MuSikat: A Mobile Audio Streaming Application for Promoting OPM (Original Pinoy Music)

A Thesis Project Presented to the Faculty of

the School of Computer Studies

University of San Jose - Recoletos

Cebu City, Philippines

In Fulfillment

of the Requirements for the Degree of

Bachelor of Science in Information Technology

By

Franz Lesly R. Rocha

Fabian Miguel F. Cañizares

Hazel Lopez

Mrs. Josephine E. Petralba, MBA
Faculty Adviser

May 2023

ENDORSEMENT

This project entitled **"MuSikat: MuSikat: A Mobile Audio Streaming Application for Promoting OPM (Original Pinoy Music)"** prepared by ROCHA, FRANZ LESLY R., CANIZARES, FABIAN MIGUEL F., LOPEZ, HAZEL S. in partial fulfillment of the requirements for the degree of BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY MAJOR IN MOBILE APPLICATIONS DEVELOPMENT, has been endorsed and is recommended for the acceptance and approval for ORAL EXAMINATION.

ENGR. CARMEL M. TEJANA, MSIT
Member

MRS. JOSEPHINE E. PETRALBA, MBA
Adviser

ENGR. VICENTE III F. PATALITA
Coordinator

MR. RODERICK A. BANDALAN, MSIT
Chairman of the Panel

DR. JOVELYN C. CUIZON
Dean

APPROVAL

Approved by the Tribune for Oral Examination with the grade of **PASSED**.

ENGR. CARMEL M. TEJANA, MSIT
Member

MRS. JOSEPHINE E. PETRALBA, MBA
Adviser

ENGR. VICENTE III F. PATALITA
Coordinator

MR. RODERICK A. BANDALAN, MSIT
Chairman of the Panel

DR. JOVELYN C. CUIZON
Dean

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to the Almighty God for giving us the strength, wisdom, and guidance to complete this research paper. We are grateful for the unwavering support and encouragement of our families and friends, who have been our constant source of inspiration throughout this journey. We also extend our deepest appreciation to the faculty of the SCS, who have imparted their knowledge, expertise, and guidance to us. Their unwavering support has been instrumental in shaping our skills and preparing us for our future careers. We would like to thank everyone who has contributed to our success, whether by providing us with feedback, resources, or simply their kind words of encouragement. We are truly grateful for your contributions.

SPECIAL MENTIONS

We express our heartfelt gratitude to our adviser, Ms. Josephine Petralba, for her invaluable guidance, mentorship, and constructive feedback. Her unwavering dedication and support have been instrumental in the completion of this research paper. Lastly, we would like to extend our appreciation to the individuals who generously allowed us to use their own recorded audio for this project. Your contributions have greatly enriched the quality of our research paper and we are truly grateful for your support.

ABSTRACT

Music is a powerful medium that can transcend cultural barriers and bring people together. In the Philippines, music is an essential part of the country's rich heritage, reflecting its diverse traditions, history, and identity. However, despite its significance, many Filipino musicians struggle to make a living from their craft, with a vast majority making no income, and limited financial support available. To address this issue, this research proposes an audio sharing app that would allow users to stream Filipino music while providing a platform for local musicians to upload their work and gain exposure. By offering a space for artists to showcase their talent, the app aims to foster a community of Filipino music enthusiasts, empowering artists and expanding the reach of Filipino music both locally and globally. The proposed app would be designed to prioritize user experience, with a user-friendly interface that allows for easy navigation and content discovery. Overall, this research aims to contribute to the growth and preservation of Filipino music by providing a platform that fosters collaboration and community building. With the proposed app, Filipino music can reach a broader audience, and local artists can gain greater recognition for their work, promoting a vibrant and thriving Filipino music industry.

CHAPTER 1

INTRODUCTION

Rationale of the Study

In recent years, the way we enjoy music has undergone a significant transformation, and this trend has continued to evolve. The conventional methods of consuming music via tangible media like radio, compact discs, cassettes, or vinyl records have given way to the streaming age. The emergence of streaming has been advantageous in numerous ways, as it has made music more accessible at an affordable price, and has also provided artists with a platform to display their talent.

The Philippines is known for its intense love of music, and OPM has a big part to play in that. OPM or Original Pinoy Music, which describes songs written and performed by Filipino musicians, rose to prominence in the late 1970s [1]. OPM evolved into a genre of music distinguished by a distinctively Filipino style. It changed over time and experimented with various musical genres. The industry has endured years of piracy and a lack of support. More than a billion was reportedly lost in 2011 to piracy in the Filipino music industry. Artists are not properly compensated for their work, and the industry loses profit, which may lead to the closure of more record companies [2]. Since then, OPM has become increasingly popular over the past several years due to the growth of major streaming services like Spotify, Apple Music, Soundcloud, etc.

Spotify, a popular streaming platform in the Philippines, offers a library of music with plenty of variety to be had. The rise of OPM on the platform has led Spotify to create the OPM hub, which consists of Filipino playlists curated by Spotify to feature what’s trending locally, up-and-coming music, and the top artists in OPM [3]. Playlisting is a crucial component of these platforms for artists because it broadens their audience and promotes their music. Through playlisting, listeners enjoy playing music based on their mood and activities while discovering new artists they have never come across. Apple Music, an audio streaming service developed by Apple Inc., is also one platform that includes OPM in its music library. In addition, Soundcloud offers a free platform for both listeners and creators. It allows users to easily upload their own work and it is free to listen to, unlike Spotify and Apple Music, where a third party is involved. These streaming services have given artists the opportunity to make their music more available and have assisted some Filipino artists in experiencing greater levels of success.

Although several platforms offer opportunities for Filipino artists to share their music and combat piracy, most up-and-coming talents from diverse ethno-linguistic backgrounds, such as Bisaya and Ilokano, struggle to gain recognition and penetrate the mainstream Filipino music scene. Despite the growing popularity of Original Pilipino Music (OPM), no music platform in the country focuses on promoting its own culture and specifically highlighting various ethnolinguistic groups in the Philippines, unlike China's QQMusic, Korea's Melon, or India's Gaana. Additionally, there is a need for personalized playlists that cater to specific moods and activities as it is critical for users.

This project aims to develop a mobile music streaming platform that focuses on promoting Original Pilipino Music (OPM) and provides a space for users to share their own music creations. The platform will feature a login and registration authentication system, a music player, and categories for songs based on genre and language. Users can interact with friends through chat, view categories based on genre, ethnic-languages from the Philippines including English, and mood, and follow their favorite artists. The platform will also utilize facial emotion recognition technology to allow users to select music that matches their current emotional state. To fully support the OPM industry, a donation system will be implemented, allowing users to donate to their favorite artists or to the OPM community. This platform will not only promote OPM but also provide a community for music enthusiasts to discover and share their own music. To further enhance the user experience, the platform will include a music insights tool for users to view analytics related to their uploaded music. Additionally, a music recommender system will be implemented, generating personalized recommendations based on the user's search history.

The combination of features will help establish a thriving music streaming platform that fosters the growth of the OPM industry and supports artists and music enthusiasts in their creative endeavors. Whether users prefer to simply listen to music or share their own creations, they will have the opportunity to connect with a community of fellow music lovers on this platform.

REVIEW OF RELATED WORKS

Spotify is a digital platform that offers access to millions of songs, podcasts, and videos created by artists all over the world. Basic features, such as playing music, are free of charge, but users can also opt for Spotify Premium. To discover new music, users can create playlists of their favorite songs or use the "radio station" feature, which generates a playlist based on their musical preferences. Spotify also enables users to synchronize playlists of specific songs to their devices, allowing them to continue listening to music even when they are offline. [4]

Apple Music is a subscription-based music streaming service that provides access to more than 90 million songs, like Spotify. It offers several features, including offline listening when you are not connected, and it consolidates all your music, including CD-ripped tracks, in one place. In addition to live radio stations, Apple Music is integrated with Siri, allowing you to access most of its features using voice commands. To attract customers, Apple Music frequently releases exclusive albums, documentaries, and music videos that are not available on other platforms. [5]

SoundCloud is a music sharing and online audio platform that lets users publish, advertise, and share audio, as well as a digital signal processor that lets users stream audio to listeners. As an entertainment firm with premium services including superior audience insights, playlist pitching tools, and promotional marketing resources for its users, SoundCloud has transformed from a conventional internet streaming platform.

Fan-powered royalties, a user-centric payment mechanism for musicians, were initially adopted and implemented by SoundCloud. [6]

Users of the well-known streaming music service Pandora can listen to music that is randomly selected for them, with an emphasis on algorithm-driven genres that connect artists based on elements like melody, rhythm, and lyrical purpose. Both online and mobile applications are available for Pandora. The business strategy of Pandora lets users choose between a free and a paid service. With the free service, Pandora randomly inserts advertisements into the streaming material. The user may or may not be able to bypass these advertisements. Additionally, users are restricted to a set number of songs skipping every period. Some of these restrictions are removed in the premium version, which provides a more individualized listening experience. [7]

REVIEW OF RELATED LITERATURE

Facial Emotion Recognition (FER)

The appeal of music to people of all ages is frequently attributed to its power to evoke specific emotions or shift our current moods. In fact, it's frequently claimed that music is the language of emotion [8]. People express their emotions primarily through facial expressions. Music has always been known to change people's moods. Capturing and recognizing a person's emotions and displaying appropriate songs matching the one's mood can gradually calm a user's mind and have an overall pleasing effect [9]. With new technology advances, recognizable features of the face can be retrieved as inputs by utilizing a camera. The gathered data helps in determining the mood, and songs are played from a customized playlist [10].

A system proposed by S Matilda and M Uma [10], a music player designed to capture human emotions through the web camera interface available on computing systems. The software captures the image of the user and then, with the help of image segmentation and image processing techniques, extracts feature from the face of a target human being and tries to detect the emotion that the person is trying to express. A paper by Muhammad, S., Ahmed, S., & Naik, D. [11], proposed an emotion detection model to recommend music based on one's mood. With the aim of achieving the highest possible accuracy while not compromising the real time aspect to apply to the real-world scenario. The researchers explored models built differently, including vanilla CNNs and pre-trained networks. One model that stood out was the GAP model. It was able to achieve an accuracy of 66.54% while reducing the number of parameters by around 80%, which is a breakthrough as such a lightweight model is easily mountable on small devices, which adds to real world scenarios' applicability.

Although most of the studies didn't implement emotion recognition for recommending music on mobile phones, one study proposed by Aditya et al. [12] developed an Android application that acts as a personalized music player for a user by analyzing and presenting songs to the user based on their mood. To implement facial recognition algorithms, the application was built with Eclipse and OpenCV. This paper also provided a comparison of various facial detection algorithms. The images of the user were taken with the mobile device's front camera. Its goal was to satisfy music fans by eliciting their emotions. Although identifying emotions is not a primary function of smartphones, their current computational capacity (or access to cloud computing) and abundance of input devices and sensors make it possible. The most crucial sensors and input tools for emotion recognition are then discussed. Additionally, the improvement in the quality of front cameras is also critical for emotion recognition, as the optical channel is one of the most important and widely used in this task [13].

Different Methods of Playlisting

As technology has improved, downloading and streaming have taken over as the primary ways that we consume music. Playlists have emerged as a crucial method of accessing sizable music libraries. Playlists are essentially collections of songs that are typically created by hand and arranged according to a logical or overarching theme [14].

As stated by Dias et al. [15], a playlist can be defined as a "sequence of songs meant to be listened to as a group". Research on playlist creation has been done according to three perspectives: i) manual creation; ii) automatic generation and recommendation; and iii) assisted/editorial playlist creation.

Manual creation

The most straightforward method for creating playlists is manual creation or user-generated playlists, which gives users complete control. Manually creating playlists is a time-consuming task. It is a time-consuming process that necessitates a thorough understanding of data collection [16]. Even though it is a simple and straightforward method, there are numerous factors to consider when creating playlists manually. An analysis of playlist creation conducted by Cunningham et al. [17] shows how people manually create and organize playlists. The study consists of several reasons why people create playlists, such as: i) to serve as the background for other activities (such as traveling, studying, or exercising at the gym); ii) to convey or express an emotion; iii) to be used in an event, like, for instance, a party or a wedding. Although the manual creation method is the simplest method for assembling a playlist, it is a complex subject with several issues that require attention from researchers. The researchers suggest that automatic creation of a playlist is a better solution since it is less expensive, less time-consuming and can be used to adapt to different listeners' tastes and behaviors, unlike manually created playlists [15].

Automatic generation and recommendation

Automated methods can create playlists almost entirely without human involvement, maximizing the creators' effort in the process [15]. The objective of automatic playlist generation is to create playlists in a manner that the user would have done themselves, resulting in endless hours of enjoyment without having to spend much time creating playlists [16]. Research on these methods has been essentially done on the algorithms that encode preferences and tastes and perform the suggestion of sets of songs. Over the past few years, numerous methods for creating playlists automatically have been put forth. Authors Bonnie et al. [18] and Sneha et al. [19] have classified algorithms into seven categories. 1) Similarity-based Algorithms; 2) Collaborative Filtering; 3) Frequent Pattern Mining; 4) Statistical Models; 5) Case-Based Reasoning; 6) Discrete Optimization and 7) Hybrid Techniques. These algorithms help solve the problems faced by automatic playlist creation, though they lack control and transparency during the creation process [15].

Assisted playlists

By visualization techniques, assisted methods help users complete browsing and playlist creation tasks [20]. The assisted method combines manual and automated approaches into one strategy. It gives creators control over playlist creation, potentially saving them time and effort. Both types of solutions can benefit from visualization techniques, which are interactive, transparent, encourage control, and engage users as they make playlists [15].

Other methods

Platform-curated playlists such as editorial and algorithmic can be found on streaming services such as Spotify, Apple Music etc [21]. Nowadays, most music streaming services have editorial playlists that are either run by the streaming services themselves or by independent brands or playlist companies. Editorial playlists were described as a feature that would assist users in need of musical guidance, and they offer a strategically important influence over music consumption [22].

SIGNIFICANCE OF THE STUDY

The study will have significant benefits for various individuals and groups.

For musicians/artists, they will have a platform to showcase and promote their own music, reaching a wider audience and potentially increasing their fan base.

For listeners, they will have access to a mobile app that enables them to listen to their favorite OPM music artists, and even have incentives for their support.

For mobile developers, they will gain knowledge on the different techniques and procedures involved in creating an audio streaming app, which they can use as a reference for future projects.

For future research, this can serve as a basis for more advanced audio streaming apps and provide a more personalized listening experience for users.

PROJECT OBJECTIVES

The primary goal of this project is to develop a mobile music sharing app using the Flutter SDK Framework that will enable users to listen to their favorite OPM music artists while also providing a platform for budding OPM musicians to promote their music.

General Objectives:

The mobile app aims to achieve the following objectives:

1. To promote the rich culture and ethnolinguistic groups in the Philippines through music.
2. To provide users with more personalized and relevant music recommendations.
3. To give exposure and help aspiring OPM musicians to broaden their audience and promote their music.

Specific Objectives:

To accomplish the general objectives, the developers aim to:

1. To integrate a recommender system based on identified criteria to provide users with personalized music recommendations.
2. To develop a user-friendly music player that enables users to listen to music, favorite specific artists, or albums, and create playlists.
3. To provide personalized playlists using different methods such as user, editorial, or algorithmic methods based on the listener's mood or emotion.
4. To categorize music based on the genre or ethnic language to make it easier for users to discover and explore new music.

SCOPE AND LIMITATION OF THE STUDY

This study aims to develop a mobile application that is compatible with Android operating system version 5.0 (Lollipop) and above. However, there are several challenges that need to be addressed. Firstly, in terms of music recommendations, users expect music streaming services to have a better understanding of their preferences than they do themselves. However, the algorithm used may occasionally suggest songs that are unrelated to the user's listening preferences, which could be problematic. This could be due to issues such as scalability and lack of data. Secondly, the use of emotion recognition in mobile devices involves utilizing the front camera to recognize facial features as inputs. However, the accuracy of this approach is still a challenge, as validation of emotion datasets is necessary to have an accurate emotion recognition system. Lastly, the app will only cater to OPM music, which means foreign or international music will not be available on the platform. Furthermore, acquiring music from artists signed to labels may be difficult as a deal or agreement with the musician's record label is required.

CHAPTER 2

SOFTWARE REQUIREMENTS AND DESIGN SPECIFICATIONS

This chapter provides a comprehensive overview of the application's functionalities and performance expectations. It details the structure and processes that will be implemented to meet the needs of the application's stakeholders, ensuring that their requirements are met efficiently and effectively. It consists of the Architectural Diagram, Use Case Diagram, Use Case Narrative, the Activity Diagram, the Class Diagram, and the User Interface Design.

APPLICATION OVERVIEW

MuSikat is a mobile app that promotes Original Pinoy Music (OPM) and supports local artists in the Philippines. It offers a wide range of OPM songs categorized by genre, language, and mood, with personalized music recommendations based on user search history. The app includes a music player accessible through a login and registration authentication, allowing users to play any music they want. Listeners can interact with friends through the chat feature, follow their favorite artists, and make playlists using manual selection and emotion recognition. The app also offers a donation feature through Patreon. For artists, MuSikat provides an easy-to-use platform to upload their original audio or cover songs and view their music's performance and earnings through a dashboard.

ARCHITECTURAL DIAGRAM

The architectural diagram shows the application architecture view, which helps the reader identify the applications, database, services, and interactions.

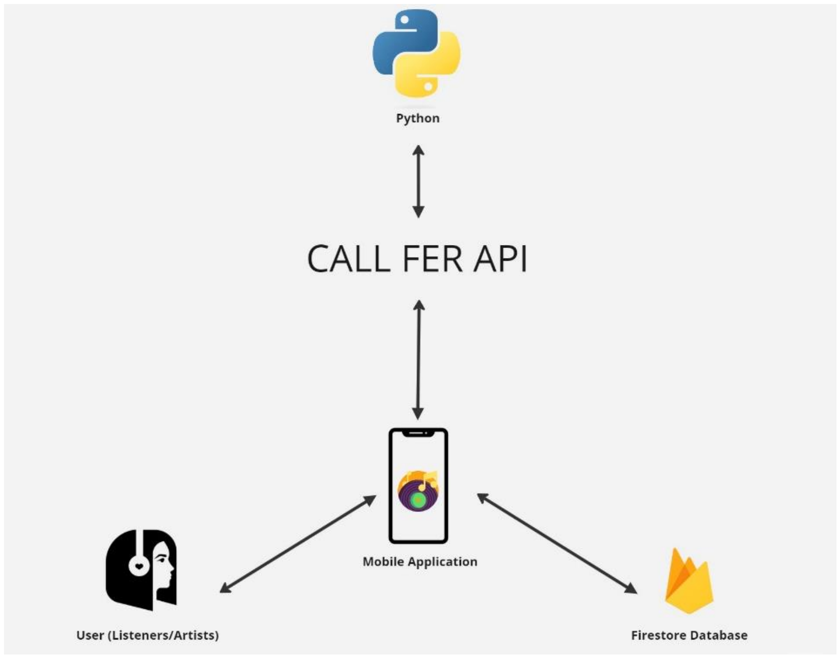


Figure 1: Architectural Diagram

The architectural diagram in Figure 1 illustrates how data and information are exchanged among different components of MuSikat. The system comprises several elements such as clients, mobile app, Firestore database, and Facial Emotion Recognition (FER) that operates through an API call from Python. MuSikat is designed for a single user, who can either be a listener or an artist. The mobile app facilitates communication and connectivity among users. The Firebase database holds and manages the data required by the system's users. The Facial Emotion Recognition (FER) system in MuSikat is integrated with an API that enables it to be called from a Python script to analyze facial expressions of users

USE CASE DIAGRAM

A use case diagram depicts the system's users and their interactions with the system. It defines the events in a system and how they flow. The use case narrative elaborates the details of each use case.

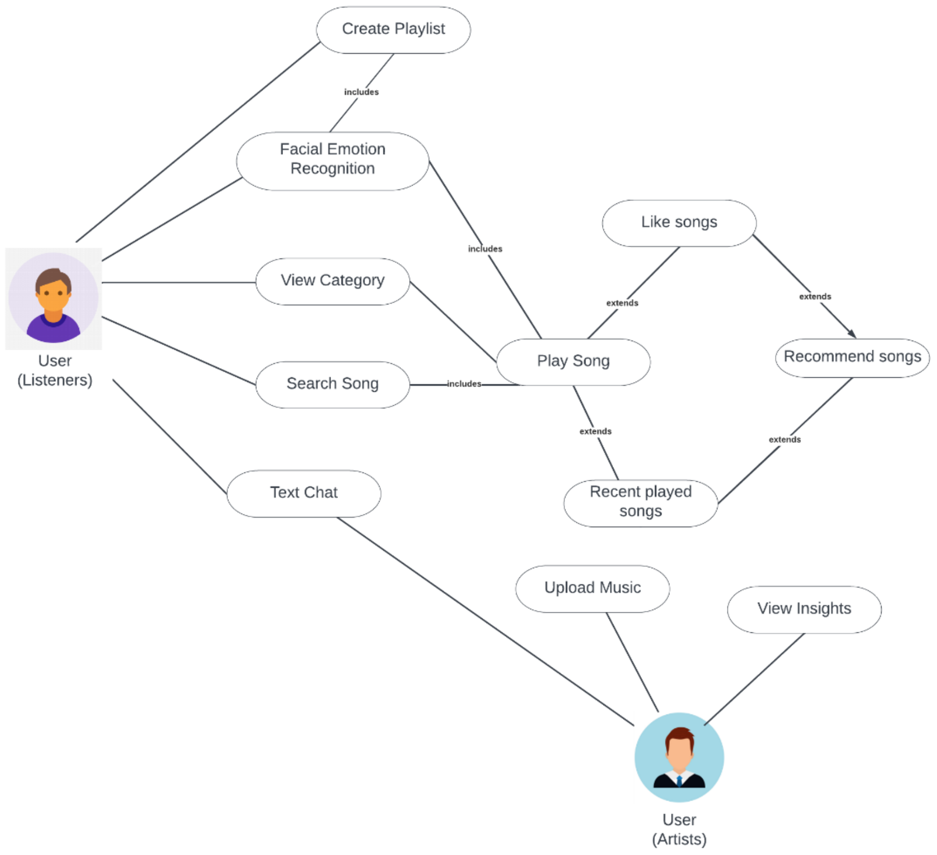


Figure 2: Use Case Diagram

Figure 2 illustrates the relationship of the actors and how they interact with the system. It describes how the MuSikat Users and the system interact with each other through different use cases.

USE CASE NARRATIVE

The use case narrative shows the main success scenario and alternative flows of a use case. It provides more details about the use cases.

Use Case 001:	***Create Account***
Actor(s):	User
Type:	Required
Overview:	This use case allows users/listeners to register and as well as authenticate their personal information, providing them access to a variety of audio streaming experience functionalities.
Precondition(s):	• The user must have a compatible device with an active internet connection capable of running the mobile application. • The application must be successfully installed on the user's device. • The user must have a valid email address to register an account. • The user must provide accurate and valid personal information upon account creation process.
Postconditions:	• The user's account information is stored securely in the app's database. • The user is logged into their account and can access all the app's features. • The user is directed to the app's home screen upon successful account creation.
Actor action	**System Response**
1. User taps the icon button in Register	The system displays the registration screen with fields for the user to input their personal information.
2. The user enters their personal information as required by the application	System validates input data for accuracy and completeness. If data is invalid, display an error message for correction. If valid, creates an account and displays a success message.

3. User taps the Checkbox to accept the terms and conditions.	The system displays a checkmark in the checkbox and allows the user to proceed to register an account, it indicates that the user has successfully accepted the terms and conditions.
4. User taps the "Submit" button	The system validates the inputted data to ensure that it is complete and accurate.
Exception Flow	
E1: The user enters invalid or incomplete information for account creation. 1.1. If a user enters an invalid email address or password upon authentication, the system displays an error message prompting the user to correct the information.	
E2: The system identifies that the account already exists. 2.1. If a user tries to create an account with an email address that is already used in the system, the system displays an error message prompting the user to correct the information.	
E3: The system is down or experiencing technical errors. 3.1. The system displays an error message prompting the user to try again.	
E4: The user has poor internet/mobile data connectivity. 4.1. The system displays an error message prompting the user to try again once the connection is stable.	

Table 1 - Use Case Narrative for User Account Creation

Use Case 002:	***Log In***
Actor(s):	User
Type:	Required
Overview:	This use case enables the users/listeners to log in to their registered account.
Precondition(s):	• The user must have a compatible device with an active internet connection

	capable of running the mobile application. • The application must be successfully installed on the user's device. • The user must have registered already. • The user's account information provided must be securely stored and retrievable from the database
Postconditions:	• The user will be able to log in to their registered account. • The user will be able to access all the app's features and functionalities.
Actor action	**System Response**
1. The user taps the Login button and Inputs the registered email ad and password.	The system will validate the inputted username and password as to whether it is a registered account information.
	The system will display a successful login message and grant access to the user's account.
Exception Flow	
E1: The system identifies the credentials as invalid. 1.1. If a user enters an invalid email address or password upon authentication, the system displays an error message prompting the user to re-enter credentials or reset password if the password is forgotten.	
E2: The system is down or experiencing technical errors.	

2.1. The system displays an error message prompting the user to try again.	
E3: The user has poor internet/mobile data connectivity. 3.1. The system displays an error message prompting the user to try again once the connection is stable.	
Alternative Flow - Forgot Password	
Actor Action	**System Response**
1. The user taps on the Forgot Password button.	The system prompts the user to input the registered email address.
2. The user will input his registered email address.	The system sends a password reset link to the registered email address.
Exception Flows	
E.1 The system detects that the entered email address is not valid or not associated with any registered account. 1.1. The system displays an error message indicating that the entered email address is invalid and prompts the passenger to re-enter a valid email address.	

Table 2 - Use Case Narrative for User Account Login

Use Case 003:	***Upload Audio***
Actor(s):	User
Type:	Required
Overview:	The user will upload an audio in the application.
Precondition(s):	• The user must have a compatible device with an active internet

	connection capable of running the mobile application. • The application must be successfully installed in the user’s device. • The user has a valid account of the app. • The user has the music file in a compatible format and meets the platform's quality standards. • The system may only accept certain audio file formats, such as MP3, WAV, or AAC. The user may need to convert their file to the correct format before uploading it.
Postconditions:	• The audio is successfully uploaded to the app. • The user will receive confirmation of the successful upload.
Actor action:	**System Response**
1. The user navigates to the "Upload" section of the platform.	The system displays the upload screen.
2. The user selects the music file they want to upload from their device.	The system verifies that the music file is in a compatible format and meets the platform's quality standards.
3. The user enters information about the new music audio, such as the title, artist name, genre, language, etc.	The system stores the new music track in the platform's database.

4. The user submits the new music for upload to the audio streaming platform.	The system sends a confirmation message to the Music Artist that the new music track has been successfully uploaded.
Exception Flow	
E1. The music file is not in a compatible format. 1.1. If the user uploads a file that does not meet the platform's quality standards, the system displays an error message prompting them to correct the issue before uploading the file.	
E2. The system is down or experiencing technical errors. 2.1. If there are any issues with the upload process, such as a loss of internet connection or an error in the platform's database, the system displays an error message to the user and prompts them to try again later.	
Alternative Flow of Events - Cancel the Upload Process	
Actor Action	**System Response**
1. The user can cancel the upload process at any time by selecting the appropriate option on the platform.	If the user chooses to cancel the upload process, the system cancels the upload and removes any partially uploaded files from the platform's database.
Exception Flow	
E.1 If there are any issues canceling the upload process 1.1. If there is such as a loss of internet connection or an error in the platform's database, the system displays an error message to the user.	

Table 3 - Use Case Narrative for Upload Audio

Use Case 004:	***Search User/Song***

Actor(s):	**User/Listener**
Type:	**Required**
Description:	This use case allows users to search for other users and songs in the app.
Precondition(s):	• The user must have an account on the audio streaming platform. • The user must have access to a device with a stable internet connection.
Postconditions:	• The user is able to find the desired user or audio track.
Actor action:	**System Response**
1. The user enters a search query for the desired user or audio track in the platform's search bar.	The system searches for the user or audio track that matches the search query.
	The system displays a list of search results that match the search query.
2. The user selects the desired search result from the displayed search results.	The system allows the user to select the desired search result to view the user or audio track.
Exception Flow	
E1. There are no search results that match the search query. 1.1. If there are no search results that match the search query, the system displays a message informing the Platform User that no results were found.	

E2. The system is down or experiencing technical errors. 2.1. If there are any technical issues with the platform's search functionality, the system displays an error message and prompts the user to try again later.

Table 4 - Use Case Narrative for Search User/Audio

Use Case 005:	***Music Player***
Actor(s):	User
Type:	Required
Overview:	The user will be able to stream/play music.
Precondition(s):	• The Music Player application is installed and running on the user's device. • The user has a collection of music files that are compatible with the Music Player application. • The user must have access to a device with a stable internet connection.
Postconditions:	• The selected music file is played by the music player.
Actor action:	**System Response**
• **The user opens the Music Player application.**	The Music Player application opens and displays the user's music collection.

• **The user selects a music file to play from their music collection.**	The selected music file is played by the application.
• **The user adjusts the volume, skips to the next or previous track, or pauses the playback as desired.**	The Music Player application responds to user actions and adjusts the playback accordingly.
Exception Flow	
E1. The system is experiencing technical errors. 1.1. If the Music Player application is unable to connect to the server due to an internet connection error, the song will not be played. 1.2. If the Music Player application is unable to access the music file due to a database error, an error message is displayed, and the file is not played.	

Table 5 - Use Case Narrative for Music Player

Use Case 006:	***Create Playlist***
Actor(s):	User
Type:	Required
Description:	This use case enables users to create and manage their own playlists of songs.
Precondition(s):	• The application is installed and running on the user's device. • The user must have a compatible device with an active internet connection capable of running the mobile application. • The user has selected the "Playlist" option in the app.

Postconditions:	• The user has created a new playlist or modified an existing one. • The user can play, edit, and delete the playlist as desired.
Actor action:	**System Response**
1. The user navigates to "Create Playlist".	The app displays the user's playlist library.
2. **The user enters a name and description for the playlist.**	The app creates a new playlist with the user's specified name and displays an empty playlist.
3. The user can choose an image cover for the playlist by selecting the Container with a Camera Icon.	The system prompts the user to choose an image from their device or from a selection of default images.
4. The user saves the playlist.	The user can save the playlist and access it later.
Exception Flow	
E1. The system is experiencing network error. 1.1. If the user loses their internet connection while creating a playlist, the app may display an error message indicating that the action could not be completed due to a network error.	
E2. The system is experiencing technical error. 2.1. If the user experiences a technical issue or error with the streaming platform while creating a playlist, the platform may display an error message and prompt the user to try again later or contact customer support.	

Table 6 - Use Case Narrative for Create Playlist

Use Case 007:	***Facial Emotion Recognition (FER)***
Actor(s):	User
Type:	Required
Overview:	This use case allows the camera to detect the user's facial expressions and emotions and selects and plays songs that match the user's mood.
Precondition(s):	• The FER system is integrated into the app. • The user has access to a camera that can capture facial expressions. • The application should have FER technology integrated or available as a feature to use in the use case. • The user's face should be well-lit to ensure that the FER algorithm can accurately detect their emotions. • Any facial data collected during the use case should be handled and stored appropriately to ensure user privacy and security.
Postconditions:	• The FER technology detected the user's facial emotions as intended. • The system will play a song or curate a playlist based on the user's detected emotion
Actor action:	**System Response**
1. The application requests permission to access the device's camera.	The FER system prompts the user to grant permission to access the camera.

2. The FER system analyzes the user's facial expressions and emotions.	The FER system accesses the camera and analyzes the user's facial expressions and emotions in real-time.
3. The system provides real-time feedback to the user, displaying the detected emotions on the screen	The FER system displays the detected emotions on the screen he FER system displays the detected emotions on the screen
4. Based on the user's emotions, the FER system displays two buttons: "Play a Song" and "Create a Playlist".	The FER system analyzes the user's emotions and displays two buttons on the screen that are relevant to the detected emotions.
5. If the user selects "Play a Song", the FER system will play a song based on the user's emotions.	The FER system will select and play a song that is appropriate for the user's detected emotions.
6. If the user selects "Create a Playlist", the FER system will create a playlist based on the user's emotions.	The FER system will create a playlist of songs that are appropriate for the user's detected emotions.
Exception Flow	
E1. The camera is not working or not accessible. 1.1. If the camera is not working or not accessible, the FER system will not be able to detect the user's facial expressions and emotions.	

E2. No visible person has been detected by the camera. 2.1. If there are no persons visible in the camera, the Facial Expression Recognition (FER) system will not be able to detect or analyze any facial expressions.

Table 7 - Use Case Narrative for Facial Emotion Recognition

Use Case 008:	***View Insights***
Actor(s):	User
Type:	Required
Overview:	This use case allows the user to view and analyze data about user activity, such as the number of plays and top songs.
Precondition(s):	• The audio streaming platform is functioning properly. • The user has uploaded at least one song. • The application is installed and running on the user's device. • The user must have a compatible device with an active internet connection capable of running the mobile application.
Postconditions:	• The user has access to data insights about song activity on the platform.
Actor action:	**System Response**
1. The user opens the insights screen.	The system displays the insights which consists the top tracks, song plays and like count.

2. The user selects the data insights they want to view.	The system retrieves and displays the requested data insights to the user or admin.
Exception Flow	
E1. The user has no song uploaded 1.1. If there's no song uploaded then there is will be no insights displayed.	
E2. The user has poor internet/mobile data connectivity. 2.1. The system displays an error message prompting the user to try again once the connection is stable.	

Table 8 - Use Case Narrative for View Insights

Use Case 009:	***Follow/Unfollow Artists***
Actor(s):	Listener
Type:	Required
Overview:	This use case allows users to follow their favorite artists and receive notifications about new releases and events.
Precondition(s):	• The audio streaming platform is functioning properly. • The user is logged in to their account.
Postconditions:	• The user has followed or unfollowed an artist on the audio streaming platform.
Actor action:	**System Response**
1. The user searches for the artist they want to follow.	The app interface displays the search bar.
2. The user selects the artist they want to follow.	The system retrieves the artist data and displays it to the user.

3. The user selects the Follow button.	The system adds the artist to the user's followed artists list.
Exception Flow	
E1. The audio streaming platform is not functioning properly. 1.1. If the audio streaming platform is not functioning properly, the Follow/Unfollow Artist feature may not be accessible.	
Alternative Flow of Events - Search Specific Profile	
Actor Action	**System Response**
1. 1. The user selects their profile and selects the Following tab.	The audio streaming platform interface displays the user's profile.
2. The user selects the Following tab.	The system retrieves the user's followed artists list and displays it to the user.
3. The user selects the artist they want to unfollow.	The system removes the artist from the user's followed artists list.
Exception Flow	
E1. The audio streaming platform is not functioning properly. 1.1. If the audio streaming platform is not functioning properly, the Follow/Unfollow Artist feature may not be accessible.	
E2. The artist does not exist in the user's following list. 2.1. If the artist is not in the user's followed artists list, the user may not be able to unfollow them.	

Table 9 - Use Case Narrative for Follow/Unfollow Artists

Use Case 010:	***Chat Messaging***
Actor(s):	Listener

Type:	Required
Overview:	This use case allows the process of chatting and sending messages between users.
Precondition(s):	• The user must have a valid account on the audio streaming platform. • The user must be logged in to their account. • The user must have a stable internet connection.
Postconditions:	• The user has successfully sent a message to another user. • The recipient user has received the message.
Actor action:	**System Response**
1. The recipient user has received the message.	The system displays the chat messaging feature and provides a list of all the user's previous conversations.
2. The user selects a recipient user to send a message to.	The system displays the list of recipient users that the user can send a message to.
3. The user types their message in the text field provided.	The system provides a text field for the user to type their message.
4. The user clicks on the "Send" button and the	The system provides a "Send" button to allow the user to send the message. After that, it sends the

message is sent to the recipient user.	message to the recipient user and confirms successful delivery.
Exception Flow	
E1. The user does not have a stable internet connection. 1.1. If the user does not have a stable internet connection, the system displays an error message informing them that the message could not be sent.	

E1. The user does not have a stable internet connection. 1.1. If the user does not have a stable internet connection, the system will not load in accessing different categories.

Table 10 - Use Case Narrative for Chat Messaging

Use Case 011:	***Choose Categories (Language, Genre, Moods)***
Actor(s):	Listener
Type:	Required
Overview:	This use case allows the user choosing categories such as language, mood, and genre to customize their audio streaming experience.
Precondition(s):	• The user must be logged in to their account. • The audio streaming platform must have a selection of music in various languages, moods, and genres.

Postconditions:	• The user's selected categories will be saved in their account settings.
Actor action:	**System Response**
1. The user navigates to the categories section of the platform.	The system displays the categories section of the platform.
2. The user selects their preferred language, mood, and genre categories.	The system presents the user with a selection of language, mood, and genre categories to choose from.
Exception Flow	
E1. The user does not have a stable internet connection. 1.1. If the user does not have a stable internet connection, the system will not load in accessing different categories.	

Table 11 - Use Case Narrative for Choose Categories (Languages, Genres, Moods)

Use Case 012:	***View Library***
Actor(s):	Listener
Type:	Required
Overview:	This use case allows the user to view their library. The library displays all the music uploaded.
Precondition(s):	• The user must be logged in to their account.

	• The user has added at least one song to their library.
Postconditions:	• The user is presented with their library containing all the songs they have added.
Actor action:	**System Response**
1. The user clicks on the "Library" button on the home screen or in the navigation menu.	The system displays the user's library.
2. 2. The system retrieves and displays the user's library.	The user can browse through their library by scrolling and can select any song available to play.
Exception Flow	
E1. The user has not added anything to their library yet. 1.1. If the user has not added anything to their library, the system displays a message saying "Your library is empty".	

Use Case 013:	***Liked Song/s***
Actor(s):	User
Type:	Required
Overview:	This use case allows users to view and manage a user's liked songs on the app.
Precondition(s):	• The user is logged in to their account.

	• Users have previously liked one or more songs.
Postconditions:	• Users can view a list of their liked songs. • Users can play, remove, or add to a playlist any of their liked songs.
Actor action:	**System Response**
1. User navigates to the "Liked Songs" section.	The app displays the user's liked songs.
2. The system displays a list of the songs the user has liked.	The system updates the user's liked songs list to reflect any changes made by the user.
Exception Flow	
E1. The user has not liked any songs yet. 1.1. If the user has not liked any songs, the "Liked Songs" section will be empty and display a message indicating that the user has no liked songs.	

Table 13 - Use Case Narrative for Liked Song/s

Use Case 014:	***Support Artist***
Actor(s):	User
Type:	Required
Overview:	The donate system for musicians processes the donation and sends the funds directly to the musician's bank account, minus any transaction fees.
Precondition(s):	• The user is logged in to their account.

	• The user would need to have a valid bank account to receive the donations and be in a country where the donate system is available. • The musician would also need to provide accurate and up-to-date personal and bank account information during the registration process.
Postconditions:	The musician can receive financial support from their fans, which helps them continue creating music and sharing it with their audience.
Actor action:	**System Response**
1. The musician goes to the donate system for musicians' website and clicks on the "Verify" button	The donate system for musicians verifies the musician's email address by sending a verification email to them. The musician clicks on the verification link in the email to activate their account.
2. After their account is activated, the musician logs in to the donate system for musicians and clicks on the "Create Donation Page" button.	The system will make the support button appear on the artist’s profile.
Exception Flow	
E1. The user tries to donate but their payment is declined due to an issue with their credit card. 1.1. The donate system for musicians displays an error message, informing the fan that their payment was declined. The system gives the fan the option to try again with a different payment method or contact customer support for assistance.	

E2. The musician receives a donation from a fan but their bank account information is incorrect.
The donate system for musicians notifies the musician of the error and provides instructions on how to update their bank account information. The system puts the donation on hold until the issue is resolved.
E3. The user makes a donation but the donate system for musician's experiences technical difficulties and is temporarily unavailable.
The donate system for musicians displays an error message, informing the fan that the system is currently unavailable.

Table 14 - Use Case Narrative for Support Artists

Use Case 015:	***Edit Playlist***
Actor(s):	User
Type:	Required
Overview:	This use case allows the user to edit a playlist they have created on the audio streaming platform. The user can add, delete, or rearrange songs on the playlist.
Precondition(s):	• The user has created a playlist. • The user is logged into their account.
Postconditions:	• The playlist has been successfully edited and saved to the user's account.

Actor action:	**System Response**
1. User navigates to their playlist page.	Users remove any song(s) they want to delete from the playlist.
2. User selects the playlist they want to edit.	The system displays the "Edit" button.
3. User selects the song(s) they want to add to the playlist.	The system shows a search bar and other filters to help the user find the songs they want to add to the playlist.
4. Users remove any song(s) they want to delete from the playlist.	The system shows a confirmation message asking the user to confirm the deletion of any song(s).
Exception Flow	
E1. The user does not have any playlists created yet. 1.1. If the user does not have any playlists, the system displays a message informing the user that they have no playlists to edit.	
E2. The user clicks the "Cancel" button. 2.1. If the user clicks the "Cancel" button, the system discards any changes and returns the user to the playlist page.	
E3. The user cancels the deletion of the selected song. 3.1. If the user cancels the deletion of a song, the system restores the song to the playlist.	

Table 15 - Use Case Narrative for Edit Playlist

Use Case 016:	***Add Songs to Playlist***
Actor(s):	User

Type:	Required
Overview:	This use case allows the user to add a song to a playlist in an audio streaming platform.
Precondition(s):	• The user is logged in to their account on the audio streaming platform. • The user has an existing playlist or has created a new playlist. • The user is currently browsing or listening to a song.
Postconditions:	• The song is added to the specified playlist of the user.
Actor action:	**System Response**
1. User selects the song that they want to add to a playlist.	The audio streaming platform displays the "Add to Playlist" option when the user selects a song.
2. User selects the playlist to which they want to add the song.	The audio streaming platform displays a list of the user's playlists for the user to choose from.
3. User confirms the addition of the song to the selected playlist.	The audio streaming platform adds the selected song to the specified playlist of the user and displays a confirmation message.
Exception Flow	
E1. The user does not have any existing playlist. 1.1. The audio streaming platform prompts the user to create a new playlist.	
E2. The user is not logged in to their account.	

2.1. The audio streaming platform prompts the user to log in or create an account.

Table 16 - Use Case Narrative for Add Song to Playlist

ACTIVITY DIAGRAM

This activity diagram shows the workflow behavior of the system by describing the sequence of actions in a process.

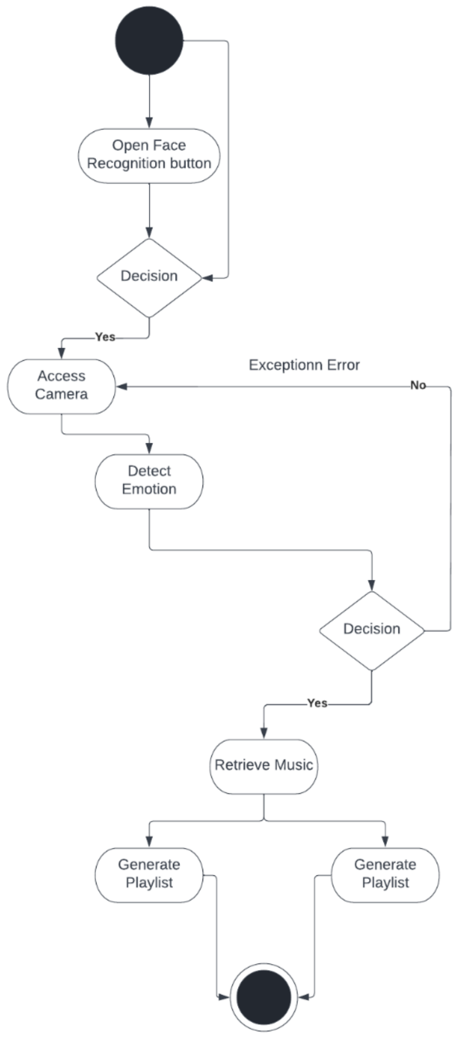


Figure 3.1: Facial Recognition Activity

Figure 2 shows the behavior of face recognition where the user can access the camera to detect emotions using the API system. By clicking the button where the user can use facial emotion recognition, it allows the user to detect their facial expression and it will curate a playlist or play a song depending on their mood.

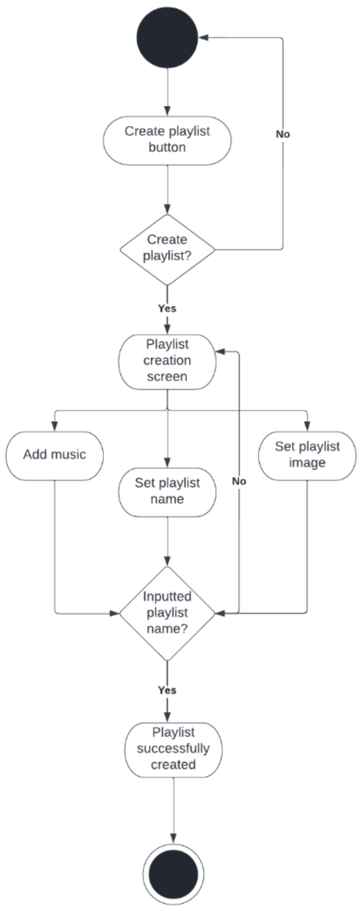


Figure 3.2: Create Playlist Activity

Figure 3 shows the activity of creating a playlist, wherein users can simply curate their own playlists for personal enjoyment.

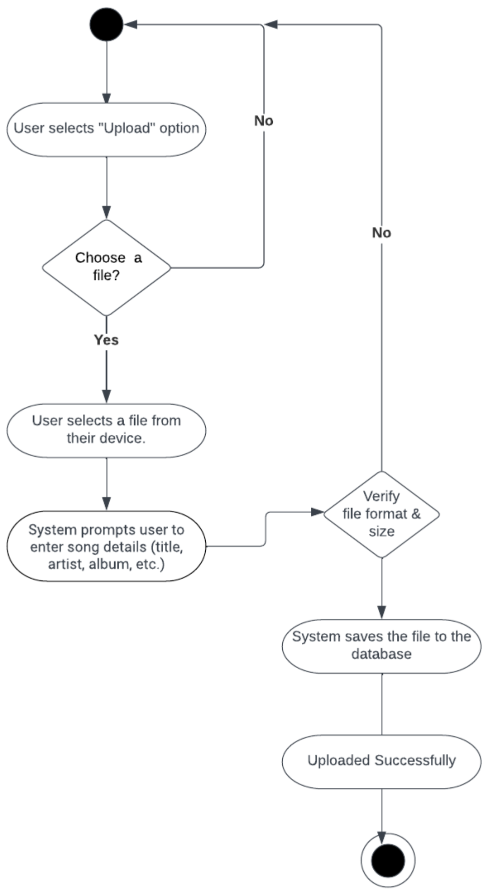

User selects "Upload" option
No
Choose a file?
No
Yes
User selects a file from their device.
System prompts user to enter song details (title, artist, album, etc.)
Verify file format & size
System saves the file to the database
Uploaded Successfully

Figure 3.3: Upload Audio/Song

Figure 3 shows the process of uploading a song. The user selects the "Upload" option, chooses a file from their device, enters song details, and the system saves the file and details to the database. A confirmation message is displayed to the user.

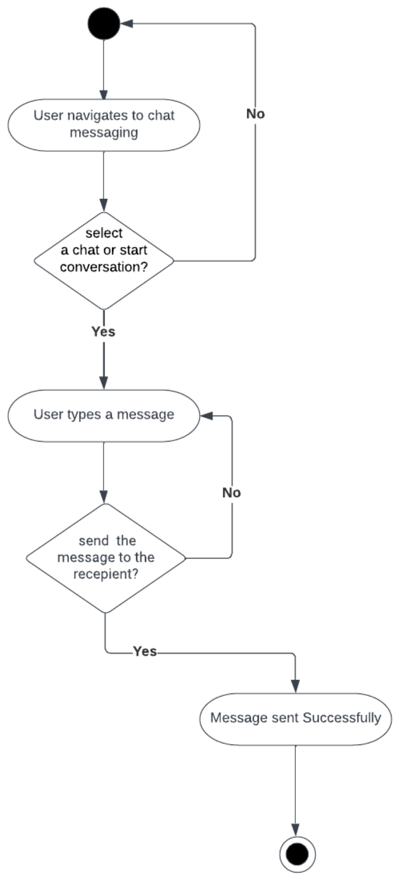

User navigates to chat messaging
No
select a chat or start conversation?
Yes
User types a message
No
send the message to the recepient?
Yes
Message sent Successfully

Figure 3.4: Chat Messaging

Figure 4 demonstrates the chat functionality of the app. The user can open the chat app and select a chat room or start a new chat. They can then type and send a message, which the system will send to the appropriate recipient or chat room.

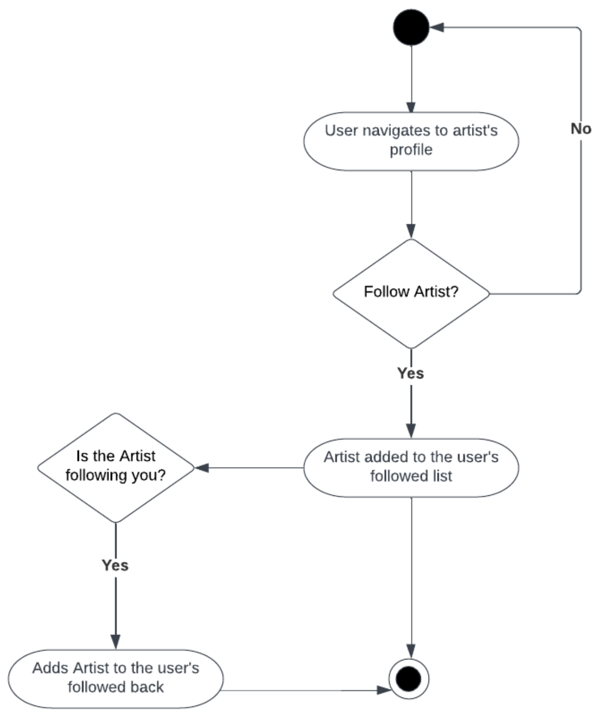

User navigates to artist's profile
No
Follow Artist?
Yes
Is the Artist following you?
Artist added to the user's followed list
Yes
Adds Artist to the user's followed back

Figure 5: Follow Artist

Figure 5 illustrates the process of following an artist on the MuSikat app. The user starts by opening the artist's page and clicking the "Follow" button. Upon clicking, the system adds the artist to the user's followed list. Additionally, the system checks if the artist is also following the user. If the artist is following the user, the system adds the artist to the user's "followed back" list. This feature enables users to stay up-to-date with their favorite artists' latest releases and updates, thereby fostering a stronger connection between artists and their fans.

CLASS DIAGRAM

The class diagram shows the structure of the system as classes with their attributes, operations, and relationships among other classes.

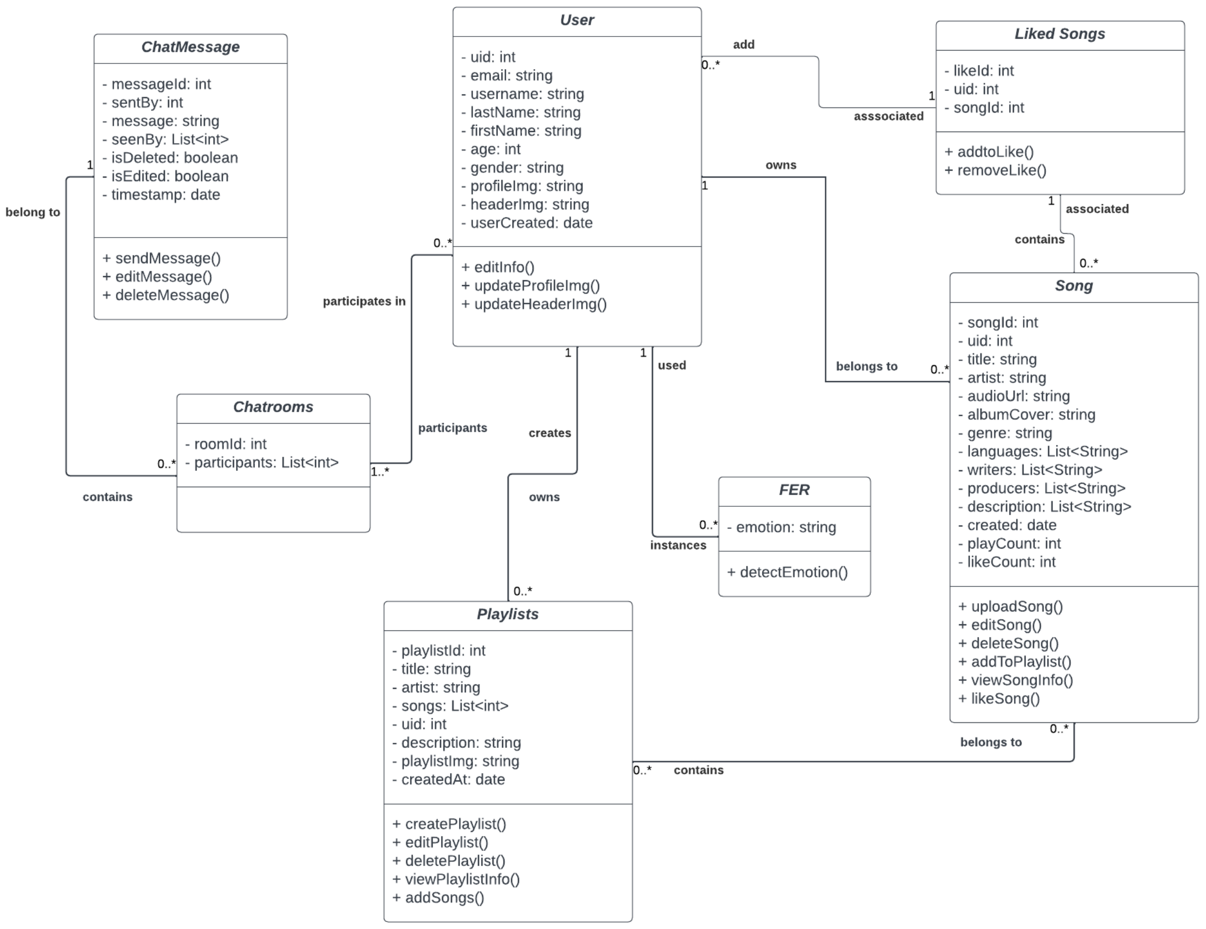


Figure 2: Class Diagram

Users

The user entity contains the basic information of each user who has registered on the application. The information stored includes the user's name, email, password, and other relevant details.

Song

The song entity stores information about each song in the app. This includes details such as the song title, artist, album, genre etc. Each song is associated with the user who uploaded it, allowing users to manage their own library of songs.

Playlists

The Playlist entity represents a collection of songs that a user can create and manage within the app. It stores information about each playlist such as its name, description, and the list of songs included in it. Users can add and remove songs from their playlists, and can also share their playlists with other users.

Liked Songs

The LikedSongs entity represents the relationship between a user and the songs they have liked. It stores the user ID and the song ID for each liked song, allowing for quick retrieval of the user's liked songs.

Chatrooms

The Chatrooms entity represents the chat rooms available in the app where users can communicate with each other in real-time. It stores information such as the users participating in the chat room, and the messages exchanged in the chat room.

Chat Messages

The ChatMessages entity stores the messages exchanged between users in a particular ChatRoom. It contains the message content, the user who sent the message, the time of the message, and the ChatRoom to which the message belongs. This entity allows for the tracking of messages and facilitates the communication between users in the app.

FER (Facial Emotion Recognition)

FER is an entity that represents the emotion detection system used in the application. It analyzes the user's facial expressions through the camera and identifies the user's current emotion, then recommends a song based on the detected emotion. The FER entity stores the necessary data for emotion detection and song recommendation.

USER INTERFACE DESIGN

This section presents the user interface and user experience design for MuSikat, displaying all the screens of the application that a potential user would encounter while using the application.

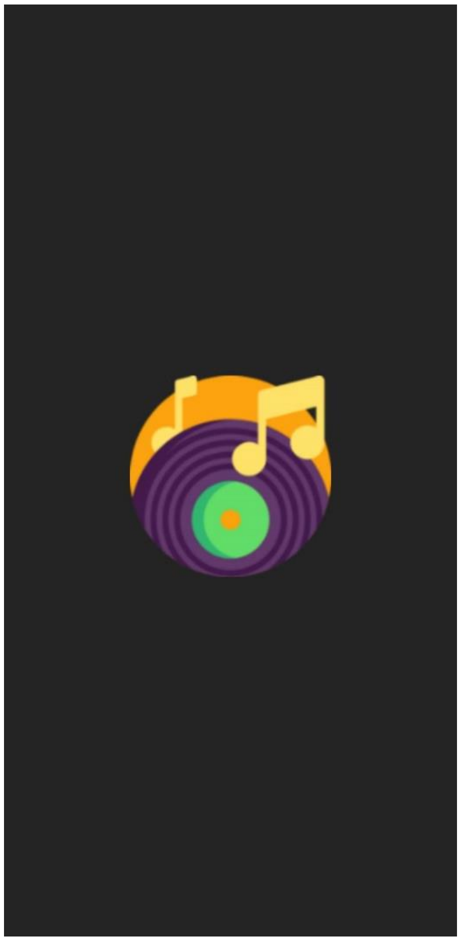


Figure 2.1: Splash Screen

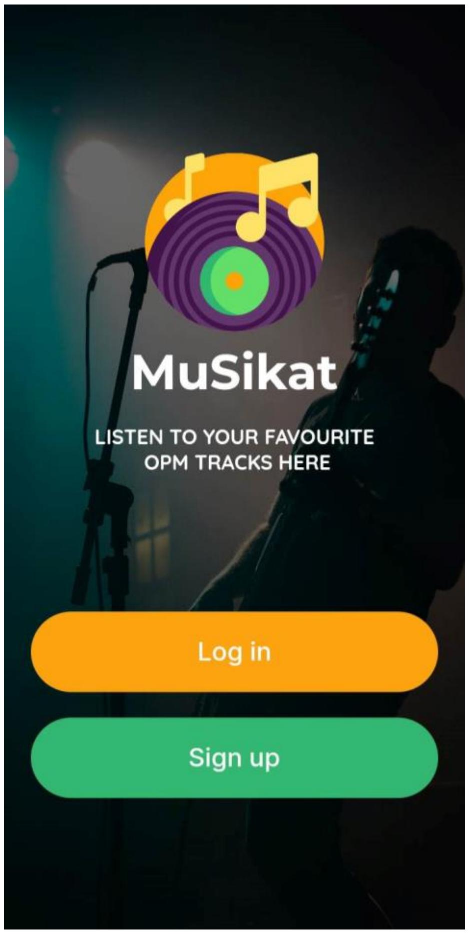
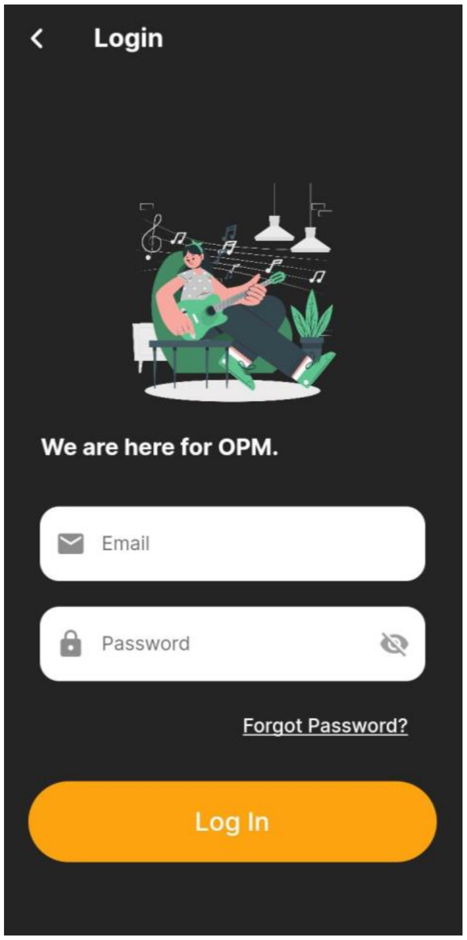


Figure 2.2: Welcome Screen

Figure 2.1, displayed above, depicts a Splash Screen which is a graphical element that briefly appears when a mobile app is launched before the primary user interface is displayed. This screen usually exhibits the logo or name of the app and functions as an introductory screen that provides feedback to the user that the app is launching.

Figure 2.2, shows the Welcome Screen which exhibits the logo of the app and two buttons for Log-in and sign-up. This screen is designed to welcome the user and give them options to either sign in to an existing account or create a new one.



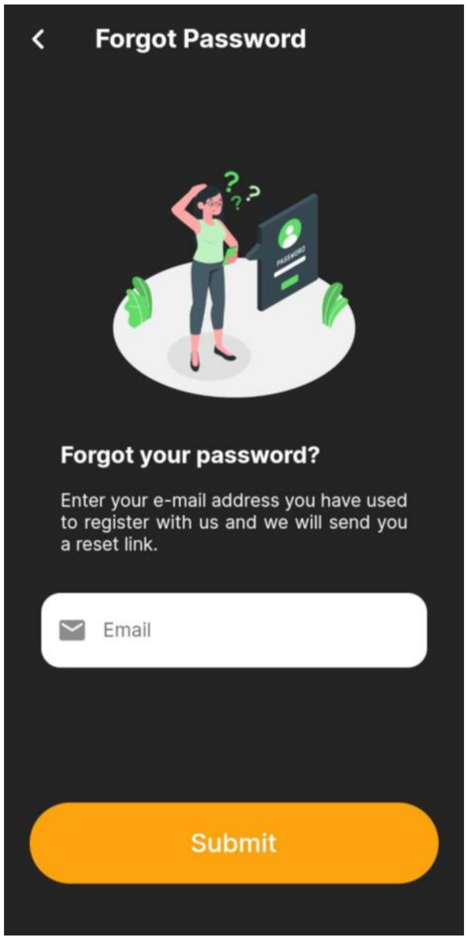
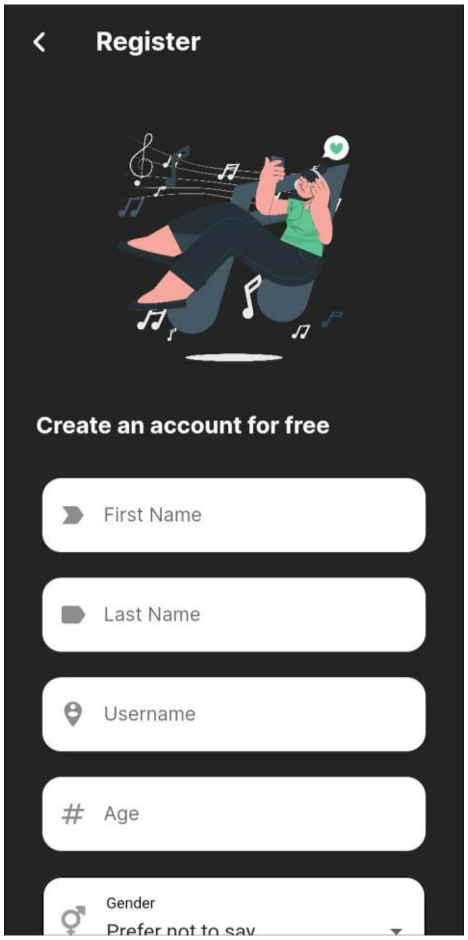

Forgot Password
Forgot your password?
Enter your e-mail address you have used to register with us and we will send you a reset link.
Email
Submit

Figure 2.3: Login Screen

Figure 2.4: Forgot Password Screen

In Figure 2.3, the Login Screen is displayed, which prompts the user to enter their email and password to gain access to the app's primary features. On the other hand, Figure 2.4 shows the Forgot Password Screen, which requests the user's email address. After providing the necessary information and clicking the submit button, the system will send a password reset link to the user's email.



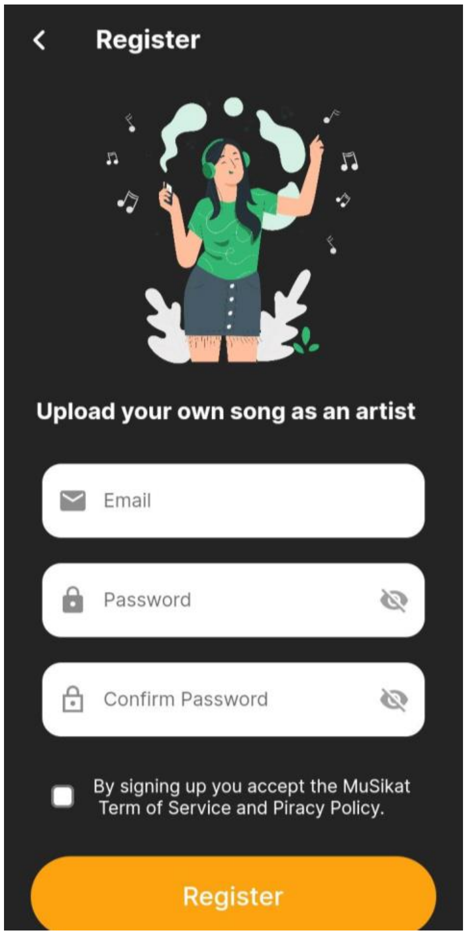

Register
Upload your own song as an artist
Email
Password
Confirm Password
By signing up you accept the MuSikat Term of Service and Piracy Policy.
Register

Figure 2.5 & 2.6: Register Screen

Figure 2.5 and 2.6 illustrate the Registration Screen, which is divided into two sections. In the first part, the user is asked to input their first name, last name, username, age, and gender. After clicking the "Next" button, the second page appears, where the user is required to enter their email, password, and confirm the password. The user must also check the box indicating that they accept the terms and conditions before clicking the "Register" button.

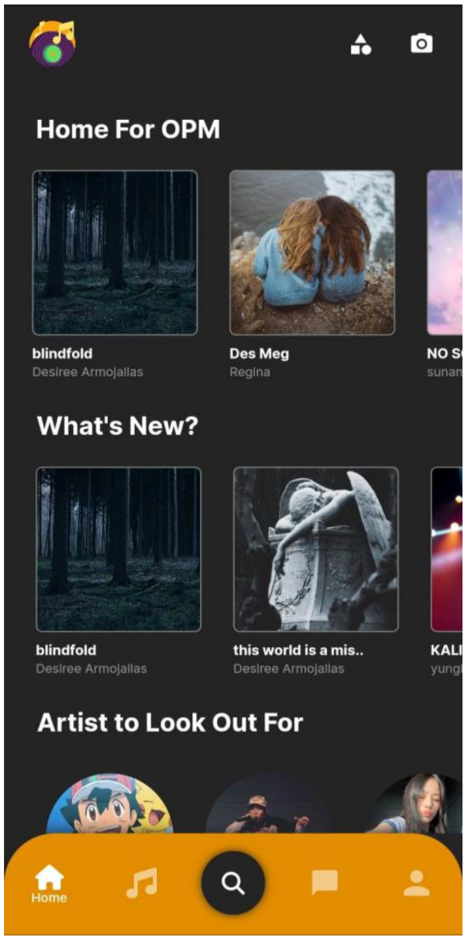


Figure 2.7: Home Screen

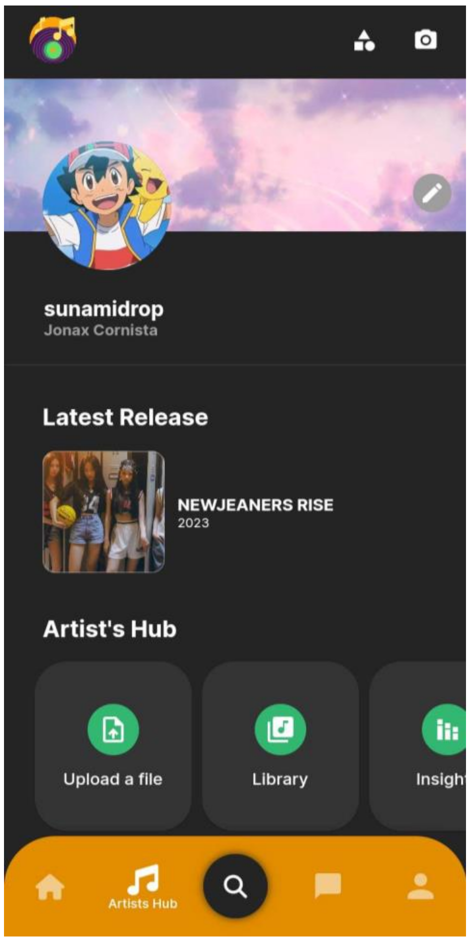


Figure 2.8: Artist's Hub

Figure 2.7 illustrates the Home Screen of MuSikat, which presents a range of music-related content. Specifically, the top row features songs uploaded by users, the middle row showcases new song releases, and the bottom row highlights artists worth exploring. This layout is designed to provide users with a diverse selection of music-related content to discover and enjoy. Figure 2.8 illustrates the Artists Hub screen, which features several components. These include the current user's profile and header, a display of the user's most recent release, and the ability for the user to upload audio content. Additionally, the screen provides access to the user's library, insights, and support system. The support system allows the user to view those who have donated to support their work and help them succeed.

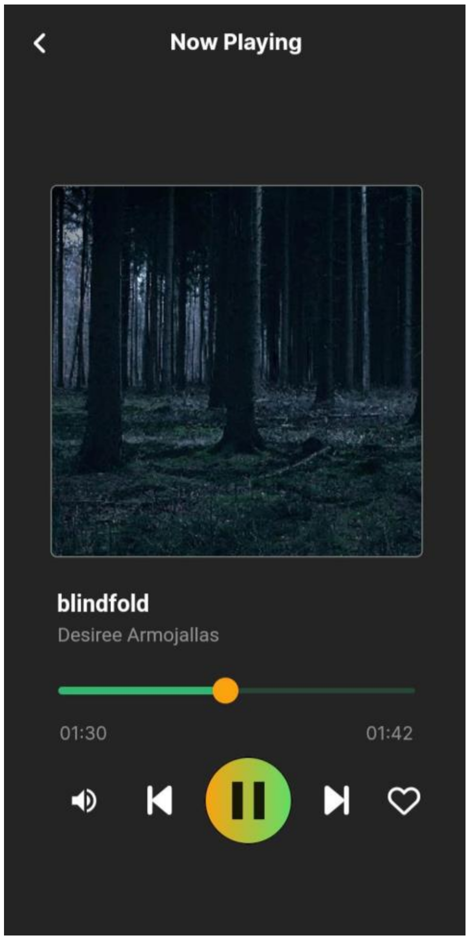


Figure 2.9: Music Player

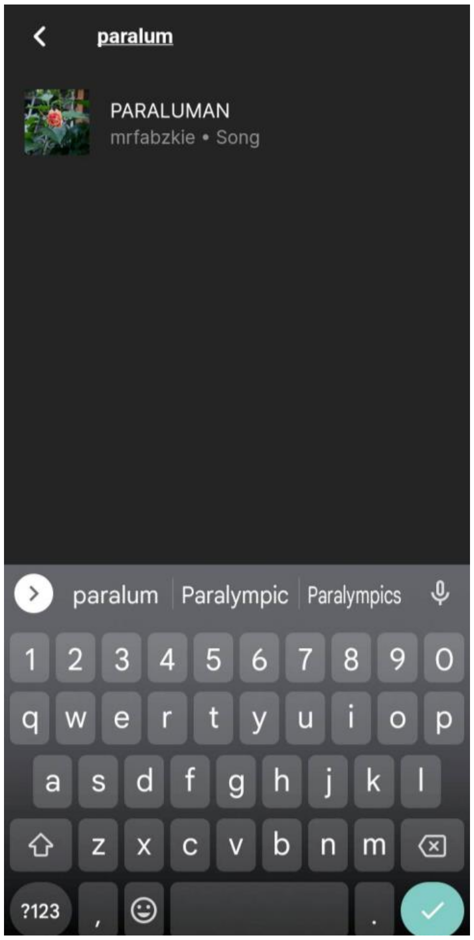


Figure 2.10 Search Screen

Figure 2.9 shows the Music Player screen provides users with a comprehensive music playback experience. At the top of the screen, users can see the song title and artist name, along with playback controls like play/pause and skip. Figure 2.10 The Search screen enables users to discover new music at the top of the screen, users can enter search terms to find specific songs and users.

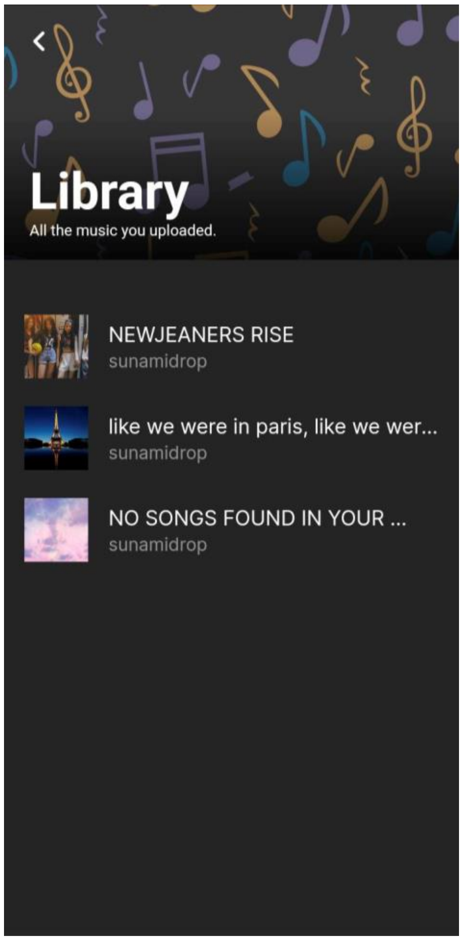

Library
All the music you uploaded.
NEWJEANERS RISE
sunamidrop
like we were in paris, like we wer...
sunamidrop
NO SONGS FOUND IN YOUR ...
sunamidrop

Figure 2.11: Library Screen

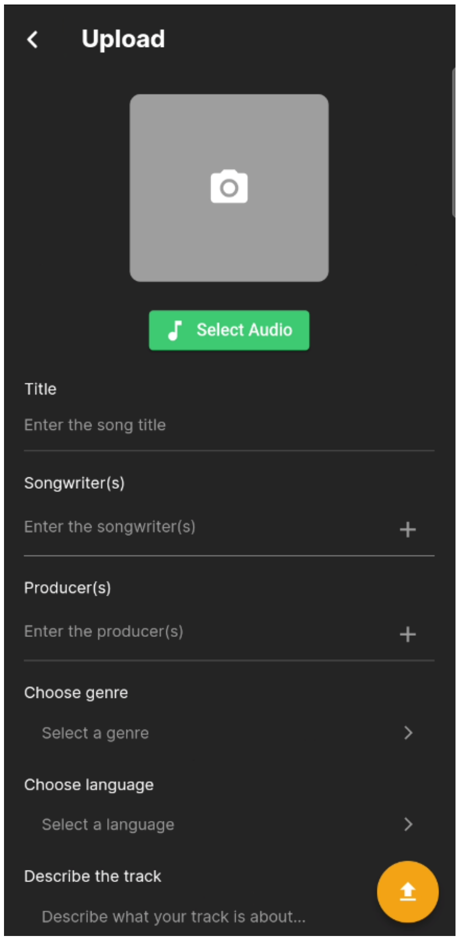
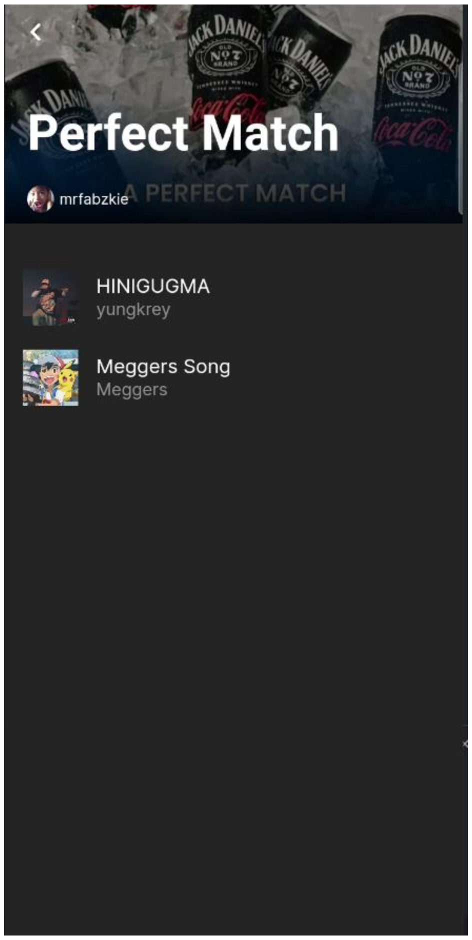


Figure 2.12 Upload Audio

Figure 2.11 shows The Library Screen that displays all the music content owned by the user. Figure 2.12 depicts the Upload Audio screen enables users to add their own music content to the platform. Users can upload a single song. The screen prompts users to enter information such as the artist's name, song title, album name, and other details such as producers, writers, and genre. Users can also upload album artwork and users can upload the audio file in various formats such as MP3, WAV, or FLAC. The screen provides feedback on the upload progress and notifies the user when the upload is complete.



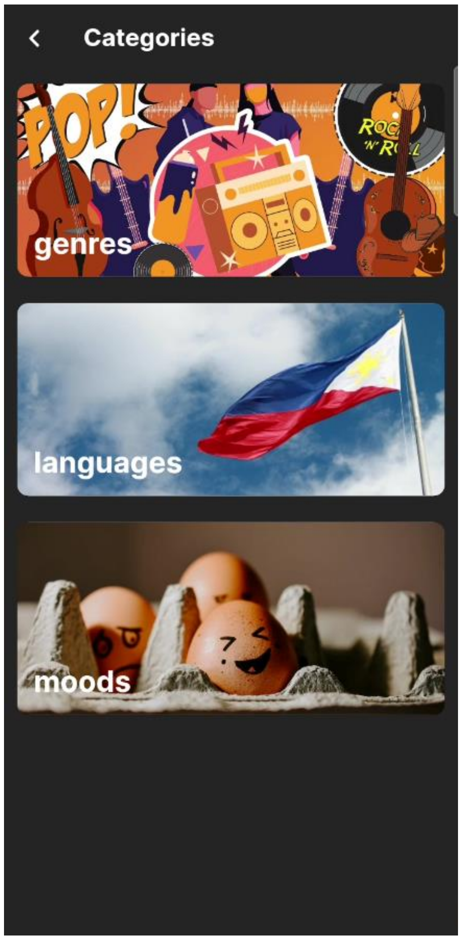
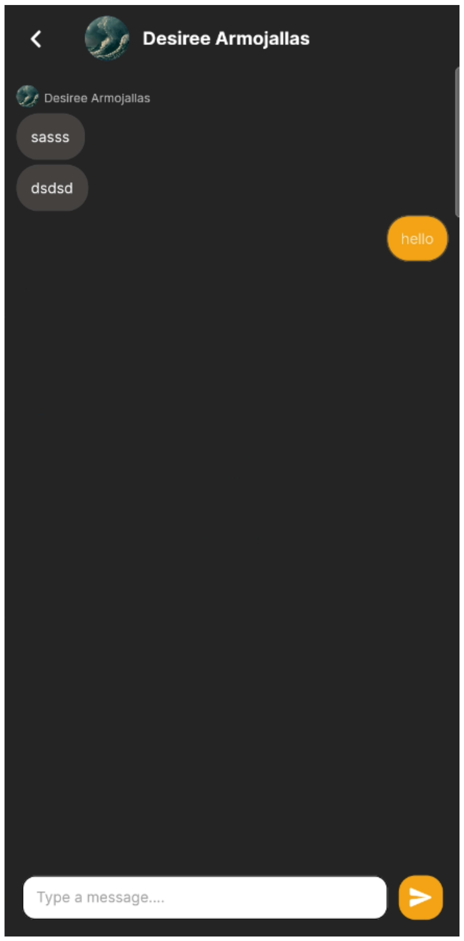


Figure 2.13: Playlist Screens

Figure 2.14: Categories Screen

Figure 2.13 illustrates the Playlist Screens displays the user's created playlists. Users can create playlists by adding songs from their library or by searching for specific songs or artists. Each playlist displays the playlist name, the number of songs in the playlist, and the playlist artwork, if available. Users can play the entire playlist or select individual songs from the playlist to play. Figure 2.14 shows the Categories Screen provides users with a curated selection of music content based on different categories such as genres, languages, moods.



sasss

dsdsd

hello

Figure 2.15: Chat Screen

The Chat Screen is a feature in a music app that allows users to chat with each other. This feature can be used to connect with friends or other music enthusiasts and share music-related content such as playlists, songs, or concert tickets.

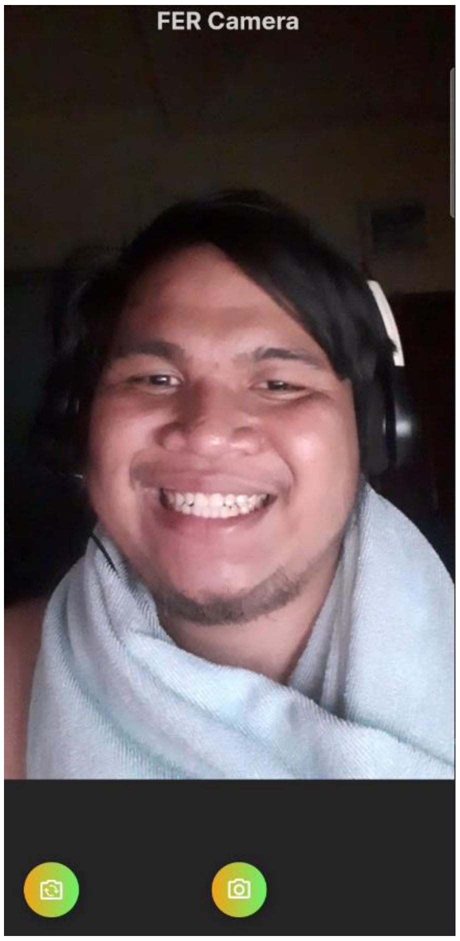


Figure 2.16: FER

Figure 2.16 depicts the FER stands for Facial Emotion Recognition, which is a technology that analyzes facial expressions to detect emotions. The FER screen in the app uses a device's camera to capture a user's facial expressions and will suggest a song based on the user’s emotion.

CHAPTER III

SOFTWARE DEVELOPMENT AND TESTING

This chapter summarizes how the researchers developed the platform, integrated different environments, used management tools for development, and tested the process. It breaks down the research's importance by identifying the application's root. It also explains the functionalities and process of development of how the researchers implemented such a development process.

DEVELOPMENT SOFTWARE PLATFORMS, DEVELOPMENT ENVIRONMENTS, AND PROJECT MANAGEMENT TOOLS

The researchers employed Visual Studio Code (VS Code) as the primary tool for coding and debugging the mobile app. VS Code is a lightweight, customizable, and user-friendly software that is widely used by mobile app developers because it supports various programming languages and frameworks.

For coding the mobile app, the researchers used the Dart programming language in combination with the Flutter framework. Dart is an object-oriented programming language created by Google that is designed to be easy to learn and use. Meanwhile, Flutter is an open-source UI software development kit (SDK) that simplifies the development process by providing pre-built tools and widgets.

To manage the development process, the researchers used GitHub, a web-based platform that allows developers to collaborate on code and manage project workflows. By doing so, the team was able to keep track of the changes made to the app's code, ensuring its stability and quality.

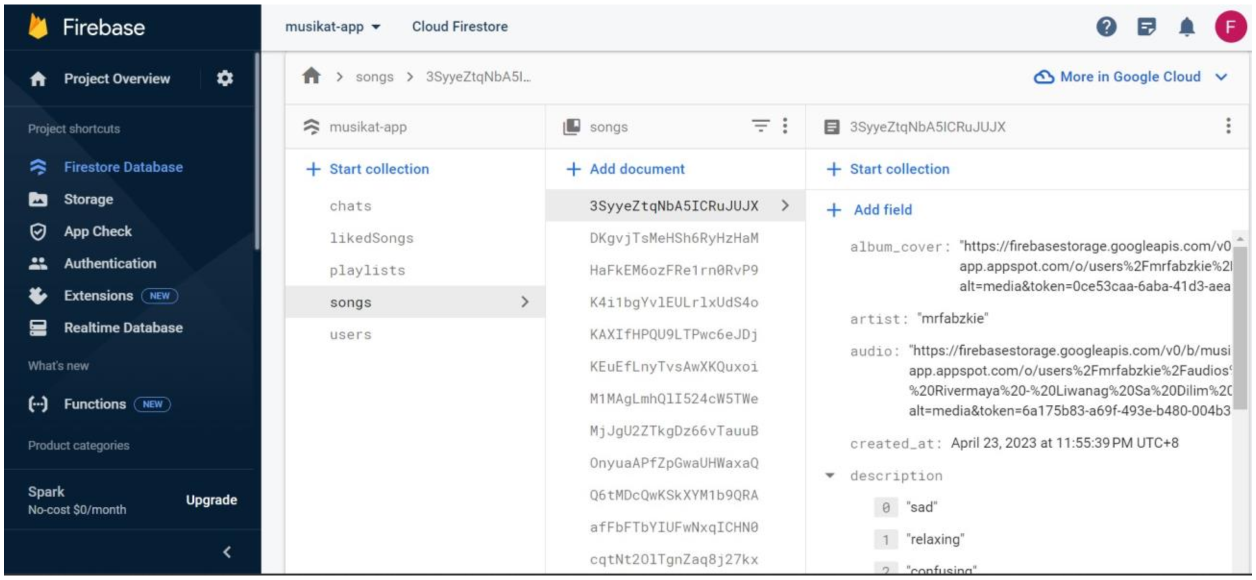
Firebase was used as the database for the mobile app development due to its ease of use, scalability, and real-time database features. The researchers leveraged Firebase's real-time database, authentication, hosting, and cloud messaging features to build a reliable and scalable mobile app with a robust backend infrastructure.

Overall, the combination of these efficient tools and technologies enabled the researchers to streamline the mobile app development process, work collaboratively, and create a robust and scalable mobile app.

DEVELOPMENT PROCESS

In this section, both the project development and testing processes are discussed and explained to better understand what is happening in the application.

Database

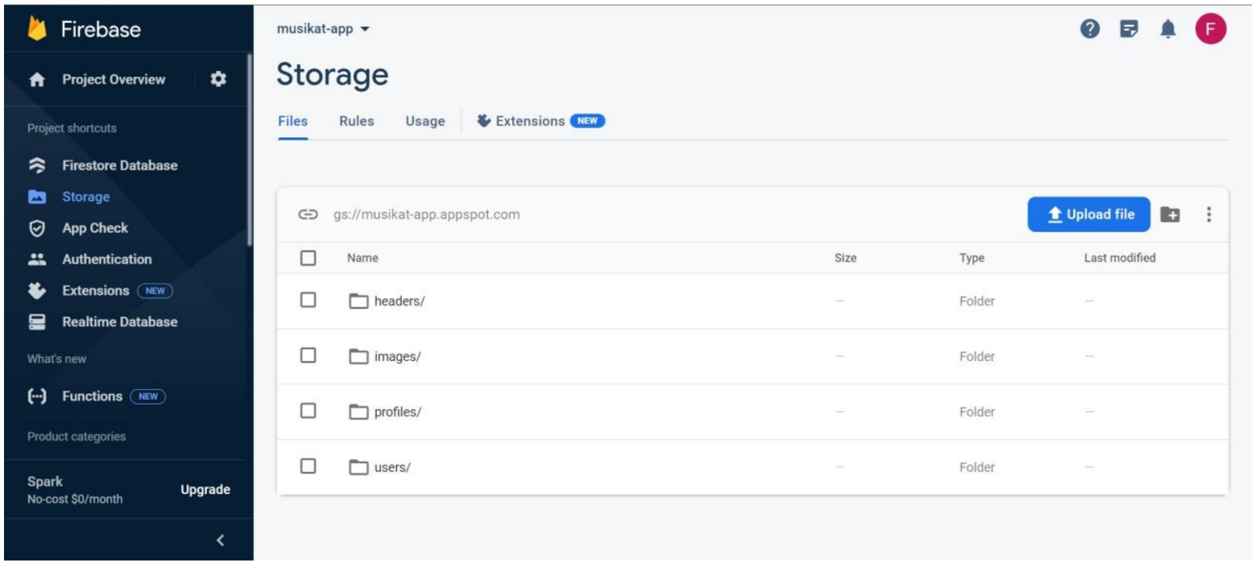


```
musikat-app          songs                   3SyyeZtqNbA5ICRuJUJX

chats                3SyyeZtqNbA5ICRuJUJX    album_cover: "https://firebasestorage.googleapis.com/v0
likedSongs           DKgvjTsMeHSh6RyHzHaM                 app.appspot.com/o/users%2Fmrfabzkie%2I
playlists            HaFkEM6ozFRe1rn0RvP9                 alt=media&token=0ce53caa-6aba-41d3-aea
songs                K4i1bgYvlEULrlxUdS4o    artist: "mrfabzkie"
users                KAXIfHPQU9LTPwc6eJDj    audio: "https://firebasestorage.googleapis.com/v0/b/musi
                     KEuEfLnyTvsAwXKQuxoi           app.appspot.com/o/users%2Fmrfabzkie%2Faudios
                     M1MAgLmhQ1I524cW5TWe           %20Rivermaya%20-%20Liwanag%20Sa%20Dilim%20
                     MjJgU2ZTkgDz66vTauuB           alt=media&token=6a175b83-a69f-493e-b480-004b3
                     OnyuaAPfZpGwaUHWaxaQ    created_at: April 23, 2023 at 11:55:39 PM UTC+8
                     Q6tMDcQwKSkXYM1b9QRA    description
                     afFbFTbYIUFwNxqICHN0      0 "sad"
                     cqtNt201TgnZaq8j27kx      1 "relaxing"
                                               2 "confusing"
```

Figure 3: Firestore Database

Firestore is a NoSQL database hosted in the cloud that provides developers with the ability to save and sync data with live updates, support for offline use, and several additional features like automatic scaling and data retrieval. By utilizing Firestore as the main database, it has resulted in a strong and dependable platform that can cater to users' requirements in diverse scenarios.

Image and Audio Management System



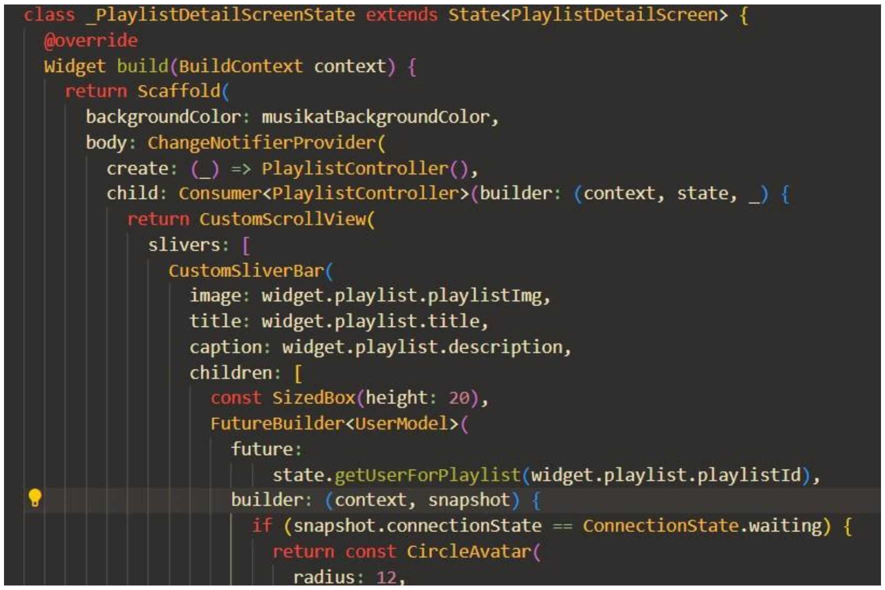
gs://musikat-app.appspot.com

Name	Size	Type	Last modified
headers/	–	Folder	–
images/	–	Folder	–
profiles/	–	Folder	–
users/	–	Folder	–

Figure 3.1: Firebase Storage

For the storage, the researchers used Firebase Storage to handle images and audios in MuSikat. Firebase Storage is a cloud storage service provided by Google as part of the Firebase suite of products. It allows developers to store and retrieve user-generated content, such as images, videos, and audio files, in a secure and scalable manner. By leveraging Firebase Storage, the researchers were able to store user-generated content in a secure and scalable manner, without the need to set up and maintain their own storage infrastructure.

State Management

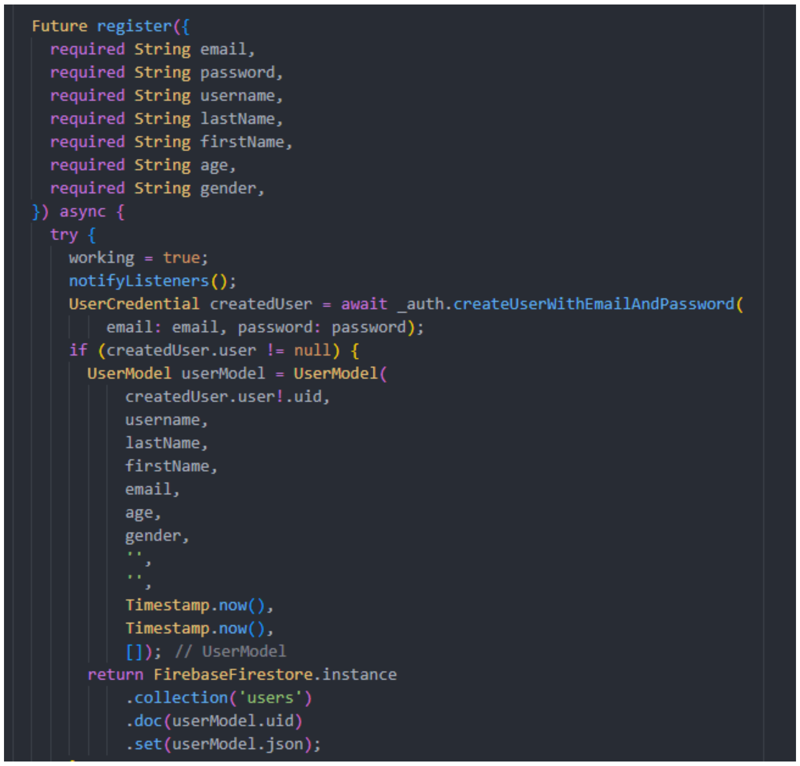


```
class _PlaylistDetailScreenState extends State<PlaylistDetailScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: musikatBackgroundColor,
      body: ChangeNotifierProvider(
        create: (_) => PlaylistController(),
        child: Consumer<PlaylistController>(builder: (context, state, _) {
          return CustomScrollView(
            slivers: [
              CustomSliverBar(
                image: widget.playlist.playlistImg,
                title: widget.playlist.title,
                caption: widget.playlist.description,
                children: [
                  const SizedBox(height: 20),
                  FutureBuilder<UserModel>(
                    future:
                        state.getUserForPlaylist(widget.playlist.playlistId),
                    builder: (context, snapshot) {
                      if (snapshot.connectionState == ConnectionState.waiting) {
                        return const CircleAvatar(
                          radius: 12,
```

Figure 3.2: Code snippet of Provider

State management is an important consideration when developing Flutter applications as it involves handling data retrieval and transfer between components. Managing the state of an application refers to managing the data that is used to update and create the user interface.

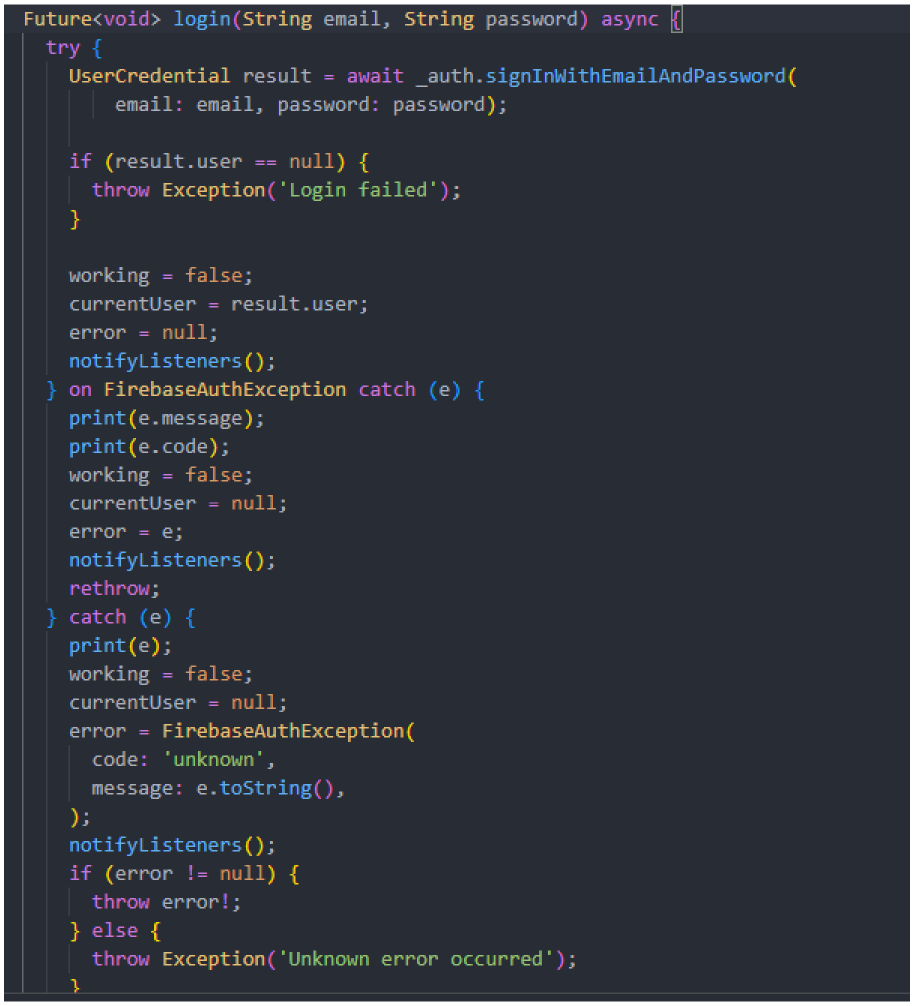
The researchers used the setState method and Provider package for state management. The setState() method is the simplest form of state management that updates the state of a widget, while Provider is a lightweight and efficient package that centralizes state management and separates business logic from the user interface. The use of state management enabled the researchers to maintain accurate and up-to-date data, resulting in a smoother user experience.



```
Future register({
  required String email,
  required String password,
  required String username,
  required String lastName,
  required String firstName,
  required String age,
  required String gender,
}) async {
  try {
    working = true;
    notifyListeners();
    UserCredential createdUser = await _auth.createUserWithEmailAndPassword(
        email: email, password: password);
    if (createdUser.user != null) {
      UserModel userModel = UserModel(
          createdUser.user!.uid,
          username,
          lastName,
          firstName,
          email,
          age,
          gender,
          '',
          '',
          Timestamp.now(),
          Timestamp.now(),
          []); // UserModel
      return FirebaseFirestore.instance
          .collection('users')
          .doc(userModel.uid)
          .set(userModel.json);
```

Figure 3.3: Code snippet of Register

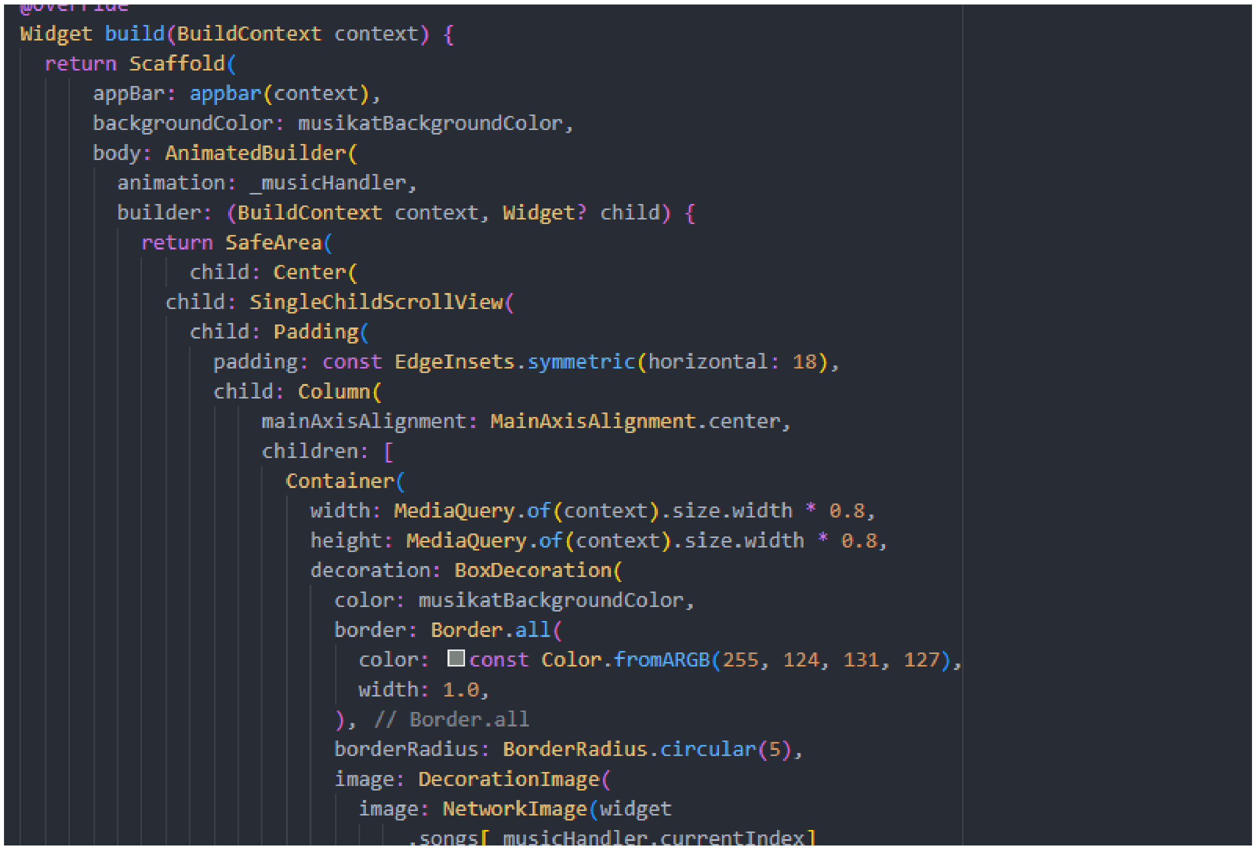
Figure 3.3 shows how to demonstrate a basic user registration flow using Firebase Authentication and Firestore in a Flutter app. It creates a new user, stores user details in Firestore, and likely performs additional actions after the registration process.



```
Future<void> login(String email, String password) async {
  try {
    UserCredential result = await _auth.signInWithEmailAndPassword(
        email: email, password: password);

    if (result.user == null) {
      throw Exception('Login failed');
    }

    working = false;
    currentUser = result.user;
    error = null;
    notifyListeners();
  } on FirebaseAuthException catch (e) {
    print(e.message);
    print(e.code);
    working = false;
    currentUser = null;
    error = e;
    notifyListeners();
    rethrow;
  } catch (e) {
    print(e);
    working = false;
    currentUser = null;
    error = FirebaseAuthException(
      code: 'unknown',
      message: e.toString(),
    );
    notifyListeners();
    if (error != null) {
      throw error!;
    } else {
      throw Exception('Unknown error occurred');
    }
```

Figure 3.4: Code snippet of Login

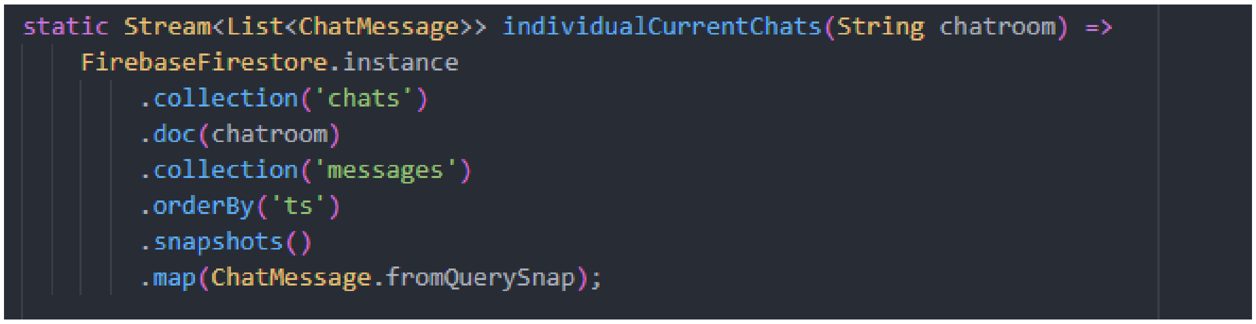
Figure 3.4 shows the provided code snippet is a login function using Firebase Authentication. It attempts to authenticate a user with their email and password. The code handles various scenarios, including successful login, failed login, and exceptions thrown during the process. It updates the state of the app accordingly and provides error information for further handling.



```
@override
Widget build(BuildContext context) {
  return Scaffold(
      appBar: appbar(context),
      backgroundColor: musikatBackgroundColor,
      body: AnimatedBuilder(
        animation: _musicHandler,
        builder: (BuildContext context, Widget? child) {
          return SafeArea(
              child: Center(
            child: SingleChildScrollView(
              child: Padding(
                padding: const EdgeInsets.symmetric(horizontal: 18),
                child: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                      Container(
                        width: MediaQuery.of(context).size.width * 0.8,
                        height: MediaQuery.of(context).size.width * 0.8,
                        decoration: BoxDecoration(
                          color: musikatBackgroundColor,
                          border: Border.all(
                            color: ▨const Color.fromARGB(255, 124, 131, 127),
                            width: 1.0,
                          ), // Border.all
                          borderRadius: BorderRadius.circular(5),
                          image: DecorationImage(
                            image: NetworkImage(widget
                                .songs[_musicHandler.currentIndex]
```

Figure 3.5 : Code snippet of Music Player

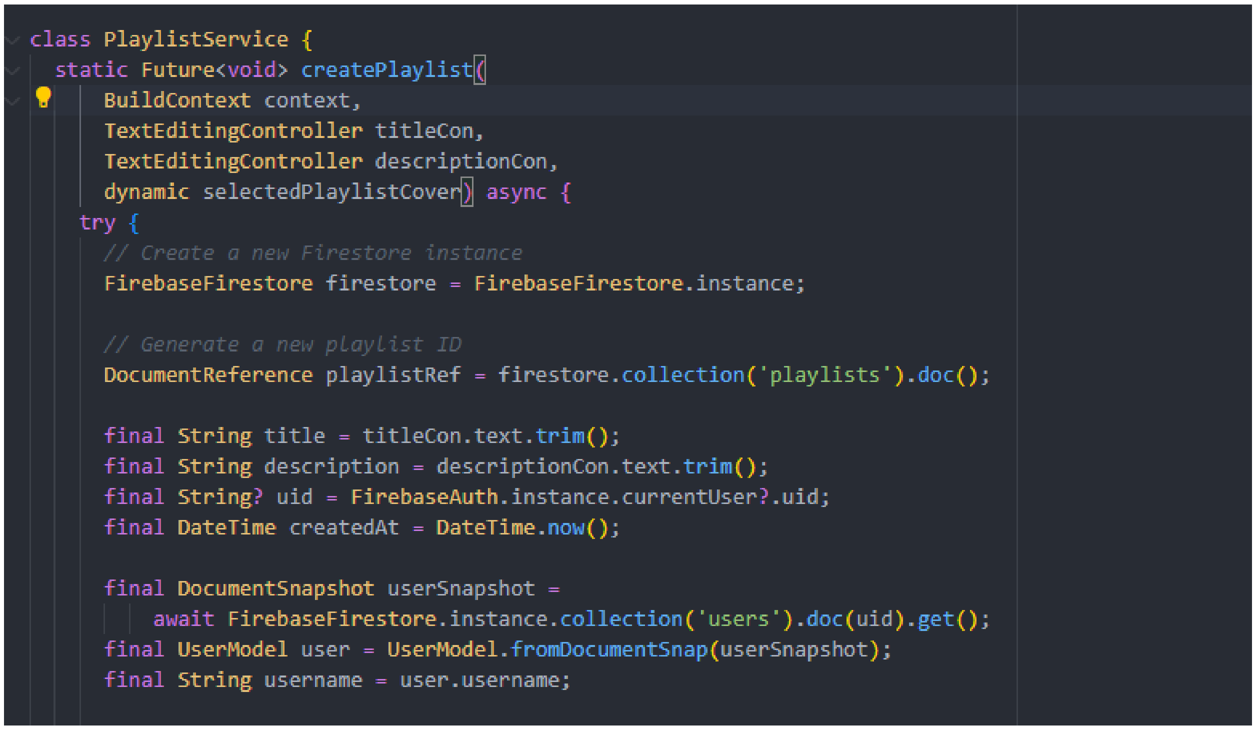
Figure 3.5 shows a basic implementation of a music player using a class called AudioPlayer. The player maintains a playlist, The play method starts playing the first song in the playlist. The next_song and previous_song methods allow you to play the next and previous songs, respectively. The stop method stops the currently playing song.



```
static Stream<List<ChatMessage>> individualCurrentChats(String chatroom) =>
    FirebaseFirestore.instance
        .collection('chats')
        .doc(chatroom)
        .collection('messages')
        .orderBy('ts')
        .snapshots()
        .map(ChatMessage.fromQuerySnap);
```

Figure 3.6: Code snippet of Chat

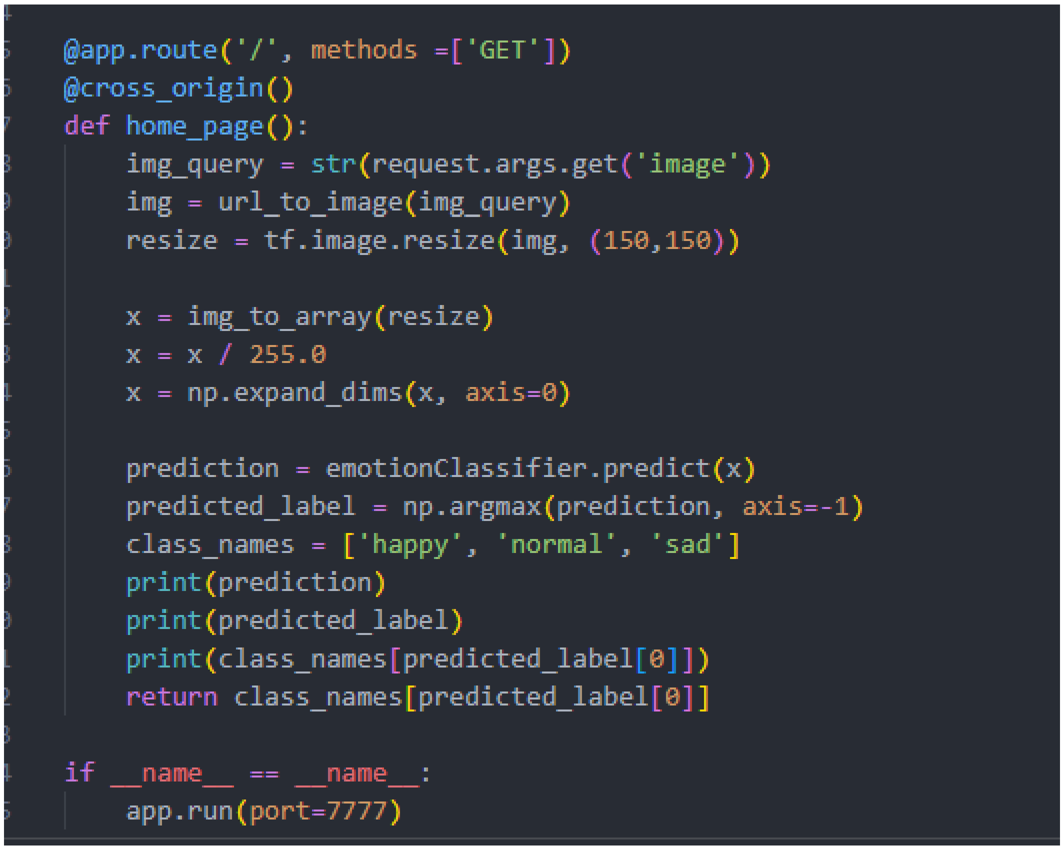
Figure 3.6 shows the code that retrieves current chat messages for a specific chat room from a Firestore database in a Flutter application using Dart. It constructs a query, orders the messages by timestamp, and returns a stream of snapshots. The snapshots are then mapped to ChatMessage objects.



```
class PlaylistService {
  static Future<void> createPlaylist(
      BuildContext context,
      TextEditingController titleCon,
      TextEditingController descriptionCon,
      dynamic selectedPlaylistCover) async {
    try {
      // Create a new Firestore instance
      FirebaseFirestore firestore = FirebaseFirestore.instance;

      // Generate a new playlist ID
      DocumentReference playlistRef = firestore.collection('playlists').doc();

      final String title = titleCon.text.trim();
      final String description = descriptionCon.text.trim();
      final String? uid = FirebaseAuth.instance.currentUser?.uid;
      final DateTime createdAt = DateTime.now();

      final DocumentSnapshot userSnapshot =
          await FirebaseFirestore.instance.collection('users').doc(uid).get();
      final UserModel user = UserModel.fromDocumentSnap(userSnapshot);
      final String username = user.username;
```

Figure 3.7 : Code snippet of Playlist

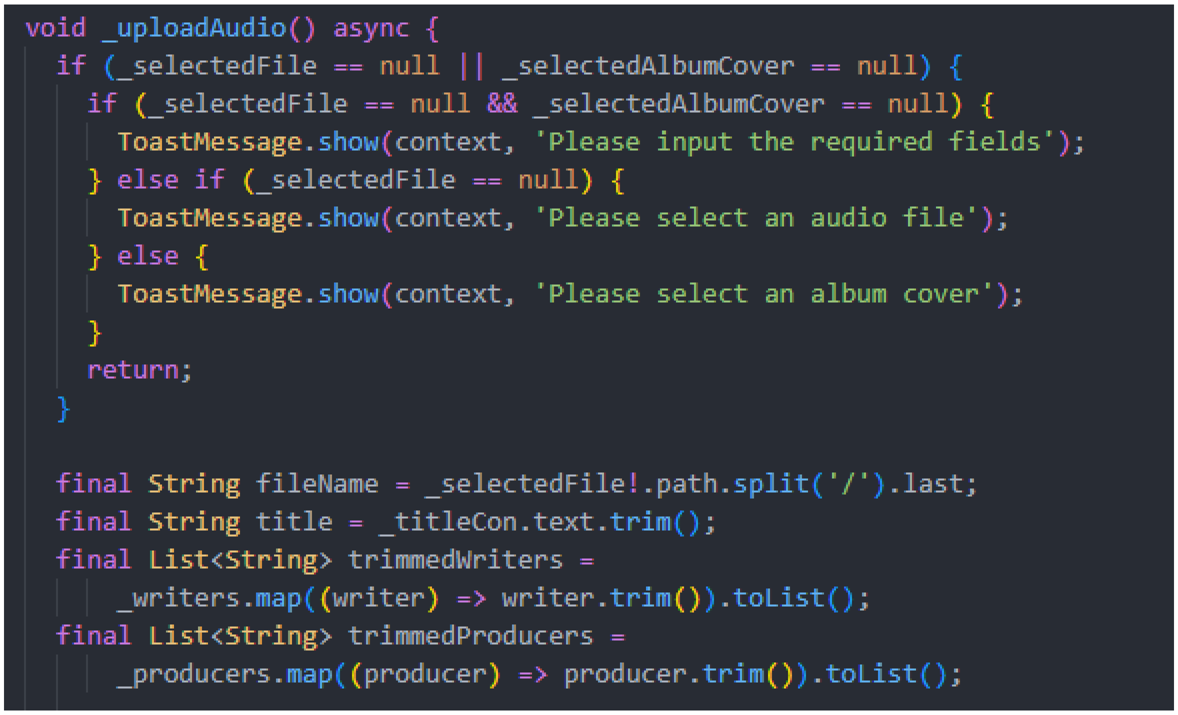
Figure 3.7 shows the createPlaylist function that creates a new playlist in a Firestore database. It retrieves the playlist details such as title, description, and cover image. It generates a unique playlist ID and uploads the cover image to Firebase Storage. It validates the input and displays error messages if necessary.



```python
@app.route('/', methods =['GET'])
@cross_origin()
def home_page():
    img_query = str(request.args.get('image'))
    img = url_to_image(img_query)
    resize = tf.image.resize(img, (150,150))

    x = img_to_array(resize)
    x = x / 255.0
    x = np.expand_dims(x, axis=0)

    prediction = emotionClassifier.predict(x)
    predicted_label = np.argmax(prediction, axis=-1)
    class_names = ['happy', 'normal', 'sad']
    print(prediction)
    print(predicted_label)
    print(class_names[predicted_label[0]])
    return class_names[predicted_label[0]]

if __name__ == __name__:
    app.run(port=7777)
```

Figure 3.8: Code snippet of FER (PYTHON)

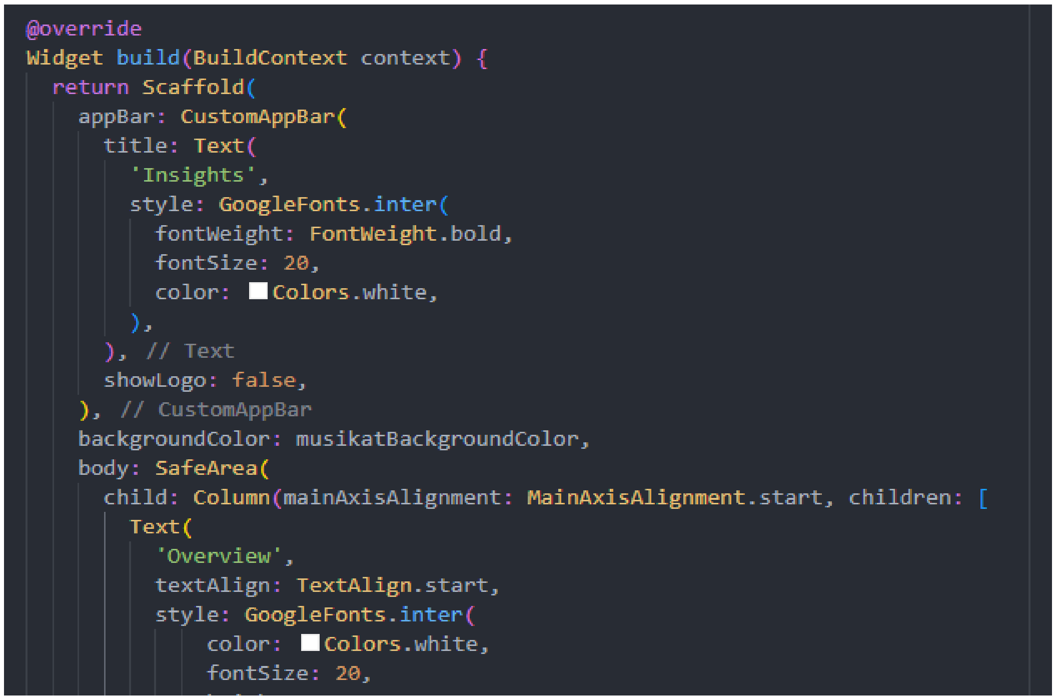
Figure 3.8 shows the code that accepts an image URL as a query parameter. It fetches the image, resizes it, and performs emotion prediction using an emotion classifier. It determines the predicted emotion label and returns it as the response to the API request. The code assumes the presence of helper functions, necessary imports, and a predefined list of emotion class names.



```
void _uploadAudio() async {
  if (_selectedFile == null || _selectedAlbumCover == null) {
    if (_selectedFile == null && _selectedAlbumCover == null) {
      ToastMessage.show(context, 'Please input the required fields');
    } else if (_selectedFile == null) {
      ToastMessage.show(context, 'Please select an audio file');
    } else {
      ToastMessage.show(context, 'Please select an album cover');
    }
    return;
  }

  final String fileName = _selectedFile!.path.split('/').last;
  final String title = _titleCon.text.trim();
  final List<String> trimmedWriters =
      _writers.map((writer) => writer.trim()).toList();
  final List<String> trimmedProducers =
      _producers.map((producer) => producer.trim()).toList();
```

Figure 3.9: Code snippet of Upload Audio

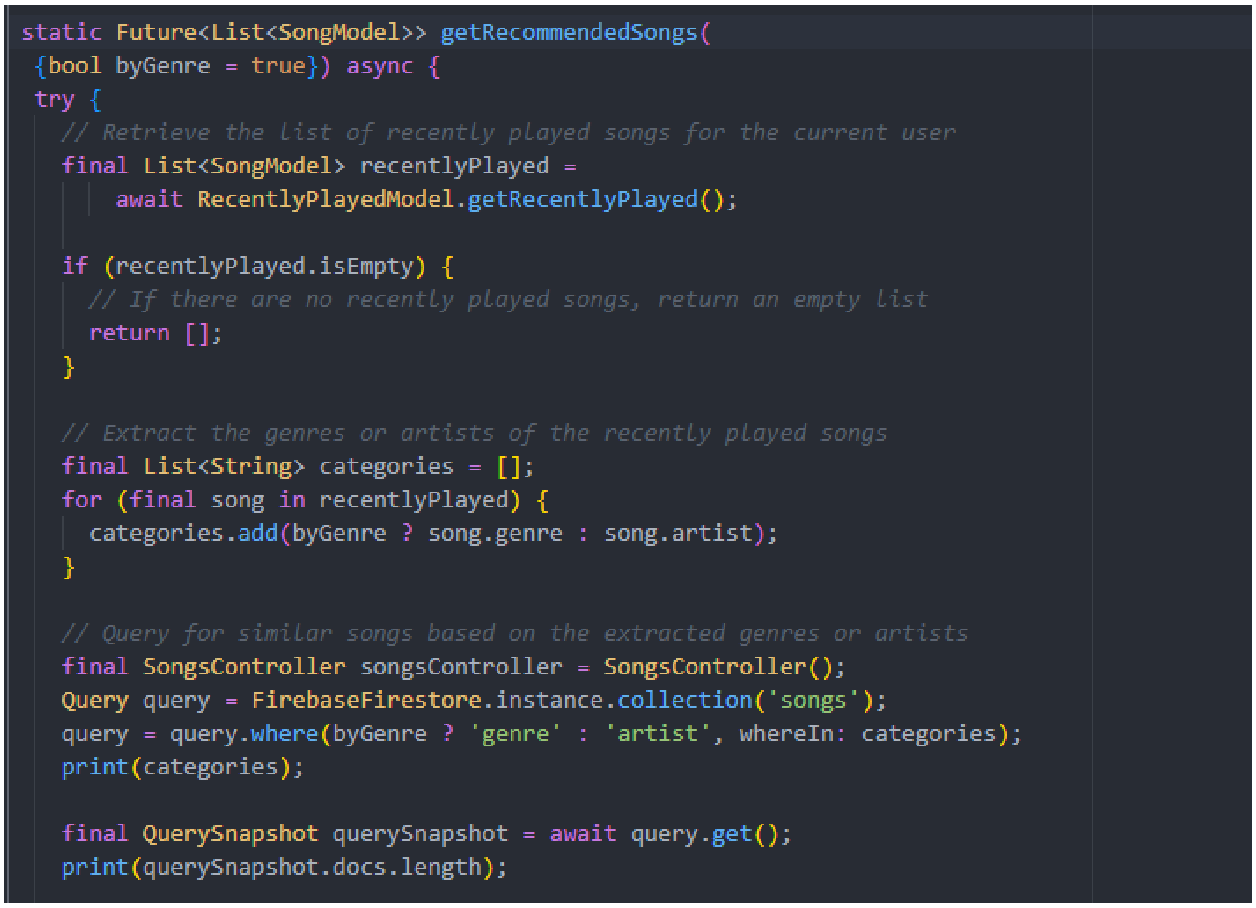
Figure 3.9 shows the function that ensures the necessary fields, such as the audio file and album cover, are not null before proceeding with the upload process. It validates the presence and content of fields like the title, writers, producers, language, genre, and description, displaying error messages if any are missing or empty. This helps ensure that all required information is provided before uploading the audio file.



```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: CustomAppBar(
      title: Text(
        'Insights',
        style: GoogleFonts.inter(
          fontWeight: FontWeight.bold,
          fontSize: 20,
          color: ■Colors.white,
        ),
      ), // Text
      showLogo: false,
    ), // CustomAppBar
    backgroundColor: musikatBackgroundColor,
    body: SafeArea(
      child: Column(mainAxisAlignment: MainAxisAlignment.start, children: [
        Text(
          'Overview',
          textAlign: TextAlign.start,
          style: GoogleFonts.inter(
            color: ■Colors.white,
            fontSize: 20,
```

Figure 3.10: Code snippet of Insights

Figure 3.10 shows the statistics for the users that provide insights into their listening habits and preferences. Some common statistics include play count, which tracks the number of times a song has been played, and like count, which indicates the number of times a user has expressed their preference for a song. Top tracks feature displays the most frequently played or liked songs, offering a snapshot of a user's musical preferences. Music platforms also often provide charts showcasing most likes and most songs played.



```
static Future<List<SongModel>> getRecommendedSongs(
 {bool byGenre = true}) async {
 try {
   // Retrieve the list of recently played songs for the current user
   final List<SongModel> recentlyPlayed =
       await RecentlyPlayedModel.getRecentlyPlayed();

   if (recentlyPlayed.isEmpty) {
     // If there are no recently played songs, return an empty list
     return [];
   }

   // Extract the genres or artists of the recently played songs
   final List<String> categories = [];
   for (final song in recentlyPlayed) {
     categories.add(byGenre ? song.genre : song.artist);
   }

   // Query for similar songs based on the extracted genres or artists
   final SongsController songsController = SongsController();
   Query query = FirebaseFirestore.instance.collection('songs');
   query = query.where(byGenre ? 'genre' : 'artist', whereIn: categories);
   print(categories);

   final QuerySnapshot querySnapshot = await query.get();
   print(querySnapshot.docs.length);
```

Figure 3.11: Code snippet of Recommender

Figure 3.11 shows a list of recommended songs based on the user's recently played songs. It retrieves the recently played songs, extracts their genres or artists, and queries the Firestore database to find similar songs. The recommended songs are then shuffled and the top five are returned. The function handles errors by printing them and returning an empty list.

TESTING PROCESS

The testing phase conducted by the developers indicates that all features and functionalities of the MuSikat application are operating correctly.

Test Cases

TEST CASE 1: USERS

Test Module	**Test Scenario**	**Expected Result**	**Actual Result**	**Status**
User Signup	User will input necessary information and click ***Sign Up*** button	User must access account and homescreen	User can access account and homescreen	**PASSED**
User Login	User will input credentials and click ***Login*** button	User must access account and homescreen	User can access account and homescreen	**PASSED**
Forgot Password	User will click the Forgot ***Password*** in the Login Page.	The system must send a password reset link to the user's registered email address. The user must be able to reset the password.	The system can send a password reset link to the user's registered email address. The user can reset the password.	**PASSED**

Create Playlist	User will navigate to Create Playlist	Users must be able to successfully upload the song.	Users can successfully upload the song.	**PASSED**
Facial Emotion Recognition	User will navigate to the camera	Users must be able to access the camera and face should be detected.	Users can access the camera and face can be detected.	**PASSED**
View Song Info	The user clicks the *Song Info*	Users must be able to view the details of the song info selected.	Users can view the details of the song info selected.	**PASSED**
View Category	User will navigate to the category	The user must be able to go through the different categories such as the Languages, Moods, and Genre.	The user can go through the different categories such as the Languages, Moods, and Genre.	**PASSED**
Play Song	User clicks on the Music Player	The user must be able to play the song of choice.	The user can play the song of choice.	**PASSED**

Chat Messaging	User clicks on the chat messaging at the navbar	The user must be able to send a message to the recipient and vice versa.	The user can send a message to the recipient and vice versa.	**PASSED**
Upload Audio	User tap on the Upload Audio.	The user must be able to fill in the necessary details of the uploaded song before publishing it. There must be no duplicate of the song being uploaded.	The user can fill in the necessary details of the uploaded song and can publish it right after. There is no duplicate of the song being uploaded.	**PASSED**
View Insights	User navigates to the *View Insights*	Users must be able to check the	User can access his ticket history and view a list of his purchased and used tickets.	**PASSED**
Recommender	User navigates to Home Screen	The system must display a list of recommended songs based on the user's listening activity and liked songs.	The system displays a list of recommended songs based on listening activity and liked songs based on genre and artist/	**PASSED**
Support artist	User will tap the Support Tile Card	Users must see people who donated.	Users can see people who donated.	**PASSED**

Follow Artist	User navigates to the Artist’s Profile	The user must be able to follow his favorite artist.	The user can follow his favorite artist.	**PASSED**
View Song Info	User taps on the Info of the Song	The user must be able to see the details of the song	The user can see the details of the song	**PASSED**
Add Song to Playlist	User adds a song to the playlist	The user must be able to add the song to the playlist	The user can add the song to the playlist	**PASSED**
Edit Song	User taps on Edit for Song	The user must be able to edit the song being uploaded.	The user can edit the song being uploaded.	**PASSED**
Delete Song	User taps on Delete for Song	The user must be able to delete the song being uploaded	The user can delete the song being uploaded	**PASSED**
View Recent Played	User navigates on the Recently Played Songs	The user must be able to view the songs that are recently played.	The user can view the songs that are recently played.	**PASSED**
View Liked Songs	User navigates on the Liked Songs	The user must be able to view the number of liked songs	The user can view the number of liked songs	**PASSED**

Usability Testing

This test is intended to identify and evaluate the user experience while utilizing the app in its completed form. This test includes all functions of the app, including implemented system features. The table below shows the satisfaction rating of 20 users who have used the application. The ratings used in the survey were as follows:

5–Strongly Agree, 4–Agree, 3–Neutral, 2 –Disagree, 1 –Strongly Disagree

QUESTION	**5**	**4**	**3**	**2**	**1**	**TOTAL**	**AVG**
The application doesn’t crash.	10	5	3	2	0	20	**4.6**
It is easy to navigate between different pages.	11	7	2	0	0	20	**4.6**
The application’s features work.	13	5	1	1	0	20	**4.5**
The application was not interrupted when using other apps and vice-versa.	13	5	1	1	0	20	**4.5**
The system gave error messages that clearly told me how to fix problems.	12	5	2	1	0	20	**4.4**
I was able to complete tasks and scenarios using this system.	14	4	1	1	0	20	**4.55**

It was easy to use this system.	13	5	1	1	0	20	**4.5**
The interface of this system is pleasant.	14	4	1	1	0	20	**4.55**
Would you likely recommend the MuSikat application to a friend?	15	3	1	1	0	20	**4.6**

Table 29 - Usability Testing

Upon completion of the testing phase, the researchers were able to achieve their project objectives successfully. The features of the application were thoroughly evaluated, and the results showed that they have passed the testing phase. The satisfaction survey's outcome indicated that the testers are pleased with their overall app experience, and the relevant app features are working correctly, as evidenced by the average rating of above 4.0 for each statement. The researchers employed different test cases to achieve a positive outcome, and the system requirements were fully realized.

CHAPTER IV

SUMMARY, CONCLUSION, AND RECOMMENDATIONS

This chapter describes the issues and conclusions encountered by the project's creators during its development, which addresses hardships and how project plans and methods were adjusted.

SUMMARY OF FINDINGS

MuSikat is an audio streaming platform that aims to promote and support Original Pinoy Music (OPM). The app provides a personalized listening experience, ensuring that users can easily discover and enjoy new music based on their emotional state. The study's findings highlight the potential of facial emotion recognition technology to enhance user engagement and satisfaction, and its application to music streaming services. This innovative feature offers a unique value proposition and can set MuSikat apart from its competitors in the market. Overall, MuSikat has the potential to become a leading audio streaming platform for OPM enthusiasts by providing a personalized and engaging user experience.

CONCLUSION

MuSikat is an innovative audio streaming platform that aims to revolutionize the way OPM is promoted and supported. By offering a unique value proposition, MuSikat can differentiate itself from its competitors and appeal to OPM enthusiasts looking for a more engaging and personalized listening experience. The integration of Firestore database has ensured secure storage and real-time updating of user data, while state management has ensured seamless application operation. The development of MuSikat demonstrates how technology can improve personalized listening experience and benefit both artists and listeners.As technology continues to shape the audio streaming industry, MuSikat's innovative approach can serve as a model for future developments that prioritize user engagement and satisfaction. Overall, MuSikat has the potential to become a leading platform in the OPM market and beyond.

RECOMMENDATIONS

Future researchers may consider exploring the use of facial emotion recognition technology in other contexts related to audio and music. The potential of this technology to enhance user engagement and satisfaction could be applied to other areas, such as podcasting or audiobooks. Additionally, further research could investigate the effectiveness of personalized listening experiences in promoting user satisfaction and loyalty. MuSikat's innovative approach provides a compelling example of how technology can be used to support local music scenes and provide a platform for emerging artists. Overall, the study of MuSikat offers valuable insights into the potential of technology to improve user experiences and support local music scenes, providing researchers with a fruitful area for future exploration.

CHAPTER V

BIBLIOGRAPHY

REFERENCES:

[1] Palmiano, M. (2019, September 3). *OPM is not dead, it is rising and developing!* Marilag's Blog. Retrieved July 24, 2022, from https://moniquepalmianoblog.wordpress.com/2019/09/03/opm-is-not-dead-it-is-rising-and-developing/

[2] Santos, T. U. (2012, May 27). Now and then: Is OPM going extinct? The Varsitarian. Retrieved July 24, 2022, from http://varsitarian.net/circle/20120513/now_and_then_is_opm_going_extinct

[3] Rappler (2019, July 12). *The rise of OPM, as seen on Spotify*. RAPPLER. Retrieved July 24, 2022, from https://www.rappler.com/brandrap/announcements/235236-opm-rise-spotify/

[4] Spotify (2016, November 8). *What is Spotify and How Does It Work?* Retrieved July 25, 2022, from https://techboomers.com/t/what-is-spotify?fbclid=IwAR22dUzxpVSot67ohoEWXmGKT0TwKChJWKIxVYCrZajk2KPJnv18K2TU3m8

[5] Tillman, M. (2022, February 4). What is Apple Music and how does it work? Retrieved July 25, 2022, from

https://www.pocket-lint.com/apps/news/apple/136725-what-is-apple-music-and-how-does-it-work?fbclid=IwAR0XQ8oYTECkJR3zjj8ThuJ076ZvBY31BGWteuTmVzu5T912SIbKRRfw_Lc

[6] *About Soundcloud*. SoundCloud. (n.d.). Retrieved July 28, 2022, from https://soundcloud.com/pages/contact

[7] Fisher, S. (2022, July 4). *Rock out with the 9 best free music apps for mobile devices*. Lifewire. Retrieved July 25, 2022, from https://www.lifewire.com/top-free-music-apps-1357073

[8] Samuvel, D. J., Perumal, B., y Elangovan, M. (2020). Music recommendation system based on facial emotion recognition. 3C Tecnología. Glosas de innovación aplicadas a la pyme. Edición Especial, Marzo 2020, 261-271. http://doi.org/10.17993/3ctecno.2020.specialissue4.261-271

[8] Mohiyeddini, C. (2013). *Handbook of psychology of emotions, recent theoretical perspectives and novel empirical findings*. Nova Publ.

[9] Matilda Florence, S., & Uma, M. (2020). Emotional detection and music recommendation system based on user facial expression. *IOP Conference Series: Materials Science and Engineering*, *912*(6), 062007. https://doi.org/10.1088/1757-899x/912/6/062007

[10] Sana, S., Sruthi, G., Suresh, D., Rajesh, G., & Subba Reddy, G. V. (2022). Facial emotion recognition based music system using Convolutional Neural Networks. *Materials Today: Proceedings*, *62*, 4699–4706. https://doi.org/10.1016/j.matpr.2022.03.131

[11] Muhammad, S., Ahmed, S., & Naik, D. (2021). Real time emotion based music player using CNN Architectures. *2021 6th International Conference for Convergence in Technology (I2CT)*. https://doi.org/10.1109/i2ct51068.2021.9417949

[12] Gupte A, Rajanarayanan A and Krishnan M Emotion Based Music Player-XBeats International Journal of Advanced Engineering Research and Science 3 236854

[13] Kołakowska, A., Szwoch, W., & Szwoch, M. (2020). A review of emotion recognition methods based on data acquired via smartphone sensors. *Sensors*, *20*(21), 6367. https://doi.org/10.3390/s20216367

[14] Bonnin, G., & Jannach, D. (2015). Automated generation of music playlists: Survey and experiments. *ACM Computing Surveys*, *47*(2), 1–35. https://doi.org/10.1145/2652481

[15] Dias, R., Gonçalves, D., & Fonseca, M. J. (2016). From Manual to Assisted Playlist Creation: A survey. *Multimedia Tools and Applications*, *76*(12), 14375–14403. https://doi.org/10.1007/s11042-016-3836-x

[16] Gartner, D. (2006). User Adaptive Music Similarity with an Application to Playlist Generation.

[17] Cunningham SJ, Bainbridge D, Falconer A (2006) More of an art than a science: Supporting the creation of playlists and mixes. In: Proceedings of 7th international conference on music information retrieval, Victoria, Canada, pp 240–245

[18] Bonnin G, Jannach D (2014) Automated generation of music playlists: Survey and experiments. ACM Comput Surv 47(2):26

[19] Sneha Antony JJN (2014) Survey on playlist generation techniques. Int J Adv Res Comput Eng Technol (IJARCET) 3:437–439

[20] Weigl DM, Guastavino C (2011) User studies in the music information retrieval literature. In: Proceedings of the international conference on music information retrieval

[21] Robert, P. (n.d.). Playlists and the Datafication of Music Formatting. *University of Groningen*.

[22] Eriksson, M. (2020). The editorial playlist as Container Technology: On Spotify and the logistical role of digital music packages. *Journal of Cultural Economy*, *13*(4), 415–427. https://doi.org/10.1080/17530350.2019.1708780

CURRICULUM VITAE

PERSONAL INFORMATION

Name: Hazel S. Lopez

Nickname: Zel

Date of Birth: June 07, 2000

Address: 1060 Abadiano Compound, Basak San Nicolas, Cebu City, Philippines

Status: Single

Citizenship: Filipino

Religion: Roman Catholic

Email Address: lopezhazel99@gmail.com

Contact Number: 09154931471

ACADEMIC INFORMATION

Tertiary: University of San Jose-Recoletos

Secondary: Don Vicente Raman Memorial National High School / Asian College of Technology

Primary: Don Vicente Raman Memorial Elementary School

PROFESSIONAL SKILLS

Language: Dart/Flutter, C, Java, HTML, CSS

Software: Figma, VS Code, Microsoft Office, Android Studio

OS: Windows

CURRICULUM VITAE

Name: Franz Lesly R. Rocha

Nickname: Pran2x

Date of Birth: October 06, 2000

Address: P. Burgos Street, Alang-Alang, 6014 Mandaue City, Philippines

Status: Single

Citizenship: Filipino

Religion: Roman Catholic

Email Address: franzleslyrocha@gmail.com

Contact Number: 09194377844

ACADEMIC INFORMATION

Tertiary: University of San Jose-Recoletos

Secondary: Labogon National High School / ACLC College of Mandaue

Primary: Basak Elementary School

PROFESSIONAL SKILLS

Language: Dart, C, Java, HTML, CSS

Software: Figma, VS Code, Microsoft Office, Android Studio

OS: Windows

CURRICULUM VITAE

PERSONAL INFORMATION

Name: Fabian Miguel F. Canizares

Nickname: Jekoy

Date of Birth: October 12, 1999

Address: St. Noel Street Mohon Talisay City, Cebu

Status: Single

Citizenship: Filipino

Religion: Roman Catholic

Email Address: jekoy99@gmail.com

Contact Number: 09569912306

ACADEMIC INFORMATION

Tertiary: University of San Jose-Recoletos

Secondary: St. Scholastica’s Academy

Primary: Saint Teresa School of Avila

PROFESSIONAL SKILLS

Language: Dart/Flutter, C, HTML, CSS, Javascript/Puppeteer&Casper

Software: Figma, VS Code, Microsoft Office, Android Studio, Photoshop, Ubuntu, Docker, Superputty

OS: Windows, Android