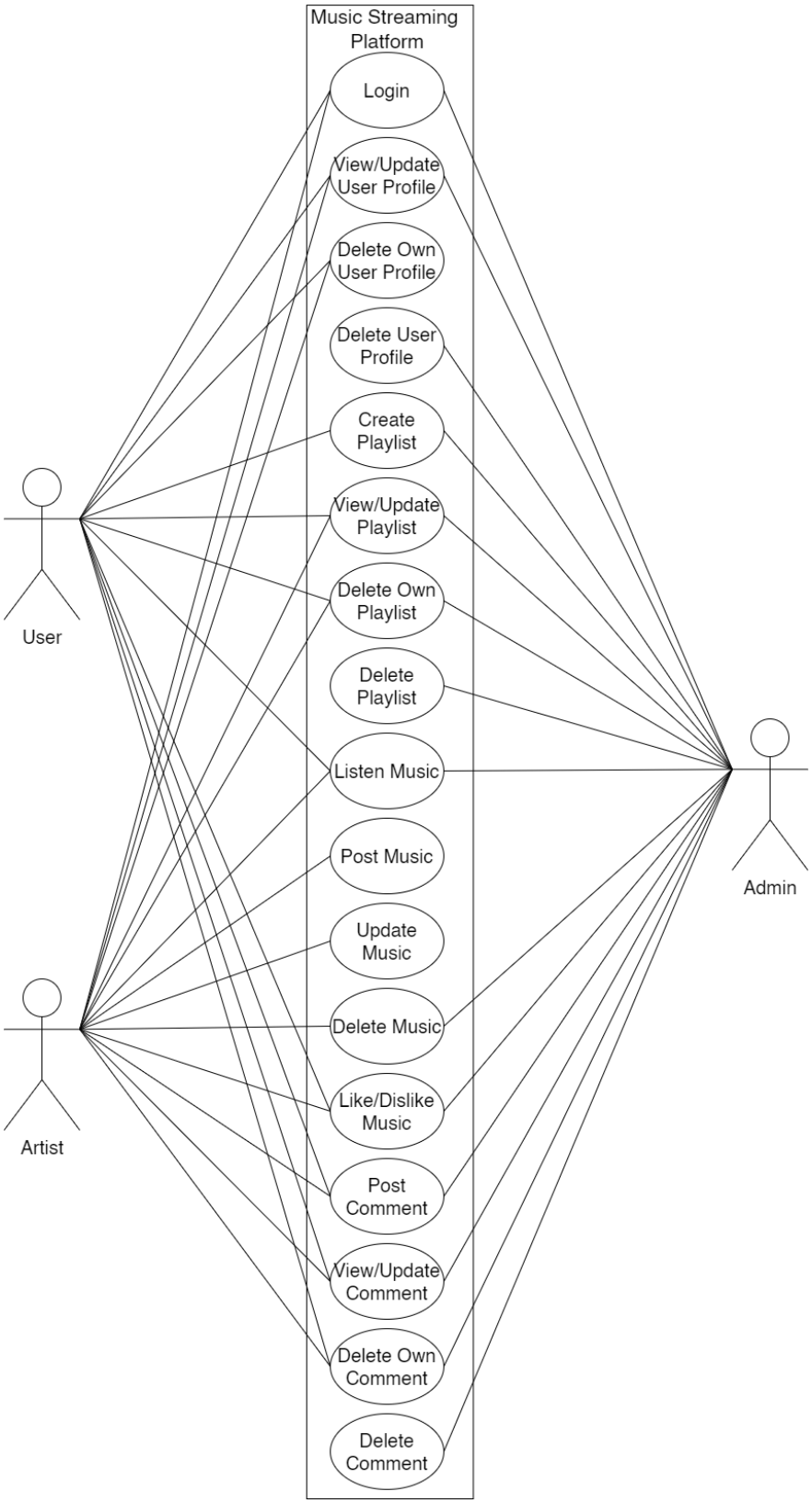

Figure 1:Use case Diagram of Online Music Library Platform

3.2.2 Non-Functional Requirement

- Portability: Our system platform independent system so it can run in any browser.
- Security: There is a high security in keeping the information of user and transaction of money.
- Backup: In case of system failure there is database which will store all the information about all the client.
- User friendly: Our system is user friendly, it can be easily use and understand by user.

3.3 Feasibility Analysis

A feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing business or proposed venture, opportunities and threats present in the natural environment, the resources required to carry through, and ultimately the prospects for success.

3.3.1 Technical Feasibility

As this system is built using the MERN stack—MongoDB, Express.js, React.js, and Node.js—it supports smooth integration, modular development, and scalability. These technologies are widely used and supported, making the system easy to deploy and maintain. MongoDB allows efficient handling of large volumes of user and music data. With this robust tech stack, the project is technically sound and feasible for implementation across various platforms.

3.3.2 Operational Feasibility

Operational feasibility examines how effectively the proposed system will run in a real-world setting. The music streaming platform is designed to address common issues like limited personalization and slow content discovery.

- The system provides smooth performance and fast load times.
- Users will experience a seamless music browsing and recommendation process.
- The system efficiently uses the available resources to ensure reliable operation.

3.3.3 Economic Feasibility

This system aims to deliver strong economic value to its users and stakeholders by minimizing costs while maximizing benefits.

- The platform is cost-efficient and uses open-source tools.
- Efficient management and automated recommendation reduce maintenance efforts.
- The long-term benefits, including user retention and satisfaction, outweigh initial development costs.

3.3.4 Scheduling Feasibility

This seeks to assess the time that the proposed system will take to develop and implement.

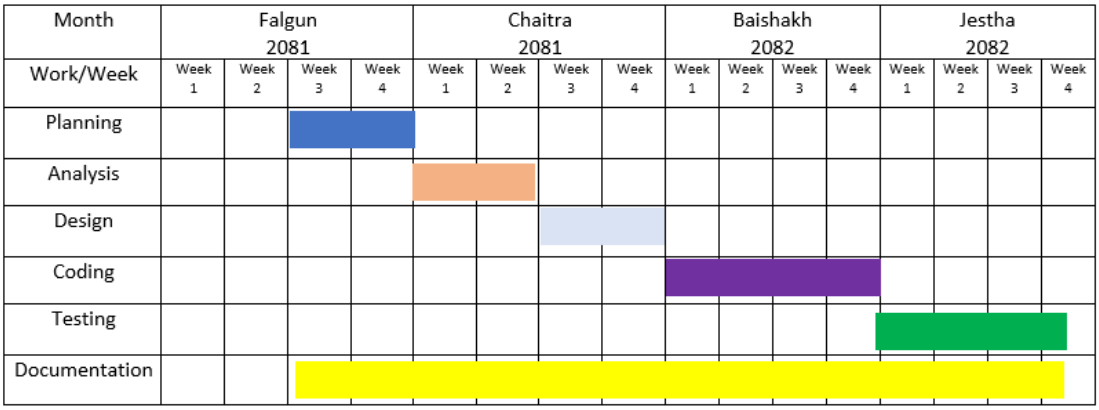

Figure 2: Gantt Chart of Online Music Library Platform

3.4 Object Modeling

3.4.1 Class Diagram

This class diagram represents the structure and relationships within the database models of Music Library Platform, including the differentiation of user roles and relationship among various classes.

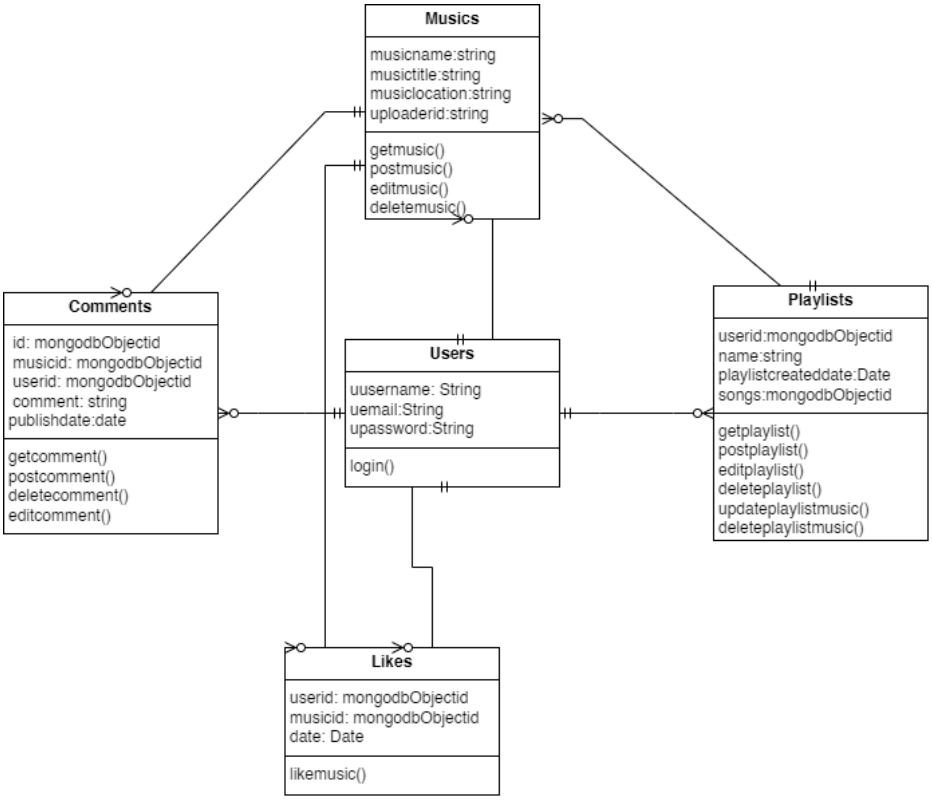

Figure 3: Gantt Chart of Online Music Library Platform

3.4.2 Object Diagram

The following object diagram illustrates a specific snapshot of Music Library Platform, showcasing the relationships and states of various objects within the system. The diagram provides a clear view of how various entities within the system are interrelated, showing the interactions between users and the courses they engage with along with other entities.

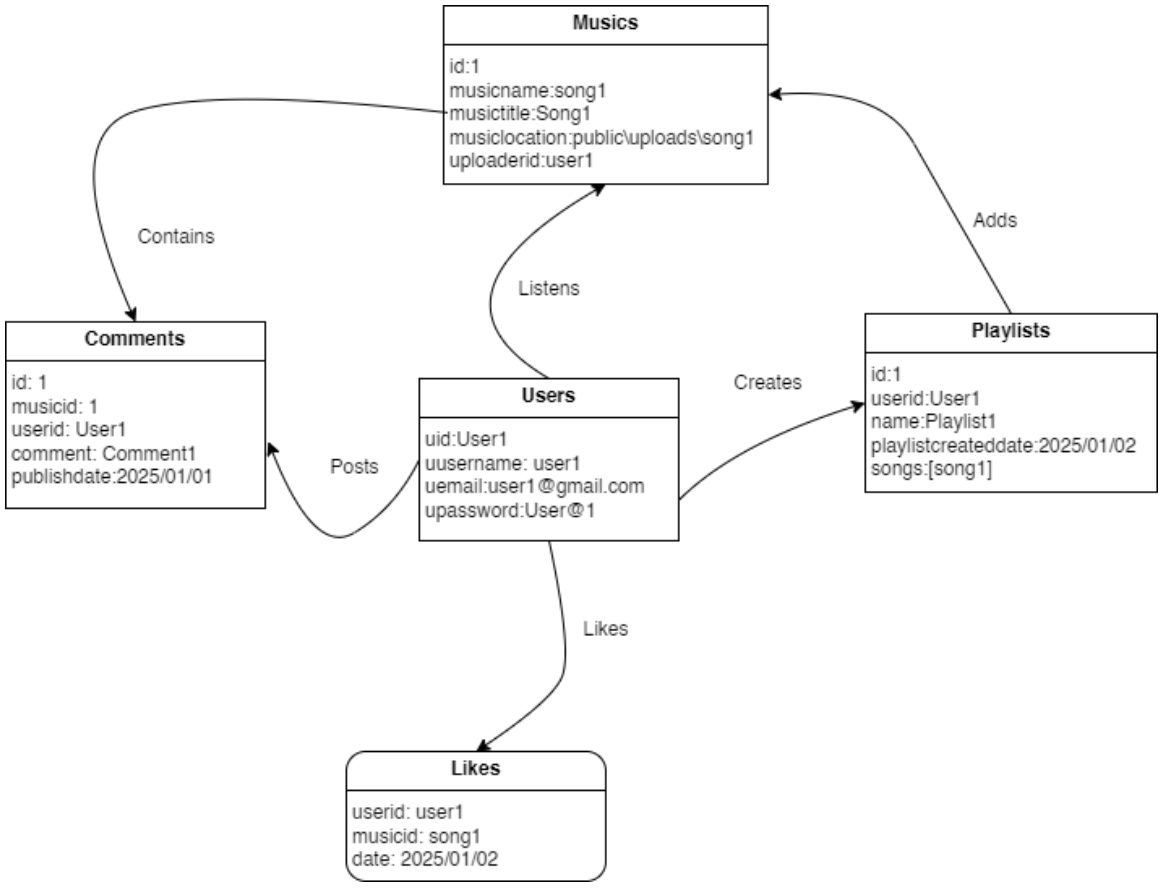

Figure 4: Object Diagram of Music Library Platform

3.5 Dynamic Modeling

3.5.1 State Diagram

The following state diagram demonstrates different lifecycle states that the different entities go through.

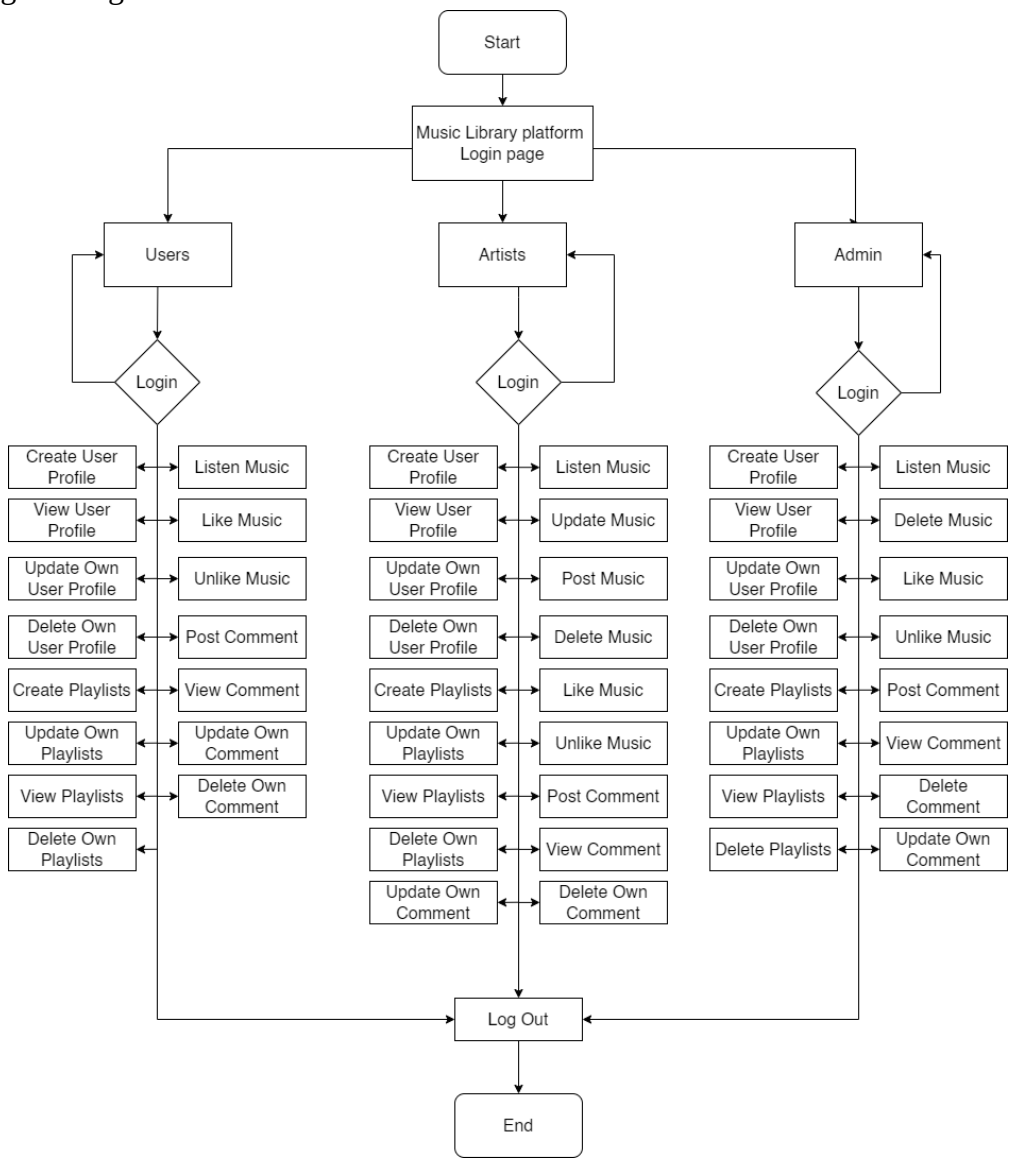

Figure 5: State diagram of music library platform

3.5.2 Sequence Diagram

This sequence diagram illustrates the interactions among objects in a sequential order. In the context of our Learning Management System (LMS), the sequence diagram captures the dynamic behavior of the system, depicting how various components interact to perform specific functionalities.

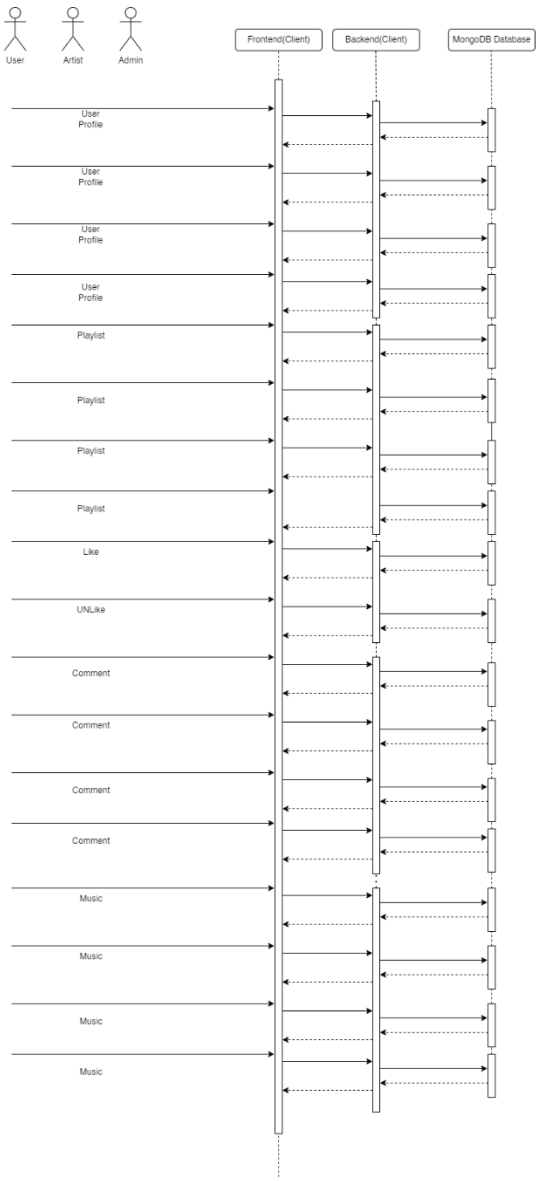

Figure 6: Sequence diagram of music library platform

3.6 Process Modeling

The following two activity diagrams illustrate the processes within a Learning Management

3.6.1 Activity Diagram of User

The above diagram shows the instructor workflow, starting from logging in, accessing the dashboard, creating and updating courses, managing sections and attachments, publishing the course and finally logging out

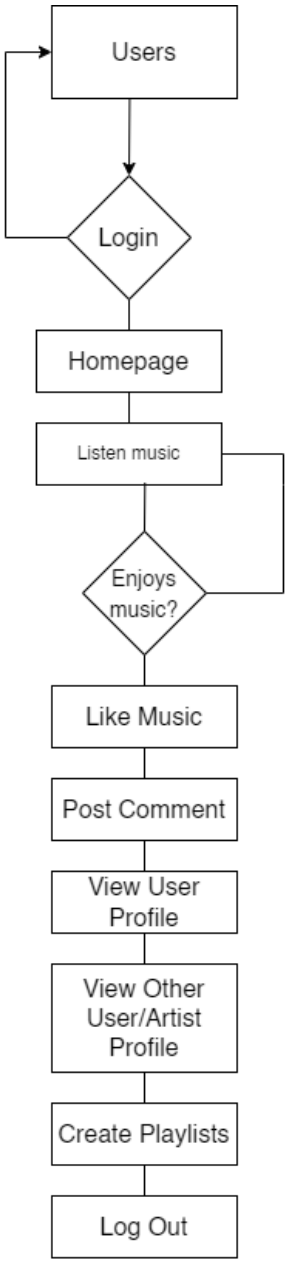

Figure 7: Activity diagram of user of music library platform

3.6.2 Activity Diagram of Artist

The above diagram shows the instructor workflow, starting from logging in, accessing the dashboard, creating and updating courses, managing sections and attachments, publishing the course and finally logging out

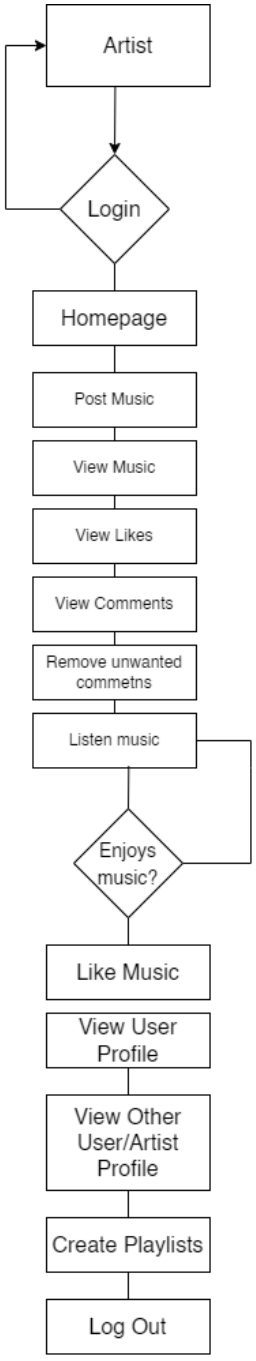

Figure 7: Activity diagram of artist of music library platform

3.6.3 Activity Diagram of Admin

The above diagram shows the instructor workflow, starting from logging in, accessing the dashboard, creating and updating courses, managing sections and attachments, publishing the course and finally logging out

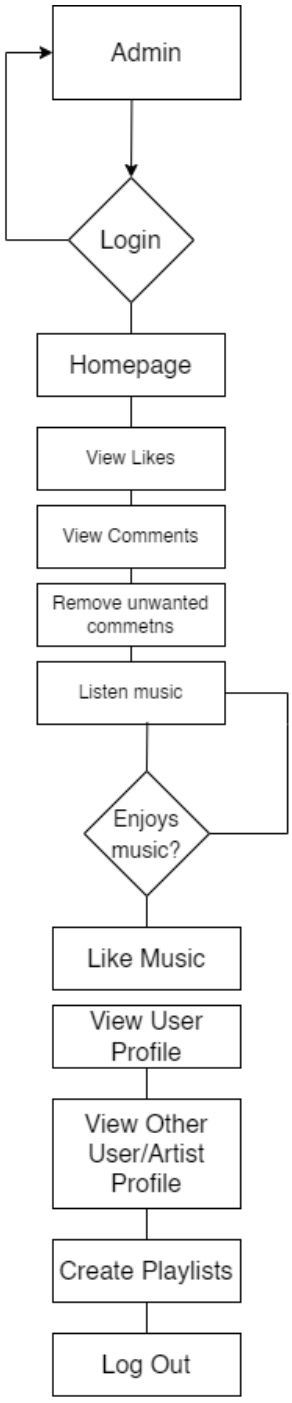

Figure 8: Activity diagram of admin of music library platform

3.7 System Design

During the system design phase, the analysis is taken to another step where existing classes and object diagrams are refined and component and deployment diagram are constructed.

3.7.1 Refinement of Class Diagram

In the following refined diagram, the existing class diagram has been refined by adding more detail. The diagram consists of method signatures, return types and parameter types. While the class diagram showed the overview of the system, the refined class diagram adds more detail for actual implementation.

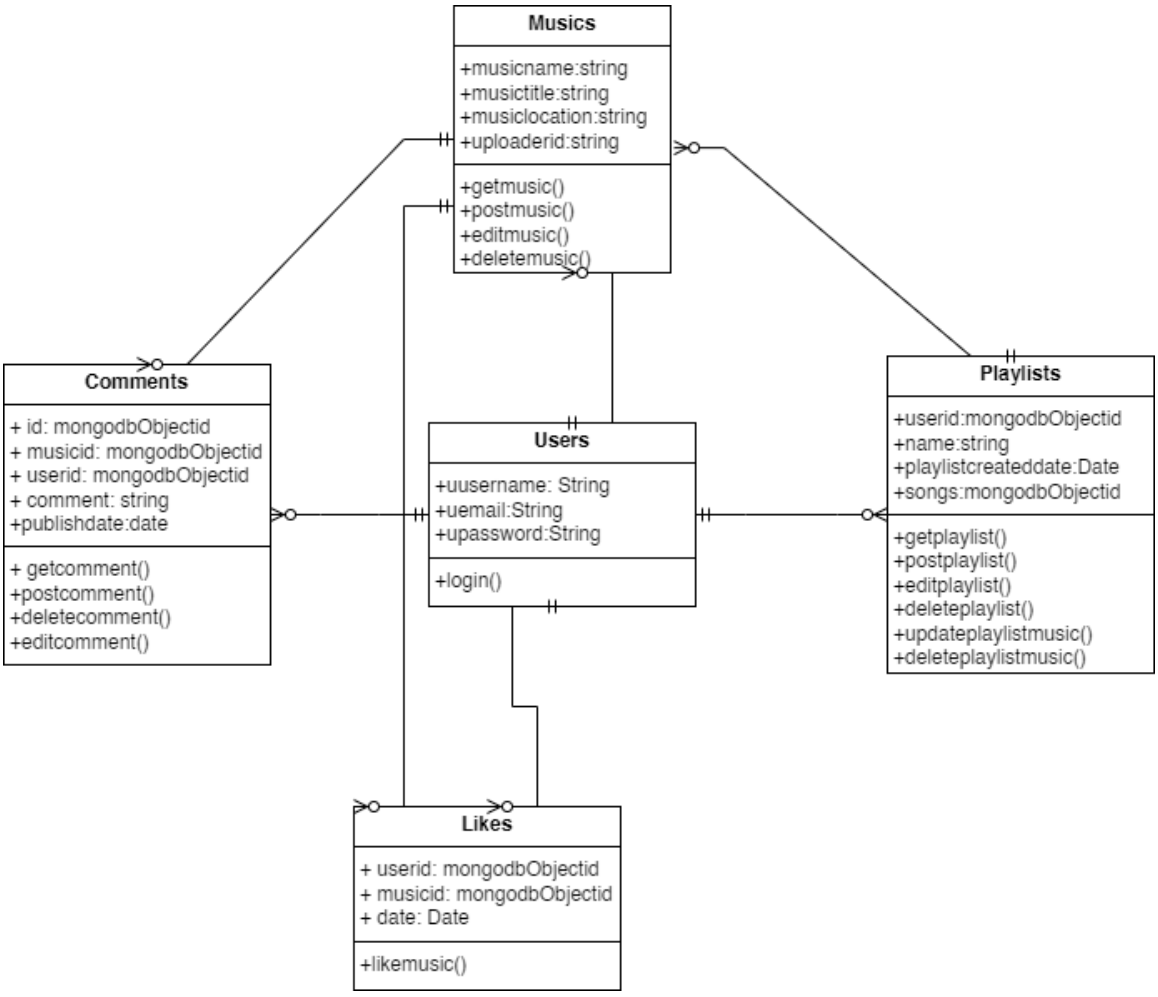

Figure 10: Refined class diagram of music library platform

3.7.2 Refinement of Object Diagram

The component diagram below illustrates the physical components of the Learning Management System and their interactions. With its help we are able to visualize the overall structure and organization of system.

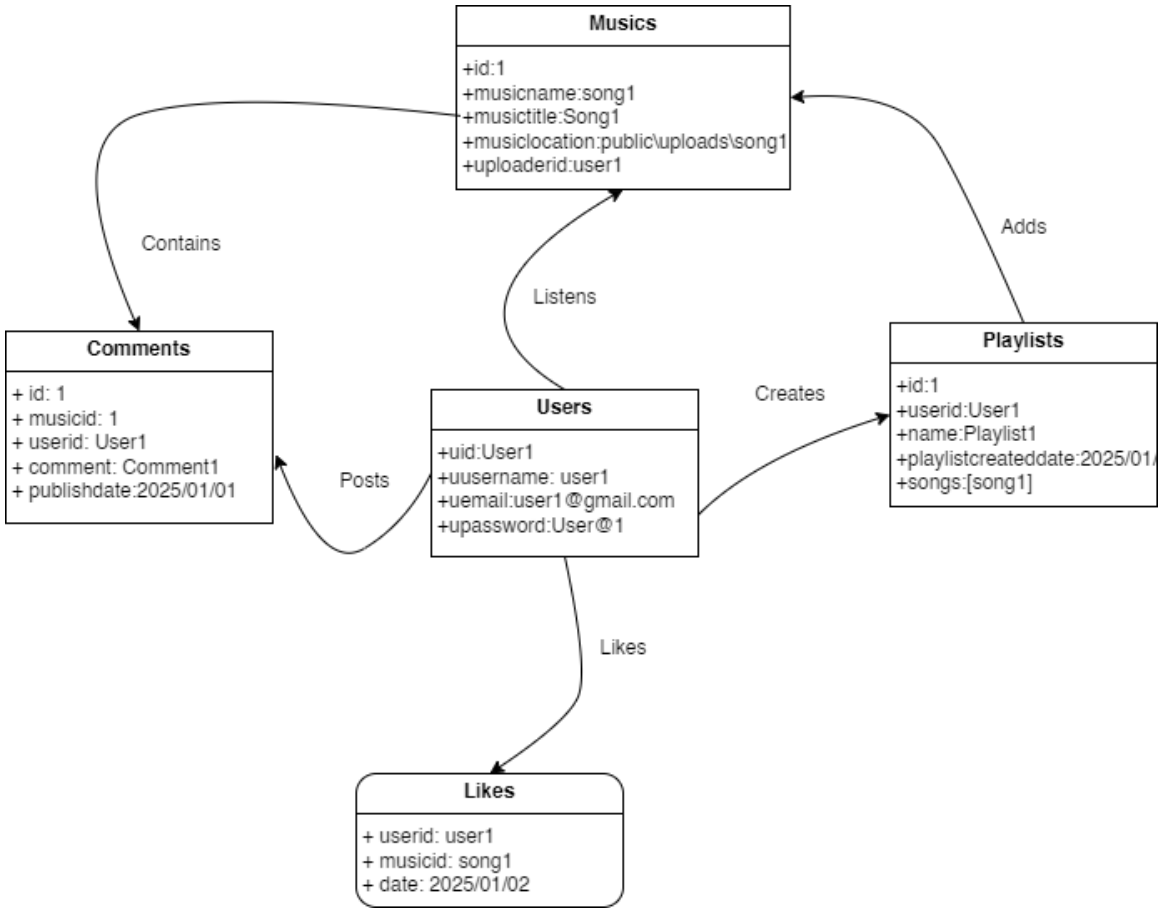

Figure 11: Refined object diagram of music library platform

3.7.3 Component Diagram

This is the component diagram of the music library platofrom

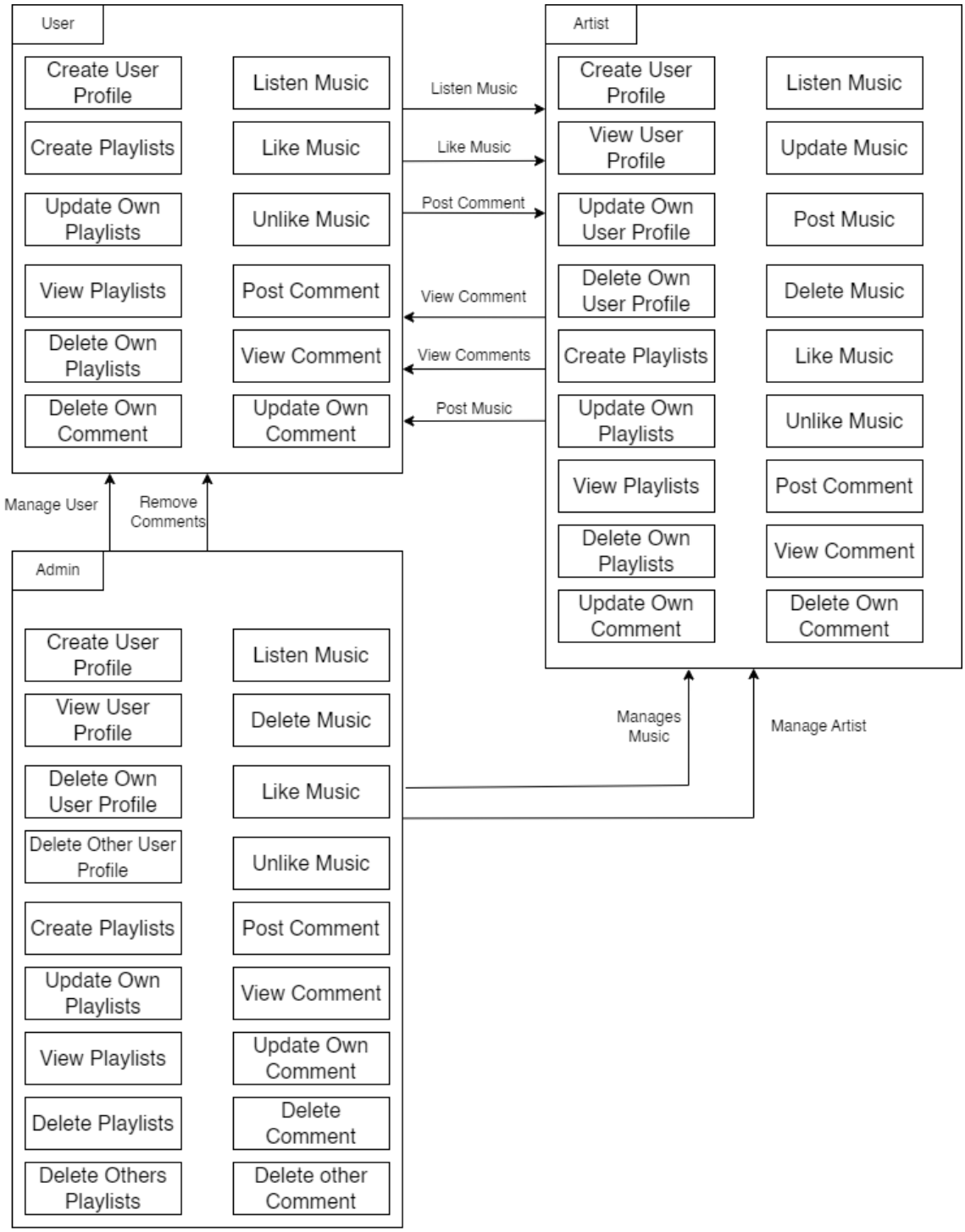

Figure 12: Component diagram of music library platform

3.7.4 Deployment Diagram

This it the deployment diagram of the music library platform

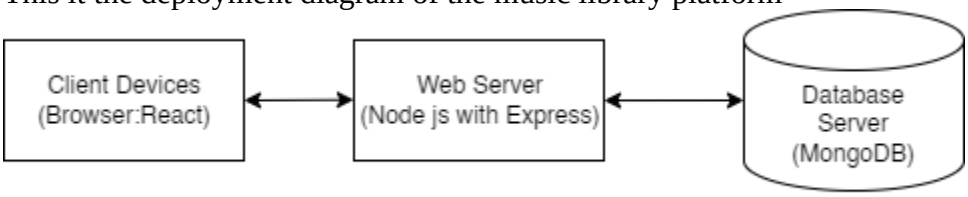

Figure 13: Deployment Diagram of music library platform

3.8 Algorithm Details: UCB1 (Upper Confidence Bound 1)

In the Spotify clone project, the UCB1 algorithm is implemented as a strategy for a multi-armed bandit problem to enhance music recommendations. The core challenge in recommendations is balancing **exploration** (suggesting new or less-played songs to discover user preferences) and **exploitation** (suggesting songs known to be liked by the user or similar users). UCB1 provides a principled way to manage this trade-off, aiming to maximize the cumulative user satisfaction (rewards) over time.

3.8.1 UCB1 Value Calculation

UCB1 works by assigning a value to each potential song (or "arm" in bandit terminology) that considers both its past performance and the uncertainty about that performance. The song with the highest UCB1 value is chosen for recommendation.

The UCB1 value for a song **a** at a given time (or trial) **t** is calculated as follows:

$$\mathbf{UCB1(a) = avgReward(a) + sqrt((2 * ln(t)) / n(a))}$$

Where:

- **avgReward(a)**: This is the current average reward received from song **a**. It is calculated as:

$$\mathbf{avgReward(a) = totalReward(a) / n(a)}$$

- **totalReward(a)**: The sum of all rewards obtained from recommending song **a** so far (e.g., 1 for a like, 0 for a skip).
- **n(a)**: The number of times song **a** has been recommended (played/pulled).

- **sqrt((2 * ln(t)) / n(a))**: This is the **exploration term** or **confidence bound**. It quantifies the uncertainty in the estimate of **avgReward(a)**.
- **ln(t)**: The natural logarithm of **t**.
- **t**: The total number of recommendation trials conducted so far across all songs. This term ensures that as more trials are performed, the exploration bonus increases, but at a decreasing rate (due to the logarithm).
- **n(a)**: The number of times song **a** has been recommended. If a song has been recommended fewer times (small **n(a)**), the exploration term becomes larger, increasing its UCB1 value and making it more likely to be chosen. This encourages trying out arms that have not been explored much.

3.8.2 Steps

The UCB1 algorithm operates iteratively as follows:

a. **Initialization Phase:**

   For each song (arm) available for recommendation:

   Recommend the song once.

   Observe the reward (e.g., user likes it or not).

   Update **totalReward(a)** and **n(a)** for that song.

   This ensures that every song has been tried at least once, providing an initial estimate for its reward.

b. Iterative Recommendation Phase (for each subsequent recommendation opportunity **t**):
   - **Calculate UCB1 Values:**

     For every song **a** in the pool of available songs:

     If **n(a)** (number of times song **a** was played) is 0, this song should have been covered in initialization. If not, it might be treated with a very high UCB value to ensure it gets played, or an error is flagged.

   - Calculate **avgReward(a)**

     totalReward(a) / n(a).

   - Calculate the exploration term

exploration_bonus = sqrt((2 * ln(t_current_round)) / n(a)),

where **t_current_round** is the total number of recommendations made so far (after the initialization phase).

- Calculate **UCB1(a)**

avgReward(a) + exploration_bonus.

c. **Select Song:**

Choose the song **a*** that has the highest **UCB1(a)** value.

If there is a tie (multiple songs have the same highest UCB1 value), break the tie arbitrarily (e.g., randomly select one of the tied songs, or pick the one with the lowest index).

d. **Recommend and Observe Reward:**

Present the selected song **a*** to the user.

Observe and record the reward **R** obtained from this recommendation (e.g., **R=1** if liked, **R=0** if not).

e. Update State

Update the statistics for the chosen song **a***:

totalReward(a*) = totalReward(a*) + R

n(a*) = n(a*) + 1

Increment the total number of recommendation trials **t** (or **t_current_round**).

f. Repeat:

Go back to step 2a for the next recommendation opportunity.

3.8.3 Concluision

This process allows UCB1 to dynamically adjust its strategy, favoring songs that have performed well (exploitation) while still giving a chance to less-explored songs that might turn out to be highly rewarding (exploration).

3.9 Algorithm Details: Epsilon-Greedy (ε-Greedy)

In the Spotify clone project, the Epsilon-Greedy (ε-Greedy) algorithm is another strategy employed for the multi-armed bandit problem to optimize music recommendations. Like UCB1, it addresses the fundamental exploration-exploitation dilemma: deciding whether to recommend a song that has performed well in the past (exploit) or to try a new or less-played song to gather more information about user preferences (explore).

3.9.1 Epsilon-Greedy Mechanism

The ε-Greedy algorithm is a simpler approach compared to UCB1. It makes its decision based on a single parameter, epsilon (ε).

With probability ε (epsilon): The algorithm chooses to explore. It selects a song randomly from all available songs, irrespective of their past performance.

With probability 1 - ε: The algorithm chooses to exploit. It selects the song that currently has the highest estimated average reward (also known as the Q-value).

This random chance of exploration ensures that all songs continue to be tried over time, preventing the algorithm from prematurely settling on a suboptimal song.

3.9.2 Parameter: Epsilon (ε)

Epsilon (ε) is a value between 0 and 1 (e.g., 0.1, 0.05). It dictates the balance between exploration and exploitation:

- High ε (e.g., ε = 0.3): More exploration. The algorithm will frequently choose random songs.
- Low ε (e.g., ε = 0.01): Less exploration, more exploitation. The algorithm will mostly choose the song it currently believes is the best.

The choice of ε is critical. It can be a fixed value, or it can be decayed over time (an "epsilon-decreasing" strategy), where exploration is higher initially and reduces as the algorithm gathers more data and becomes more confident in its estimates.

Q-Value (Estimated Average Reward)

For each song a, the algorithm maintains an estimate of its average reward, denoted as Q(a). This is calculated as:

$$Q(a) = totalReward(a) / n(a)$$

Where:

- totalReward(a): The sum of all rewards obtained from recommending song a so far (e.g., 1 for a like, 0 for a skip).
- n(a): The number of times song a has been recommended.

When exploiting, the algorithm picks the song a* such that Q(a*) is maximal.

3.9.3 Steps

The Epsilon-Greedy algorithm operates as follows:

1. Initialization:

   For each song a in the pool of available songs:

   Initialize its estimated Q-value: Q(a) = 0 (or some optimistic initial value).

   Initialize its play count: n(a) = 0.

   Set the value for epsilon (ε).

2. Iterative Recommendation Phase (for each recommendation opportunity):
   a. Decide to Explore or Exploit:

      Generate a random number p from a uniform distribution between 0 and 1.

   b. Select Song:

      If $p < \varepsilon$ (Explore):

      Select a song a randomly from the set of all available songs.

      Else (Exploit, $p >= \varepsilon$):

      Select the song a* that has the highest current Q-value: a* = argmax_a Q(a).

      If there are multiple songs with the same highest Q-value, break ties arbitrarily (e.g., randomly select one of them).

c. Recommend and Observe Reward:

Present the selected song (either from exploration or exploitation) to the user.

Observe and record the reward R obtained from this recommendation (e.g., R=1 if liked, R=0 if not).

d. Update State for the Chosen Song a_chosen:

Increment the play count for the chosen song:

$$n(a\_chosen) = n(a\_chosen) + 1.$$

Update the total reward for the chosen song:

$$totalReward(a\_chosen) = totalReward(a\_chosen) + R.$$

Update the Q-value for the chosen song using the incremental average formula:

$$Q(a\_chosen) = Q(a\_chosen) + (1 / n(a\_chosen)) * (R - Q(a\_chosen))$$

Alternatively

$$Q(a\_chosen) = totalReward(a\_chosen) / n(a\_chosen).$$

e. Repeat

Go back to step 2a for the next recommendation opportunity.

3.9.4 Conclusion

This straightforward process allows the Epsilon-Greedy algorithm to balance trying out new options with leveraging its current knowledge, making it a popular and effective baseline for many reinforcement learning problems.

CHAPTER 4. Implementation and Testing

4.1 Implementation

The implementation phase involves the application of design specifications done before. The implementation involves coding of the system designs if this project, systems testing are live running. During implementation we start coding according to our requirement.

4.2 Tools Used

This project is developed using the tools, which are most suited for development of the MERN web-based system. These tools are as follows:

- MongoDB – A NoSQL database that stores data in flexible, JSON-like documents, making it ideal for handling dynamic and unstructured data.
- Express.js – A fast, minimalist web application framework for Node.js, used to build robust APIs and handle server-side logic.
- React – A JavaScript library developed by Facebook for building dynamic user interfaces, particularly single-page applications (SPAs) with reusable UI components.
- Node.js – A JavaScript runtime built on Chrome's V8 engine that allows execution of JavaScript code server-side, enabling full-stack development with JavaScript.

4.2.1 Implementation Details of Modules

There are various modules present in this system. They are

Login Module

Allows existing users to securely access their accounts using their registered credentials.

Register Module

Enables new users to create an account by providing necessary details such as username, email, and password.

Post Music Module

Allows artists or users to upload new music tracks along with relevant metadata (e.g., title, genre, preview segments).

Display Music Module

Handles the presentation of available music to users, including previews, likes/dislikes, and song metadata.

Playlist Module

Enables users to create, manage, and listen to custom playlists containing their liked songs.

Comment Module

Allows users to comment on music tracks, facilitating engagement and feedback within the community.

4.3 Testing

The testing section is accomplished to validate the News portal System. The News Portal System is examined to test if the final system can work in keeping with what we have been waiting for and is free from any programming and logical errors. It additionally makes sure whether or not all of the system and requirements are met or not.

4.3.1 Test Cases for Unit Testing

Unit testing is a software program development method in which the smallest testable components of an application, know as units, are individually and independently scrutinized for correct operation. Below are the numerous tables for distinctive test case.

4.3.1.1 For login

S.N	Test Case	Input	Expected Outcome	Outcome
1.	Valid credential	Valid username and password	Success	User logged in successfully
2.	Invalid credential	Invalid password	Failure	Not logged in
3.	Non existent credential	Non-existent username	Failure	Not logged in
4.	Empty credential	Empty username and password	Failure	Not logged in

4.3.1.2 Register Module

S.N	Test Case	Input	Expected Outcome	Outcome
1.	Valid credential	Valid username, email, and password	Success	User registered successfully
2.	Duplicate credential	Duplicate username	Failure	Not logged in
3.	Invalid credential	Invalid email format	Failure	Not logged in
4.	Invalid credential	Password too short	Failure	Not logged in

5.	Empty credential	Empty input fields	Failure	Not logged in

4.3.1.3 Post Music Module

S.N	Test Case	Input	Expected Outcome	Outcome
1.	Valid credential	Valid music file with title and genre	Success	Music Posted successfully
2.	Invalid credential	Missing title	Failure	Not posted
3.	Invalid credential	Unsupported audio format	Failure	Not posted
4.	Invalid credential	Upload with no preview segments	Failure	Not posted
5.	Invalid credential	Large file upload	Failure	Not posted

4.3.1.4 Display Music Module

S.N	Test Case	Input	Expected Outcome	Outcome
1.	Valid credential	Load homepage with music feed	Success	Music loaded successfully
2.	Valid credential	Like a song	Success	Not posted
3.	Valid credential	Dislike a song	Success	Not posted

4.3.1.5 Playlist Module

S.N	Test Case	Input	Expected Outcome	Outcome
1.	Valid credential	Create new playlist	Success	Playlist created successfully
2.	Valid credential	Add liked song to playlist	Success	Not posted
3.	Valid credential	Remove song from playlist	Success	Not posted
4.	Valid credential	Rename playlist	Success	Not posted
5.	Valid credential	Delete playlist	Success	Not posted

4.3.2 Test Cases for system Testing

System Testing is a from of software testing that is executed on a complete integrated system to assess the compliance of the system with the corresponding requirements.

Sn	Test Case	**Test Data**	**Expected Outcome**	**outcome**
1.	Log in	Username and password	Logged in	User logged in
2.	Register	Username, Email and password	User registered	User registered
3.	Listen to music	Swipe	Music played	Music played
4.	Comment	Mouse click	Comment posted	Comment posted
5.	Like	Mouse click	Liked music	Music liked
6.	View profile	Mouse Click	Profile viewed	Profile not views
7.	Post music	Title, music description	Music posted	Music posted

CHAPTER 5. Conclusion and Future Recommendations

5.1 Lesson Learnt

In creating the online music streaming platform, we have gained valuable insights. First, user-centric design and real-time feedback are vital for building an intuitive and enjoyable experience. We learned that integrating smart recommendation algorithms significantly enhances user engagement and satisfaction. Additionally, proper structuring of user roles (admin, artist, listener) and ensuring secure authentication are essential for a scalable and trustworthy platform. Finally, efficient handling of media content and optimization of music playback features are crucial for smooth and responsive performance.

5.2 Conclusion

In conclusion, the development of a MERN-based music streaming system has demonstrated the potential of modern web technologies in delivering seamless audio streaming, personalized user experiences, and community interaction. This platform empowers users to explore, enjoy, and interact with music content while providing artists with tools to showcase their work. Despite the challenges in implementing features like recommendation systems and preview logic, the project reflects the evolving needs of digital entertainment. With robust foundations and user-focused design, this system serves as a scalable and adaptive solution for the online music streaming domain.

5.3 Future Recommendations

While the current system provides core functionalities, there are several areas where future enhancements can be made:

- In-app music video playback should be supported to enrich the user experience.
- AI-based mood and context-aware recommendations could be implemented for more personalized listening.
- Offline listening capabilities should be added for convenience.
- Live streaming features for artists to connect with fans in real time.
- Social features such as sharing playlists and following friends/artists.
- Advanced analytics for artists to track engagement and listener preferences.