A Project Report On

MUSIC PLAYER

Submitted in partial fulfillment of the requirement for the award of the degree

Bachelor of Computer Application (BCA)

Academic Year 2023 – 24

<Dishant Nakrani >	**<92200527180>**
<Manish Toyata>	**<92200527225>**
< Nikunj Gondaliya>	**<92200527244>**

Internal Guide
(Milan Doshi)

Marwadi University

Rajkot-Morbi Road, At & PO : Gauridad, Rajkot 360 003. Gujarat. India.

Faculty of Computer Applications (FCA)

Certificate

This is to certify that the project work entitled

Music player

submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Computer Application

of the

Marwadi University

is a result of the bonafide work carried out by

Student Name:Dishant Nakrani (92200527180)

Student Name:ManisToyata. (92200527225)

Student Name:Nikunj Gondaliya (92200527244)

during the academic year 2023-24

Faculty Guide	HOD	Dean

DECLARATION

I/We hereby declare that this project work entitled **Music Player** is a record done by me.

I also declare that the matter embodied in this project is genuine work done by us and has not been submitted whether to this University or to any other University / Institute for the fulfillment of the requirement of any course of study.

Place

Date :

Dishant Nakrani (92200527180) **Signature :** _____

Manish Toyata (92200527225) **Signature** : _____

Nikunj Gondaliya (92200527244) **Signature** : _____

1. Project Details

1.1 Synopsis:

A musical synopsis is a summary of a show that familiarizes the reader (producers, investors, directors, theatres, agents, etc.) with the plot, characters, and placement of songs within the musical.

1.2 Project Description:

The music player is a software project supporting all known media files and has the ability to play them with ease. The project features are as follows: User may attach Folder to Play add various media files within it. User may see track lists and play desired ones accordingly.

1.3 Project Modules Detailed Description:

1 .User Interface (UI) Module:

Purpose: Manages the visual presentation and user interactions.
Components: Main screen, playback controls (play, pause, stop, skip, volume control), playlist management, and settings.
Details: Includes graphical elements such as buttons, sliders, and menus. Handles user input and provides visual feedback.

2 .Audio Playback Module:

Purpose: Handles the actual playing of audio files.
Components: Audio player engine, media codecs, and buffering system.
Details: Responsible for decoding audio files (e.g., MP3, AAC), managing playback (play, pause, stop), and controlling audio output.

3 .Media Library Module:

Purpose: Manages the collection of audio files and metadata.
Components: Database or file system access, metadata extraction, and indexing.

Details: Scans directories for audio files, extracts metadata (e.g., artist, album, track name), and organizes files into a searchable library.

4 .Playlist Management Module:

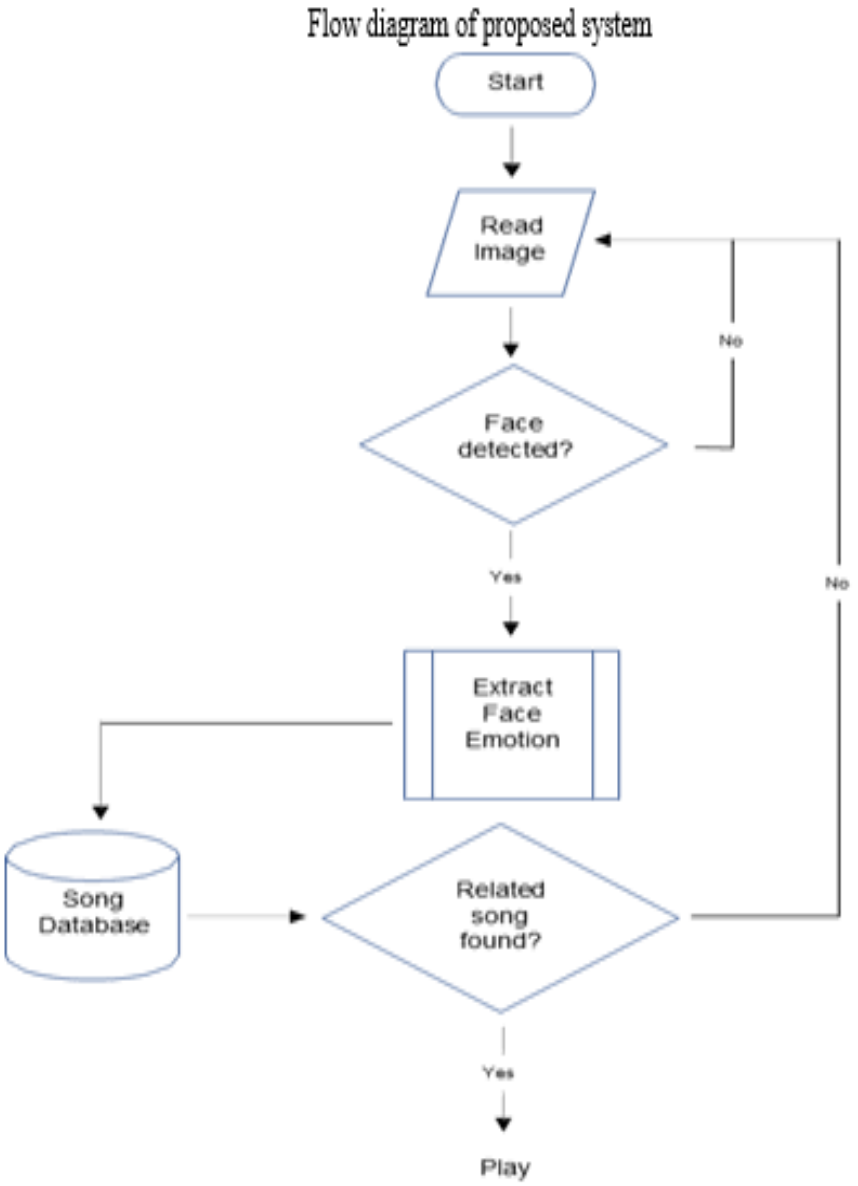
Purpose: Allows users to create, edit, and manage playlists.
Components: Playlist editor, playlist storage, and playlist playback.
Details: Provides functionality for adding/removing tracks from playlists, saving playlists, and loading playlists for playback.

5 .Search and Filtering Module:

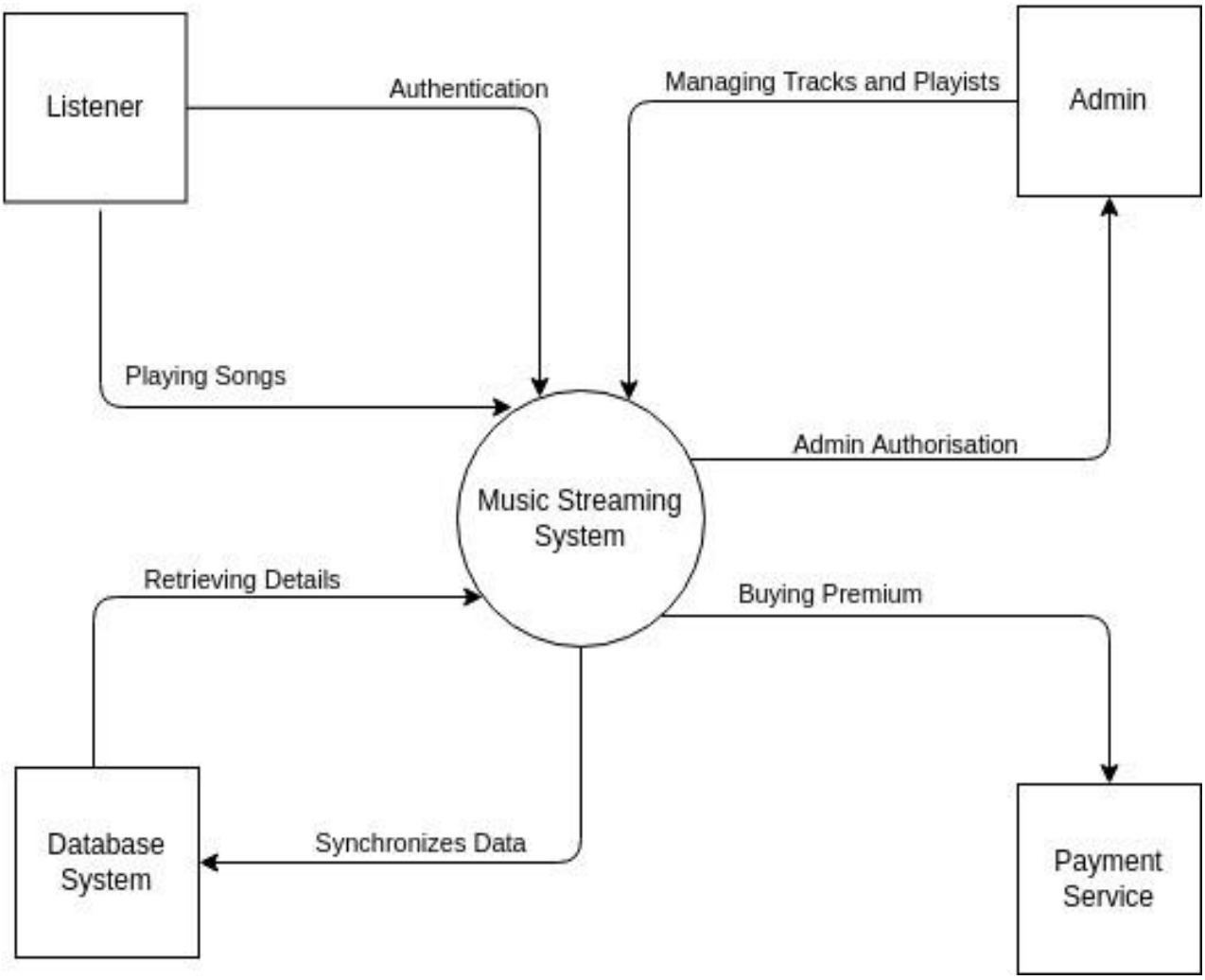
Purpose: Facilitates searching and filtering within the media library.
Components: Search engine, filtering options, and search results display.
Details: Allows users to search for specific tracks, artists, or albums and apply filters to narrow down results.

2.Diagrams as applicable :-

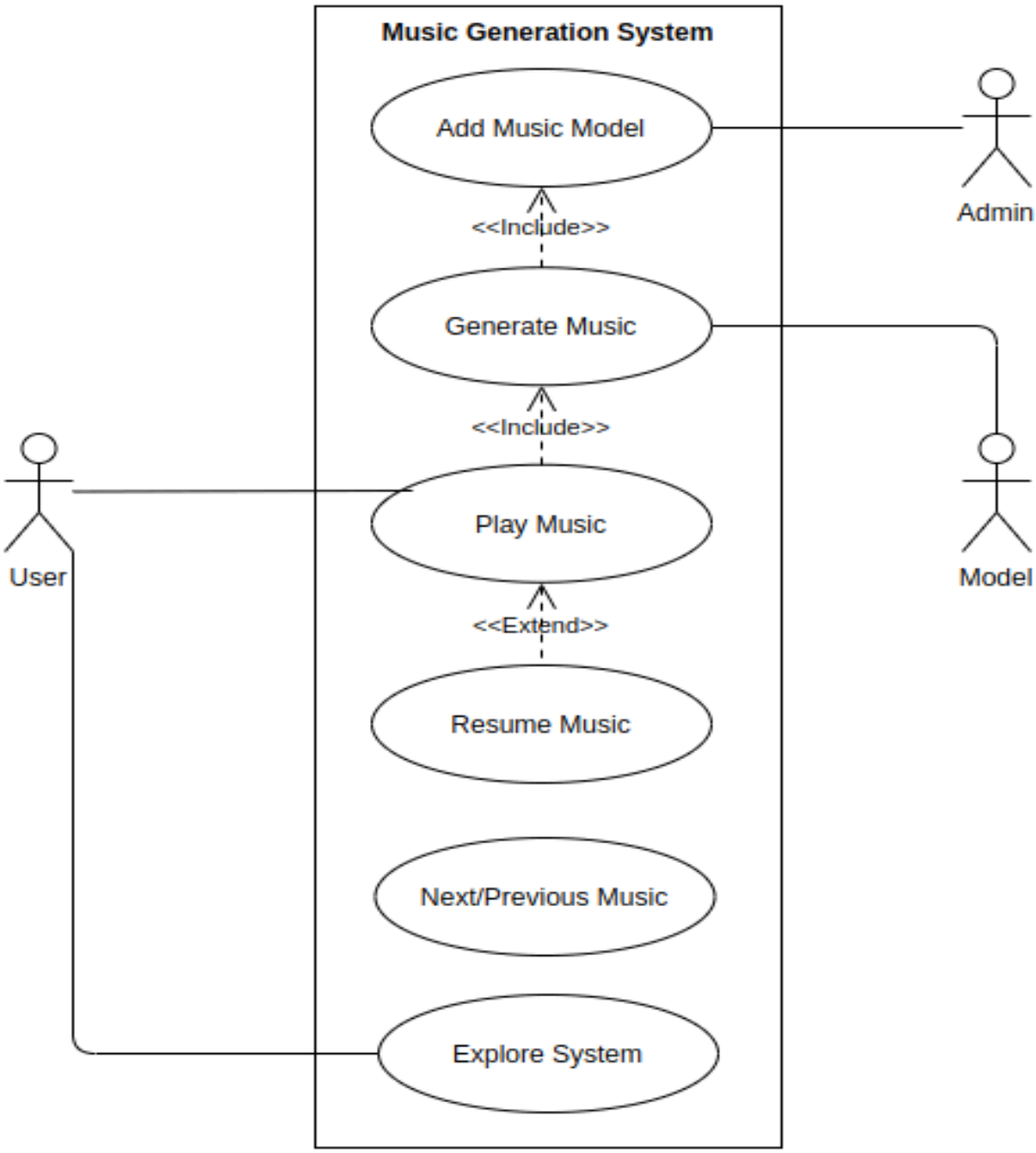
2.1 Flowchart :-



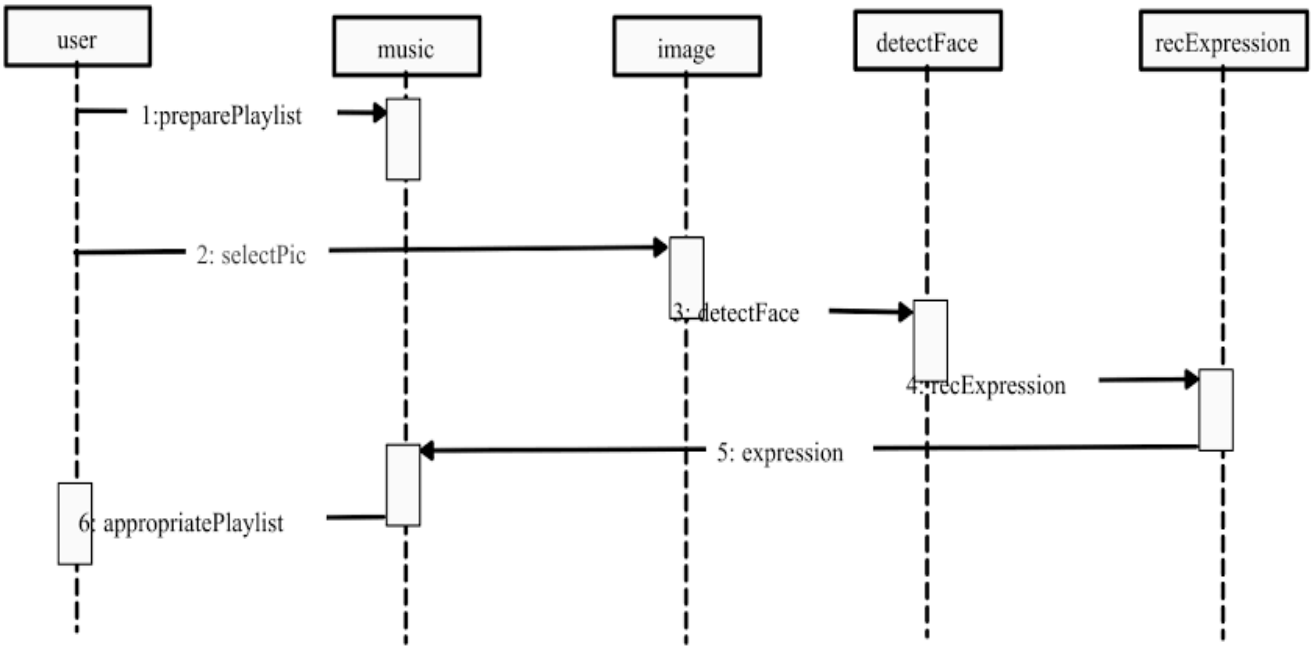
2.2 I



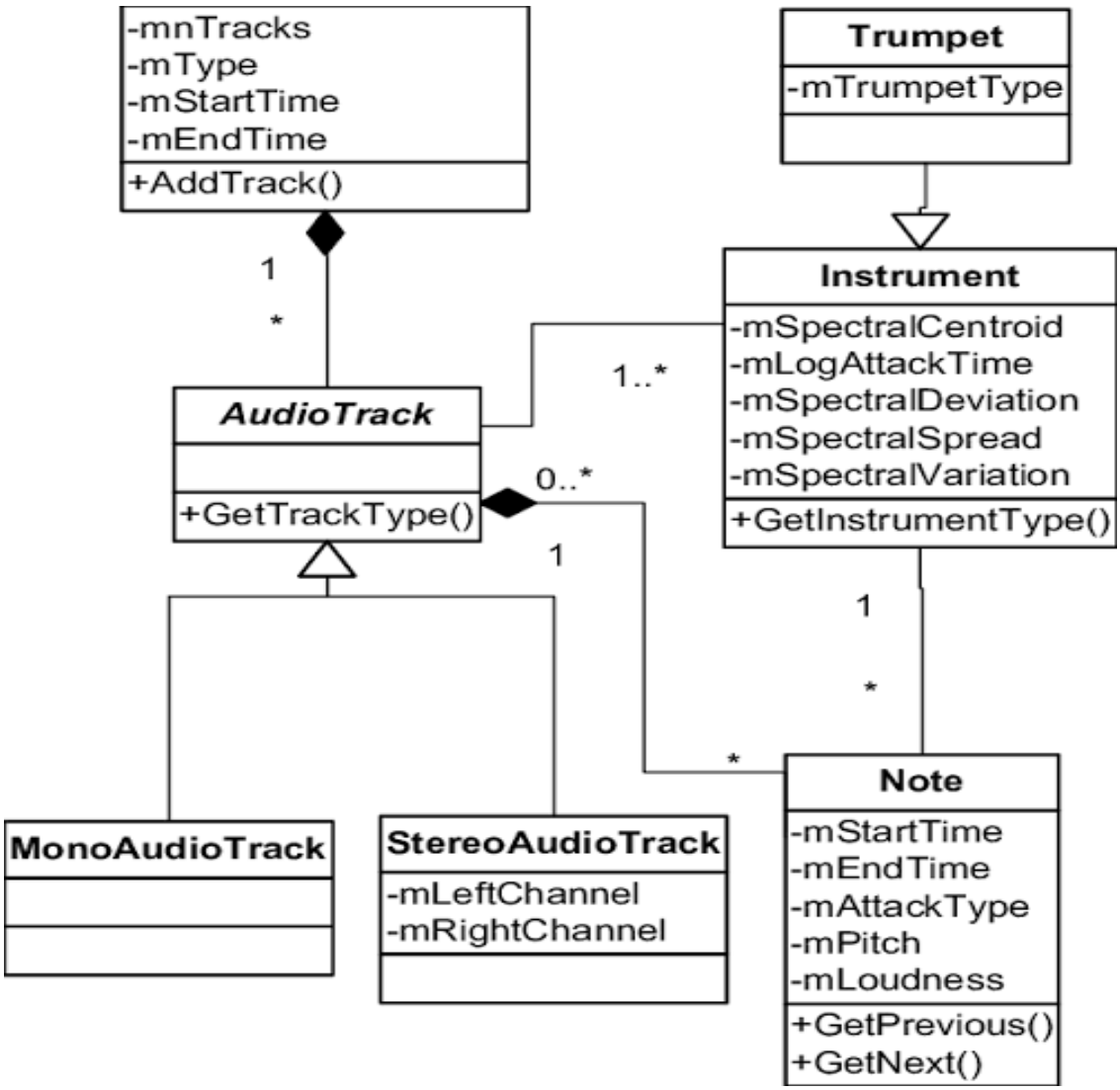
2.3 USE CASE DIAGRAM



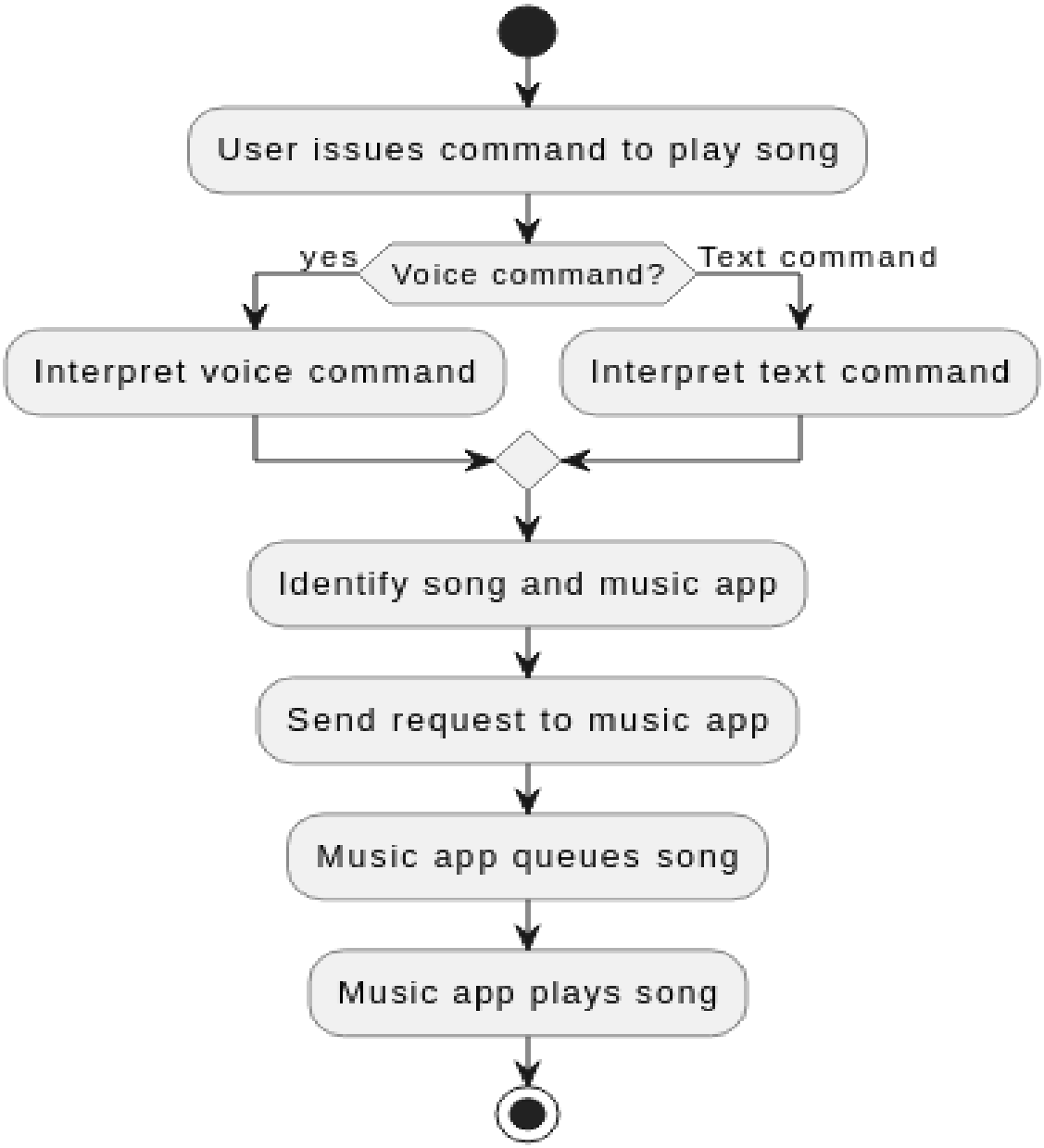
2.4 Sequential Diagram



2.5 Class Diagram



2.6 Activity Diagram:-



3. TECHNICAL DESCRIPTION

3**.1 -** Hardware Requirement

Processor - A modern multi-core processor to ensure smooth execution of the application.

RAM – 4GB (Minimum)

Storage – 100M

Display - 1280x800

3.2 Software Requirement

- **Programming Language:**

- **Python 3.x**: Ensure you have Python 3 installed. Python 3.6 or later is recommended to support all the libraries used in your code.

4 Database Structure (as applicable):-

This project aims to develop a relational database system for a music streaming platform to ensure that the music inventory is properly managed, playlists created and user interaction enabled. Below are the main key terms of the Music Streaming and Playlist Management are as follows:

Access to vast music libraries: Users can access a wide range of songs, albums, and artists from various genres.
On-demand listening: Users can play songs instantly without the need to download them.
Offline listening: Some platforms allow users to download music for offline listening.
High-quality audio: Many streaming services offer high-quality audio formats for a better listening experience.
Create and edit playlists: Users can create custom playlists based on their mood, genre preferences, or activity.
Collaborative playlists: Some platforms allow users to create playlists with friends, where multiple users can add and edit songs.
Import/export playlists: Users can import playlists from other platforms or export their playlists for use on different devices or services.

5. REVIEW OF LITERATURE:-

1.Python Libraries for Audio Playback

pygame

- **Overview**: pygame is a popular library for game development that includes a mixer module for audio playback.
- **Capabilities**: It supports loading, playing, pausing, and stopping audio files in various formats, primarily WAV and MP3.
- **References**:
  - *Official pygame Documentation*: pygame.org/docs
  - *Tutorial*: "Creating a Simple Music Player with Pygame" on Real Python or similar tutorial sites.

pydub

- **Overview**: pydub is a high-level library for manipulating audio files, built on ffmpeg or libav. It provides functionality for audio manipulation, including cutting, concatenation, and exporting.
- **Capabilities**: While pydub itself does not play audio, it can be used in conjunction with other libraries like simpleaudio to handle playback.
- **References**:
  Tutorial: "How to Use Pydub for Audio Manipulation" on Medium or similar platforms.

2. Graphical User Interface (GUI) Libraries

tkinter

- **Overview**: tkinter is the standard GUI library for Python. It is used for creating desktop applications with interactive interfaces.
- **Capabilities**: It provides widgets like buttons, labels, and dialogs, which can be used to create user interfaces for music players.
- **References**:
  - *Tutorial*: "Building a Music Player GUI with Tkinter" on platforms like GeeksforGeeks or Tutorialspoint.

PyQt/PySide

- **Overview**: PyQt and PySide are bindings for the Qt application framework, which provides advanced GUI capabilities.
- **Capabilities**: They offer more sophisticated widgets and tools for creating modern and responsive interfaces compared to tkinter.

- **References**:
  - *Official PyQt Documentation*: riverbankcomputing.com/software/pyqt/intro
  - *Tutorial*: "Creating a Music Player with PyQt" available on platforms like Real Python or PyQt tutorials.

3. Advanced Features and Extensions

Audio Visualization

- **Overview**: Incorporating visual elements such as waveforms or frequency spectrums can enhance the music player experience.
- **Libraries**: Libraries like matplotlib for static plots or PyQtGraph for real-time interactive plots can be used.
- **References**:
  - *Tutorial*: "Visualizing Audio Data in Python" on Medium or Towards Data Science.

Streaming and Network Features

- **Overview**: Advanced music players may support streaming audio from the web or networked sources.
- **Libraries**: Libraries like requests for HTTP streaming or gstreamer for more complex streaming needs.
- **References**:
  - *Tutorial*: "Streaming Audio with Python" on sites like Real Python or specific library documentation.

Playlist Management

- **Overview**: Adding playlist functionality allows users to manage and play multiple audio files in sequence.
- **Libraries**: Can be implemented using Python's standard data structures or libraries like playlist for more structured approaches.
- **References**:
  - *Tutorial*: "Building a Playlist Manager in Python" available on various Python development blogs.

THANK YOU