A

Project Report

On

Moodify (MP3 player)

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

BACHELORS IN COMPUTER SCIENCE AND ENGINEERING

(ARTIFICIAL INTELLIGENCE and MACHINE LEARNING)

(Session-2023-2027)

Submitted To:	Submitted by:
Mr. Paramjeet Singh	Ishmeet Kaur & Ajay
HOD	CSE(AIML) 5th Sem.
Dept. of Computer Sci. & Engineering (AI ML)	231950019 & 231950002

MAHARAJA RANJIT SINGH PUNJAB TECHNICAL UNIVERSITY, BATHINDA

PREFACE

In the age of digital transformation, music has become more than just entertainment—it is a way of life. With evolving user expectations, the demand for intelligent and interactive media players has seen an exponential rise. Traditional MP3 players offer basic functionalities but often lack intelligent features and appealing interfaces. In light of this, the development of an enhanced and user-centric audio player application named "Moodify MP3 Player" is proposed.

Moodify is designed not only to play music but to enhance user experience through an interactive, visually appealing, and mood-responsive interface. The fusion of backend logic in Python and frontend technology like Python, Tkjinter, SQLite3 makes Moodify a hybrid solution that caters to modern user needs. This document outlines the rationale, objectives, and technical components of the Moodify MP3 Player project in detail.

ACKNOWLEDGEMENT

We take this opportunity to express our deepest gratitude to those who have generously helped us in providing the valuable knowledge and it is with real pleasure, that we record our indebtedness to our academic guide, ***Mr. Paramjeet Singh***, HOD of Computer Science Engineering (AI ML) for his counsel and guidance during the preparation of this project and other staff of Maharaja Ranjit Singh Punjab Technical University, Bathinda.

I would like to take this opportunity to convey my deep sense of gratitude to my External guide ***Ms. Neha***, for her direction, support and steady motivation. I am extremely obliged to her for her continuous suggestions, help and cooperation, without which it was impossible to achieve the success in the work allocated to me.

At the end I would love to thank all of staff members of Softwizz Technologies Pvt. Ltd. for providing an excellent and healthy environment during work.

And finally, we would like to thank each and every person who has contributed in any of the ways in our project.

Ishmeet Kaur (231950019)

Ajay (231950002)

INDEX

CHAPTER 1

COMPANY PROFILE

About Softwizz

Softwizz is a software development company providing solutions in the field of education, construction, publishing and many more. Our specialists are truly sensitive and responsive to the needs of our clients due to their unwavering dedication and unequaled professionalism, which help our company to deliver wide-ranging and proficient custom web design services.

Softwizz Pvt. Ltd. is leading software development and training group in various domains across the industry like CSE IT, Non IT (ECE, ME, ETC, EEE) and Management based programs. We are in software development industry, corporate training as well as individual training program to meet our market segments.

Softwizz Pvt. Ltd. customized program is to provide the training Software industry with project ready candidates by filling the gap between theoretical knowledge and industry requirement.

Vision:

To be the most admired and respected Technology company providing "Best value" software solution and education system.

Mission:

To achieve the leadership position in our focus domains and to become a reason for smile for everyone related with the organization in the form of staff, clients and trainees by providing the best services with all the commitment and dedication of our work.

Core Value:

Clients first- We exist because of our clients.

Reliability and transparency- committed to be ethical, transparent and reliable in all our operation.

Services:

SOFWIZZ offers a wide range of Engineering Projects and Information Technology services. These include:

- Software Development Process
- Software Development Quality
- Software Development Solutions
- Social Media
- Ecommerce Solution
- Web Hosting <u>Approach</u>

SOFTWIZZ sees teamwork on every project as the key success ingredient and are responsible for creating this environment. The overall project manager is duty-bound in ensuring effective communication to all stakeholders and at all levels. Teams are built by combining the strengths of the individual members and refining skills to meet and exceed the Client's expectations.

We also believe in a hands-on approach on all our projects. This is why the Director of the firm will always be in control of the key functions on a project. SOFTWIZZ is equipped with the latest technology and has the necessary staff and resources to ensure that best professional service is provided at all times.

CHAPTER 2

INTRODUCTION OF PROJECT

2.1 INTRODUCTION

About the Project:

The Moodify MP3 Player is an intelligent, user-centric desktop application designed to enhance the way users interact with music. Built using Python and the Tkinter library, this multimedia tool not only offers a clean and responsive interface but also integrates emotional intelligence by adapting playlists based on the user's current mood.

The idea behind Moodify stems from the belief that music is deeply connected to human emotion. Unlike traditional media players that offer static playback, Moodify brings a personalized listening experience where users can select their mood—such as Happy, Sad, Energetic, Calm, Focused, etc.—and instantly receive a curated list of songs that match that emotional state.

Through features like mood-based song filtering, login/signup functionality, user-specific preferences, and potential integration with platforms like YouTube for online streaming, Moodify transforms simple audio playback into a dynamic and interactive emotional companion.

This document outlines the motivation, features, design, and technical implementation behind the Moodify MP3 Player and explores how it aims to bridge the gap between emotion and technology.

2.2 OBJECTIVE OF THE PROJECT

To develop a user-friendly MP3 player that provides essential and advanced music playback functionalities.

- To implement a mood-based music selection system that recommends songs based on the user’s emotional state.
- To create an interactive and visually appealing GUI using Python’s Tkinter library.
- To integrate login and signup functionalities for personalized user experiences and data security.
- To enable Spotify song streaming for a dynamic and extensive music library (optional/extended feature).
- To provide customized themes or UI styles that reflect the user's mood and enhance engagement.
- To maintain a lightweight and responsive application suitable for low-end systems without compromising performance.
- To help users improve their emotional well-being by offering music as a therapeutic aid through curated mood-based playlists.

2.3 FUNCTIONALITIES

• Mood-Based UI Themes

○ Manually select moods (happy, sad, relaxed, energetic) to change the interface theme and display matching song categories.

• Login/Signup with SQLite Database

○ Secure user authentication using SQLite to store and manage user credentials.

• User Greeting with Username

○ Displays the logged-in user's name on the top navigation bar to personalize the experience.

• Image-Based UI Interface

○ Use of themed images and buttons (such as "close eyes", "welcome", etc.) to create an engaging visual experience.

• Fixed Song Categories

○ Songs are pre-organized into mood-specific folders and presented accordingly when a mood is selected.

• Tkinter GUI Application

○ Fully developed desktop interface using Python’s Tkinter library for all windows, buttons, and navigation.

CHAPTER 3
REQUIREMENT ANALYSIS

This process is also known as feasibility study. In this phase, the development team visits the customer and studies their system. They investigate the need for possible software automation in the given system. By the end of the feasibility study, the team furnishes a document that holds the different specific recommendations for the candidate system. To understand the nature of the program(s) to be built, the system engineer or "Analyst" must understand the information domain for the software, as well as required function, behavior, performance and interfacing.

Requirements analysis in systems engineering and software engineering, encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product, taking account possibly conflicting requirements of the various stakeholders, analyzing, documenting, validating and managing software or system requirements.

Requirements analysis is critical to the success of a systems. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

Types of Requirements

There are some common type of software requirements are used as:

1. Customer Requirements:

Statements of fact and assumptions that define the expectations of the system in terms of mission objectives, environment, constraints, and measures of effectiveness and suitability (MOE/MOS). The customers are those that perform the eight primary functions of systems engineering, with special emphasis on the operator as the key customer. Operational requirements will define the basic need and, at a minimum, answer the questions posed in the following listing:

- Operational distribution or deployment: Where will the system be used?
- Mission profile or scenario: How will the system accomplish its mission objective?
- Performance and related parameters: What are the critical system parameters to accomplish the mission?
- Utilization environments: How are the various system components to be used?
- Effectiveness requirements: How effective or efficient must the system be in performing its mission?
- Operational life cycle: How long will the system be in use by the user?
- Environment: What environments will the system be expected to operate in an effective manner?

2. Architectural Requirements:

Architectural requirements explain what has to be done by identifying the necessary system architecture of a system.

3. Structural Requirements:

Structural requirements explain what has to be done by identifying the necessary structure of a system.

4. Behavioural Requirements:

Behavioural requirements explain what has to be done by identifying the necessary behaviour of a system.

5. Functional Requirements:

Functional requirements explain what has to be done by identifying the necessary task, action or activity that must be accomplished. Functional requirements analysis will be used as the toplevel functions for functional analysis.

6. Non-functional Requirements:

Non-functional requirements are requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviors.

7. Performance Requirements:

The extent to which a mission or function must be executed; generally measured in terms of quantity, quality, coverage, timeliness or readiness. During requirements analysis, performance (how well does it have to be done) requirements will be interactively developed across all identified functions based on system life cycle factors; and characterized in terms of the degree of certainty in their estimate, the degree of criticality to system success, and their relationship to other requirements.

8. Design Requirements:

The "build to," "code to," and "buy to" requirements for products and "how to execute" requirements for processes expressed in technical data packages and technical manuals

3.1 PROBLEM ANALYSIS

Problem Statement

- **Lack of Personalization:**

  Most traditional MP3 players do not provide features that adapt based on the user's mood or preferences.

- **Outdated User Interface:**

Many existing music players have outdated designs that are not optimized for modern responsive screens or user engagement.

- **Limited Interactivity:**
  Conventional MP3 players focus solely on play/pause functionalities and do not include interactive visual elements like animations or mood-based themes.
- **Poor User Experience:**
  Audio players often lack playlist management, intuitive navigation, or responsive layouts, especially for beginners.
- **No Mood Integration:**
  There is no option in standard players to reflect or adapt to the emotional state or listening behavior of users.
- **Limited File Compatibility and Handling:**
  Users often face errors or crashes when trying to load unsupported or corrupt audio files.

Proposed System

- **Modern and Responsive UI:**
  Built with Python, Tkinter, SQLite3 for a clean, mobile-friendly, and attractive interface.
- **Python-Integrated Backend**:
  Uses Python libraries (like tkinter, PIL, webbrowser, sqlite3) to handle music playback, file management, and audio processing.
- **Mood Based Song Categorization:**
  Allows users to select their current mood(e.g.,Happy, Sad, Calm, Energetic,etc.) and view playlists accordingly.
- **Personalized User Greeting:**
  Displays the user's name after login, making the experience more personalized.
- **User Authentication with SQLite3:**
  Provides secure login and signup functionality using an integrated SQLite database.
- **Interactive Graphical Interface:**
  GUI is built using Tkinter with visually engaging buttons and Image-based elements.

- **Predefined Playlist Integration:**
  Instead of live streaming or dynamic song fetching, the app uses predefined Spotify links categorized by mood.
- **Simple, Lightweight desktop App:**
  No internet connection required for core functionality; runs smoothly on any basic system.
- **Expandable Mood Options:**
  Designed to support new moods in the future like Focus, Dreamy etc..
- **Modular Code Structure:**
  Organized code with separate functions for login, mood selection, playlist windows etc., making it easy to maintain and expand.

3.2 REQUIREMENT SPECIFICATION DOCUMENT

3.2.1 SOFTWARE REQUIREMENT SPECIFICATION

A software requirements specification (SRS) is a complete description of the behavior of the system to be developed. It includes a set of use cases that describe all of the interactions that the users will have with the software. Use cases are also known as functional requirements. In addition to use cases, the SRS also contains nonfunctional (or supplementary) requirements. Non-functional requirements are requirements that impose constraint on the design or implementation (such as performance requirements, quality standard or design constraint).

In system engineering and software engineering, requirements analysis encompasses those tasks that go into determining the requirements of a new or altered system, taking account of the possibly conflicting requirements of the various stakeholders, such as users. Requirements analysis is critical to the success of project. The document that contains all the requirements of the project is determined as “Software Requirements Specification”.

Software requirements specification states the goals and objectives of the software, describing it in the context of the computer based system.

The Information Description provides a detailed description of the problem that the software must solve. Information content, flow and structure are documented.

A description of each function required to solve the problem is presented in the Functional Description. Validation Criteria is probably the most important and ironically the most often neglected section of the software requirement specification.

An SRS minimizes the time and effort required by developers to achieve desired goals and also minimizes the development cost. A good SRS defines how an application will interact with system hardware, other programs and human users in a wide variety of real-world situations. Parameters such as operating speed, response time, availability, portability, maintainability, footprint, security and speed of recovery from adverse events are evaluated. Methods of defining an SRS are described by the IEEE (Institute of Electrical and Electronics Engineers) specification 830-1998.

3.2.2 SPECIFIC REQUIREMENTS

1. **Processing Requirements**

In this step we analyzed the processing capabilities of the system on which the proposed system would be developed and the specification is divided into two categories namely:

 Minimum Hardware Requirements

1	System Type	64-bit operating System
2	RAM	4 GB
3	Hard Disk	1 TB of free HDD space for Internet Cache.

4	Internet Connection	512Kbps
5	Processor Speed	5.2 GHz

 Minimum Software Requirements

3.2.4 TECHNOLOGY USED

I.	Operating System		
III.	Back End		
V.	Designing Software		
V1.	Framework		

3.2.4.1 PYTHON

- Python is an object-oriented, high-level programming language with integrated dynamic semantics primarily for web and app development. It is extremely attractive in the field of Rapid Application Development because it offers dynamic typing and dynamic binding options.
- Python is relatively simple, so it's easy to learn since it requires a unique syntax that focuses on readability.
- One of the most promising benefits of Python is that both the standard library and the interpreter are available free of charge, in both binary and source form.

- Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast.

History:

- Python was originally conceptualized by Guido van Rossum in the late 1980s as a member of the National Research Institute of Mathematics and Computer Science in Netherland.
- Python 0.9.0 was first released in 1991.
- In 1994, Python 1.0 was released with new features like: lambda, map, filter, and reduce.
- In 2000, Python 2.0 was released. This version was more of an open-source project with new features like: list comprehensions, garbage collection system.
- Python 3.0 was the next version and was released in December of 2008 (the latest version of Python is 3.6.4).
- ABC programming language is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.
- The name "Python" was adopted from the comedy series "Monty Python's Flying Circus".

Features:

- Simple and elegant syntax which is easier to learn : Easy to read and write Python programs compared to other languages.

- Free and open source : Can freely use and distribute Python.

- Portability : Can move Python programs from one platform to another.

- Extensible and Embeddable : Can easily combine pieces of C, C++ or other languages with Python code.

- High-level Interpreted Language : Don’t have to worry about memory management or garbage collection.

- Large Standard Libraries to solve common tasks : Like MySQLdb to connect to database.

- Object-Oriented : With OOP, you can divide complex problems into smaller sets by creating objects.

3.2.4.2 **TKINTER**

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the Tkinter module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table −

Sr. No.	Operator & Description
1	Bu on The Button widget is used to display buttons in your application.
2	Canvas The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.

3	Checkbu on The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.
4	Entry The Entry widget is used to display a single-line text field for accepting values from a user.
5	Frame The Frame widget is used as a container widget to organize other widgets.
6	Label The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
7	Listbox The Listbox widget is used to provide a list of options to a user.
8	Menubu on The Menubutton widget is used to display menus in your application.

9	Menu The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.
10	Message The Message widget is used to display multiline text fields for accepting values from a user.
11	Radiobu on The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.
12	Scale The Scale widget is used to provide a slider widget.
13	Scrollbar
	The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.

14	Text The Text widget is used to display text in multiple lines.
15	Toplevel The Toplevel widget is used to provide a separate window container.
16	Spinbox The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.
17	PanedWindow A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.
18	LabelFrame A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
19	tkMessageBox This module is used to display message boxes in your applications.

3.2.4.3 SQLite3 Database

(used in backend)

SQLite is a C-language library that implements a small, fast, self-contained, highreliability, full-featured, SQL database engine. SQLite is the most used database engine in the world. SQLite is built into all mobile phones and most computers and comes bundled inside countless other applications that people use every day.

The SQLite file format is stable, cross-platform, and backwards compatible and the developers pledge to keep it that way through the year 2050.

SQLite database files are commonly used as containers to transfer rich content between systems and as a long-term archival format for data .

There are over 1 trillion SQLite databases in active use.

SQLite source code is in the public-domain and is free to everyone to use for any purpose.

CHAPTER 4
SYSTEM ANALYSIS

INTRODUCTION

System analysis is the process of studying the business processors and procedures, generally referred to as business systems, to see how they can operate and whether improvement is needed. This may involve examining data movement and storage, machines and technology used in the system, programs that control the machines, people providing inputs, doing the processing and receiving the outputs.

INVESTIGATION PHASE

The investigation phase is also known as the fact-finding stage or the analysis of the current system. This is a detailed study conducted with the purpose of wanting to fully understand the existing system and to identify the basic information requirements. Various techniques may be used in factfinding and all fact obtained must be recorded. A thorough investigation was done in every effected aspect when determining whether the purposed system is feasible enough to be implemented.

Investigation

As it was essential for us to find out more about the present system, we used the following methods to gather the information: -

1. Observation: -Necessary to see the way the system works first hand.
2. Document sampling: - These are all the documents that are used in the system. They are necessary to check all the data that enters and leaves the system.
3. Questionnaires: - These were conducted to get views of the other employees who are currently employed in the system.

System Security

System security is a vital aspect when it comes to developing a system. The system should ensure the facility of preventing unauthorized personnel from accessing the information and the data within the system. The system should provide total protection for each user's information so that the integrity of data is sustained and also prevent hackers from hacking the system.

The proposed system ensures the security and the integrity of data. This is done by providing a password login system . And for example the System Administrator has access to all kinds of information.

By providing this facility information is properly managed and information is protected.

FEASIBILTY STUDY

Feasibility study is done so that an ill-conceived system is recognized early in definition phase. During system engineered, however we concentrate our attention on four primary areas of interest. This phase is really important as before starting with the real work of building the system it is very important to find out whether the idea thought is feasible or not.

- Economic Feasibility: An evaluation of development cost weighted against the ultimate income or benefit derived from the developed system.
- Technical Feasibility: A study of function, performance and constraints that may affect the ability to achieve an acceptable system.
- Operational Feasibility: A study about the operational aspects of the system.

ECONOMIC ANALYSIS

Among the most important information contained in feasibility study is Cost Benefit Analysis and assessment of the economic justification for a computer based system project. Cost Benefit Analysis delineates costs for the project development and weights them against tangible and intangible benefits of a system. Cost Benefit Analysis is complicated by the criteria that vary with the characteristics of the system to be developed, the relative size of the project and the expected return on investment desired as part of companies strategic plan. In addition, many benefits derived from a computer-based system are intangible (e.g. better design quality through iterative optimization, increased customer satisfaction through programmable control etc.)

TECHNICAL ANALYSIS

During technical analysis, the technical merits of the system are studied and at the same time additional information about the performance, reliability, maintainability and predictability is collected. Technical analysis begins with an assessment of the technical reliability of the proposed system.

 What technologies are required for accomplished system function and performance? What new materials, methods, algorithms, or processes are required and what is their development risk? How will these obtained from technical analysis from the basis for another go/no-go decision on the test system? If the technical risk is severe, if models indicate that the desired function cannot be achieved, if the pieces just won’t fit together smoothly, it’s back to the drawing board.

As the software is very much economically feasible, then it is really important for it to be technically sound. The software will be built among:

- Backend: Sqlite3 Server
- Frontend: Python /Tkinter

OPERATIONAL ANALYSIS

The project is opera onally feasible. This product is being made for the convenience of persons. This system will greatly reduce a huge burden of them. So because of the above stated advantages the users of the system will not be reluctant at all.

CHAPTER 5
SOFTWARE DESIGN

Design is the first step in the development phase for any techniques and principles for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization. Once the software requirements have been analyzed and specified the software design involves three technical activities - design, coding, implementation and testing that are required to build and verify the software.

The design activities are of main importance in this phase, because in this activity, decisions ultimately affecting the success of the software implementation and its ease of maintenance are made. These decisions have the final bearing upon reliability and maintainability of the system. Design is the only way to accurately translate the customer's requirements into finished software or a system.

Design is the place where quality is fostered in development. Software design is a process through which requirements are translated into a representation of software. Software design is conducted in two steps. Preliminary design is concerned with the transformation of requirements into data.

Design process for the software system has two levels, namely:

a) System Design

b) Detailed Design

5.1 SYSTEM DESIGN

This design level is also called top level design. At this level, the focus is on:

- Deciding which modules are needed for the system.
- The specifications of the system
- How the modules should be interconnected?

5.1.1 <u>ARCHITECTURAL DESIGN</u>

Analysts collect a great deal of unstructured data through interviews, questionnaires, on-site observations, procedural manuals and like. It is required to organize and convert the data through system flowcharts, data flow diagrams, structured English, decision tables and the like which support future developments of the system.
The Data flow diagrams and various processing logic techniques show how, where, and when data are used or changed in an information system, but these techniques do not show the definition, structure, and relationships within the data.
It is a way to focus on functions rather than the physical implementation. This is analogous to the architect's blueprint as a starting point for system design. The design is a solution, a "how to" approach, compared to analysis, a "what is" orientation.
System design is a highly creative process. This System design process is also referred as data modeling. The most common format used for data modeling is entity-relationship (E-R) diagramming. Data modeling using the E-R notation explains the characteristics and structure of data independent of how the data may be stored in computer memories.

1. The External Design

External design consists of conceiving, planning out and specifying the externally observable characteristics of the software product. These characteristics include user displays or user interface forms and the report formats, external data sources and the functional characteristics, performance requirements etc. External design begins during the analysis phase and continues into the design phase.

2. Physical design

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified/ authenticated, how it is processed, and how it is displayed as output. Physical design, in this context, does not refer to the tangible physical design of an information system. To use an analogy, a personal computer's physical design involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots, etc.

3. Logical design

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modeling, which involves a simplistic (and sometimes graphical) representation of an actual system. In the context of systems design, modeling can undertake the following forms, including:

- Data flow diagrams
- Entity Relationship Diagrams

Prototyping Model has been used for software development according to which a throwaway prototype of the proposed system, based on the currently known requirements, is given to the user so that he has a fair idea about how the proposed system is going to be like. This will help him in deciding the interface, input and output requirements.

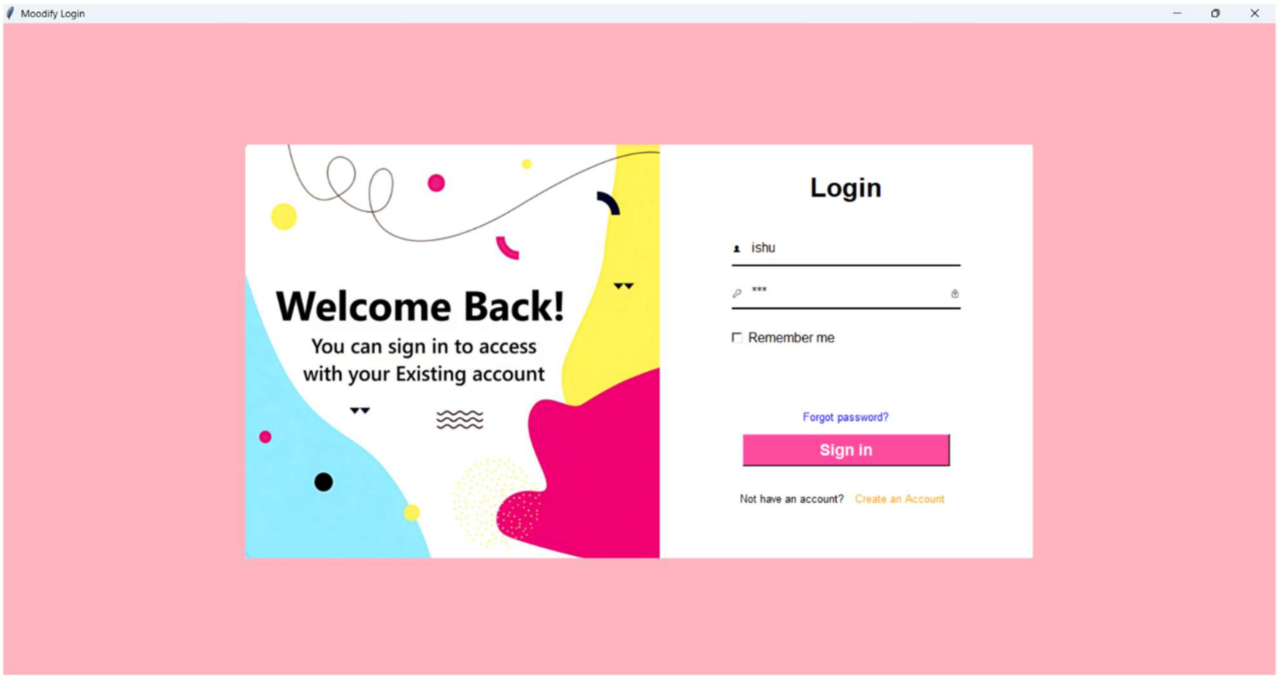
It can be easily adjudged that inputs and outputs are big in number, can increase exponentially and may create a big chaos if not restricted properly. As the user spends some time on the prototype, he will become more precise about his own input and output requirements. This prototype will provide him with an environment analogous to the proposed system's environment.

Because of object oriented support in Python, various concepts (like reusability, polymorphism, isolation etc.) are already there but for the efficient management of system components, Component based Software Engineering will also be exercised which will help in a resultant library of components, the benefit of which will be reusability and fast development.

Because of lack of hierarchical structure in object oriented approach, there is no meaning of Bottom-up or Top-down testing. Testing will begin from the most rudimentary levels of the system and will move towards higher level components which will be based on design phase rather than coding phase. In little words, it can be said that 'CLUSTER Testing' will be exercised to scrutinize all the parts and their associative functionality.

5.1.2 USER INTERFACE DESIGN

Login Window:


Moodify Login

Welcome Back!
You can sign in to access
with your Existing account

Login

ishu

***

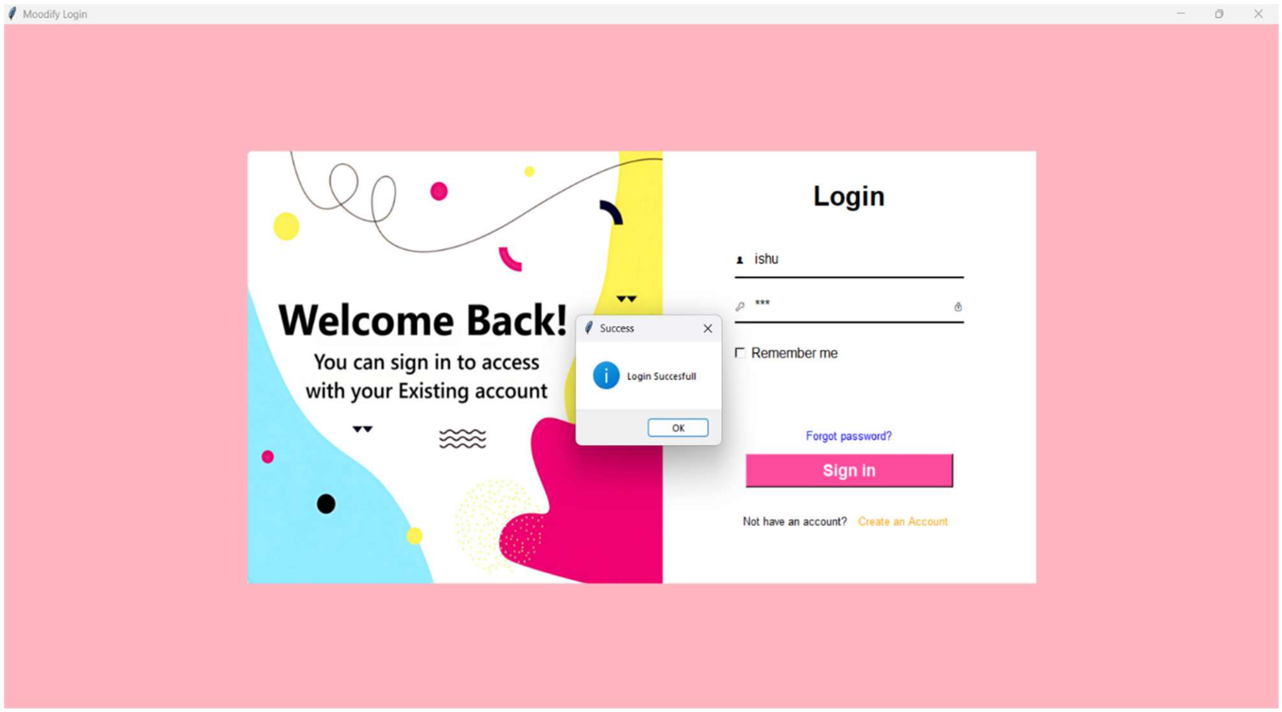
Remember me

Forgot password?

Sign in

Not have an account? Create an Account

Login Successful:


Moodify Login

Welcome Back!
You can sign in to access
with your Existing account

Login

ishu

***

Remember me

Forgot password?

Sign in

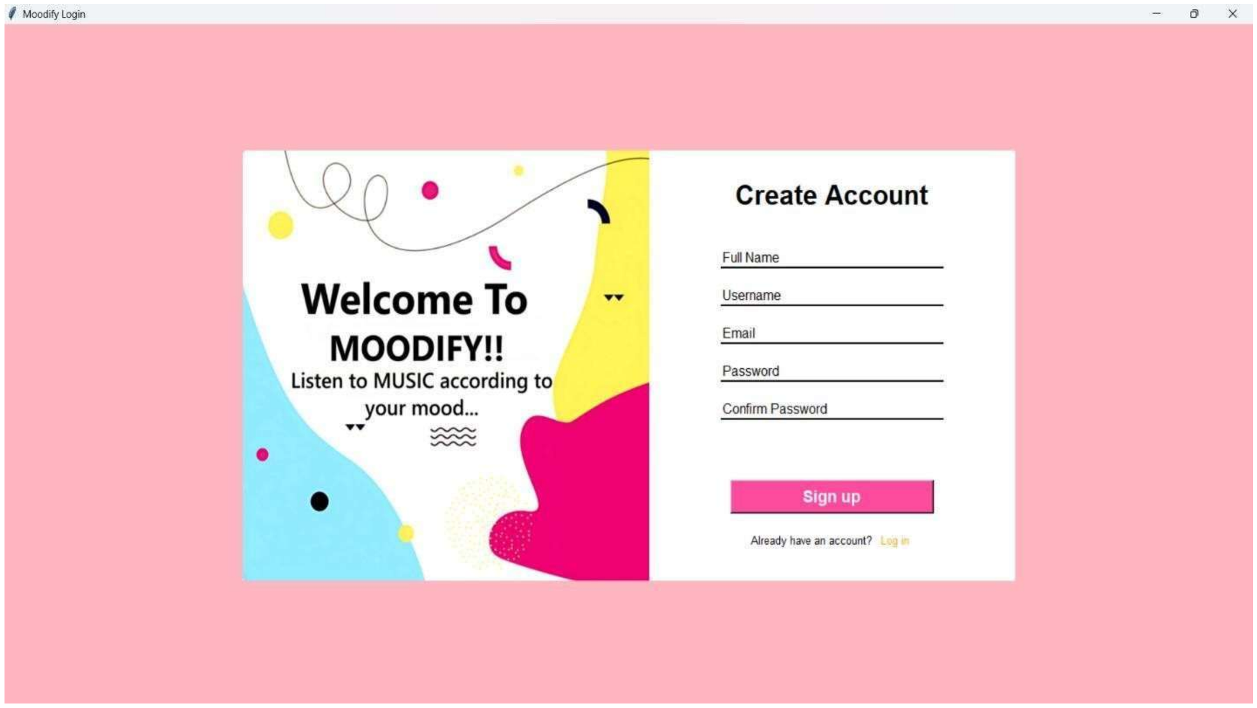
Not have an account? Create an Account

Success

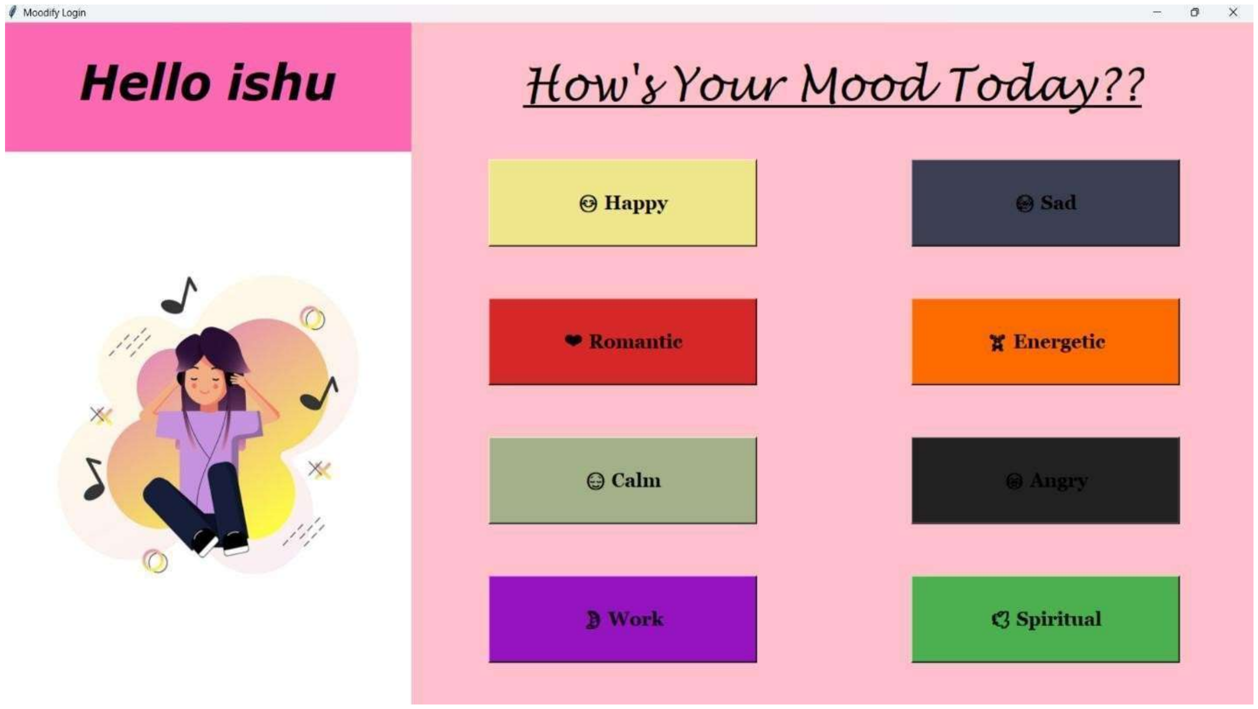
Login Succesfull

OK

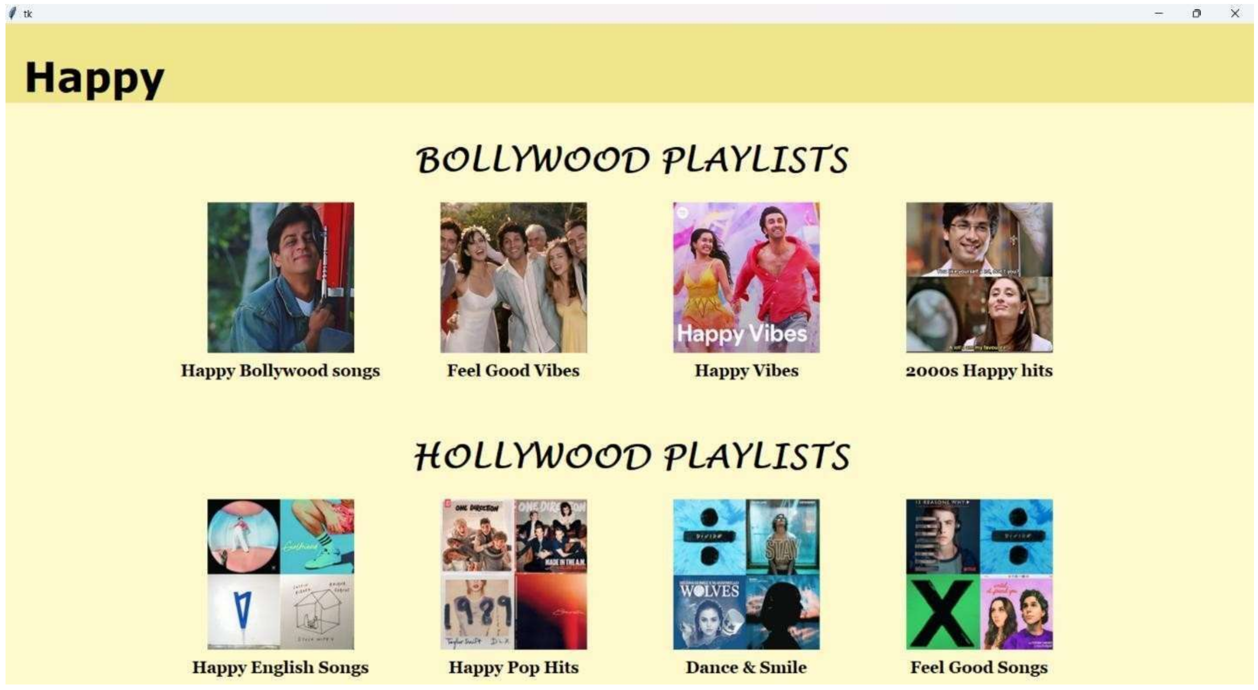
Create Account Window:



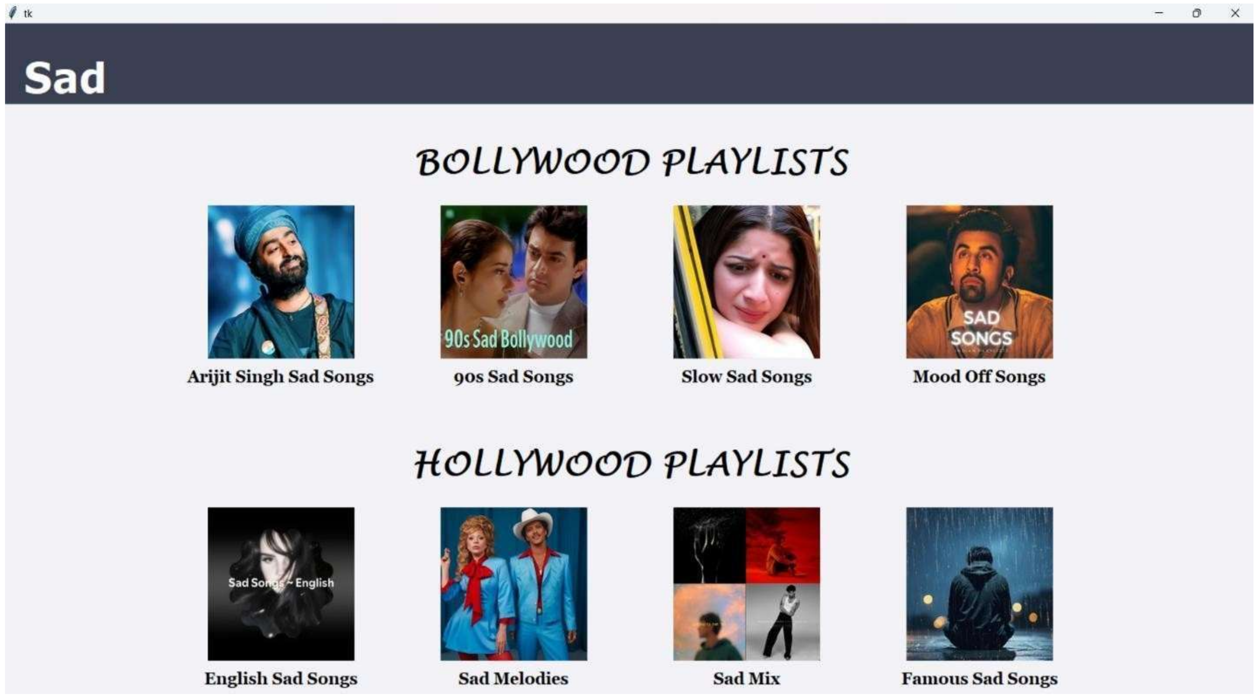
Mood Selector Window:



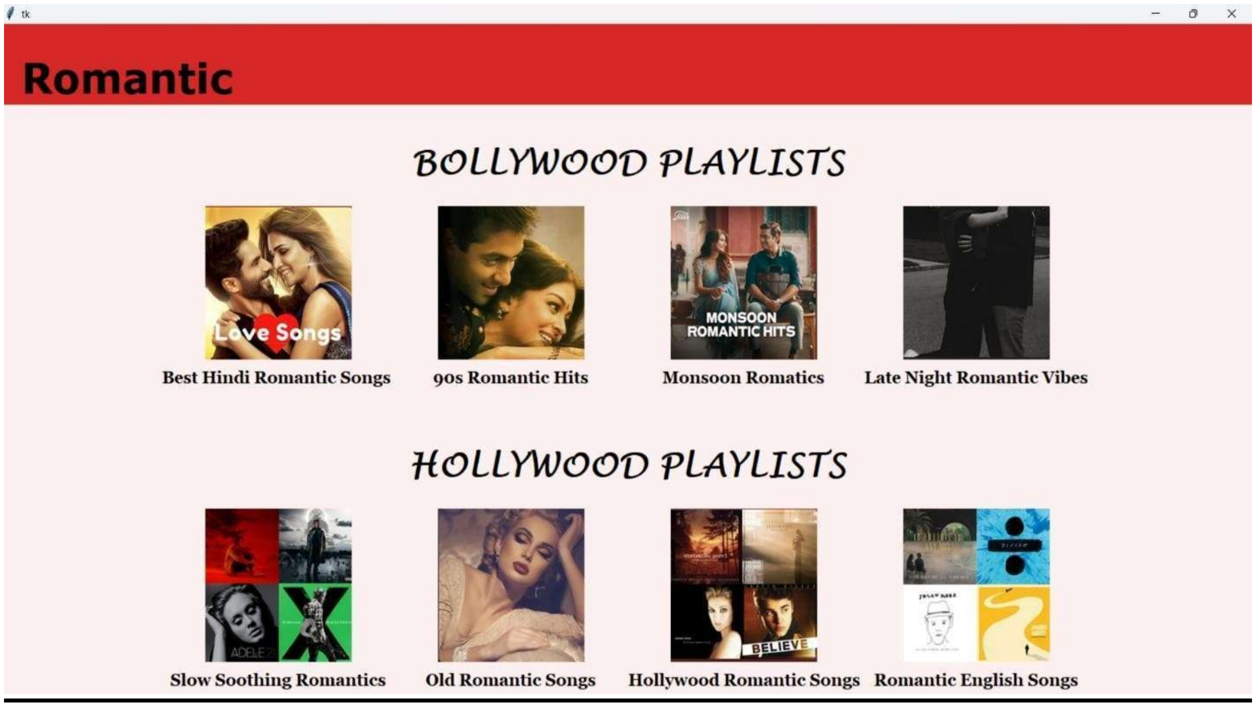
Happy Mood Window:



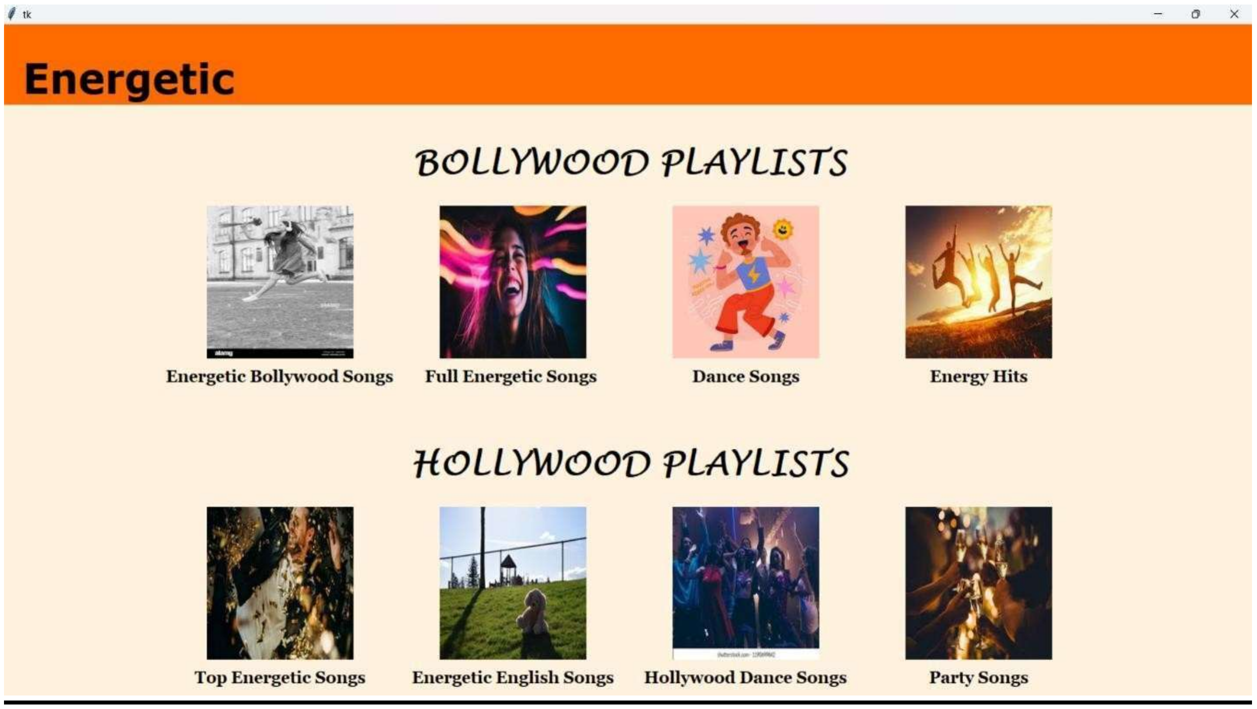
Sad Mood Window:



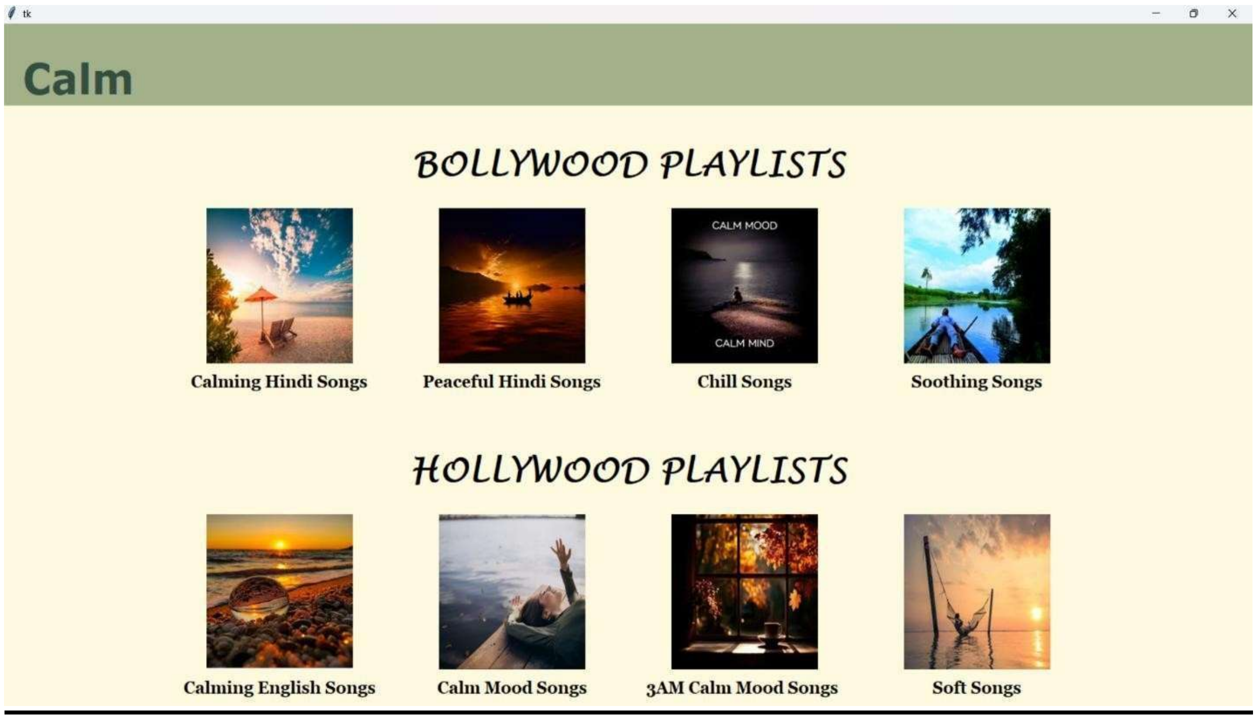
Romantic Mood Window:



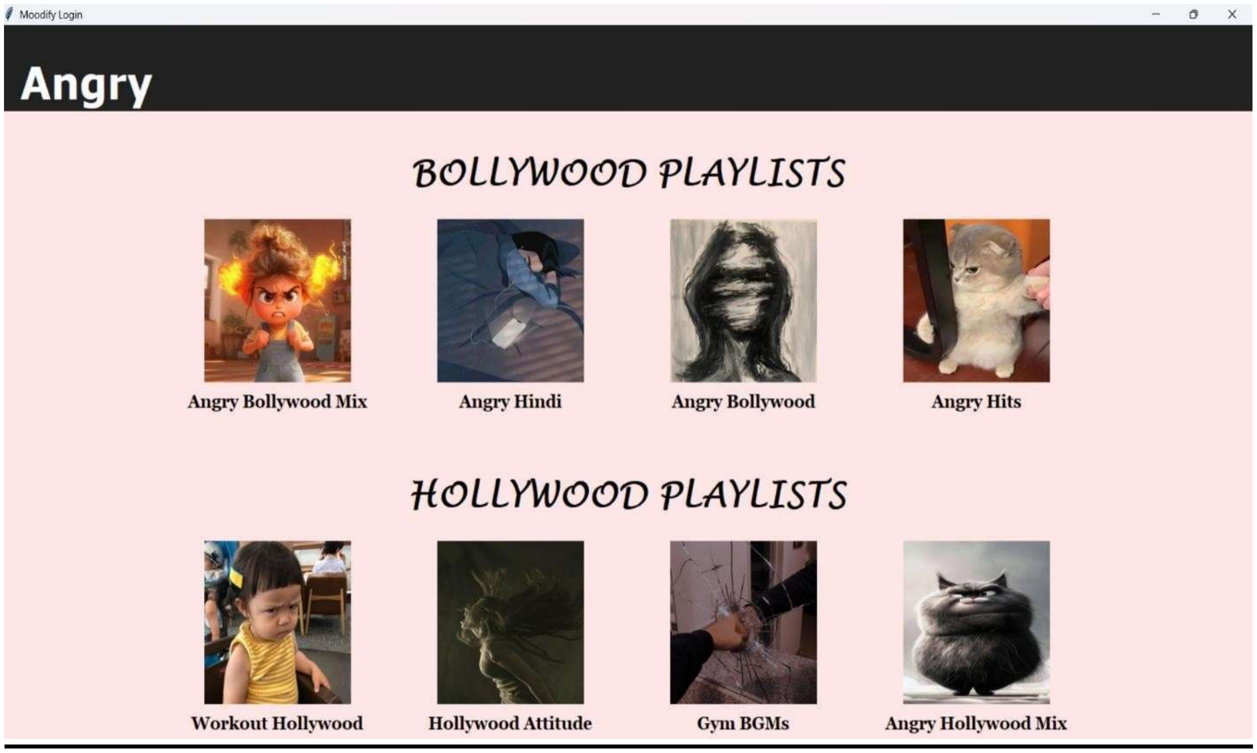
Energetic Mood Window:



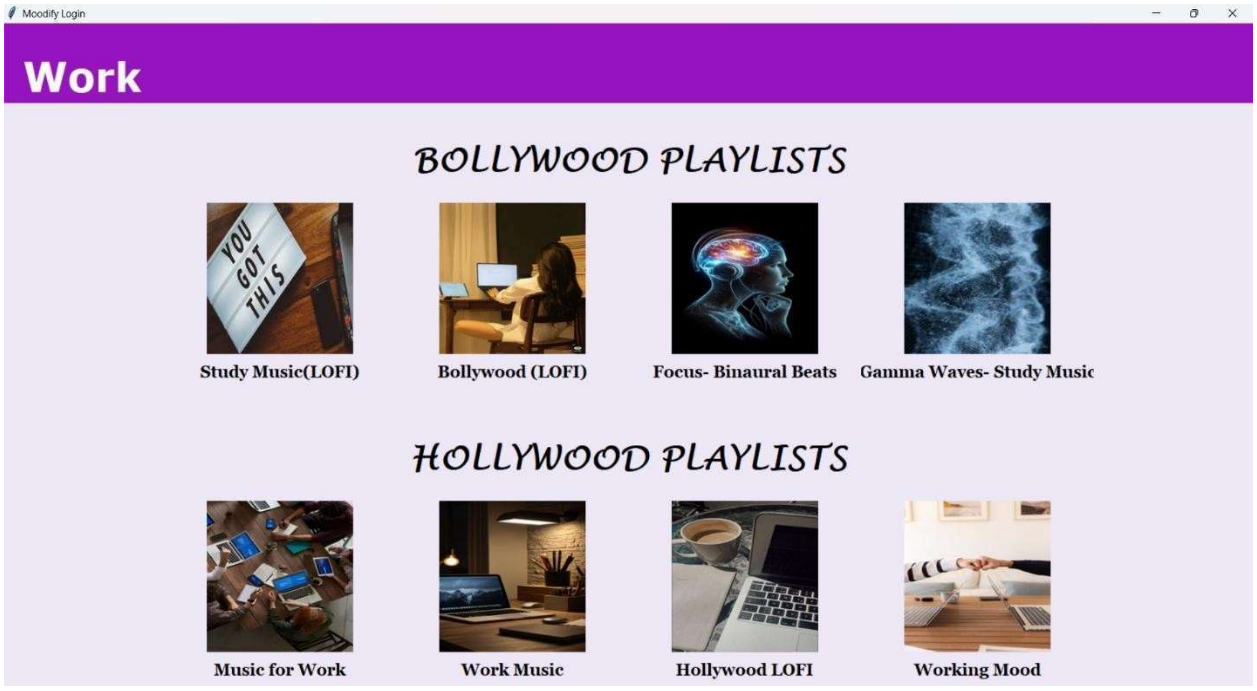
Calm Mood Window:



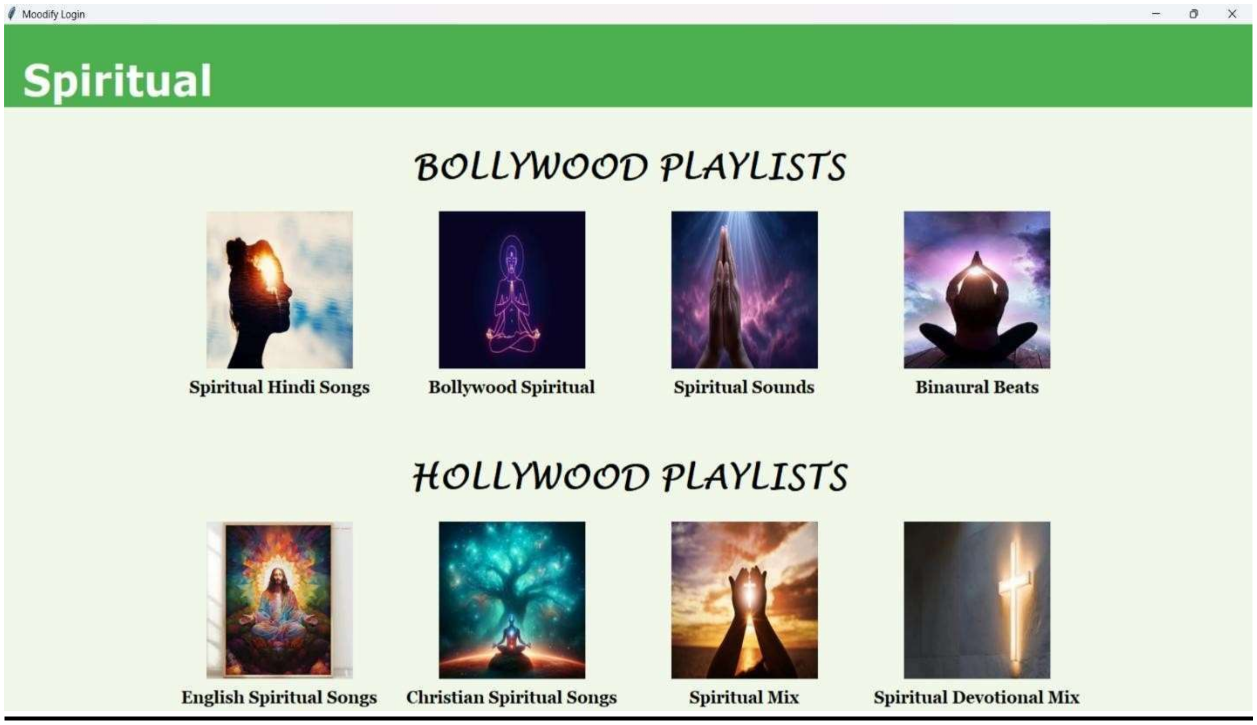
Angry Mood Window:



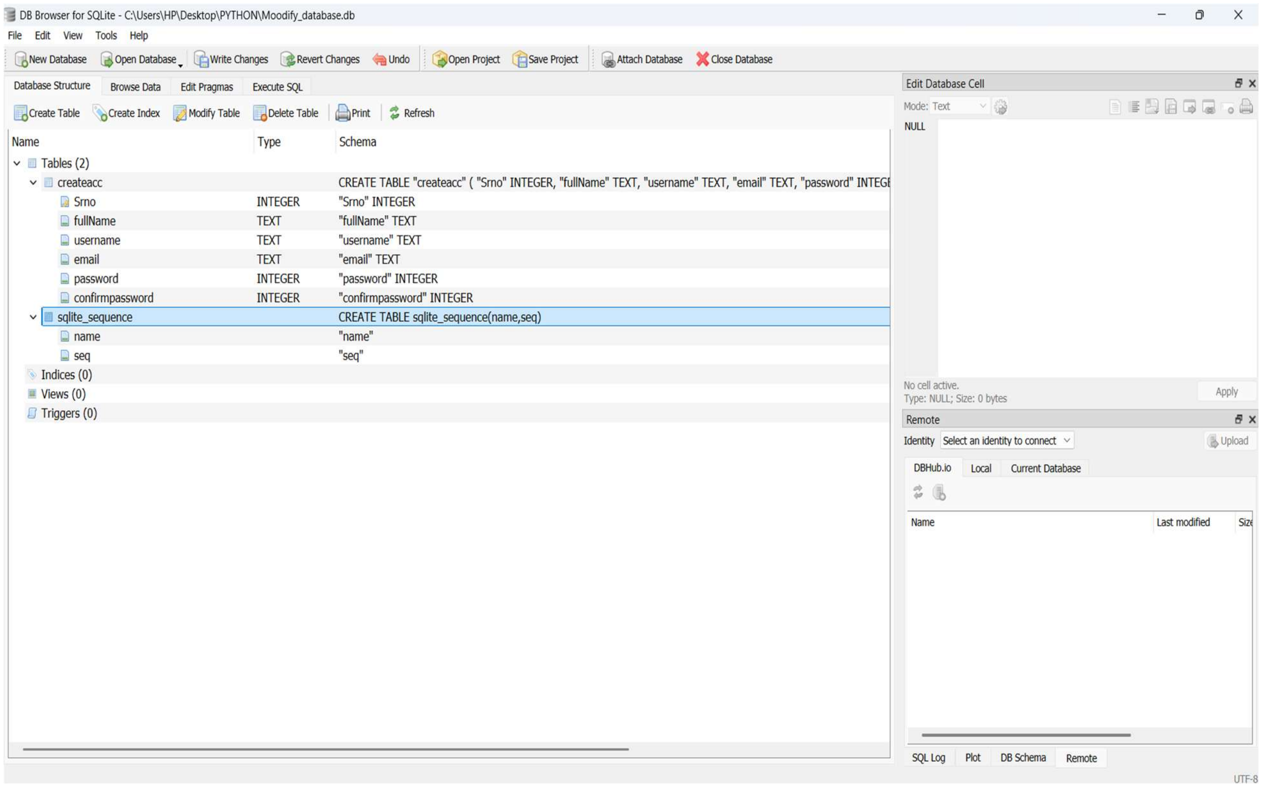
Work Mood Window:



<u>Spiritual Mood Window:</u>



Moodify Database:


DB Browser for SQLite - C:\Users\HP\Desktop\PYTHON\Moodify_database.db
File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database
Database Structure Browse Data Edit Pragmas Execute SQL
Create Table Create Index Modify Table Delete Table Print Refresh
Edit Database Cell
Mode: Text
NULL
No cell active.
Type: NULL; Size: 0 bytes
Apply
Remote
Identity Select an identity to connect
Upload
DBHub.io Local Current Database
Name Last modified Size
SQL Log Plot DB Schema Remote
UTF-8

Name	Type	Schema
Tables (2)		
createacc		CREATE TABLE "createacc" ("Srno" INTEGER, "fullName" TEXT, "username" TEXT, "email" TEXT, "password" INTEGE
Srno	INTEGER	"Srno" INTEGER
fullName	TEXT	"fullName" TEXT
username	TEXT	"username" TEXT
email	TEXT	"email" TEXT
password	INTEGER	"password" INTEGER
confirmpassword	INTEGER	"confirmpassword" INTEGER
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
name		"name"
seq		"seq"
Indices (0)		
Views (0)		
Triggers (0)		

5.2 DETAILED DESIGN

This design level is also called Logic Design. In this level, the internal design of the modules or how the specifications of the module can be satisfied is decided.

Much of the design effort for designing software is spent creating the system design. The input to the design phase is the specifications for the system to be designed. The output of the top level design phase is the architectural design for the software to be built.

A design methodology is a systematic approach to create a design by applying set of techniques and guidelines. A design can be either object oriented or function oriented. In function oriented design, the design consists of modules definitions with each module supporting a functional abstraction. In object oriented design, the modules in the design represents data abstraction.

The snapshots of step by step forms designed to achieve the goal through the proposed system along with their brief description are shown below:

The main objective of the system design is to make the system user friendly. System design involves various stages as:

- User Interface Design
- Database Design
- Navigation Flow design
- Playlist Display and Action
- Modular Function Design
- Aesthetic and Theme Design

System design is the creative act of invention, developing new inputs, a database, offline files, procedures and output for processing business to meet an organization objective. System design builds information gathered during the system analysis.

CHAPTER 6

CODING & DEVELOPMENT

The purpose of coding is to express the program logic in the best possible way and to the check it. The main reasons for coding are:

1. Unique Identification. Each item in a system should be identified uniquely and correctly.

2. Cross referencing. Diverse activities in an organization give rise to Transactions in different sub systems but affect the same item

3. Efficient storage. Code is a concise representation it reduces data entry me and improves reliability, Code as a key reduces storage space required for the data Retrieval based on a ssssskey search is faster in a computer.

Requirements of coding scheme:

The number of digits / characters used in a code must be minimal to reduce storage space of the code and retrieval efficiency. It should be expandable, that is it must allow new items to be added easily.

Type of Codes:

1. Serial Numbers. This method is that it is concise, precise and expandable. It is however not meaningful.

2. Block Codes. The block codes use blocks of serial numbers. This code is Expandable and more meaningful than the serial number coding. It is precise but not comprehensive.

3. Code Efficiency: It is often said that readability of a program is much more important than the intricacies of its code.

Main emphasis while coding was on style so that the end result was an optimized code.
The following points were kept into consideration while coding:

i) Coding Style: - The Structured programming method was used in all the modules of the project. It incorporated the following features:

- The code has been written so that the definition and implementation of each function is contained in one file.
- A group of related function was clubbed together in one file to include it when needed and save us from the labor of writing it again and again.

ii) Naming Convention: - As the project size grows, so does the complexity of recognizing the purpose of the variables. Thus the variables were given meaningful names, which would help in understanding the context and the purpose of the variable. The function names are also given meaningful names that can be easily understood by the user.

iii) Indentation: - Judicious use of indentation can make the task of reading and understanding a program much simpler. Indentation is an essential part of a good program. If code id intended without thought it will seriously affect the readability of the program. The higher-level statements like the definition of the variables, constants and the function are indented, with each nested block indented, stating their purposes in the code. Blank line is also left between each function definition to make the code look neat. Indentation for each source file stating the purpose of the file is also done.

6.1 CODING APPROACH

Top Down Approach:

A top-down approach (also known as stepwise design and in some cases used as a synonym of decomposition) is essentially the breaking down of a system to gain insight into its compositional sub-systems. In a top-down approach an overview of the system is formulated, specifying but not detailing any first-level subsystems. Each subsystem is then refined in yet greater detail, sometimes in many additional subsystem levels, until the entire specification is reduced to base elements. A top-down model is often specified with the assistance of "black boxes", these make it easier to manipulate. However, black boxes may fail to elucidate elementary mechanisms or be detailed enough to realistically validate the model. Top down approach starts with the big picture. It breaks down from there into smaller segments.

CHAPTER 7
TESTING

During testing the program to be tested is executed with the set of test cases and have the output of the program for the test cases is evaluated to determine if the program is performing as expected. Due to its approach dynamic testing can only ascertain the presence of errors in the program, the exact nature of errors is not usually decided by testing. Testing forms is the first in determining errors in the program.

Once a programs are tested individually then the system as a whole needs to be tested. During testing the system is used experimentally to ensure that the software does not fail i.e. it will run according to its specification. The programs executed to check for any syntax and logical errors. The Errors are corrected and test is made to determine whether the program is doing what it is supposed to do.

This system is tested using unit testing firstly then all the modules are integrated and again the system is tested using integrated testing and it was find that system is working according to its expectation.

Why testing is done??

- Testing is the process of running a system with the intention of finding errors.
- Testing enhances the integrity of a system by detecting deviations in design and errors in the system.
- Testing aims at detecting error-prone areas. This helps in the prevention of errors in a system.
- Testing also add value to the product by confirming to the user requirements.

Causes of Errors

The most common causes of errors in a software system are:

- Communication gap between the developer and the business decision maker: A communication gap between the developer and the business decision maker is normally due to subtle differences between them. The differences can be classified into five broad areas: Thought process, Background and Experience, Interest, Priorities, Language.
- Time provided to a developer to complete the project: A common source of errors in projects comes from time constraints in delivering a product. To keep to the schedule, features can be cut. To keep the features, the schedule can be slipped. Failing to adjust the feature set or schedule when problems are discovered can lead to rushed work and flawed systems.

- Over Commitment by the developer: High enthusiasm can lead to over commitment by the developer. In these situations, developers are usually unable to adhere to deadlines or quality due to lack of resources or required skills on the team.

- Insufficient testing and quality control: Insufficient testing is also a major source of breakdown of e-commerce systems during operations, as testing must be done during all phases of development.

- Inadequate requirements gathering: A short time to market results in developers starting work on the Web site development without truly understanding the business and technical requirements. Also, developers may create client-side scripts using language that may not work on some client browsers.

- Keeping pace with the fast changing Technology: New technologies are constantly introduced. There may not be adequate time to develop expertise in the new technologies. This is a problem for two reasons. First, the technology may not be properly implemented. Second, the technology may not integrate well with the existing environment.

Testing Principles

- To discover as yet undiscovered errors.
- All tests should be traceable to customer's requirement.
- Tests should be planned long before the testing actually begins.
- Testing should begin "in the small" & progress towards "testing in the large".
- Exhaustive Testing is not possible.
- To be most effective training should be conducted by an Independent Third Party

Testing Objectives

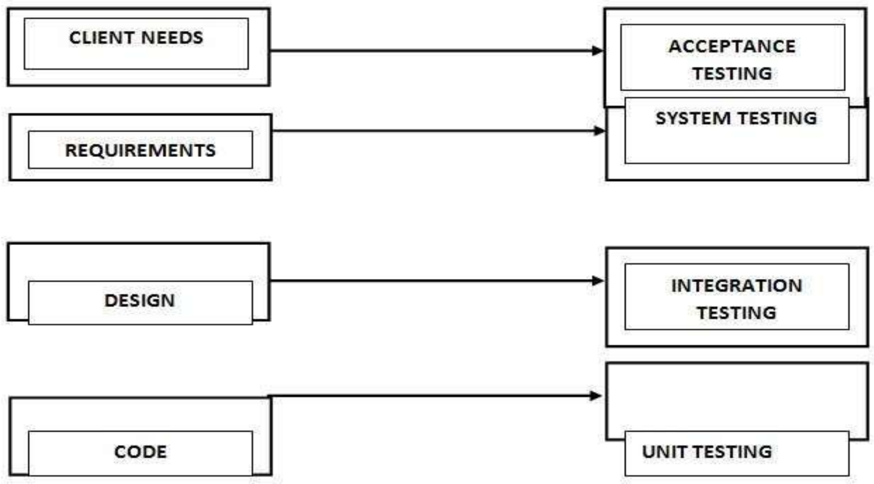
- Testing is a process of executing a program with the intent of finding errors.
- A good test case is one that has a high probability of finding an as yet undiscovered error.

- A successful test is one that uncovers an as yet undiscovered error.

Developing a Test Plan:

The first step in testing is developing a test plan based on the product requirements. The test plan is usually a formal document that ensures the product meets the following standards:

- Is thoroughly tested? Untested code adds an unknown element to the product and increases the risk of product failure.
- Meets product requirements: To meet customer needs, the product must provide the features and behavior described in the product specification. For this reason, product specifications should be clearly written and well understood.
- Does not contain defects: Features must work within established quality standards, and those standards should be clearly stated within the test plan.



A good test plan answers the following questions:

- How are tests written? Describe the languages and tools used for testing.
- Who is responsible for the testing? List the teams or individuals who write and perform the tests.
- When are the tests performed? The testing schedule closely follows the development schedule
- Where are the tests and how are test results shared? Tests should be organized so that they can be rerun on a regular basis.

- What is being tested? Measurable goals with concrete targets let you know when you have achieved success.

Some of these questions might have more than one answer, depending on the type of test. For instance, individual developers are often responsible for writing the first level of tests for their own code, while a separate testing team might be responsible for ensuring that all code works together. The following sections describe the different types of tests and the techniques used with Visual Studio .NET to perform these tests.

TYPES OF TESTS

The test plan specifies the different types of tests that will be performed to ensure the product meets customer requirements and does not contain defects. Table 10-1 describes the most common test types.

Test type	Ensures that
Unit test	Each independent piece of code works correctly
Integration test	All units work together without errors
Regression test	Newly added features do not introduce errors to other features that are already working
Load test (also called stress test)	The product continues to work under extreme usage
Platform test	The product works on all of the target hardware and software platforms

A test plan has the following steps:

- Prepare test plan
- Specify conditions for user acceptance testing
- Prepare test data for program testing
- Prepare test data for transaction path testing
- Plan user testing
- Compile/Assemble program
- Prepare job performance aids
- Prepare operational documents

Test Subject	Test Method	Expected Result	Actual Result	Remarks
Complete testing of system on different browsers	The whole project is executed using different browsers like opera, Mozilla firefox, and internet explorer.	The project should give the same output on all the browsers.	Because High HD picture Quality, the size of the Website Become Large & on executing It takes Time to View.	Browser can sometime effect the working and performance of execution of the project.

Table : Test cases for system testing

- Component testing
- Integration testing
- User testing

UNIT TESTING

Unit testing focuses verification effort on the smallest unit of software i.e. the module. Using the detailed design and the process specifications, testing is done to uncover errors within the boundary of the module. All modules must be successful in the unit test before the start of the integration testing begins.

INTEGRATION TESTING

After unit testing, we have to perform integration testing. The goal here is to see if modules can be integrated properly, the emphasis being on testing interfaces between modules. This testing activity can be considered as testing the design and hence the emphasis on testing module interactions.

SYSTEM TESTING

Here the entire software system is tested. The reference document for this process is the requirements document, and the goal is to see if software meets its requirements. Here entire 'HRRP' has been tested against requirements of project and it is checked whether all requirements of project have been satisfied or not.

ACCEPTANCE TESTING

Acceptance Testing is performed with realistic data of the client to demonstrate that the software is working satisfactorily. Testing here is focused on external behavior of the system; the internal logic of program is not emphasized.

WHITE BOX TESTING

This is a unit testing method, where a unit will be taken at a time and tested thoroughly at a statement level to find the maximum possible errors

White-box test focuses on the program control structure. Test cases are derived to ensure that all statement in the program control structure. Test cases are derived to ensure that all statement in the program control structure.

Test cases are derived to ensure that all statement in the program has been executed at least once during testing and that all logical conditions have been exercised. Basis path testing, a white box technique, makes use of program graphs (or graph matrices) to derive the set of linearly independent test that will ensure coverage. Condition and data flow testing further exercising degrees of complexity.

BLACK BOX TESTING

This testing method considers a module as a single unit and checks the unit at interface and communication with other modules rather getting into details at statement level. Here the module will be treated as a block that will take some input and generate output. Output for a given set of input combinations are forwarded to other modules.

Black-box testing techniques focus on the information domain of the software, deriving test cases by partitioning the input and output

TEST INFORMATION FLOW

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vortex of the spiral and, concentrates on each unit, component of the software as implemented in source code.Testing progresses moving outward along the spiral to integration testing, where the focus is on designed the construction of the software architecture.

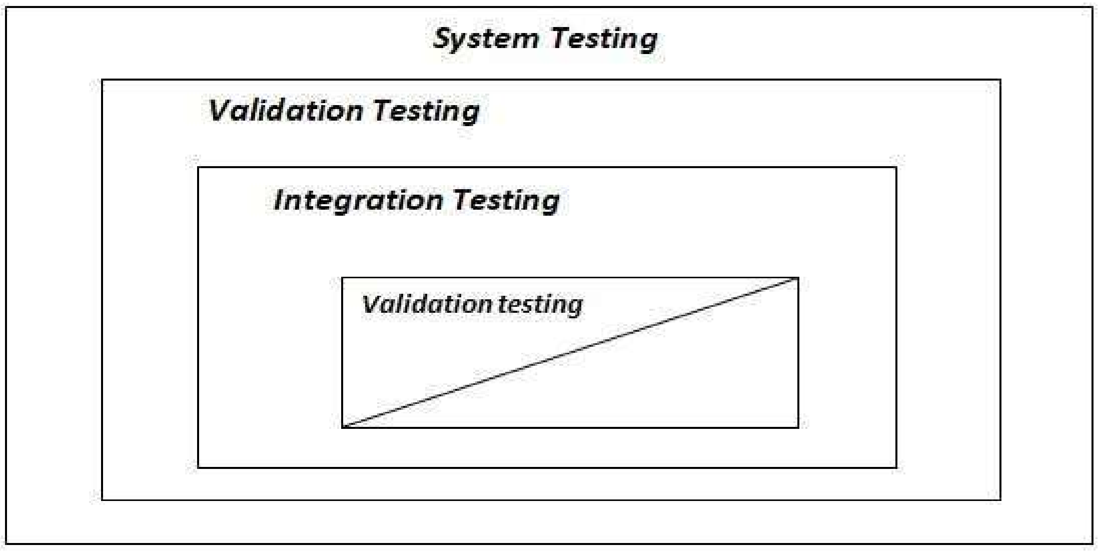


Fig : Stages of Testing

7.1 TEST CASES AND TEST CRITERIA

The main aim of integration test cases is that it tests the multiple modules together. By executing these test cases the user can find out the errors in the interfaces between the Modules.

In Website Testing, there will be User Panel. In this section the user can validate a specific URL.

No. of Test	Description	Test Inputs	Expected Results
1	Check for Login Screen	Enter values in Email and Password For Eg: Email -abc@gmail.com Password-******	Inputs should be accepted.
	Backend Verification	Select email, password from tbl_admin	The entered Email and Password should be displayed at sqlprompt or phpmyadmin.
2	Check for validation	Click submit or upload option	It should display complete details of the errors & warnings
3	Check for Registration Fill the detail in given form. Inputs should be accepted.		
4			

	Backend verification	Insert data in members	The entered data should be displayed at the sql prompt.

CHAPTER 8
IMPLEMENTATION AND EVALUATION

An Implementation plan is a management tool for a specific policy measure, or package of measures, designed to assist agencies to manage and monitor implementation effectively. Implementation plans are intended to be scalable and flexible; reflecting the degree of urgency, innovation, complexity and/or sensitivity associated with the particular policy measure. Agencies are expected to exercise judgment in this area; however, the level of detail should be sufficient to enable the agency to effectively manage the implementation of a policy measure.
At a minimum, plans should reflect the standards outlined in the Guide to Preparing Implementation Plans.

Implementation is the stage in the project where the theoretical design is turned into the working system and is giving confidence to the new system for the users i.e. will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of method to achieve the change over, an evaluation, of change over methods. A part from planning major task of preparing the implementation is education of users. The more complex system is implemented, the more involved will be the system analysis and design effort required just for implementation. An implementation coordinating committee based on policies of individual organization has been appointed. The implementation process begins with preparing a plan for the implementation for the system. According to this plan, the activities are to be carried out, discussions may regarding the equipment has to be acquired to implement the new system.

Implementation is the final and important phase. The most critical stage is in achieving a successful new system and in giving the users confidence that the new system will work and be effective. The system can be implemented only after thorough testing is done and if it found to working according to the specification. This method also offers the greatest security since the old system can take over if the errors are found or inability to handle certain types of transaction while using the new system. At the beginning of the development phase a preliminary implementation plan is created to schedule and manage the many different activities that must be integrated into plan. The implementation plan is updated throughout the development phase, culminating in a changeover plan for the operation phase. The major elements of implementation plan are test plan, training plan, equipment installation plan, and a conversion plan.

8.1 IMPLEMENTATION AND OUTPUTS

A crucial phase in the system development life cycle is the successful implementation of the new system design. Implementation simply means converting a new system design into operation. Implementation phase is used to translate the design phase into programming constructs. actual implementation of the project is done or we can say that in this phase we develop all the aspect of the project. In this phase the programmer also does user documentation of the project.

Implementation Phases: - These are the following implementation different phases.

Phase 1: - During this phase, the project water quality goals and plant capacity are set. Then, with assistance from membrane manufacturers and specialty consultants, a critique of various technologies is conducted to assess feasibility and cost-effectiveness of membrane options. Many utilities can complete this phase with their own staff. It is crucial to give a "yes" or "no" to membranes in this phase. Remember, membranes may not be the best option for all types of waters and in every application.

Phase 2: - In this phase, advice from a specialized consultant is a must. This is when layouts and conceptual design are done to evaluate membrane options. This is also the last practical and costeffective phase where you can go back to the feasibility study if the membrane is not found to be the best alternative. Detailed water quality investigation and sometimes piloting is done in this phase to verify membrane applicability and type of systems to use, as well as setting design parameters for the next phase. Depending on the piloting requirements and periods, this phase could take as little as two or three months to more than a year; if seasonal, water quality changes are substantial. If a pilot study is required, a detailed test protocol should be prepared to not only evaluate various manufacturers but also as a basis for operations and maintenance (O&M) cost evaluation. It is highly recommended to prepare this test protocol with guidance from the permitting agencies and make them a part of the decision process.

The conclusion of Phase 2 should determine what type of membrane to use and the membrane manufacturer. If manufacturers were invited to pilot test, you must ensure that they are being evaluated in a fair and open environment. Test protocol is the key evaluation tool. It is also recommended to get them involved early in the draft test protocol so there are no surprises.

Phase 3: - Before starting Phase 3, all design parameters, plant capacity, reliability and redundancy factors, stand-by provisions, temperature and water quality considerations must be established. They will then become the design basis for the specialty consultant. Phase 3 is essentially when the local engineers working with the specialty consultants to perform detail designs and preparing the bidding documents while the local engineer is focusing on the site work, building, incoming power, etc. The specialty consultant is doing detail design and layout for the process equipment and setting the bidding requirements for the membrane system.

Depending on the project schedule and local requirements, typically three major submittals are prepared: 20% to 30%, 60% to 70% and 100% design.

It is critical to establish the type of procurement and short list manufacturers, and identify all key process needs during the 20% to 30% phase. Even with the same membrane technology, the system layout, process needs and power/chemical requirements are very different.

Phase 4: - This phase is the most complex phase in membrane system implementation. There are many different methods and ways of bidding membrane systems, each with its own advantages/disadvantages.

Phase 5: - The success and smoothness of Phase 5 depends on phases 3 and 4. The single most important factor becomes how detailed the bid document is and who is responsible for what material and equipment, as well as testing and guarantees.

Phases 6 and 7: - Typically, each entity performs its own function in phases 6 and 7, except the overall controls, for which one entity should be taking charge.

Phase 8: - This phase is preparing as- built, final O&M manuals and each entity completing its punch lists. The specialty consultant can be of great assistance to compile all O&M and shop drawings and provide a comprehensive operator training on the overall plant process, while each supplier provides training of individual components.

Post Implementation:-

A Post-Implementation Review (PIR) is an assessment and review of the completed working solution. It will be performed after a period of live running, sometime after the project is completed.

There are three purposes for a Post-Implementation Review:

- To ascertain the degree of success from the project, in particular, the extent to which it met its objectives, delivered planned levels of benefit, and addressed the specific requirements as originally defined.
- To examine the efficacy of all elements of the working business solution to see if further improvements can be made to optimize the benefit delivered.
- To learn lessons from this project, lessons which can be used by the team members and by the organization to improve future project work and solutions.

In some cases, the first of these objectives can be a contractual issue. Where that is the case, it may be safer to run separate reviews - one focused on contractual compliance and the other seeking to derive further benefit from a no-blame review.

TYPES OF IMPLEMENTATION

- Implementation of a computer system to replace a manual system.
- Implementation of a new computer system to replace an existing system.
- Implementation of a modified application to replace an existing one, using the same computer.

Successful implementation may not guarantee improvement in the organization using the new system, but improper installation will prevent it. It has been observed that even the best system cannot show good result if the analysts managing the implementation do not attend to every important detail. This is an area where the systems analysts need to work with utmost care.

8.2 MAINTENANCE

Once the website is launched, it enters the maintenance phase. All systems need maintenance. Maintenance is required because there are often some residual errors remaining in the system that must be removed as they are discovered. Maintenance involves understanding the effects of the change, making the changes to both the code and the documents, testing the new parts and retesting the old parts that were not changed. Maintenance is mainly of two types:

1. Corrective Maintenance

2. Adaptive Maintenance

8.2.1 Corrective Maintenance:

Almost all software that is developed has residual errors or bugs in them. Many of these surfaces only after the system have been in operation, sometimes for a long time. These errors once discovered need to be removed, leading to the software to be changed. This is called Corrective Maintenance.

8.2.2 Adaptive Maintenance:

Even without bugs, software frequently undergoes change.. This requires modification of the software. This type of maintenance is known as the Adaptive Maintenance

CHAPTER 9

CONCLUSION

The Moodify MP3 Player is a simple yet innovative desktop application that enhances the music experience by allowing users to listen to songs based on their current mood. By combining the power of Python's Tkinter for GUI and SQLite for data handling, Moodify provides a clean, interactive, and user-friendly platform for music lovers.

The application successfully implements features such as mood-based playlist selection, user login/signup system, and an aesthetic interface with visual mood themes. It opens curated Spotify or YouTube playlists directly from the app, creating an emotionally resonant music journey.

Though Moodify currently uses static playlists, it lays the foundation for future improvements like dynamic song recommendations, user-added playlists, and real-time music streaming. Overall, the project reflects how creativity and coding can come together to build applications that are both functional and emotionally engaging.

CHAPTER 10

SCOPE OF PROJECT

The Moodify MP3 Player is not just a simple desktop music player—it is a creative step toward building a music experience that understands human emotions. The project has been designed to associate music with mood, aiming to bring comfort, energy, and relaxation based on how the user feels at a given time.

Future Scope:

While the current system provides a functional mood-based player, there is immense potential for growth. The future scope includes:

1. Dynamic Music Integration

   - Replace static playlist links with real-time API integration from platforms like Spotify, YouTube Music, Apple Music, etc.
   - Allow the app to fetch trending or most-played songs based on selected moods.

2. Offline Music Playback

   - Incorporate local media playback capabilities with support for MP3 files.
   - Add controls like Play, Pause, Stop, Volume, Next, Previous, etc.

3. AI-Powered Mood Detection

   - Use facial expression recognition, voice tone analysis, or text sentiment analysis to automatically detect user mood and suggest music accordingly.

4. Mobile App Version

- Extend Moodify into Android and iOS platforms using frameworks like Kivy or Flutter to increase accessibility and user base.

5. User Personalization

- Track user mood history and listening patterns to give personalized recommendations.
- Introduce profile customization and favorite playlists feature.

6. Custom Playlist Management

- Allow users to create, save, and manage their own playlists.
- Enable adding/removing songs and syncing across devices.

7. Community & Sharing Features

- Let users share playlists or moods with friends.
- Introduce ratings and mood-based song discussions or chat.

CHAPTER 11

BIBLIOGRAPHY

Books:

Python Crash Course, 3rd Edi on: A Hands-On, Project-Based Introduc on to Programming by Eric Matthes

Python for Everybody: Exploring Data Using Python 3 by Charles Severance

Websites: www.w3schools.com

www.google.com

www.tutorialspoint.com www.python.org