MINOR PROJECT
ON
MELODY -MUSIC PLAYER

Submitted In Partial Fulfilment for Award of Degree In

BACHELOR OF COMPUTER APPLICATIONS

(BATCH 2021-2024)

Submitted By:

DRISHTI:35120602021

DEEPANSHU:01620602021

UNDER THE ESTEEMED GUIDANCE OF
Ms. Nishika Mam

TRINITY INSTITUTE OF PROFESSIONAL STUDIES,DWARKA
(Affiliated to GGSIPU,DWARKA)

ACKNOWLEDGEMENT

We are preparing the **"MINOR PROJECT"**.

I would like to take this opportunity to express my gratitude towards all the people who have in various ways, helped in the successful completion of my project.

I must convey my gratitude to Ms. Nishika Mam for giving me a constant source of inspiration and help in preparing the project, personally correcting my work and providing encouragement throughout the project.

I also thank all my faculty members for steering me through the tough as well as easy phases of the project in a result-oriented manner with concern attention.

ATTESTATION OF AUTHORSHIP

We hereby declare that this Submission for partial fulfillment of the requirements for the degree BCA course is our work and that to the best of my knowledge and belief,

It contains no written material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning except where due acknowledgment is made.

ABSTRACT

The "Melody" music player for Windows is a sophisticated and user-friendly application designed as a minor project to enhance the digital music listening experience. With a sleek and intuitive interface, Melody prioritizes simplicity and functionality, providing users with a seamless platform to organize, explore, and enjoy their music collection.

Key features of Melody include an advanced playlist management system, enabling users to curate personalized playlists effortlessly. The player supports various audio formats, ensuring compatibility with diverse music libraries. Melody also incorporates a user-friendly search and sorting mechanism, facilitating quick and efficient navigation through extensive music collections.

One of the standout features of Melody is its robust audio playback engine, delivering high-quality sound reproduction and supporting various audio enhancements. The application is optimized for performance, ensuring a smooth and responsive user experience even with large music libraries.

In addition to its core functionalities, Melody offers a visually appealing and customizable interface, allowing users to personalize their music player according to their preferences. With a focus on user satisfaction and accessibility, Melody aims to provide a seamless and enjoyable music listening experience on the Windows platform, making it an ideal choice for music enthusiasts seeking a reliable and feature-rich music player.

TABLE OF CONTENTS

3. SYSTEM DESIGN

3.1 INPUT AND OUTPUT DESIGN
3.1.1 INPUT DESIGN
3.1.2 OUTPUT DESIGN
3.2 DATABASE
3.3 SYSTEM TOOLS
3.3.1 FRONT END
3.3.2 BACK END
3.4 TABLES
3.5 E-R DIAGRAMS
3.6 DATA FLOW DIAGRAMS (DFD)
3.7 SCREEN SHOTS
3.8 SAMPLE CODE

4. CONCLUSION

CHAPTER 1
INTRODUCTION

Introducing "Melody," a visionary endeavor in the realm of digital music, poised to redefine the Windows music player experience. Developed as a comprehensive minor project, Melody represents the fusion of innovation and user-centric design, with the singular goal of enhancing how users interact with and enjoy their music collections on the Windows platform.

The genesis of Melody stems from a recognition of the diverse needs and preferences of music enthusiasts. The contemporary digital landscape demanded a music player that seamlessly blended sophistication with simplicity, offering a feature-rich experience without sacrificing user-friendliness. In response to this need, the Melody project was conceived as an ambitious initiative to craft a music player that transcends conventional boundaries.

At its core, Melody aspires to be more than just a player; it seeks to be an immersive platform that resonates with users on a personal level. The project's journey commenced with a well-defined objective: to create a user-friendly interface that marries advanced functionalities with an intuitive design. The user experience was placed at the forefront, steering the project towards the development of a music player that not only meets but exceeds the expectations of Windows users.

Central to Melody's design philosophy is its advanced playlist management system. Recognizing the importance of customization and personalization in the modern music consumption landscape, Melody empowers users to curate and organize their music collections effortlessly. The playlist management system is designed to be intuitive, allowing users to navigate through their libraries, create dynamic playlists, and tailor their musical journey to their unique preferences.

Melody's commitment to versatility is further demonstrated through its robust support for various audio formats. In acknowledging the diverse nature of digital music files, the project ensures that Melody can seamlessly handle a plethora of formats, providing users with the freedom to enjoy their music without the constraints of compatibility.

The auditory experience lies at the heart of Melody's development. A sophisticated audio playback engine has been integrated to deliver a high-fidelity sound reproduction, ensuring that users can immerse themselves in the richness of their favorite tracks. The project recognizes that true enjoyment of music goes beyond organization and extends into the quality of the listening experience.

The interface design of Melody serves as a visual testament to the commitment to user satisfaction. Striking a harmonious balance between aesthetics and functionality, Melody offers a visually appealing, customizable interface. Users have the freedom to personalize their Melody experience, from themes to layouts, enhancing the overall enjoyment of the application.

As Melody embarks on its journey as a minor project, it seeks to not only meet but exceed the expectations of Windows users seeking a refined and versatile music player. The introduction of Melody is an invitation to explore a new dimension in digital music, where simplicity meets sophistication, and the user is at the center of the melody.

1.1 PROJECT OBJECTIVE:

The overarching objective of the "Melody" project is to create an exceptional music player tailored for Windows, redefining the digital music listening experience. This minor project seeks to address the evolving needs of music enthusiasts by developing a feature-rich, user-friendly application that seamlessly integrates innovation and accessibility.

At the forefront of Melody's objectives is the design and implementation of an intuitive user interface, prioritizing ease of use and a visually appealing layout. The project aims to empower users with a sophisticated yet accessible platform, ensuring that both casual listeners and avid music collectors can navigate and engage with their music seamlessly.

A key focus is the development of an advanced playlist management system. Melody aims to provide users with a powerful tool to organize, curate, and personalize their music collections effortlessly. The playlist management system is designed to be intuitive, allowing users to create, modify, and navigate playlists with ease, thereby enhancing the overall user experience.

Compatibility and versatility are paramount in the project's objectives. Melody is engineered to support a wide array of audio formats, acknowledging the diversity of music files available. This ensures that users can enjoy their music without concerns about format restrictions, fostering a more inclusive and flexible music playback experience.

The auditory experience is a central facet of Melody's objectives. The project endeavors to implement a robust audio playback engine, delivering high-quality sound reproduction. This focus on audio quality aims to elevate the enjoyment of music, recognizing that the essence of a music player lies not only in its organizational capabilities but also in the fidelity with which it presents the audio content.

Furthermore, Melody seeks to provide users with a customizable interface, allowing personalization of themes and layouts to enhance the visual experience. The project acknowledges that users have diverse preferences, and the ability to tailor the interface ensures that Melody caters to a broad spectrum of individual tastes.

In summary, the project's objectives for Melody encompass the creation of a sophisticated, user-centric music player for Windows. By combining advanced playlist management, support for various audio formats, a robust audio playback engine, and a customizable interface, Melody aims to redefine how users interact with their music libraries, promising a versatile and enriching music listening experience on the Windows platform.

1.2 PROJECT OVER VIEW:

The "Melody" project represents a transformative endeavor focused on crafting a cutting-edge music player for the Windows platform. This minor project is driven by a comprehensive overview, encompassing the development of a sophisticated and user-centric application designed to redefine the digital music listening experience.

Melody is conceived with the primary objective of creating a music player that strikes a harmonious balance between innovation and user-friendliness. The project is rooted in the understanding that the modern digital music landscape demands more than just a playback tool – it calls for a seamlessly integrated

platform that caters to the diverse needs of users, from casual listeners to avid music enthusiasts.

At its core, Melody seeks to revolutionize the user interface, introducing a visually appealing and intuitive design that transcends traditional expectations. The project recognizes the importance of accessibility, ensuring that users can effortlessly navigate through their music collections and interact with the application in a way that feels both natural and engaging.

A pivotal component of Melody's project overview is the development of an advanced playlist management system. This feature aims to empower users by providing a powerful tool for organizing and customizing their music collections. Melody seeks to streamline the playlist creation process, allowing users to curate dynamic playlists with ease and ensuring that the application adapts to the unique preferences of each user.

Compatibility and versatility are key pillars of the project, with Melody engineered to support a diverse range of audio formats. This ensures that users can seamlessly integrate their existing music libraries into the application, eliminating concerns about format restrictions and fostering a more inclusive and flexible music playback experience.

Melody's commitment to an immersive auditory experience is reflected in the implementation of a robust audio playback engine. This engine is designed to deliver high-fidelity sound reproduction, recognizing that the true essence of a music player lies in the quality with which it presents the audio content.

Furthermore, Melody places a strong emphasis on a customizable interface, allowing users to personalize their experience through themes and layouts. This ensures that Melody caters to a broad spectrum of individual tastes, enhancing the visual and interactive aspects of the music playback journey.

In summary, the project overview for Melody encompasses a holistic approach to developing a Windows music player that transcends conventional boundaries. By integrating advanced playlist management, support for various audio formats, a robust audio playback engine, and a customizable interface, Melody aspires to be more than just a music player—it aims to be a transformative platform that redefines how users engage with and enjoy their music on the Windows platform.

1.3 PROJECT SCOPE

The project scope for "Melody," a music player tailored for the Windows platform, encompasses a comprehensive range of features and functionalities aimed at delivering a superior digital music listening experience. This minor project outlines the parameters within which Melody will be developed, focusing on key aspects that contribute to its success and user satisfaction.

1. User-Centric Interface:

Melody's development revolves around creating an intuitive and visually appealing user interface. The scope includes designing an interface that is not only aesthetically pleasing but also user-friendly, ensuring easy navigation for users of varying technical expertise.

2. Advanced Playlist Management:

A central component of Melody's scope is the implementation of an advanced playlist management system. This feature is designed to empower users by providing a robust tool for organizing and customizing their music collections. Melody will allow users to effortlessly create, edit, and manage playlists, fostering a personalized and dynamic music listening experience.

3. Audio Format Compatibility:

Recognizing the diverse nature of digital music files, the project scope includes ensuring Melody's compatibility with various audio formats. This feature aims to eliminate restrictions on the types of music files users can enjoy, promoting inclusivity and flexibility in managing their music libraries.

4. Robust Audio Playback Engine:

The project emphasizes the integration of a high-performance audio playback engine. This engine is pivotal in delivering a superior auditory experience, focusing on high-fidelity sound reproduction. Melody aims to elevate the quality of music playback, contributing to a more immersive and enjoyable listening environment.

5. Performance Optimization:

The scope involves a dedicated effort towards performance optimization to guarantee a smooth and responsive user experience. Melody is designed to

efficiently handle large music libraries, ensuring that users can navigate and interact with the application seamlessly, even with extensive collections.

6. Customizable Interface:
Recognizing the importance of personalization, Melody will feature a customizable interface. Users will have the ability to tailor the visual aspects of the application, including themes and layouts, to suit their individual preferences, enhancing the overall user experience.

7. Documentation and User Guidance:
The project scope extends to the creation of comprehensive documentation. This documentation will serve as a user guide, providing clear instructions on Melody's features and functionalities, ensuring users can maximize the utility of the music player.

8. Compatibility Across Windows Versions:
Melody is scoped to be compatible across various Windows versions, aiming to reach a broad user base and ensuring accessibility to users regardless of their operating system version.

In addition to the outlined project scope for "Melody," there are specific enhancements and features that will be analyzed and incorporated into the development process. These additions are aimed at making Melody not only a music player but also a personalized and secure platform for users. The analysis and integration of these features will contribute to the major project's goal of differentiating Melody from existing platforms, with a particular focus on user authentication and innovative functionalities.

9. User Authentication and Login System:
A critical aspect of the major project involves the implementation of a user authentication and login system for Melody. This will add a layer of security and personalization, allowing users to create accounts, log in securely, and access personalized features such as saved playlists and preferences.

10. Enhanced User Profiles:
The analysis will delve into the development of enhanced user profiles, enabling users to customize their Melody experience further. This may include options to add profile pictures, display names, and other personalized details, fostering a sense of ownership and individuality within the platform.

11. Unique Feature Set Differentiating from Spotify:
The major project aims to identify and implement significant differentiators from existing platforms like Spotify. Through in-depth analysis, Melody will introduce innovative features that set it apart in the market, potentially reshaping how users interact with and experience their music.

12. Social Integration and Sharing:
The analysis will explore the integration of social features, allowing users to share their favorite tracks, playlists, or even collaborate on playlists with friends. This social aspect aims to enhance the community aspect of Melody, fostering a sense of connection among users.

13. Seamless Cross-Platform Integration:
Consideration will be given to ensuring seamless integration across various Windows versions, creating a consistent user experience. This analysis will involve optimizing Melody for different Windows environments, ensuring a uniform and reliable performance.

14. User Feedback and Iterative Development:
The major project will incorporate a robust feedback loop, encouraging user input and actively iterating on Melody's features based on user suggestions and preferences. This iterative development approach ensures that Melody remains responsive to user needs and preferences over time.

15. Accessibility and Inclusivity:
The analysis will focus on ensuring Melody is accessible to a diverse user base. This includes considerations for users with varying levels of technical expertise and potentially incorporating features that enhance the inclusivity of the platform.

In summary, the major project aims to extend the project scope for Melody by incorporating user authentication, enhanced profiles, unique differentiators from Spotify, social integration, cross-platform optimization, and an iterative development approach based on user feedback. These additions will contribute to the creation of a distinctive and user-centric music player that not only meets but exceeds the expectations of users in the dynamic digital music landscape.

1.4 STUDY OF THE SYSTEM

1.4.1 MODULES:

The system after careful analysis has been identified to be presented with the following modules and roles.

The modules involved are:

- Melody code file
- Songs Playlist – Categorical Songs (Bollywood/Hollywood)-

  Sub Categories
- Images

1.4.1.1 MELODY:

The administrator is the super user of this application. Only admin have access into this admin page. Admin may be the owner of the shop.
The administrator has all the information about all the users and about all products.

This module is divided into different sub-modules.

1. Manage Moderators
2. Manage Products
3. Manage Users
4. Manage Orders

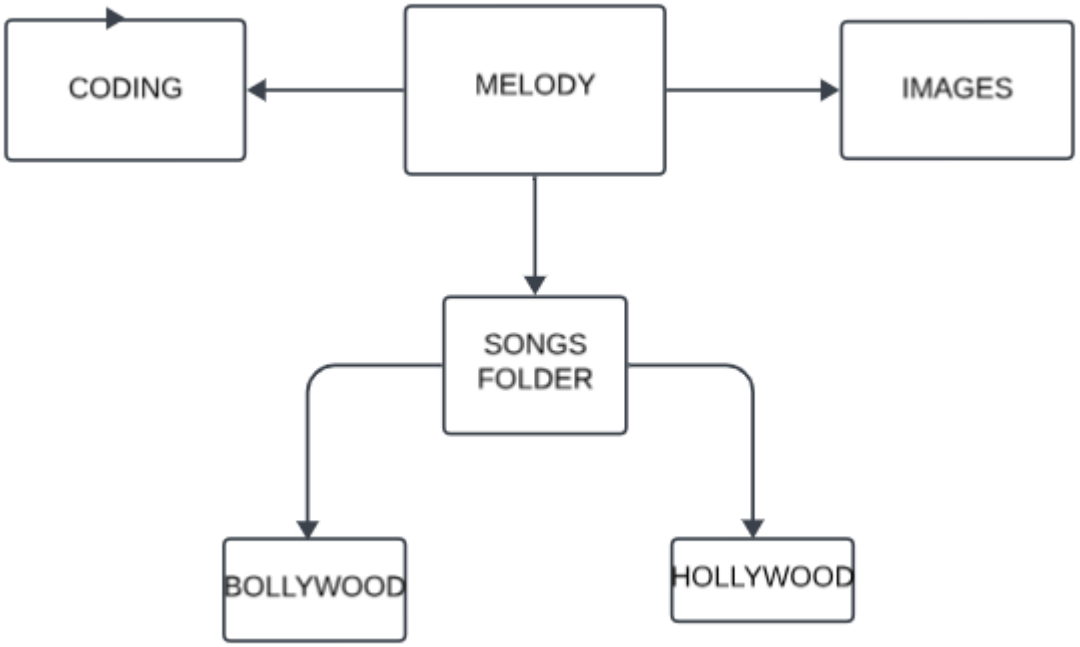


Fig 1.4.1: Admin Module

Fig 1.4.2: Manage Moderator

- Add Moderator

Only admin is having the privilege to add a moderator. A moderator can be considered as a staff who manages the orders or owner of a group of products.

- Block moderator

Admin can restrict a moderator from managing the orders by blocking them. Admin can unblock a blocked user if needed.

- Remove Moderator

Admin has privilege to delete a moderator who was added.

- Search moderator:

All existing moderators can be viewed by the administrator as a list. If there is number of moderators and admin need to find one of them, the admin can search for a moderator by name.

MANAGE PLAYLIST

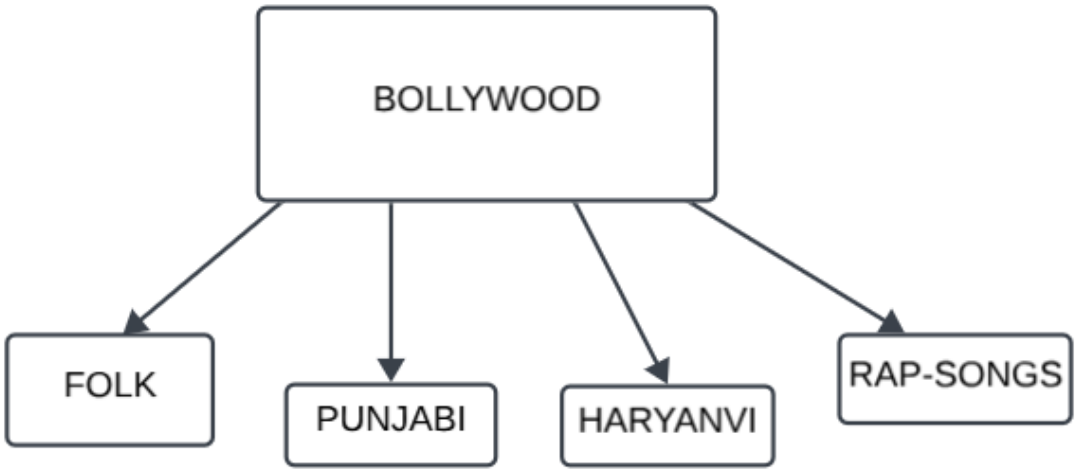


Fig 1.4.3: Manage Playlist

- ADD PRODUCTS

The shopping cart project contains different kind of products. The products
can be classified into different categories by name. Admin can add new products into the existing system with all its details including an image.

 Delete Products

Administrator can delete the products based on the stock of that particular
product.

 Search products

MANAGE HOLLYWOOD

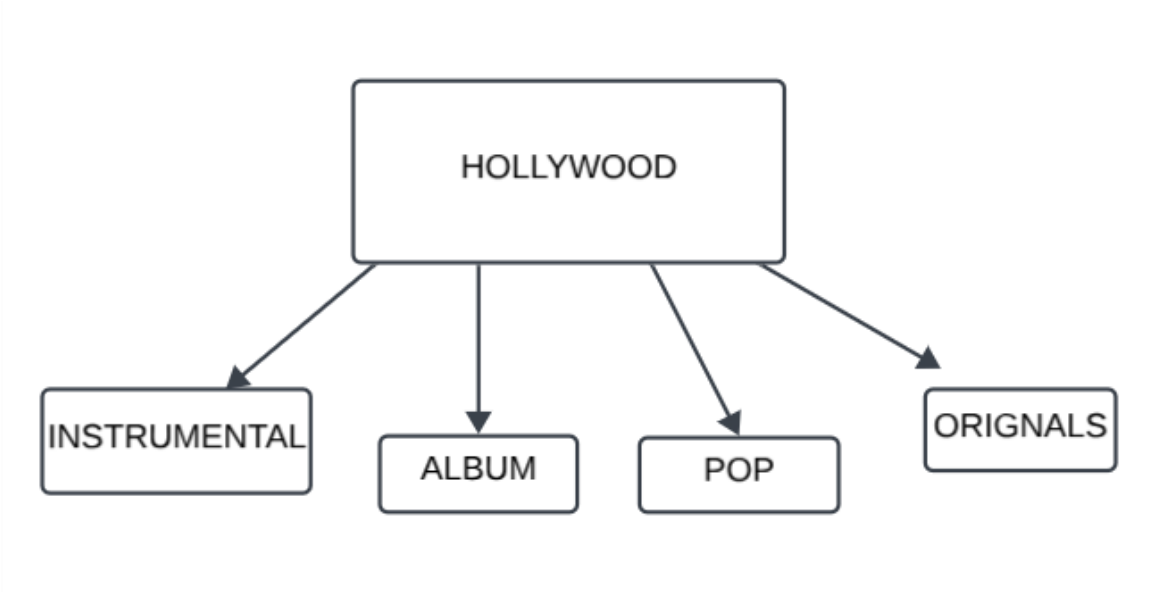


Fig 1.4: Manage HOLLYWOOD

Admin will have a list view of all the existing products. He can also search for a particular product by name.

 View Users

The admin will have a list view of all the users registered in the system. Admin can view all the details of each user in the list except password.

 Add Users

Admin has privileges to add a user directly by providing the details.

 Delete &Block Users

Administrator has a right to delete or block a user. The default status of a new user registered is set as blocked. The admin must accept the new user by unblocking him.

A moderator is considered as a staff who can manage orders for the time being. As a future update moderator may give facility to add and manage his own products. Moderators can reduce the work load of admin. Now moderator has all the privilege an admin having except managing other

moderators. He can add products and users. He can also check the orders and edit his profile.

- Manage products
- Manage users
- Manage orders

1.4.1.2 USERS

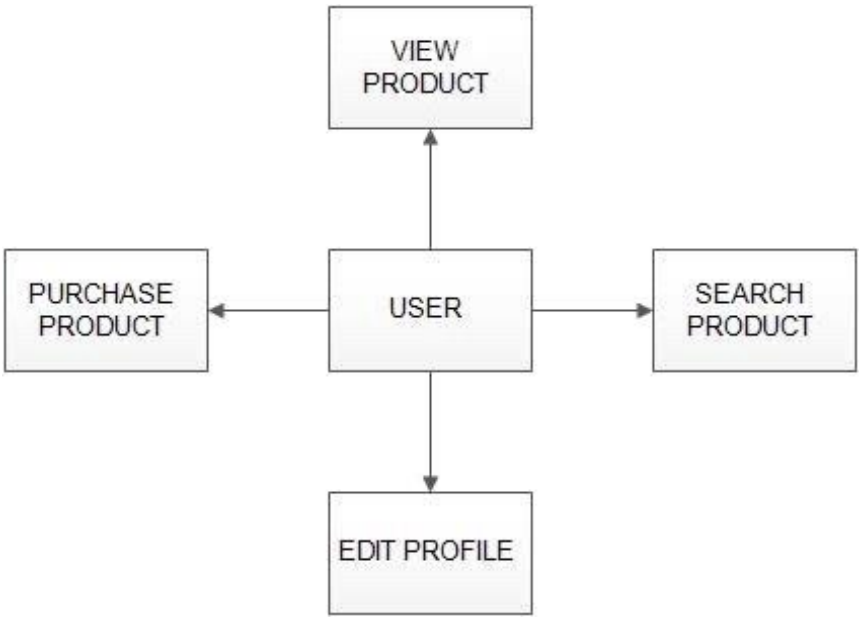


Fig 1.5: User Module

 Registration

A new user will have to register in the system by providing essential details in order to view the products in the system. The admin must accept a new user by unblocking him.

 Login

A user must login with his user's name and password to the system after registration.

 View Products

User can view the list of products based on their names after successful login. A detailed description of a particular product with product name, products details, product image, price can be viewed by users.

 Search Product

Users can search for a particular product in the list by name.

 Add to cart:

The user can add the desired product into his cart by clicking add to cart option on the product.

He can view his cart by clicking on the cart button. All products added by cart can be viewed in the cart. User can remove an item from the cart by clicking remove.

 Submit Cart:

After confirming the items in the cart, the user can submit the cart by providing a delivery address. On successful submitting the cart will become empty.

 History

In the history the user will have a view of pending orders.

 Edit Profile

The user can view and edit the profile.

CHAPTER 2

2.1 SYSTEM ANALYSIS

System study is the first stage of a system development life cycle. This gives a clear picture of what the physical system actually is. The system is done in two phases. In the first phase, the preliminary survey of the system is done which helps in identifying the scope of the system. The second phase of the system study is a more detailed and in-depth study in which the identification of user's requirements and limitations and problems of the present system are studied. After completing the system study, a proposal is prepared by the user.

System analysis is the detailed study of the various operations performed by the system and relationships within and outside the system. It is the most essential part of the development of the project. During system analysis data is collected on the available file, decisions points and transactions handled by the presence system. System analysis must carry out a customary approach to the use of computers for problem solving.

EXISTING SYSTEM

The current system for Melody is to visit the software applications like - youtube, spotify manually and from the available product choose the item customer want and buying the item by payment of the price of the item .

1. It always require Internet
2. User must to go online and select songs.
3. It is a time consuming process
4. Not in reach of distant users.
5. Contains Advertisement
6. Not pocket friendly as requires Subscriptions

2..0 PROPOSED SYSTEM

The proposed system for "Melody" envisions a state-of-the-art music player for Windows that transcends traditional boundaries. Melody will feature a meticulously designed user interface, providing an intuitive and visually appealing platform for seamless navigation and interaction. The core of the proposed system lies in the development of an advanced playlist management system, empowering users to curate and personalize their music collections effortlessly.

Furthermore, the system will prioritize compatibility by supporting a diverse range of audio formats, ensuring users can enjoy their music without constraints. Melody aims to incorporate a robust audio playback engine to deliver a high-fidelity listening experience. Additionally, the proposed system includes a customizable interface, allowing users to tailor the visual aspects according to their preferences. The culmination of these features will result in Melody, a sophisticated music player that sets new standards for user-friendliness, versatility, and audio quality on the Windows platform.

2.1 SYSTEM REQUIREMENT SPECIFICATION

2.1.1 GENERAL DESCRIPTION

Product Description:

Melody, the innovative music player for Windows, redefines digital audio playback. With a sleek interface, advanced playlist management, and compatibility with various audio formats, Melody offers a user-centric experience. The robust audio playback engine ensures high-fidelity sound, while a customizable interface allows personalization. Elevate your music journey with Melody—where simplicity meets sophistication on the Windows platform.

Problem Statement:

In the context of Melody, the music player for Windows, the problem statement revolves around the limitations of current online-based systems in providing a seamless and user-friendly music playback experience. Many existing platforms prioritize online streaming, often neglecting the needs of users with locally stored music libraries. This presents a significant gap for users who prefer offline access to their extensive collections. The prevalent focus on cloud-based solutions also poses challenges related to privacy and security concerns, as users may be hesitant to upload their entire music repositories to online servers. Additionally, the dependency on continuous internet connectivity excludes users in areas with limited or no internet access.

Melody aims to address these issues by offering an innovative and locally focused music player for Windows. By providing advanced playlist management, support for various audio formats, and a robust playback engine, Melody aims to fill the void left by online-based systems, catering to users who value offline access, data privacy, and a more tailored, localized music experience on the Windows platform. The problem statement for Melody is an exploration of how the current trend towards online-based music systems falls short for certain users, and how a dedicated Windows music player can bridge these gaps.

2.1.2 SYSTEM OBJECTIVES

- To provide offline based system in an existing system.
- To provide an offline player for music without compromising user entertainment

2.1.3 SYSTEM REQUIREMENTS

2.1.3.1 NON-FUNCTIONAL REQUIREMENTS

Performance: Responsiveness: Melody should provide a highly responsive user interface, ensuring immediate feedback to user inputs such as playback control and playlist modifications.

Loading Time: The application should have fast loading times to enable users quick access to their music libraries upon launching Melody.

Streaming Efficiency: If online features are incorporated, efficient streaming with minimal buffering should be a priority to optimize the user experience.

Stability: Melody should be stable and reliable, minimizing crashes, freezes, or unexpected terminations during regular usage.

Error Handling: The system should feature effective error handling mechanisms, providing users with clear and actionable messages to resolve issues encountered during operation.

Recovery Mechanism: In the event of unexpected shutdowns or errors, Melody should have a recovery mechanism to restore the application to its previous state.

Compatibility:

Cross-Version Compatibility: Melody should be compatible with various Windows versions, ensuring a consistent and reliable user experience across different operating system environments.

Audio Format Support: The application should support a diverse range of audio formats commonly used by users, ensuring compatibility with different types of music files.

Hardware Compatibility: Melody should seamlessly work with various hardware configurations, accommodating users with different computer setups and specifications.

Security:

User Authentication: If a login system is implemented, it should follow best practices to ensure the security of user accounts and personal information.

Data Privacy: Melody should prioritize user data privacy, implementing measures to securely handle and store any user-related information.

Encryption: Sensitive information, such as login credentials and user preferences, should be encrypted to protect against unauthorized access.

Usability: Intuitive Interface: The user interface should be intuitive and user-friendly, catering to users of different technical expertise levels.

Customizability: Melody should allow users to customize the interface, including themes, layouts, and other visual elements, to enhance personalization.

Accessibility: The application should be designed to be accessible to users with

disabilities, incorporating features such as keyboard shortcuts and compatibility with screen readers.

Scalability:

Future Expandability: The system should be designed with flexibility for future enhancements and feature additions, ensuring Melody can adapt to evolving user needs.
Database Scalability: If applicable, the database architecture should be scalable to accommodate a growing user base and increasing volumes of music data.

User Guides: Clear and comprehensive user documentation should be provided, offering instructions on using Melody's features, settings, and functionalities.
Developer Documentation: Internal documentation should be available for developers, aiding in system maintenance, updates, and future development efforts.
Technical Support Documentation: Detailed documentation addressing common user issues and providing technical support resources should be made available.

These non-functional requirements form the foundation of Melody's development, ensuring it is a performant, reliable, secure, user-friendly, and scalable music player for Windows that caters to a diverse user base.

2.1.3.2 FUNCTIONAL REQUIREMENTS

User Authentication:

Implement a secure user authentication system to allow users to create accounts, log in securely, and access personalized features.

Playlist Management:

Develop an advanced playlist management system allowing users to create, edit, and organize playlists efficiently.

Enable drag-and-drop functionality for easy playlist customization.

Provide options for playlist sharing and collaborative playlist creation.

Audio Playback:

Implement a robust audio playback engine to ensure high-fidelity sound reproduction.

Support basic playback controls such as play, pause, skip, and volume adjustment.

Include features for repeat and shuffle modes to enhance the listening experience.

Audio Format Support:

Support a wide range of audio formats including MP3, FLAC, WAV, and others to ensure compatibility with various music files.

User Interface:

Design an intuitive and visually appealing user interface with easy navigation for users of varying technical proficiency.

Allow users to customize the interface, including themes, color schemes, and layout preferences.

Search and Sorting:

Implement a robust search mechanism to allow users to quickly locate specific songs, albums, or artists within their music library.

Provide sorting options based on criteria such as title, artist, genre, and date added.

Offline Mode:

Develop an offline mode allowing users to access their music library without an internet connection.

Ensure seamless functionality in offline mode, including playback, playlist management, and search capabilities.

Cross-Platform Integration:

Ensure compatibility with various Windows versions, providing a consistent and reliable user experience across different operating system environments.

Enable synchronization of user preferences and playlists across multiple devices if applicable.

Social Integration:

Integrate social features, allowing users to share their favorite tracks, playlists, or collaborate on playlists with friends.

Provide options for users to connect and share music experiences on social media platforms.

Accessibility:

Design the application to be accessible to users with disabilities, incorporating features

such as keyboard shortcuts and compatibility with screen readers.

Documentation:

Create comprehensive user documentation outlining features, settings, and functionalities to guide users.

Develop internal developer documentation to facilitate system maintenance, updates, and potential future development efforts.

Feedback Mechanism:

Implement a user feedback mechanism allowing users to provide input on their experience and suggest improvements.

Consider user feedback in iterative development processes to enhance the application based on user preferences and needs.

These functional requirements collectively define the core features and capabilities of Melody, ensuring it becomes a versatile and user-centric music player for Windows.

CHAPTER 3

SYSTEM DESIGN

System design is the solution for the creation of a new system. This phase focuses on the detailed implementation of the feasible system. It emphasis on translating design. Specifications to performance specification. System design has two phases of development

- Logical design
- Physical design

During the logical design phase, the analyst describes inputs (sources), outputs (destinations), databases (data sores), and procedures (data flows) all in a format that meets the user requirements. The analyst also specifies the needs of the user at a level that virtually determines the information flow in and out of the system and the data resources. Here the logical design is done through data flow diagrams and database design. The physical design is followed by physical design or coding. Physical design produces the working system by defining the design specifications, which specify exactly what the candidate system must do. The programmers write the necessary programs that accept input from the user, perform necessary processing on accepted data and produce the required report on a hard copy or display it on the screen

.

3.1 INPUT AND OUTPUT DESIGN

3.1.1 INPUT DESIGN:

Input design is the link that ties the information system into the world of its users. The input design involves determining the inputs, validating the data, minimizing the data entry, and providing a multi-user facility. Inaccurate inputs are the most common cause of errors in data processing. Errors entered by the data entry operators can be controlled by input design. The user-originated inputs are converted to a computer-based format in the input design. Input data are collected and organized into groups of similar data. Once identified, the appropriate input media are selected for processing. All the input data are validated and if any data violates any conditions, the user is warned by a message. If the data satisfies all the conditions, it is transferred to the appropriate tables in the database. In this project, the student details are to be entered at the time of registration. A page is designed for this purpose which is user-friendly and easy to use. The design is done such that users get appropriate messages when exceptions occur.

3.1.2 OUTPUT DESIGN:

Computer output is the most important and direct source of information to the user. Output design is a very important phase since the output needs to be in an efficient manner. Efficient and intelligible output design improves the system relationship with the user and helps in decision making. Allowing the user to view the sample screen is important because the user is the ultimate judge of the quality of output. The output module of this system is the selected notifications.

3.2 DATABASE

DATABASE DESIGN:

Databases are the storehouses of data used in the software systems. The data is stored in tables inside the database. Several tables are created for the manipulation of the data for the system. Two essential settings for a database are

- the field that is unique for all the record occurrences.

-the field used to set relation between tables.

Normalization is a technique to avoid redundancy in the tables.

3.3 **TOOLS AND TECHNOLOGIES**

These are the necessary tools and materials needed to build the website both the front-end and the back-end. These include software and open-source materials.

PYTHON

Python is a high-level, dynamically-typed programming language known for its simplicity, readability, and versatility. Created by Guido van Rossum, Python emphasizes code readability and ease of use, making it an excellent choice for beginners and experienced developers alike. Its extensive standard library and vibrant community contribute to its popularity in various domains, including web development, data science, artificial intelligence, and more.

LIBRARIES

1) Tkinter:

Tkinter is the standard GUI (Graphical User Interface) library for Python. It provides tools for creating windows, buttons, labels, and other GUI elements.

SYNTAX:

```
from tkinter import *
root = Tk()
```

2) themed_tk (from ttkthemes):

themed_tk is an extension of Tkinter that provides additional themes for Tkinter applications. It enhances the visual appearance of GUI elements.

Syntax example:

```
from ttkthemes import themed_tk as tk
root = tk.ThemedTk()
```

3) filedialog (from tkinter):

The filedialog module, part of the Tkinter library, offers dialogs for file-related operations like opening and saving files

Syntax:

```
from tkinter import filedialog
filename = filedialog.askopenfilename()
```

4) mutagen.mp3:

mutagen.mp3 is a Python library for handling audio metadata, particularly for MP3 files. It allows for extracting information about the audio.

Syntax:

```
from mutagen.mp3 import MP3
audio = MP3("filename.mp3")
```

5) pygame.mixer:

pygame.mixer, part of the Pygame library, provides an interface to control sound playback. It's commonly used for playing and managing music in Python applications.

Syntax:

```
from pygame import mixer
mixer.init()
```

6) PIL (Python Imaging Library):

PIL, or the Python Imaging Library, is a library for opening, manipulating, and saving various image file formats. It is now succeeded by the Pillow library.

Syntax:

```
from PIL import Image
img = Image.open("image.jpg")
```

3.4 APPLICATION AND DESCRIPTION

3.1 Overview of the Various Parts

This project has several parts to it, but the most essential are three listed

in Table 1. **Table 1**: The overview of the three major parts of the shop

Administrators	**User**
Login access	*Cannot login*
Can add products	*Can add to cart*
Can edit products	*Can edit product in carts*
Can view products	*Cannot check out*
Can delete customer	*None*

3.2 Administrators Detailed Attribute

➢ **Admin register**

The administrator needs to register before they can have access to the core data of the playlist.

➢ **Admin login**

The admin logs in and can view, add music , manage customers.

➢ **Admin Edit**

The admin can make changes the code such as delete song, add a song or, upload new song.

➢ **Manage Customer**

The administrator has the authority to delete or add a song to playlist.

3.3 User Detailed Attribute

➢ **sign up**

This refers to registering as a customer. The registered member has a lot of privileges associated with the music when one becomes a customer.

➢ **Login**

After the user has registered, the user becomes a customer, and he or she can log in with their personal information.

➢ **View**

The customer can see all the playlist type in the catalog and able to look at the song and some features on the homepage.

➢ **Edit**

The customer can make changes to their data displayed on the customer page.

➢ **Update Cart**

This refers to putting or removing products from a playlist .

The Various Management Unit

The Administrators play the management role. They make sure everything in the shop runs smoothly. Table 2 lists the various management units.

Product Management Unit
This is the Unit that is responsible for keeping records, product name, description, price, products image and many others.
Customer Management Unit
This Unit involves some activities such the control all of the registered members, view all the members.
Admin login Unit
The Administrator can log in to the management webpage and make use of the features on the website such as adding product,view customers.
Payment View Unit
This enables the administrator to view all the bills made via the customers.
Admin logout Unit
The administrator will be able to logged out with this function

Table 2: Administrators management table

- **User Registration:**

Users will utilize their exclusive information for registration through a Python-based web application. After filling the form and submitting it, server-side Python code, possibly using a web framework like Flask or Django, checks the entered fields' correctness. If the data is incorrect, the user stays on the same page; otherwise, the information is stored in the "customers" table in a database. This transition signifies the user becoming a customer, and they are then directed to the shop's login webpage.

- **Customer Login:**

The customer, utilizing their email and password, logs into the shop through a Python-powered system. Upon form submission, server-side Python code, integrated with a web framework, validates the fields. If there's an issue, the customer remains on the login page; if not, the login information is verified against the database. Successful verification redirects the customer to their homepage, enabling them to browse products and proceed to checkout. The entire process is implemented using Python, leveraging libraries such as Flask or Django, replacing other languages mentioned in the original HTML5-based scenario.

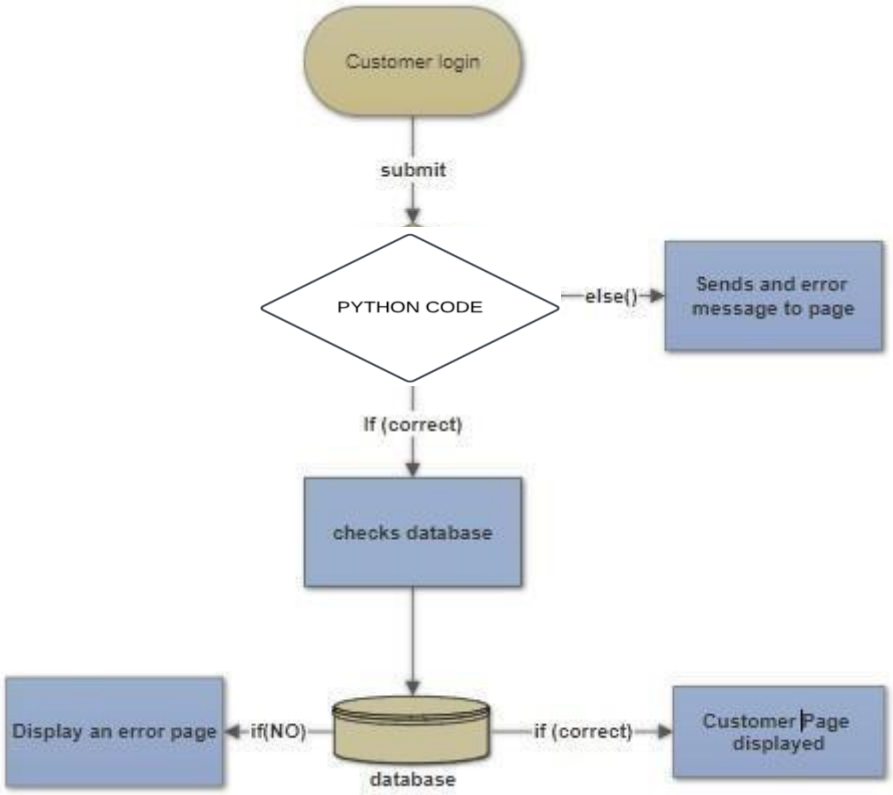


Figure 7: Customer login function

➢ +

➢ **Administrator Registration**

The Administrator will use his particular data such as name, e-mail, and password. After submitting the form, the Python code checks to see if all the fields entered by the administrator correct. If the filed are not correctly filed, or conditions are not met the admin remains on the same page but if all requirements are met admin's information goes to the database and saves the data in the "*Admins*" in the database. After that, the administrator is directed to the admin webpage to log in. The Figure 8 shows the administrator registration diagram.

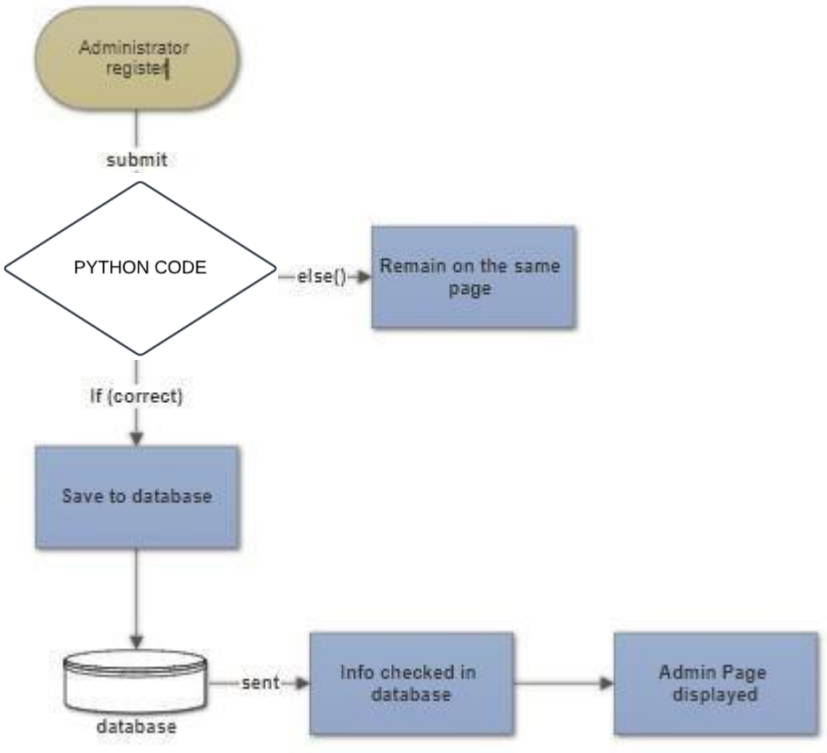


Figure 8: Administration registration diagram

➢ **Administrator Login**

When the admin logs into the Administrators webpage the Python code checks to see if conditions are met when logging in. If all the information provided are correct, the data is sent to the database to check if the data corresponds to the information used to register. If it matches to the information provided by the Administrator, a page opens, and the admin can have access to the administrator's webpage if not the administrator is restricted from having access to management webpage

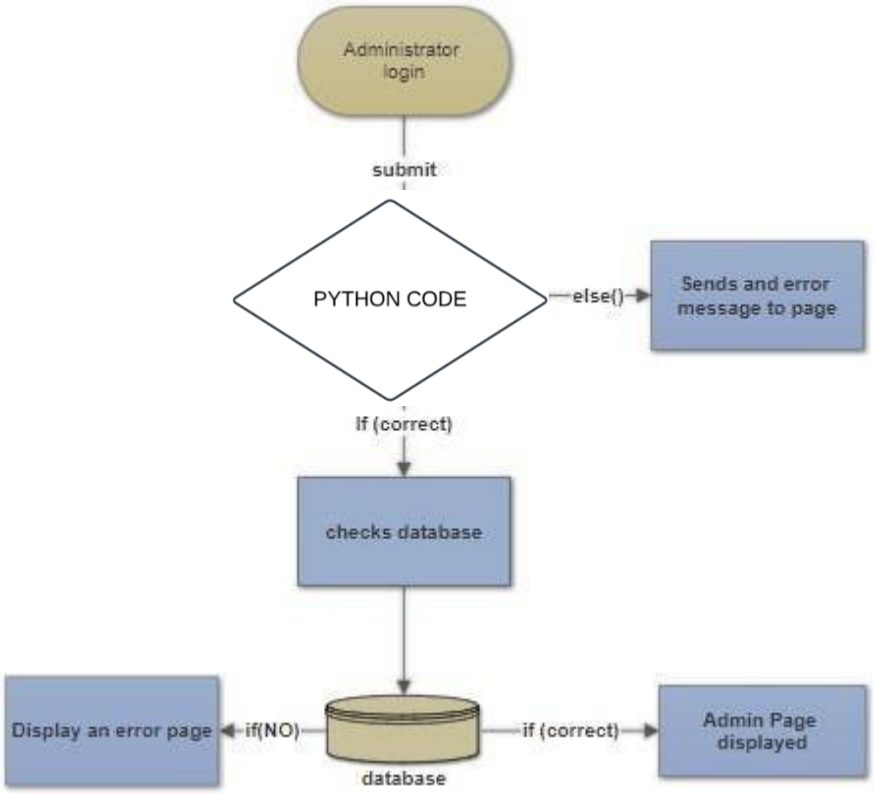


Figure 9: Administrator Login diagram

4 MVC UNIT OF PLAYLIST

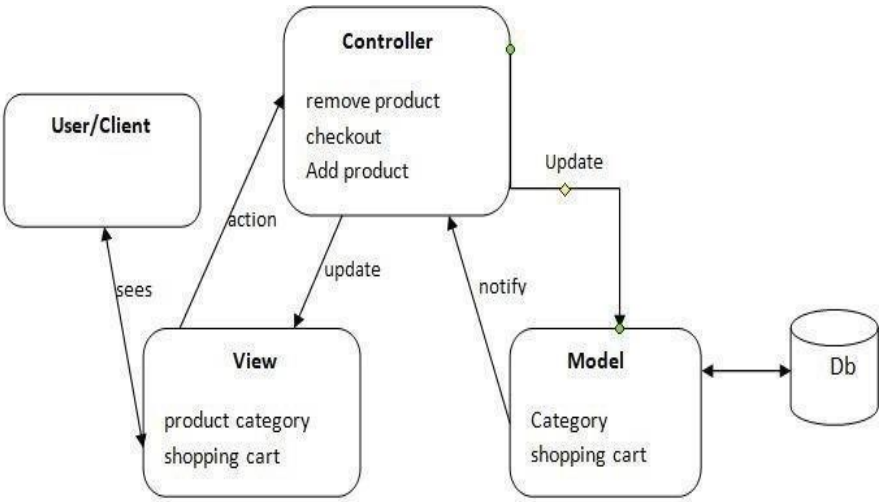


Figure 10: MVC diagram of the online shop

The three parts of the MVC software structure perform the following:

View - shows the interface that the person sees (usually, An GUI). The view additives provide records to the user and moves to the Controller for manipulating data.

Model - defines the statistics for the software (typically, the facts is saved in a data-base (DB)).

The controller provides the interface between the View and the model.

4.1 Back-end Module layout

This includes Units such as products, brand, category, orders and, customer management modules. Figure 11 shows the diagram of the back-end module

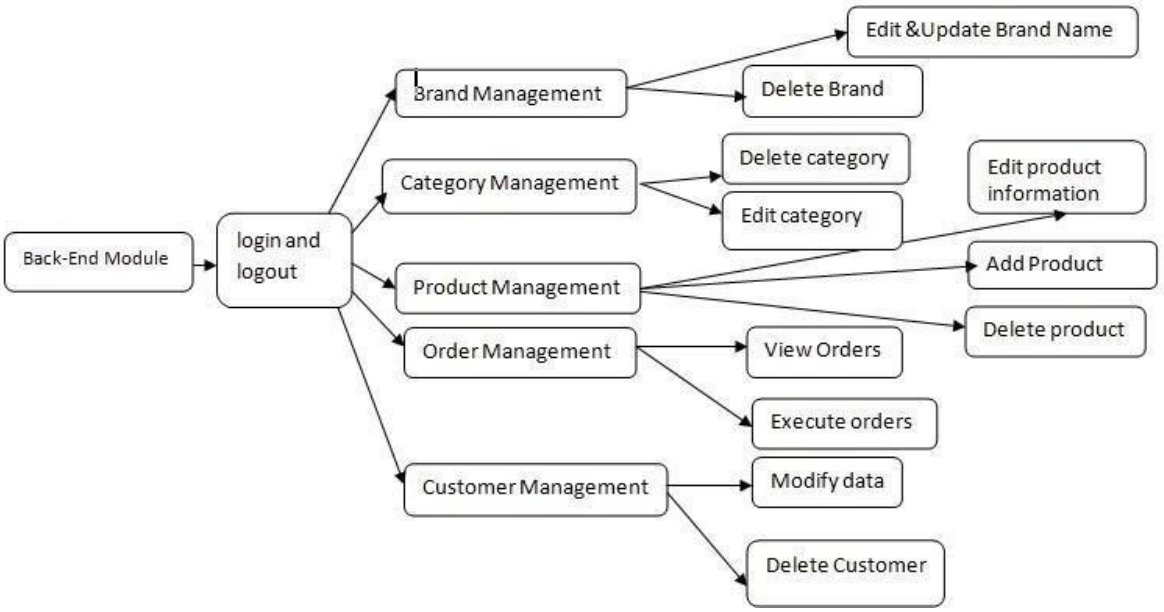
Figure 11: Back-end system management diagram.

The back-end module/Unit is used to manage the backend of the code . This is only available to the administrators. They can manipulate the code to suit the conditions they have set for the playlist. They also make sure that customers have a good experience when visiting the GUI based Music player by updating products, deleting products, executing, and managing playlist.

2 DATABASE DESIGN

5.1 Database

MySQL database is used to save software data for this project. MySQL is relational database management, and it is free of charge. All of the information is kept in a selected table, and every table has particular range columns and rows. It has eight tables named as admins, brands, cart, categories, customers, orders, payments and products. Figure 12 shows the ER-diagram of the eight tables in the database.



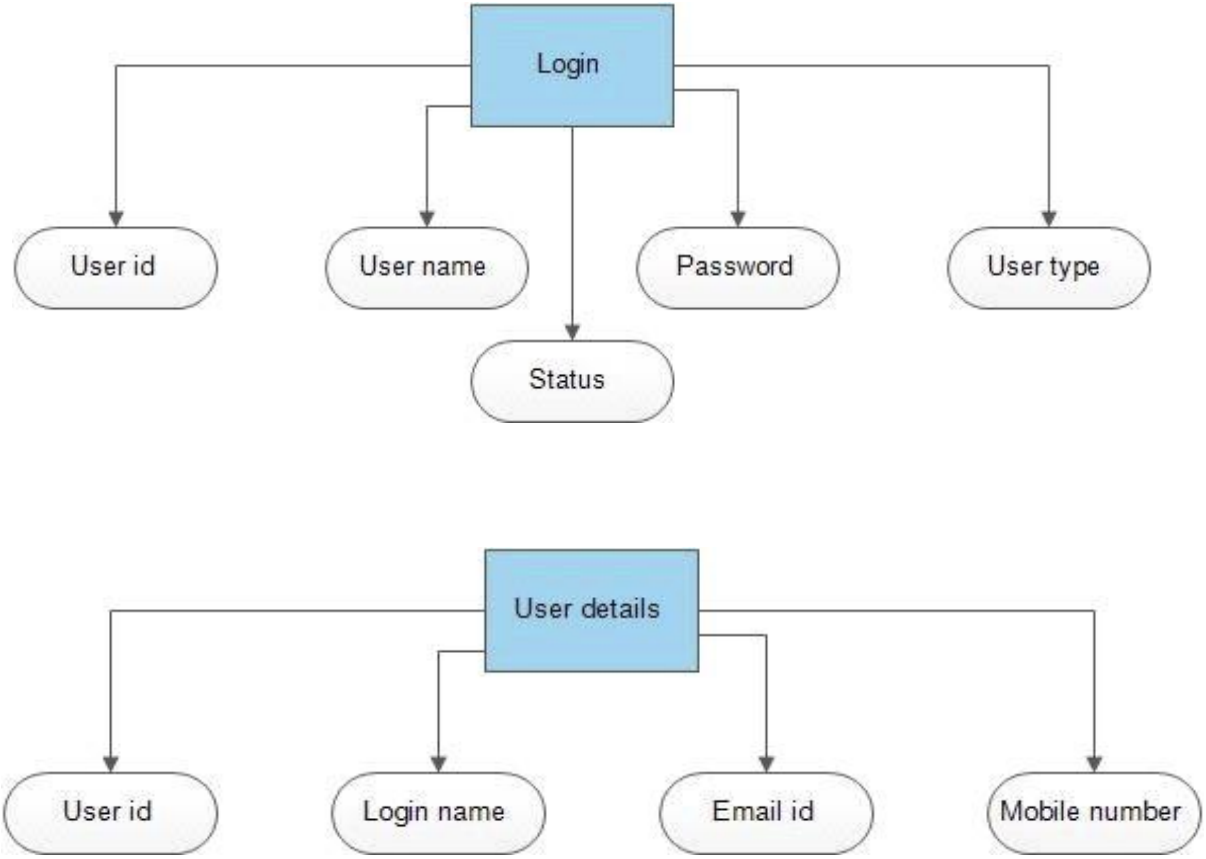
The customers who have registered to the online shop have their data automatically stored in the database. This information is only available to the technical administrators. The administrator can delete, edit, and update customer information.

5.2 Customer Interface layout

Consumer refers to customers and non-customers. These are individuals who visit the shop either to buy products or to browse. There are two categories of interfaces namely the Consumer and the Administrator interface. The administrator has higher authority over the customer in the shop. The admin can edit, replace a product and, manipulate data in the shop. The customer can browse a product, add a product to the cart, change personal information, check shopping history and checkout or log out. The User, on the other hand, can only browse and add a product to cart. The homepage or interface is the index page of the shop so can be accessed when the address is typed into a browser. The webpage has products images, names, prices, product categories and product brands. The webpage has a registration link, login link, cart, company contact information. A picture of the homepage is shown in Figure

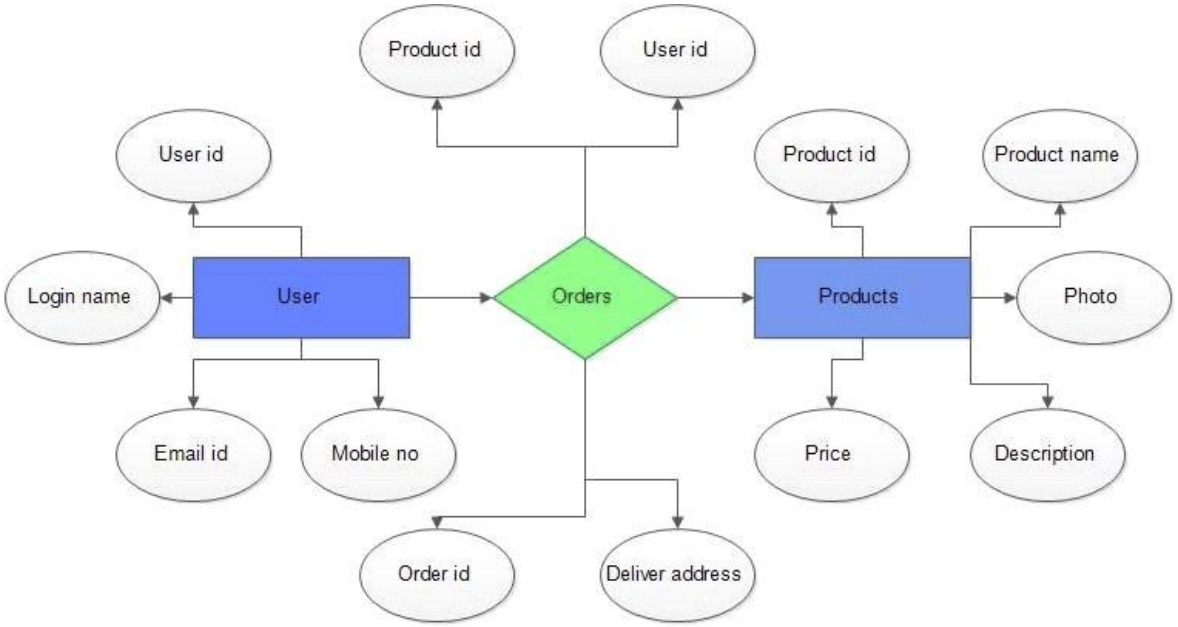
 PRODUCT DETAILS

Fig 3.4: Product Orders table



3.5 ER-DIAGRAM

 COMPLETE DIAGRAM



3.6 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a structured analysis and design tool that can be used for flowcharting. A DFD is a network that describes the flow of data and the processes that change or transform the data throughout a system. This network is constructed by using a set of symbols that do not imply any physical implementation. It has the purpose of clarifying system requirements and identifying major transformations. So it is the starting point of the design phase that functionally decomposes the requirements specifications down to the lowest level of detail.

DFD can be considered to an abstraction of the logic of an information-oriented or a process-oriented system flow-chart. For these reasons DFD's are often referred to as logical data flow diagrams.

EXTERNAL ENTITY

An external entity is a source or destination of a data flow. Only those entities which originate or receive data are represented on a data flow diagram. The symbol used is a rectangular box.

PROCESS

A process shows a transformation or manipulation of data flow within the system. The symbol used is an oval shape.

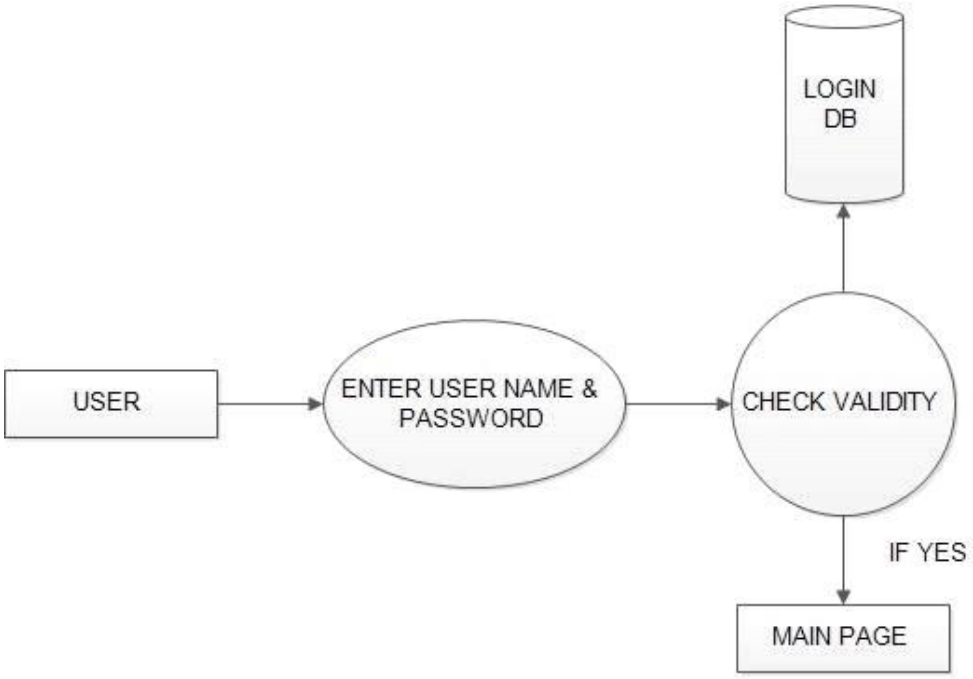
DATAFLOW

The data flow shows the flow of information from a source to its destination. Data flow is represented by a line, with arrowheads showing the direction of flow. Information always flows to or from a process and may be written, verbal or electronic. Each data flow may be referenced by the processes or data stores at its head and tail, or by a description of its contents.

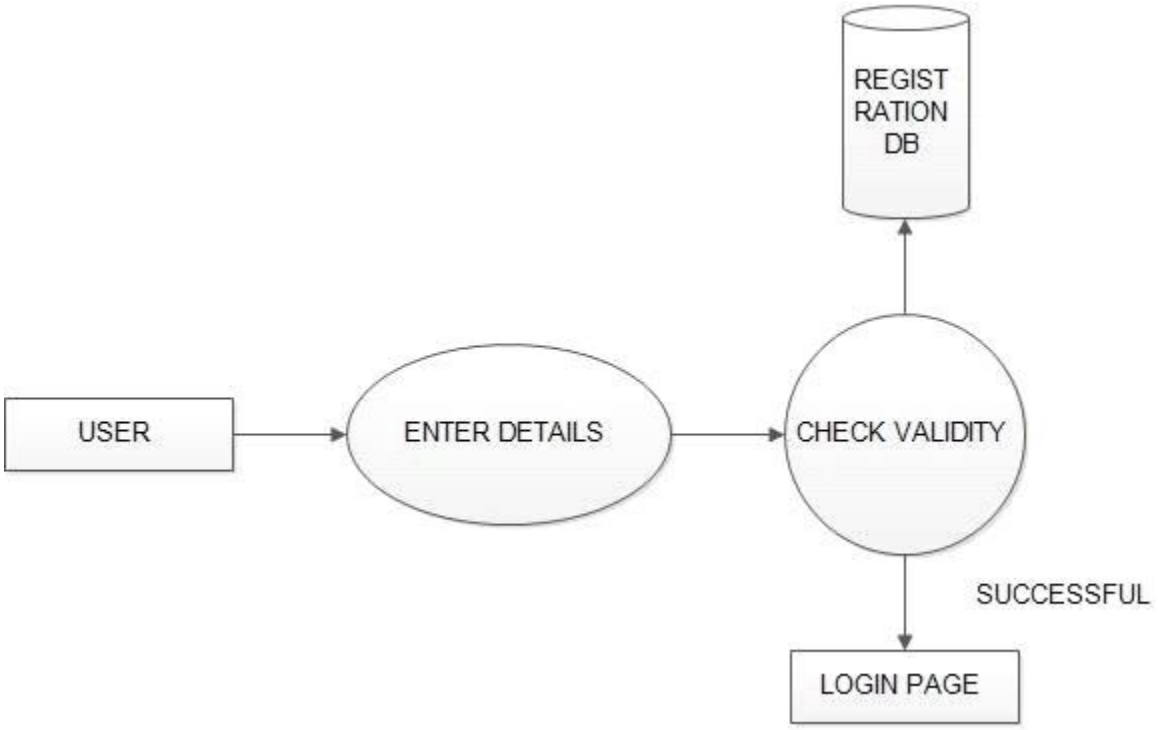
DATA STORE

A data store is a holding place for information within the system: It is represented by an open ended narrow rectangle. Data stores may be long-term files such as sales ledgers, or may be short-term accumulations: for example batches of documents that are waiting to be processed. Each data store should be given a reference followed by an arbitrary number.

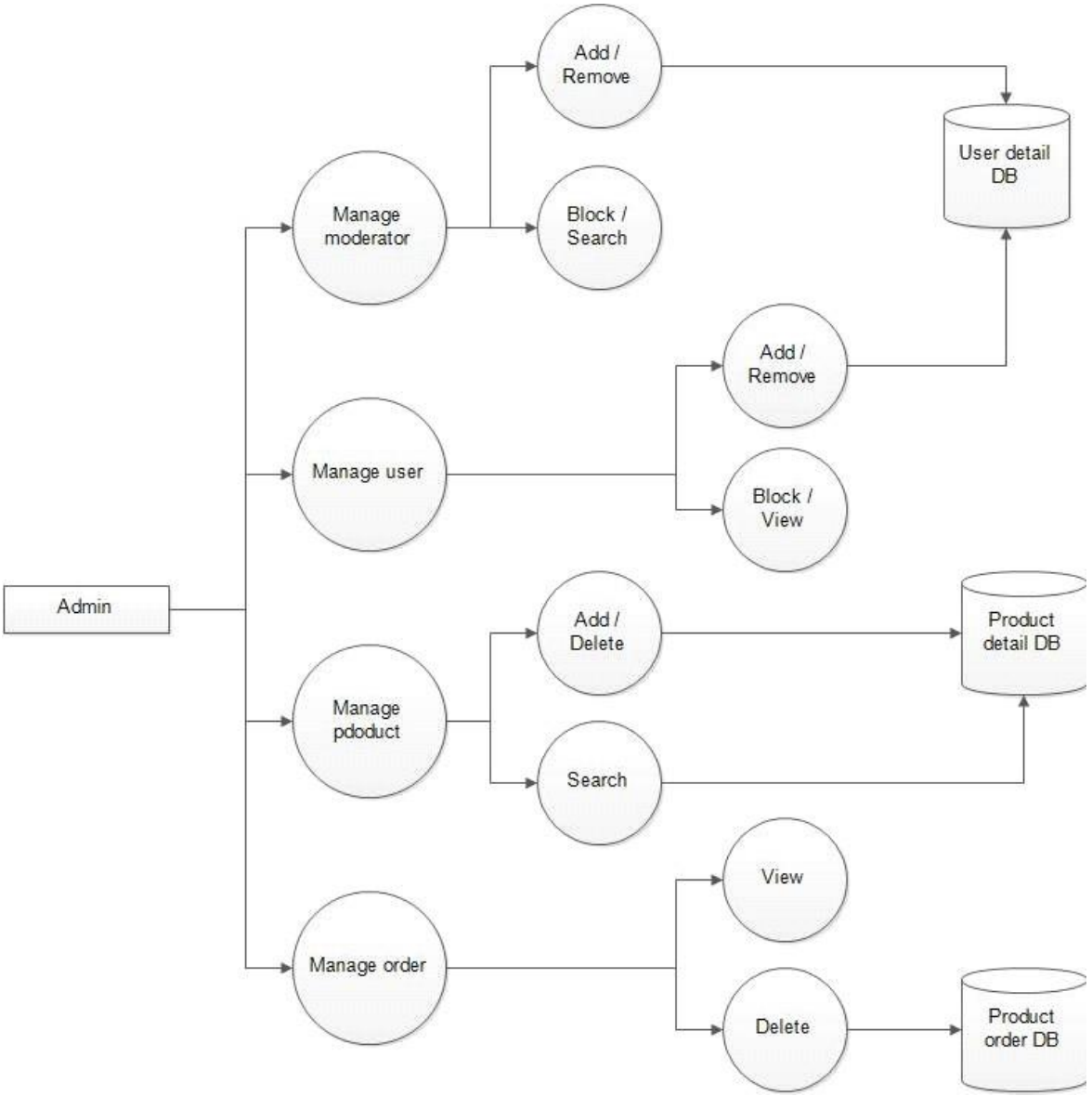
LOGIN DFD


LOGIN
DB
USER
ENTER USER NAME &
PASSWORD
CHECK VALIDITY
IF YES
MAIN PAGE

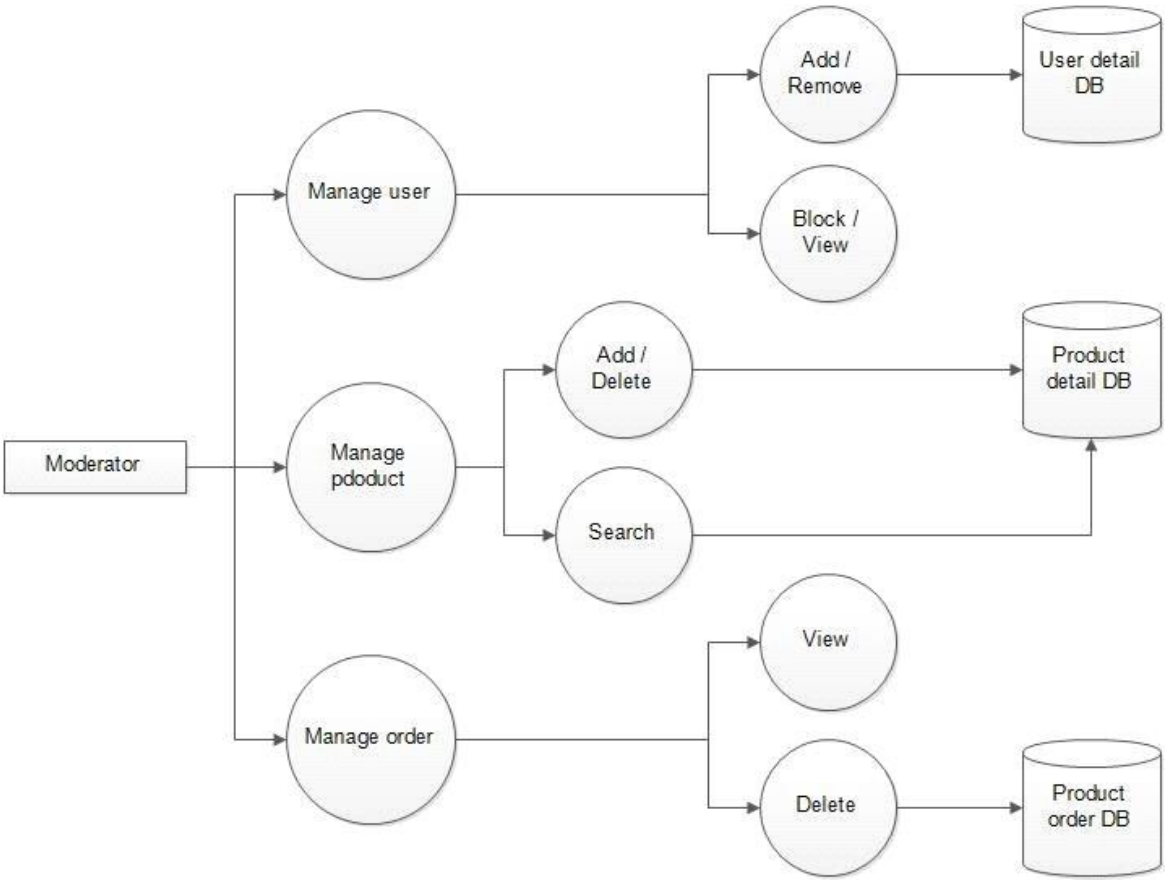
REGISTERATION DFD



➢ ADMIN DFD

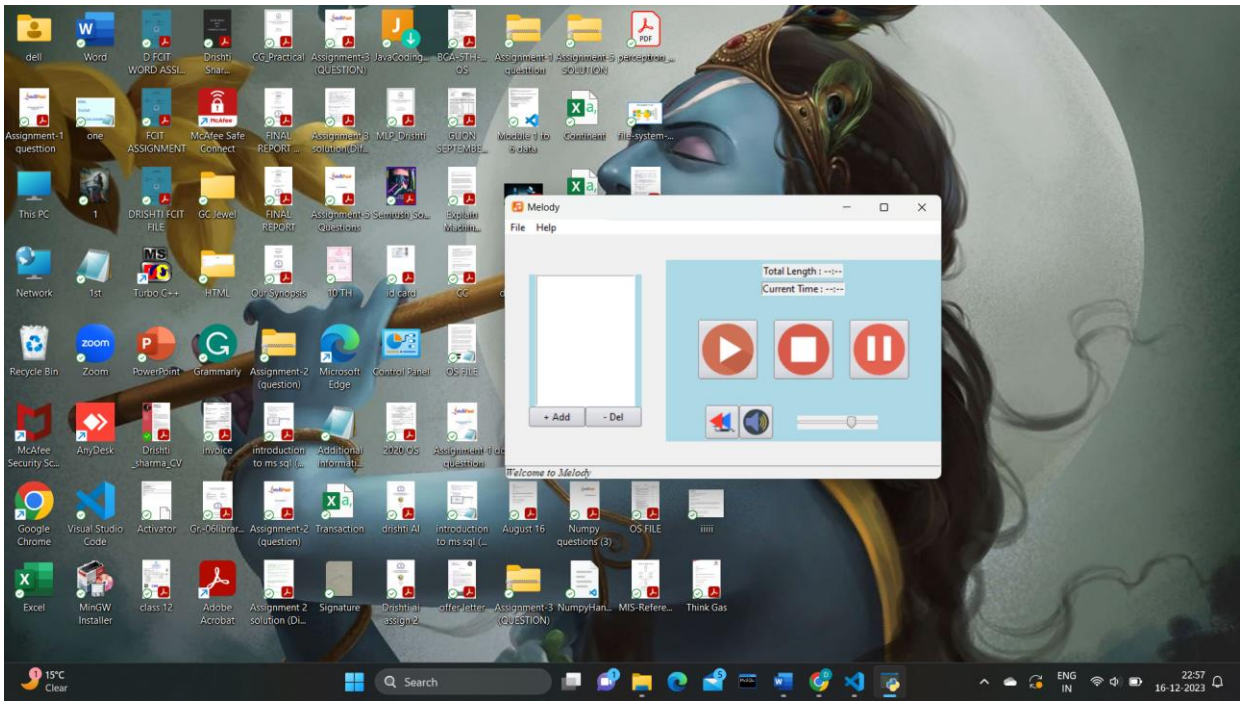

Admin
Manage moderator
Manage user
Manage pdoduct
Manage order
Add / Remove
Block / Search
Add / Remove
Block / View
Add / Delete
Search
View
Delete
User detail DB
Product detail DB
Product order DB

MODERATOR DFD

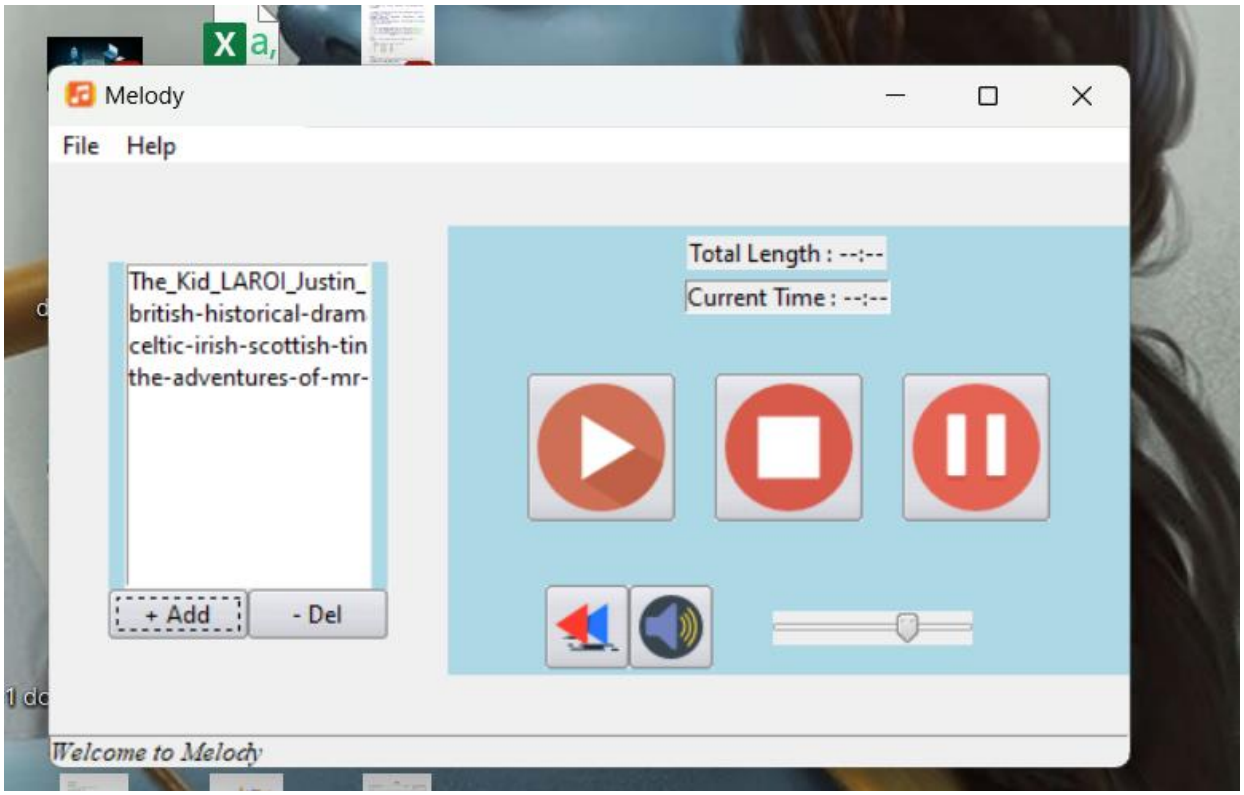


SCREENSHOTS

1) Melody offline usage on home screen



2) Melody playlist songs being added



3) Melody playlist song being Deleted

3.7 SAMPLE CODE

```
import os
import threading
import time
import tkinter.messagebox
from tkinter import *
from tkinter import filedialog
from tkinter import ttk
from ttkthemes import themed_tk as tk
from mutagen.mp3 import MP3
from pygame import mixer

root = tk.ThemedTk()
root.get_themes()
root.set_theme("plastik")  # Change the theme to 'plastik'

statusbar = ttk.Label(root, text="Welcome to Melody",
relief=SUNKEN, anchor=W, font='Times 10 italic')
statusbar.pack(side=BOTTOM, fill=X)

menubar = Menu(root)
root.config(menu=menubar)

subMenu = Menu(menubar, tearoff=0)

playlist = []


def browse_file():
    global filename_path
    filename_path = filedialog.askopenfilename()
    add_to_playlist(filename_path)
    mixer.music.queue(filename_path)
```

```
def add_to_playlist(filename):
    filename = os.path.basename(filename)
    index = 0
    playlistbox.insert(index, filename)
    playlist.insert(index, filename_path)
    index += 1


menubar.add_cascade(label="File", menu=subMenu)
subMenu.add_command(label="Open",
command=browse_file)
subMenu.add_command(label="Exit", command=root.destroy)


def about_us():
    tkinter.messagebox.showinfo('About Melody', 'This is a
music player build using Python Tkinter by @attreyabhatt')


subMenu = Menu(menubar, tearoff=0)
menubar.add_cascade(label="Help", menu=subMenu)
subMenu.add_command(label="About Us",
command=about_us)

mixer.init()

root.title("Melody")
root.iconbitmap(r'images/melody.ico')

leftframe = Frame(root, bg='#ADD8E6')  # Light blue
background color
leftframe.pack(side=LEFT, padx=30, pady=30)
```

```
playlistbox = Listbox(leftframe)
playlistbox.pack()

addBtn = ttk.Button(leftframe, text="+ Add",
command=browse_file)
addBtn.pack(side=LEFT)


def del_song():
    selected_song = playlistbox.curselection()
    selected_song = int(selected_song[0])
    playlistbox.delete(selected_song)
    playlist.pop(selected_song)


delBtn = ttk.Button(leftframe, text="- Del",
command=del_song)
delBtn.pack(side=LEFT)

rightframe = Frame(root, bg='#ADD8E6')  # Light blue
background color
rightframe.pack(pady=30)

topframe = Frame(rightframe, bg='#ADD8E6')  # Light blue
background color
topframe.pack()

lengthlabel = ttk.Label(topframe, text='Total Length : --:--')
lengthlabel.pack(pady=5)

currenttimelabel = ttk.Label(topframe, text='Current Time : --:--
', relief=GROOVE)
currenttimelabel.pack()
```

```
def show_details(play_song):
    file_data = os.path.splitext(play_song)
    if file_data[1] == '.mp3':
        audio = MP3(play_song)
        total_length = audio.info.length
    else:
        a = mixer.Sound(play_song)
        total_length = a.get_length()

    mins, secs = divmod(total_length, 60)
    mins = round(mins)
    secs = round(secs)
    timeformat = '{:02d}:{:02d}'.format(mins, secs)
    lengthlabel['text'] = "Total Length" + ' - ' + timeformat

    t1 = threading.Thread(target=start_count,
args=(total_length,))
    t1.start()


def start_count(t):
    global paused
    current_time = 0
    while current_time <= t and mixer.music.get_busy():
        if paused:
            continue
        else:
            mins, secs = divmod(current_time, 60)
            mins = round(mins)
            secs = round(secs)
            timeformat = '{:02d}:{:02d}'.format(mins, secs)
            currenttimelabel['text'] = "Current Time" + ' - ' +
```

```
timeformat
            time.sleep(1)
            current_time += 1




def play_music():
    global paused
    if paused:
        mixer.music.unpause()
        statusbar['text'] = "Music Resumed"
        paused = FALSE
    else:
        try:
            stop_music()
            time.sleep(1)
            selected_song = playlistbox.curselection()
            selected_song = int(selected_song[0])
            play_it = playlist[selected_song]
            mixer.music.load(play_it)
            mixer.music.play()
            statusbar['text'] = "Playing music" + ' - ' +
os.path.basename(play_it)
            show_details(play_it)
        except:
            tkinter.messagebox.showerror('File not found', 'Melody
could not find the file. Please check again.')


def stop_music():
    mixer.music.stop()
    statusbar['text'] = "Music Stopped"
```

```
paused = FALSE




def pause_music():
    global paused
    paused = TRUE
    mixer.music.pause()
    statusbar['text'] = "Music Paused"



def rewind_music():
    play_music()
    statusbar['text'] = "Music Rewinded"


def set_vol(val):
    volume = float(val) / 100
    mixer.music.set_volume(volume)


muted = FALSE


def mute_music():
    global muted
    if muted:
```

```
        mixer.music.set_volume(0.7)
        volumeBtn.configure(image=volumePhoto)
        scale.set(70)
        muted = FALSE
    else:
        mixer.music.set_volume(0)
        volumeBtn.configure(image=mutePhoto)
        scale.set(0)
        muted = TRUE


middleframe = Frame(rightframe, bg='#ADD8E6')  # Light blue
background color
middleframe.pack(pady=30, padx=30)

playPhoto = PhotoImage(file='images/play.png')
playBtn = ttk.Button(middleframe, image=playPhoto,
command=play_music)
playBtn.grid(row=0, column=0, padx=10)


stopPhoto = PhotoImage(file='images/stop.png')
stopBtn = ttk.Button(middleframe, image=stopPhoto,
command=stop_music)
stopBtn.grid(row=0, column=1, padx=10)

pausePhoto = PhotoImage(file='images/pause.png')
pauseBtn = ttk.Button(middleframe, image=pausePhoto,
command=pause_music)
pauseBtn.grid(row=0, column=2, padx=10)

bottomframe = Frame(rightframe, bg='#ADD8E6')  # Light
blue background color
bottomframe.pack()
```

```
rewindPhoto = PhotoImage(file='images/rewind.png')
rewindBtn = ttk.Button(bottomframe, image=rewindPhoto,
command=rewind_music)
rewindBtn.grid(row=0, column=0)

mutePhoto = PhotoImage(file='images/mute.png')
volumePhoto = PhotoImage(file='images/volume.png')
volumeBtn = ttk.Button(bottomframe, image=volumePhoto,
command=mute_music)
volumeBtn.grid(row=0, column=1)

scale = ttk.Scale(bottomframe, from_=0, to=100,
orient=HORIZONTAL, command=set_vol)
scale.set(70)
mixer.music.set_volume(0.7)
scale.grid(row=0, column=2, pady=15, padx=30)




def on_closing():
    stop_music()
    root.destroy()


root.protocol("WM_DELETE_WINDOW", on_closing)
root.mainloop()
```

CHAPTER 4

CONCLUSION

The project entitled MELODY -MUSIC PLAYER was completed successfully.

In conclusion, the development of "Melody" as a music player for Windows has been a transformative minor project, aiming to redefine the digital music listening experience on the Windows platform. The journey commenced with a clear objective: to create a user-friendly and feature-rich application that caters to the diverse needs of music enthusiasts. As the project unfolded, various key components were meticulously integrated to achieve this vision.

Melody's playlist management system emerged as a cornerstone, empowering users to effortlessly organize and curate their music collections. Supporting a myriad of audio formats ensures compatibility with a wide range of music files, offering users the flexibility to enjoy their favorite tunes without constraint. The robust audio playback engine incorporated into Melody elevates the auditory experience, delivering high-quality sound reproduction.

The project also emphasized the importance of a responsive and customizable interface. Melody's visually appealing design not only enhances the aesthetic appeal but also provides users with the ability to tailor the application to their preferences. The search and sorting mechanisms enhance the efficiency of navigating through extensive music libraries, ensuring a seamless and enjoyable user experience.

Performance optimization has been a continuous focus throughout the project, with the goal of providing users with a smooth and responsive application, even when handling extensive music libraries. Compatibility across various Windows versions has been a priority, aiming to reach a broad user base and ensuring accessibility to a diverse audience.

As Melody reaches its culmination, the project's success lies not only in the creation of a sophisticated music player but also in the establishment of a foundation for potential future enhancements. User documentation has been prepared to guide users through the application's features, and considerations for ongoing maintenance and updates based on user feedback are integral to the project's conclusion.

In essence, Melody stands as a testament to the fusion of simplicity and sophistication, offering Windows users a versatile and enduring music player that transcends conventional expectations. The project's conclusion marks the beginning of a new chapter, where Melody takes its place as a reliable companion in the digital realm of music appreciation.