CHATBOT-SONG RECOMMENDATION SYSTEM

A PROJECT REPORT
for
Mini Project (KCA353)
Session (2023-24)

Submitted by

Riya rai
(University Roll No 2200290140129)
Riya khurana
(University Roll No 2200290140128)

Submitted in partial fulfillment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Miss Komal Salgotra
Teaching Assistant

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206
(JANUARY 2024)

CERTIFICATE

Certified that **Riya rai<2200290140129>, Riya Khurrana<2200290140128>** have carried out the project work having "**Chatbot-Song Recommendation System**" (**Mini Project-KCA353**) for **Master of Computer Application** from Dr.A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU),Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:
Riya Rai (2200290140129)
Riya Khurana (2200290140128)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Mr Praveen Kumar Gupta	**Dr. Arun Tripathi**
Assistant Professor	**Head**
Department of Computer Applications	**Department of Computer Applications**
KIET Group of Institutions, Ghaziabad	**KIET Group of Institutions, Ghaziabad**

Riya Rai
Riya Khurana

ABSTRACT

MoodTunes is an innovative project that introduces an intelligent chatbot designed to play songs based on users' moods, aiming to enhance the music listening experience. Leveraging natural language processing and machine learning techniques, MoodTunes engages users in text-based conversations to understand and interpret their emotions and moods. By collecting valuable information about the user's emotional state, preferences, and contextual factors, the chatbot curates a selection of songs that align with their current mood. The user-friendly interface allows seamless interaction, enabling users to express their mood naturally and receive personalized song recommendations. MoodTunes adapts and learns from user feedback, continually improving its recommendations over time, and provides personalized playlists, creating a tailored and immersive music experience. This project revolutionizes the way users engage with music, offering a dynamic and intuitive platform that understands and responds to their emotions, playing the perfect song for every mood.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Miss Komal Salgotra** for her guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Arun Kumar Tripathi, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Riya Rai

Riya Khurana

TABLE OF CONTENTS

Certificate	i
Abstract	ii
Acknowledgement	iii
Table of Contents	iv-vi
List of Tables	vii
List of figures	viii
1.Introduction	1-4
1.1 Overview	1
1.2 Need	2
1.3 Proposed Description	2
1.4 Proposed System	2
1.5 Features of the system	3
1.6 Project scope	3-4
2. Literature review	5-7
2.1 Abstract	5
2.2 Introduction	5
2.3 Literature review	5-7
2.3.1 Working mechanism	6
2.3.2 Issues in existing system	6
2.3.3 Future scope	7
2.4 Literature Survey	7
3. Feasibility study	8-10
3.1 Technical feasibility	8
3.1.1Hardware feasibility	8
3.1.2 Software feasibility	

3.2 Economic feasibility 9

3.3 Legal feasibility 9

3.4 Operational feasibility 9

3.5 Behavioural feasibility 9

3.6 Risk Analysis 10

4. Design 11-21

4.1 Design goals 11

4.2 Use case diagram 11-12

4.3 Functional flow of the system 13

4.4 Entity-Relationship diagram 14

4.5 Flowchart 15-16

4.6 DFD 17-18

4.7 Structure chart 18-20

4.8 System Design 20-21

5. Requirement 22-23

5.1 User requirement 22

5.2 Hardware and software requirement 23

6.Testing and debugging 24-26

6.1 Unit testing 24

6.2 Integration testing 25

6.3 System testing 25

6.4 Acceptance testing 26

6.5 Debugging 26

7. Testing and test results 27-28

7.1 Negative testing table 27

7.2 Positive testing table 28

8. Implementation

8.1 Modules 29

9. Screenshot 30-33

Conclusion 34

References 35

LIST OF TABLES

Table no.	Name of table	Page no.
Table 7.1	Negative testing table	27
Table 7.2	Positive testing table	28

LIST OF FIGURES

Figure no.	Name of figure	Page no.
3.1	Feasibility study	8
4.2	Use case diagram	12
4.3	Activity diagram	13
4.4	E-R diagram	14
4.5	DFD	18
4.6	System specification	21

CHAPTER 1

INTRODUCTION

1.1 Overview

Natural language processing is the ability of a computer programme to understand human language and has a lot of applications in today's world. The most popular application is the recognition of emotions from text. Emotion Detection and Recognition from Text analysis is a recent field of research that is closely associated with sentiment analysis. Sentiment analysis is a branch of Natural Language Processing (NLP) that seeks to detect positive, neutral, or negative emotions in text. It is dedicated to the exploration of subjective opinions or feelings collected from various sources about a particular subject, whereas Emotion Analysis is a process of detecting and recognising different types of feelings through the expression of texts, such as anger, fear, disgust, happiness, sadness, and surprise. Detecting a person's emotions is a difficult task but detecting emotions using text written by a person is even more difficult as a person can express their emotions in any form. The context-dependence of emotions inside the text is one of the most difficult aspects of determining emotion. For example, consider the phrase "Shut up!" It has an element of anger without using the word "anger" or any of its counterparts. Emotion detection is a crucial process in our project. Recognizing these emotions from a text plays a vital role in our application. In our project, the process of emotion detection of the user is done with the help of four supervised machine learning algorithms, namely Support Vector Machine (SVM), Linear Support Vector Machine (LSVM), Random Forest, and Decision Tree. The proposed model will detect six basic emotions as "happy", "sad", "anger", "surprise", "fear" and "neutral". The Chatbot module of the application makes use of Deep Learning techniques for its implementation. We created a simple, retrieval-based chatbot that employs pre-programmed input patterns and responses. To classify the category of a user's message and obtain an appropriate response from the chatbot, we utilized a special recurrent neural network called Long Short-Term Memory.

1.2 Need

Imagine a virtual companion that not only engages in friendly conversations but also curates a personalized playlist tailored to your emotional context. Here's how this innovative system works:

1. **Chatbot Interaction**:
    - Picture this: You're chatting with a friendly AI chatbot. It's like having a digital buddy who listens to your thoughts and feelings.
    - As you converse, the chatbot keenly analyzes your text, paying attention to the **tone and emotion** expressed in your messages.
2. **Emotion Analysis**:
    - Our chatbot buddy is no ordinary listener. It extract the emotional tone from your words.
    - Whether you're feeling **happy**, **sad**, **excited**, or somewhere in between, the chatbot deciphers it.
3. **Song Recommendations**:
    - Based on your mood, the chatbot suggests songs that resonate with your feelings. Feeling upbeat? It's got a track for that. Need something soothing? It's got your back.
4. **Personalization at Its Finest**:
    - Unlike generic music streaming services, our chatbot tailors recommendations just for you.
    - It considers your current emotional state and serves up tunes that match your vibe.
    - It's like having a DJ who knows your heart's desires.
5. **Enhanced User Experience**:
    - Beyond mere conversation, chatbots elevate the user experience.
    - With the added feature of music recommendations, you're not just chatting; you're discovering new songs aligned with your soul.

So, whether you're sipping coffee, pondering life's mysteries, or dancing like nobody's watching, our chatbot has a song recommendation waiting for you. 🎶🤘

1.3 Project description

The project aims to create a chatbot that recommends music based on the user's text tone. The chatbot analyzes the tone of the text expressed by the user to identify the mood. Once the mood is identified, the application will play songs in the form of a web page based on the user's choice as well as their current mood. The main goal of the proposed system is to reliably determine a user's mood based on their text tone with an application that can be installed on the user's desktop.

1.4 Proposed system

The proposed system is a machine learning-based chatbot song recommender system that aims to recommend songs based on the user's current emotion or mood. The chatbot analyzes the user's sentiment by asking some general questions and then generates a playlist based on the input provided by the user.

The proposed system is personalized, where the user's current emotion is analyzed with the help of the chatbot. The chatbot identifies the user's sentiment by asking some general questions. Based on the input provided by the user, current emotion or mood is analyzed by the chatbot and it will generate the playlist.

1.5 Features of the system

A chatbot-song recommendation system can have the following features:

1. Personalized recommendations**:** The chatbot can generate personalized song recommendations based on the user's current emotion or mood. The chatbot can analyze the user's sentiment by asking some general questions and then generate a playlist based on the input provided by the user.
2. Tone analysis**:** The chatbot can use IBM Tone Analyzer API to check the text tone of the user and to predict the mood based on the text of the user [123].
3. Song recommendation**:** The chatbot can use Last.FM API to recommend songs based on the mood of the user .
4. Ease of use**:** The chatbot can be installed on the user's desktop and can be used to recommend songs based on the user's mood [1].
5. Machine learning-based**:** The chatbot can be a machine learning-based chatbot song recommender system that can recommend songs based on the user's current emotion or mood.

1.6 Project scope

A chatbot-song recommendation system can have the following scope:

1. Personalized recommendations**:** The chatbot can generate personalized song recommendations based on the user's current emotion or mood. The chatbot can analyze the user's sentiment by asking some general questions and then generate a playlist based on the input provided by the user .
2. Tone analysis**:** The chatbot can use IBM Tone Analyzer API to check the text tone of the user and to predict the mood based on the text of the user [123].

3. Song recommendation**:** The chatbot can use Last.FM API to recommend songs based on the mood of the user [123].
4. Ease of use**:** The chatbot can be installed on the user's desktop and can be used to recommend songs based on the user's mood [1].
5. Machine learning-based**:** The chatbot can be a machine learning-based chatbot song recommender system that can recommend songs based on the user's current emotion or mood .

CHAPTER-2

LITERATURE REVIEW

2.1 Abstract

MoodTunes is an innovative project that introduces an intelligent chatbot designed to play songs based on users' moods, aiming to enhance the music listening experience. Leveraging natural language processing and machine learning techniques, MoodTunes engages users in text-based conversations to understand and interpret their emotions and moods. By collecting valuable information about the user's emotional state, preferences, and contextual factors, the chatbot curates a selection of songs that align with their current mood. The user-friendly interface allows seamless interaction, enabling users to express their mood naturally and receive personalized song recommendations. MoodTunes adapts and learns from user feedback, continually improving its recommendations over time, and provides personalized playlists, creating a tailored and immersive music experience. This project revolutionizes the way users engage with music, offering a dynamic and intuitive platform that understands and responds to their emotions, playing the perfect song for every mood.

2.2 Introduction

A chatbot-song recommendation system is a machine learning-based chatbot song recommender system that can recommend songs based on the user's current emotion or mood . The proposed system is personalized, where the user's current emotion is analyzed with the help of the chatbot. The chatbot identifies the user's sentiment by asking some general questions. Based on the input provided by the user, current emotion or mood is analyzed by the chatbot and it will generate the playlist [1].

There are several research papers available on this topic. For example, a research paper titled "A Machine Learning Based Chatbot Song Recommender System" proposes a chatbot song recommender system that recommends songs based on the user's current emotion or mood. The proposed system uses IBM Tone Analyzer API to check the text tone of the user and to predict the mood based on the text of the user, and Last.FM API to recommend songs based on the mood of the user [1]. Another research paper titled "Chatbot with Song Recommendation based on Emotion" proposes a chatbot that interacts with the user, analyzes the emotions of chats, and recommends a song playlist based on the user's emotions. The objective of the application is to identify the emotion perceived by the user, and once the emotion is identified, a list of songs is suggested based on the emotion.

2.3 Literature review

2.3.1 Working mechanism

A chatbot-song recommendation system is a machine learning-based chatbot song recommender system that can recommend songs based on the user's current emotion or mood [123]. The proposed system is personalized, where the user's current emotion is analyzed with the help of the chatbot. The chatbot identifies the user's sentiment by asking some general questions.
Based on the input provided by the user, current emotion or mood is analyzed by the chatbot and it will generate the playlist.

2.3.2 Issues in existing system

I found some research papers on chatbot-song recommendation systems, but I couldn't find any specific issues with the existing systems. However, some of the research papers propose improvements to the existing systems. For example, a research paper titled "A Machine Learning Based Chatbot Song Recommender System" proposes a chatbot song recommender system that recommends songs based on the user's current emotion or mood. The proposed system uses IBM Tone Analyzer API to check the text tone of the user and to predict the mood based on the text of the user, and Last.FM API to recommend songs based on the mood of the user . Another research paper titled "Chatbot with Song Recommendation based on Emotion" proposes a chatbot that interacts with the user, analyzes the emotions of chats, and recommends a song playlist based on the user's emotions. The objective of the application is to identify the emotion perceived by the user, and once the emotion is identified, a list of songs is suggested based on the emotion.

2.3.3 Future scope

- Integration with Voice Assistants: Enhance the chatbot's functionality by integrating it with popular voice assistants like Amazon Alexa, Google Assistant, or Apple Siri. This integration would enable users to interact with the chatbot using voice commands, making the experience more natural and convenient.
- Emotion Recognition: Incorporate advanced emotion recognition techniques, such as sentiment analysis and facial recognition, to better understand the user's emotions and tailor song recommendations accordingly. This would provide a more personalized and immersive music listening experience.
- Multi-lingual Support: Expand the chatbot's capabilities to support multiple languages, allowing users from diverse linguistic backgrounds to interact with the chatbot and receive song recommendations in their preferred language.
- Social Integration: Integrate the chatbot with social media platforms and music streaming services to leverage social data and user preferences for more accurate and

relevant song recommendations. This would enable users to discover music that is popular among their friends or within their social network.

- Context-aware Recommendations: Enhance the chatbot's recommendation engine by considering additional contextual factors such as location, time of day, and user activity. This would allow the chatbot to provide more relevant song suggestions that align with the user's specific context and enhance the overall music listening experience.

2.4 Literature review

There are several research papers available on the topic of chatbot-song recommendation systems. A research paper titled “CHAT BOT SONG RECOMMENDER SYSTEM” proposes a machine learning-based chatbot song recommender system that recommends songs based on the user’s current emotion or mood. The proposed system uses IBM Tone Analyzer API to check the text tone of the user and to predict the mood based on the text of the user, and Last.FM API to recommend songs based on the mood of the user . Another research paper titled “Music Recommender System Using ChatBot” proposes a chatbot that recommends songs based on the tone of the user’s voice. The system uses Last.fm API to recommend songs based on the user’s mood, and IBM Tone Analyzer API to analyze the tone of the user’s voice .
A research paper titled “A Machine Learning Based Chatbot Song Recommender System” proposes a chatbot song recommender system that recommends songs based on the user’s current emotion or mood. The proposed system uses IBM Tone Analyzer API to check the text tone of the user and to predict the mood based on the text of the user, and Last.FM API to recommend songs based on the mood of the user . Another research paper titled “Chatbot with Song Recommendation based on Emotion” proposes a chatbot that interacts with the user, analyzes the emotions of chats, and recommends a song playlist based on the user’s emotions. The objective of the application is to identify the emotion perceived by the user, and once the emotion is identified, a list of songs is suggested based on the emotion.

CHAPTER-3

FEASIBILITY STUDY

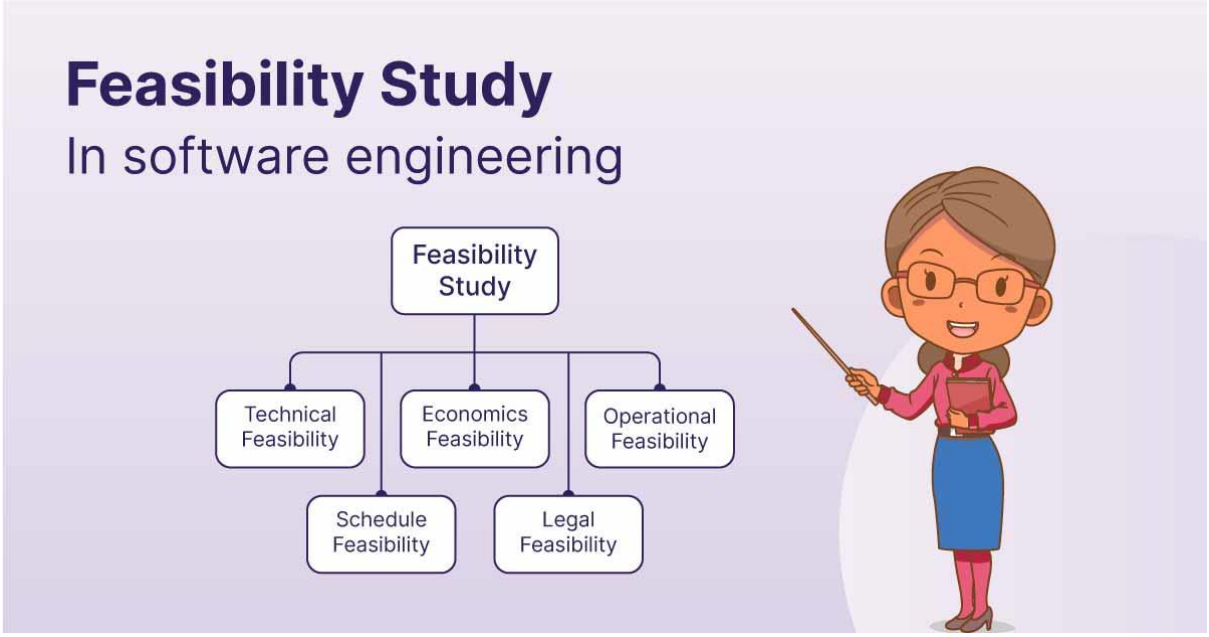


Fig 3.1 feasibility study

3.1 Technical feasibility

A technical feasibility study is a standard practice for companies to conduct feasibility studies before commencing work on a project. The study assesses the practicality and viability of a product or service before launching it. Technical feasibility helps determine the efficacy of the proposed plan by analyzing the process, including tools, technology, material, labor, and logistics. The study identifies potential challenges and uncovers ways to overcome them. It also helps in long-term planning, as it can serve as a flowchart for how products and services evolve before they reach the market .In technical feasibility current resources both hardware software along with required technology are analyzed/assessed to develop the project. This technical feasibility study reports whether there exists correct required resources and technologies which will be used for project development.

The technical feasibility study in software engineering is conducted in various ways depending on the company. Some people may do it in a precise and organized method, while others may do it as needed. However, you must have the following resources -

- Hardware

- Software
- Technology
- Skills and knowledge
- Time and budget for development
- Specialists
- Software development tools

3.1.1 Hardware feasibility

Hardware feasibility is a part of technical feasibility that assesses the ability of computer hardware to handle workloads adequately. It involves evaluating the hardware requirements of the project and ensuring that the available hardware is capable of meeting those requirements .

3.1.2 Software feasibility

A feasibility study is performed on a software project to understand the viability of the product. Understanding a project's feasibility has a lot to do with how it will perform in the market, what will work, what competitors have created and how will this product survive.

3.2 Economic feasibility

In economic feasibility study the cost and benefit of the project is analyzed. It means under this feasibility study a detailed analysis is carried out of what will be the cost of the project for development which includes all required cost for final development like hardware and software resource required, design and development cost and operational cost and so on. After that it is analyzed whether the project will be beneficial in terms of finance for the organization or not. It offers reduced costs, improved efficiency, increased profits with transparency and flexibility.

3.3 Legal feasibility

In Legal Feasibility study project is analyzed in legality point of view. This includes analyzing barriers of legal implementation of project, data protection acts or social media laws, project certificate, license, copyright etc. Overall it can be said that Legal Feasibility Study is study to know if proposed project conform legal and ethical requirements.

3.4 Operational feasibility

In operational feasibility, the degree of providing service to requirements is analyzed along with how easy the product will be to operate and maintain after deployment. Along with this other operational scopes are determining usability of product, determining suggested solution by software development team is acceptable or not etc.

3.5 Behavioural feasibility

Behavioural feasibility is studied in order to check, whether the human or employees in the business will use it or not. Operational feasibility relies on human resources and analyses whether the software will operate after it is developed properly or not.

Behavioural feasibility is a scale of how the proposed system solves the problem, to what extent it takes the advantage of the opportunities identified during scope definition and how much it satisfies the requirements identified in the requirement analysis of the software development.

3.6 Risk analysis

Risk Analysis in project management is a sequence of processes to identify the factors that may affect a project's success. These processes include risk identification, analysis of risks, risk management and control, etc. Proper risk analysis helps to control possible future events that may harm the overall project. It is more of a pro-active than a reactive process.

It is the process of prioritizing risks for further analysis of project risk or action by combining and assessing their probability of occurrence and impact. It helps managers to lessen the uncertainty level and concentrate on high priority risks.

Plan risk management should take place early in the project, it can impact on various aspects for example: cost, time, scope, quality and procurement.

The inputs for qualitative Project Risk Analysis and Management includes

- Risk management plan
- Scope baseline
- Risk register
- Enterprise environmental factors
- Organizational process assets

The output of this stage would be

- Project documents updates

CHAPTER-4

DESIGN

Designing is the most important phase of software development. It requires a careful planning and thinking on the part of the system designer. Designing software means to plan how the various parts of the software are going to achieve the desired goal. It should be done with utmost care because if the phase contains any error, then that will affect the performance of the system, as a result it may take more processing time, more response time, extra coding workload etc.

Software design sits at the technical kernel of the software engineering process and is applied regardless of the software process model that is used. After the software requirements have been analysed and specified, software design is the first of the three technical activities Designing, Coding and Testing that are required to build and verify the software. Each activity transforms information in such a manner that ultimately results in validated computer software.

4.1 Design goals

The following goals were kept in mind while designing the system:

- Make system user-friendly. This was necessary so that system could be used efficiently and system could act as catalyst in achieving objectives.
- Make system compatible i.e. It should fit in the total integrated system. Future maintenance and enhancement must be less.
- Make the system compatible so that it could integrate other modules of system into itself.
- Make the system reliable, understandable and cost-effective.

4.2 Use case diagram

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

Purpose of use case diagram

The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

Following are the purposes of a use case diagram given below:

1. It gathers the system's needs.
2. It depicts the external view of the system.
3. It recognizes the internal as well as external factors that influence the system.
4. It represents the interaction between the actors.

Use case diagram for chatbot-song recommendation system:

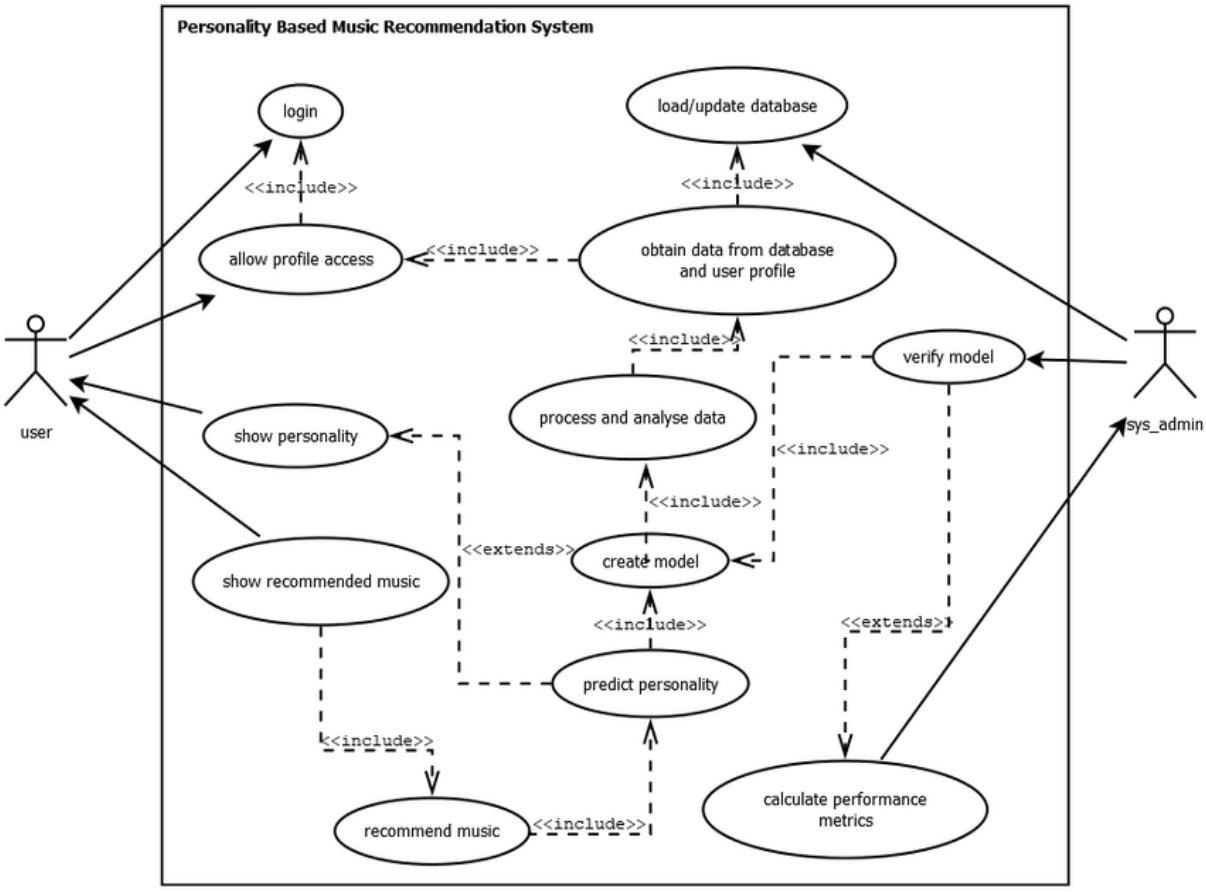


Fig 4.2 use case diagram

4.3 Functional flow of the system

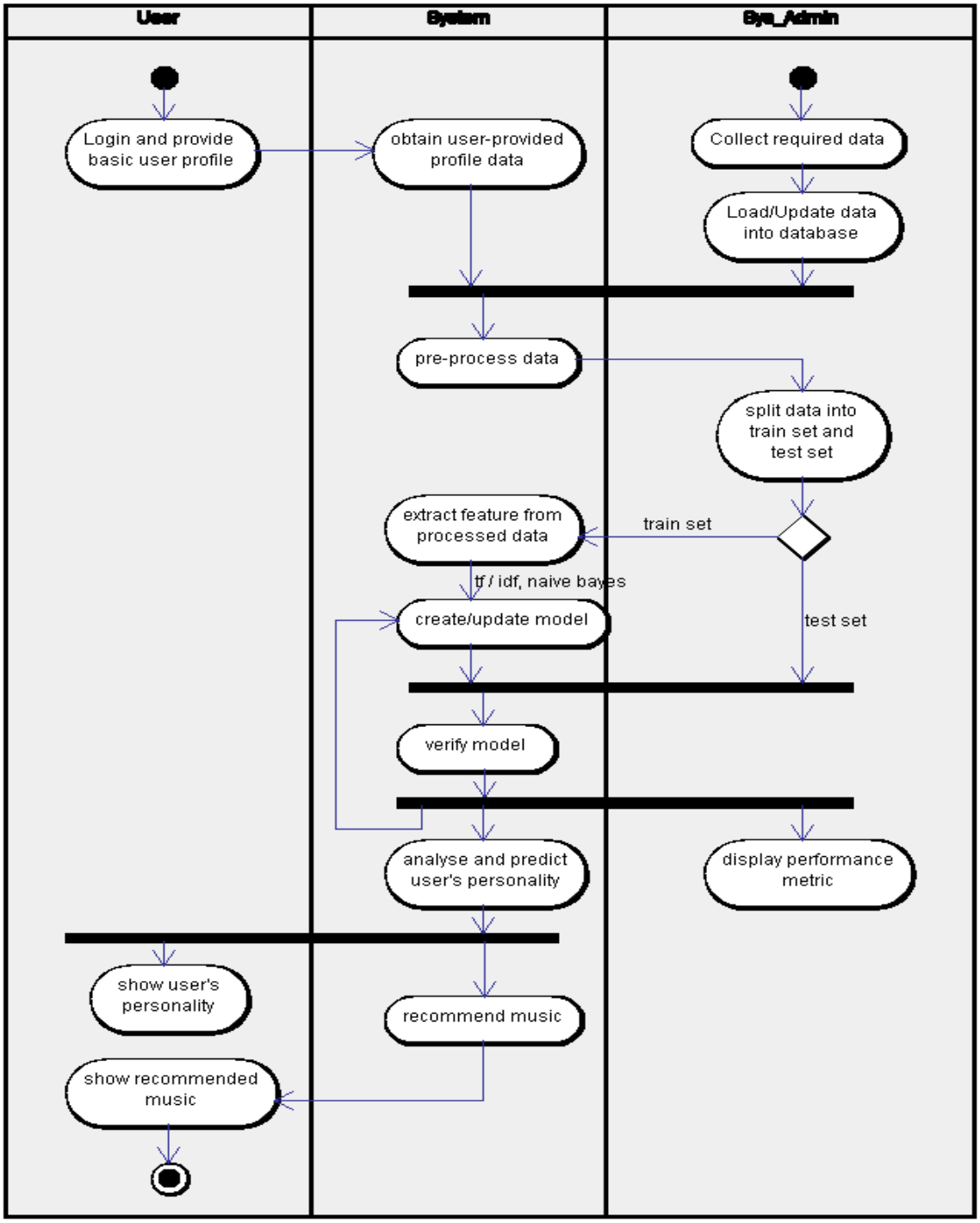


Fig 4.3 Activity diagram

4.4 Entity relationship diagram

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

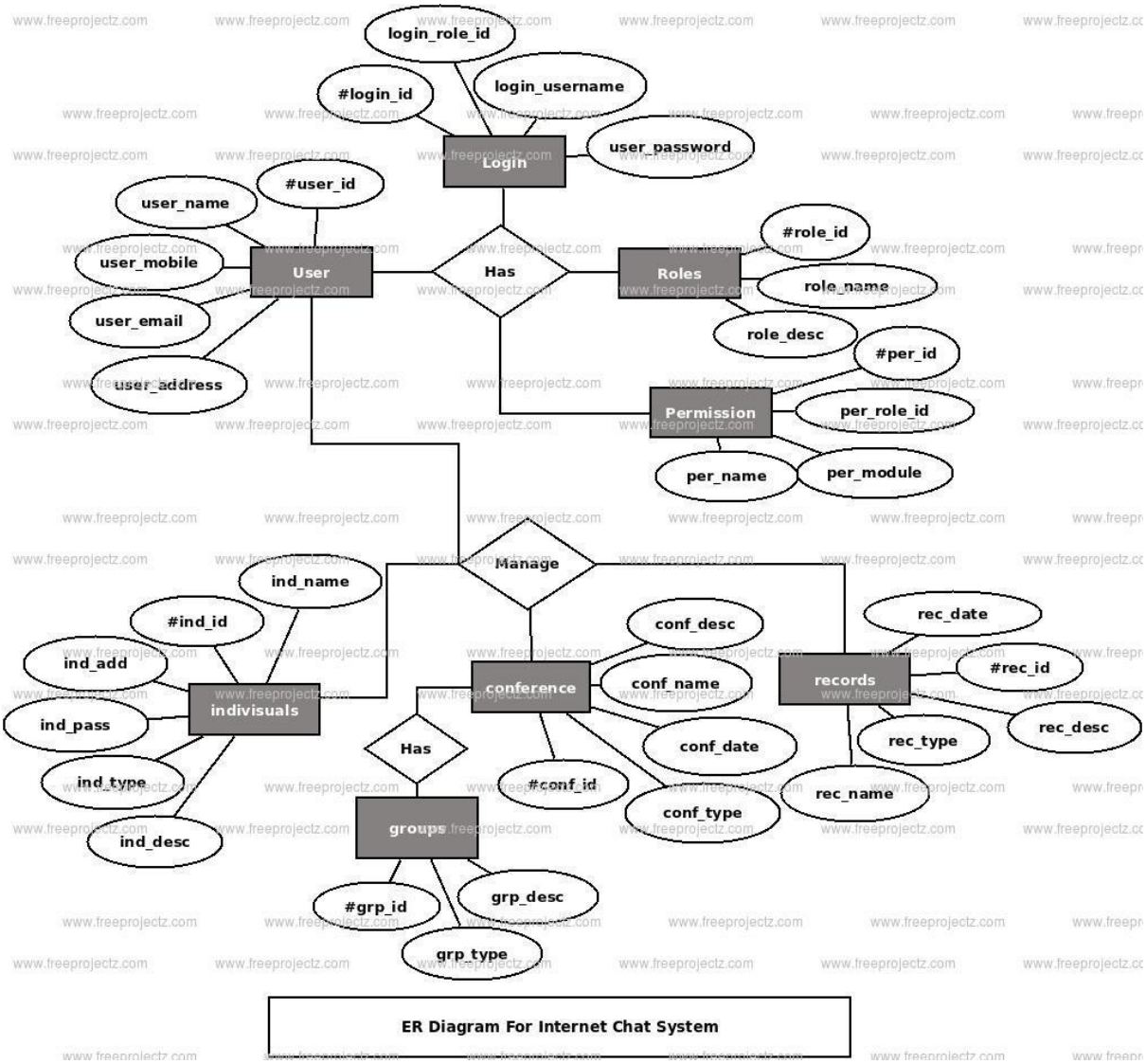


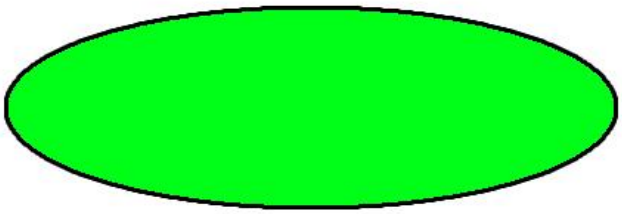
Fig:4.4 ER-diagram

4.4 Flowchart

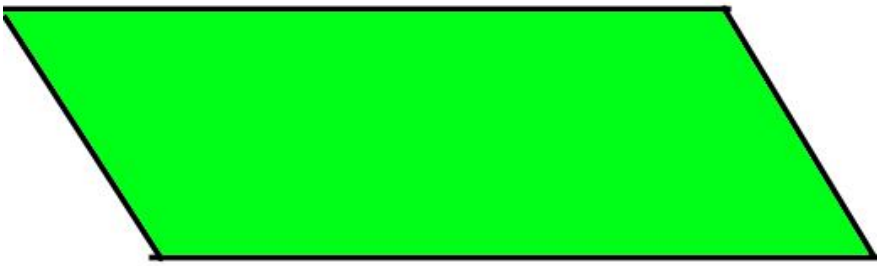
Flowchart is a graphical representation of an algorithm. Programmers often use it as a program-planning tool to solve a problem. It makes use of symbols which are connected among them to indicate the flow of information and processing. The process of drawing a flowchart for an algorithm is known as "flowcharting".

4.1.1 Basic symbols of flowchart

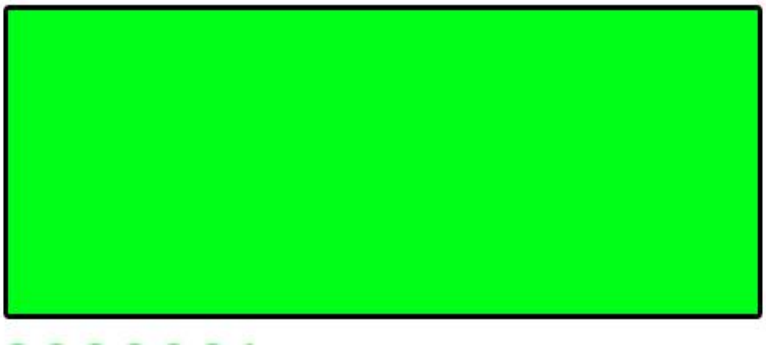
1. **Terminal:** The oval symbol indicates Start, Stop and Halt in a program's logic flow. A pause/halt is generally used in a program logic under some error conditions. Terminal is the first and last symbols in the flowchart.



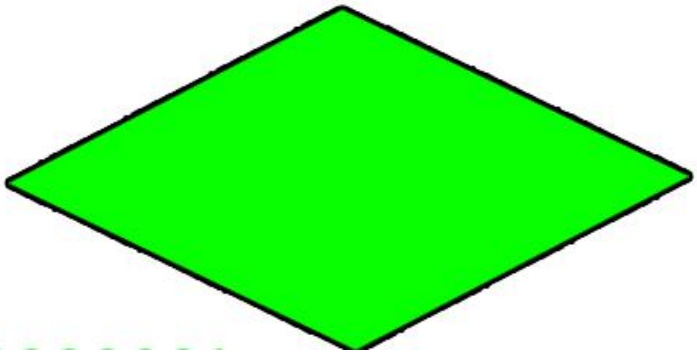
- **Input/Output:** A parallelogram denotes any function of input/output type. Program instructions that take input from input devices and display output on output devices are indicated with parallelogram in a flowchart.



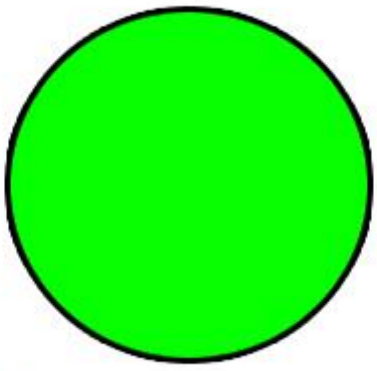
- **Processing:** A box represents arithmetic instructions. All arithmetic processes such as adding, subtracting, multiplication and division are indicated by action or process symbol.



- **Decision:** Diamond symbol represents a decision point. Decision based operations such as yes/no question or true/false are indicated by diamond in flowchart.



- **Connectors:** Whenever flowchart becomes complex or it spreads over more than one page, it is useful to use connectors to avoid any confusions. It is represented by a circle.



- **Flow lines:** Flow lines indicate the exact sequence in which instructions are executed. Arrows represent the direction of flow of control and relationship among different symbols of flowchart.

4.5 Data flow diagrams

Data flow diagram is graphical representation of flow of data in an information system. It is capable of depicting incoming data flow, outgoing data flow and stored data. The DFD does not mention anything about how data flows through the system.
There is a prominent difference between DFD and Flowchart. The flowchart depicts flow of control in program modules. DFDs depict flow of data in the system at various levels. DFD does not contain any control or branch elements.

Types of DFD

Data Flow Diagrams are either Logical or Physical.

- Logical DFD - This type of DFD concentrates on the system process, and flow of data in the system. For example in a Banking software system, how data is moved between different entities.
- Physical DFD - This type of DFD shows how the data flow is actually implemented in the system. It is more specific and close to the implementation.

DFD components

DFD can represent Source, destination, storage and flow of data using the following set of components:

- Entities - Entities are source and destination of information data. Entities are represented by a rectangles with their respective names.
- Process - Activities and action taken on the data are represented by Circle or Round-edged rectangles.
- Data Storage - There are two variants of data storage - it can either be represented as a rectangle with absence of both smaller sides or as an open-sided rectangle with only one side missing.
- Data Flow - Movement of data is shown by pointed arrows. Data movement is shown from the base of arrow as its source towards head of the arrow as destination.

Levels of DFD

- Level 0 - Highest abstraction level DFD is known as Level 0 DFD, which depicts the entire information system as one diagram concealing all the underlying details. Level 0 DFDs are also known as context level DFDs.
- Level 1 - The Level 0 DFD is broken down into more specific, Level 1 DFD. Level 1 DFD depicts basic modules in the system and flow of data among various modules. Level 1 DFD also mentions basic processes and sources of information.
- Level 2 - At this level, DFD shows how data flows inside the modules mentioned in Level 1. Higher level DFDs can be transformed into more specific lower level DFDs

with deeper level of understanding unless the desired level of specification is achieved.

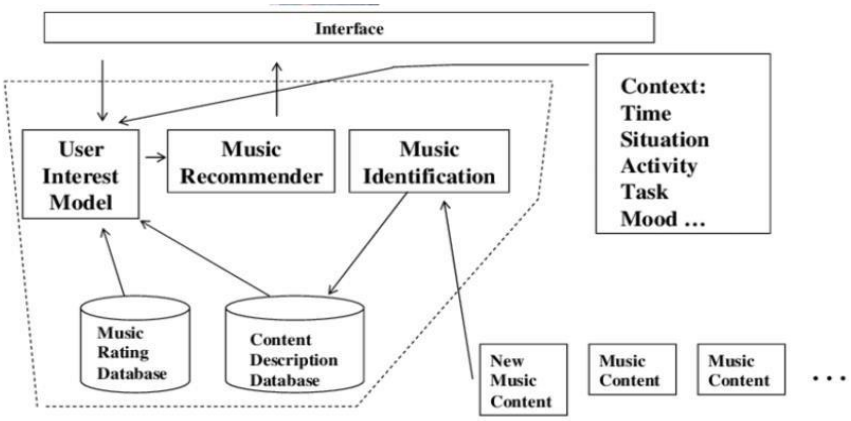
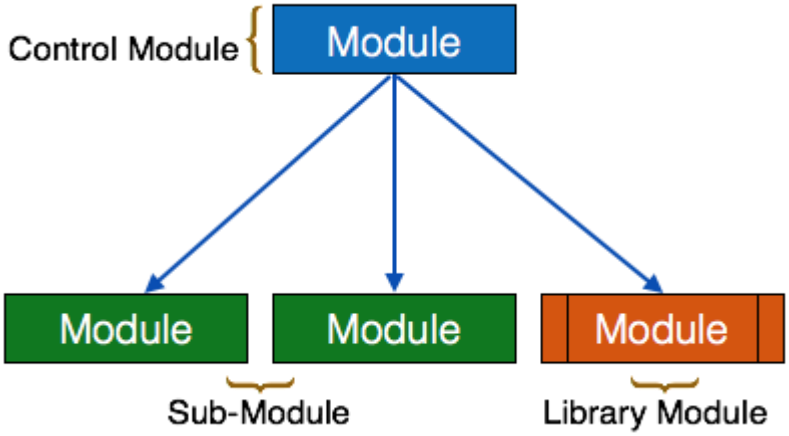


Fig 4.5 DFD

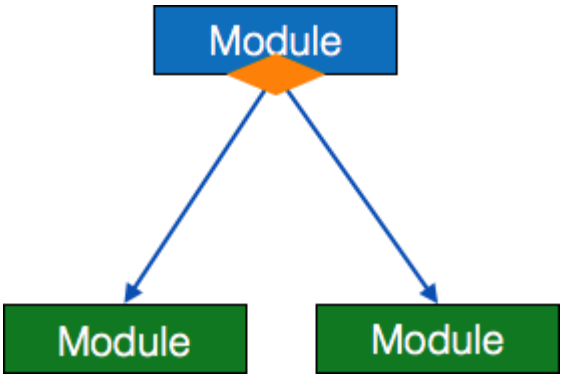
4.6 Structure charts

Structure chart is a chart derived from Data Flow Diagram. It represents the system in more detail than DFD. It breaks down the entire system into lowest functional modules, describes functions and subfunctions of each module of the system to a greater detail than DFD.
Structure chart represents hierarchical structure of modules. At each layer a specific task is performed. Here are the symbols used in construction of structure charts -

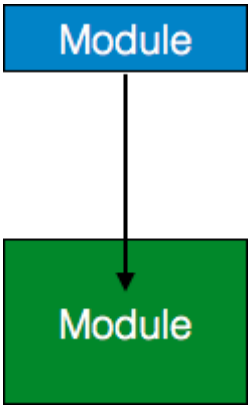
- Module - It represents process or subroutine or task. A control module branches to more than one submodule. Library Modules are re-usable and invokable from any module.



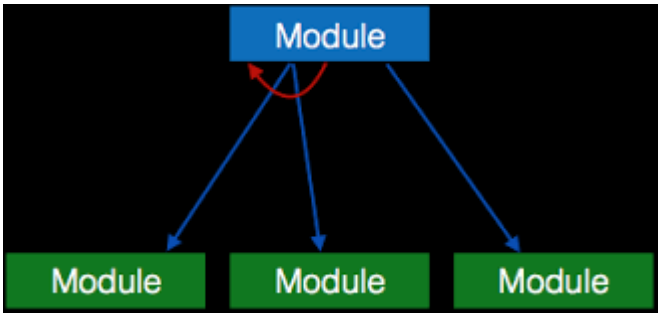
- Condition - It is represented by small diamond at the base of module. It depicts that control module can select any of sub-routine based on some condition.



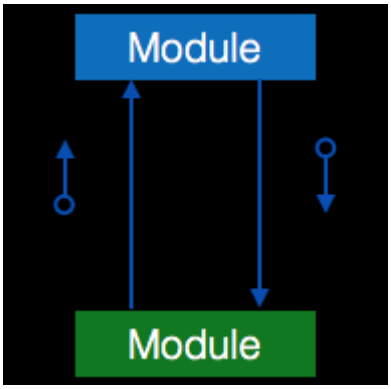
- Jump - An arrow is shown pointing inside the module to depict that the control will jump in the middle of the sub-module.



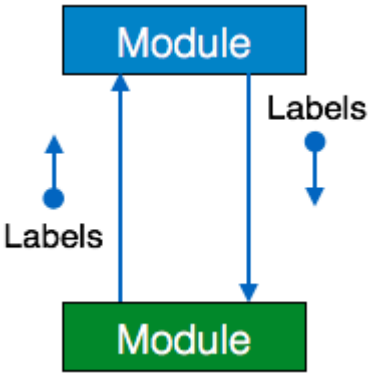
- Loop - A curved arrow represents loop in the module. All sub-modules covered by loop repeat execution of module.



- Data flow - A directed arrow with empty circle at the end represents data flow.



- Control flow - A directed arrow with filled circle at the end represents control flow.



4.7 System design

System specification

Process model used: Agile Model

The meaning of Agile is swift or versatile. Agile process model refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

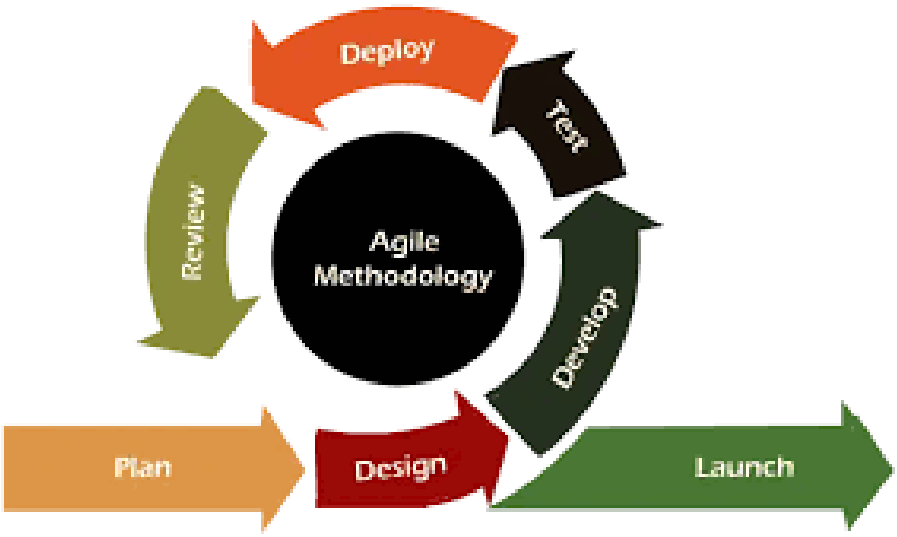


Fig 4.6 System specification

Phases of agile model:

1. Requirements gathering: In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

2. Design and requirements: When you have identified the project, work with stakeholders to define requirements. You can use flow diagrams or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.

3. Construction / iteration: When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

4. Testing: In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

5. Deployment: In this phase, the team issues a product for the user's work environment.

6. Feedback: After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

CHAPTER-5

REQUIREMENTS

5.1 USER REQUIREMENTS

User Requirements for Song Recommendation Chatbot:

- Personalization: Users expect the chatbot to provide personalized song recommendations based on their unique musical preferences, tastes, and past listening history.
- Mood-based Recommendations: Users want the chatbot to recommend songs that align with their current mood or emotions, allowing them to find music that suits their state of mind.
- Music Genre Selection: Users desire the ability to specify their preferred music genres or styles to receive recommendations that cater to their specific musical preferences.
- Contextual Factors: Users appreciate the chatbot considering contextual factors such as the time of day, weather, or activity to generate more relevant song suggestions that fit the current situation.
- Exploration and Discovery: Users expect the chatbot to introduce them to new songs and artists, providing recommendations beyond their existing favorites to encourage music exploration and discovery.
- Customization and Control: Users want the ability to provide feedback on recommendations, refine their preferences, and have control over the recommendations they receive. They may also seek options to create custom playlists or save favorite songs.
- Integration with Music Platforms: Users desire seamless integration with popular music streaming platforms or services, allowing them to easily listen to recommended songs within their preferred music app.
- Natural Language Interaction: Users expect the chatbot to engage in natural language conversations, understanding their queries, commands, and preferences without the need for rigid or specific input formats.
- Real-Time Responsiveness: Users appreciate a chatbot that provides quick and real-time responses, minimizing latency and ensuring a smooth and efficient user experience.

- User Privacy and Data Protection: Users want assurance that their personal information, music listening history, and preferences will be kept confidential and protected by the chatbot.

5.2 Hardware and software requirements

- Processor: Intel i3 and above
- Language used : Python , kivy Framework
- RAM: 4.00 GB and above
- CPU: 2.00 GHz and above
- OS: Windows 10 or iOS
- System Type: 64 bit operating system
- IDE: Pycharm

CHAPTER-6

TESTING AND DEBUGGING

Software testing is a critical element of the ultimate review of specification design and coding. Testing of software leads to the uncovering of errors in the software functional and performance requirements are met. Testing also provides a good indication of software reliability and software quality as a whole. The result of different phases of testing are evaluated and then compared with the expected results. If the errors are uncovered, they are debugged and corrected. A strategy approach to software testing has the generic characteristics:

- Testing begins at the module level and works "outwards" towards the integration of the entire computer-based system.
- Different testing techniques are appropriate at different points of time.
- Testing and debugging are different activities, but debugging must be accommodated in the testing strategy.

6.1 Unit testing

The module interface is tested to ensure that information properly flows into and out of the program unit under test. The unit testing is normally considered as an adjunct step to coding step. Because modules are not a standalone program, drivers and/or stubs software must be developed for each unit. A driver is nothing more than a "main program" that accepts test cases data and passes it to the module. A stub serves to replace the modules that are subordinate to the modules to be tested. A stub may do minimal data manipulation, prints verification of entry and returns.

Approaches used for Unit Testing were:

Functional Test: Each part of the code was tested individually and the panels were tested individually on all platforms to see if they are working properly.

Performance Test: These determined the amount of execution time spent on various parts of units and the resulting throughput, response time given by the module.

Stress Test: A lot of test files were made to work at the same time in order to check how much workloads can the unit bear.

Structure Test: These tests were made to check the internal logic of the program and traversing particular execution paths.

6.2 Integration testing

If they all work individually, they should work when we put them together. The problem of course is "putting them together". This can be done in two ways:

Top-down integration: Modules are integrated by moving downwards through the control hierarchy, beginning with main control module are incorporated into the structure in either a depth first or breadth first manner.

Bottom-up integration: It begins with construction and testing with atomic modules i.e. modules at the lowest level of the program structure. Because modules are integrated from the bottom up, processing required for the modules subordinate to a given level is always available and the need of stubs is eliminated.

Testing includes Verification and Validation

Verification:-is a process of confirming that software meets its specification.

Validation:- is the process of confirming that software meets the customer's requirements.

6.3 System testing

System testing is a type of software testing that evaluates the overall functionality and performance of a complete and fully integrated software solution. It tests if the system meets the specified requirements and if it is suitable for delivery to the end-users. This type of testing is performed after the integration testing and before the acceptance testing.

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behaviour of a component or a system when it is tested.

System testing process:

System Testing is performed in the following steps:

- Test Environment Setup: Create testing environment for the better quality testing.
- Create Test Case: Generate test case for the testing process.
- Create Test Data: Generate the data that is to be tested.

- Execute Test Case: After the generation of the test case and the test data, test cases are executed.
- Defect Reporting: Defects in the system are detected.
- Regression Testing: It is carried out to test the side effects of the testing process.
- Log Defects: Defects are fixed in this step.
- Retest: If the test is not successful then again test is performed

6.4 Acceptance testing

It is formal testing according to user needs, requirements, and business processes conducted to determine whether a system satisfies the acceptance criteria or not and to enable the users, customers, or other authorized entities to determine whether to accept the system or not. Acceptance Testing is the last phase of software testing performed after System Testing and before making the system available for actual use.

6.5 Debugging

Debugging occurs as a consequence of successful testing i.e. when a test case uncovers an error, debugging is the process that results in identifying the location of error ad the removal of error. The poorly understood mental process that connects a symptom to cause is debugging.

This process will always have one of the two outcomes.

- The cause will be found, corrected and then removed or
- The cause will not be found. In the latter case the person performing debugging may suspect a cause, design a test case to help validate his suspicion, and then work towards the correction of errors in the interactive fashion.

Following three approaches of debugging were used:

- Debugging by Induction
- Debugging by Deduction
- Backtracking

CHAPTER-7

TESTING AND TEST RESULTS

7.1 Negative testing table

Module Name	Component Tested	Action on component	Outcome
Mood Tunes	Send button	Unable to click	Error Prompt
Mood Tunes	Play button	Unable to click or play music	Error Prompt
Mood Tunes	Pause button	Unable to pause music	Error Prompt
Mood Tunes	Stop button	Unable to stop music	Error Prompt
Mood Tunes	Shuffle button	Unable to shuffle	Error Prompt

7.2 Positive testing table

Module name	Component tested	Action on component	Outcome
Mood Tunes	Send button	Click	Process user input and generate a response in the chatbot interface.
Mood Tunes	Play button	Click	The chatbot enables seamless and interactive music playback
Mood Tunes	Pause button	Click	Pause song playback and display paused status in the chatbot

			interface.
Mood Tunes	Stop button	Click	Stop song playback and reset the playback position in the chatbot interface.
Mood Tunes	Shuffle button	Click	Randomize playlist with single click on shuffle button.

CHAPTER-8

IMPLEMENTATION

Once the system was tested, the implementation phase started. A crucial phase in the system development life cycle is successful implementation of new system design. Implementations simply mean converting new system design into operation. This is the moment of truth the first question that strikes in every one's mind that whether the system will be able to give all the desires results as expected from system. The implementation phase is concerned with user training and file conversion. The term implementation has different meanings, ranging from the conversion of a basic application to a complete replacement of computer system Implementation is used here to mean the process of converting a new or revised system design into an operational one. Conversion is one aspect of implementation. The other aspects are the post implementation review and software maintenance. There are three types of implementations:

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

8.1 Modules

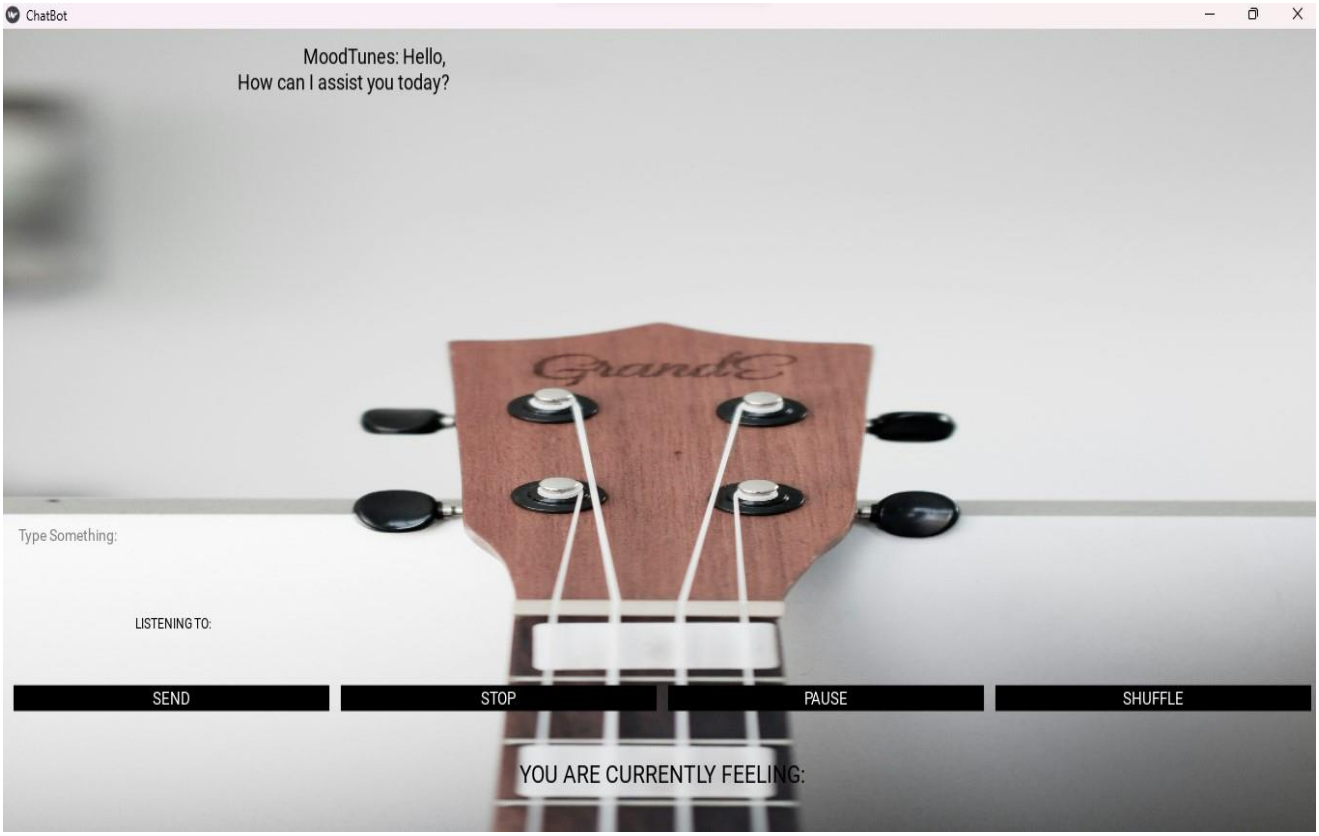
In computer software, a module is an extension to a main program dedicated to a specific function. In programming, a module is a section of code that is added in as a whole or is designed for easy reusability

The proposed system of "Chatbot-song recommendation system" has the following modules:

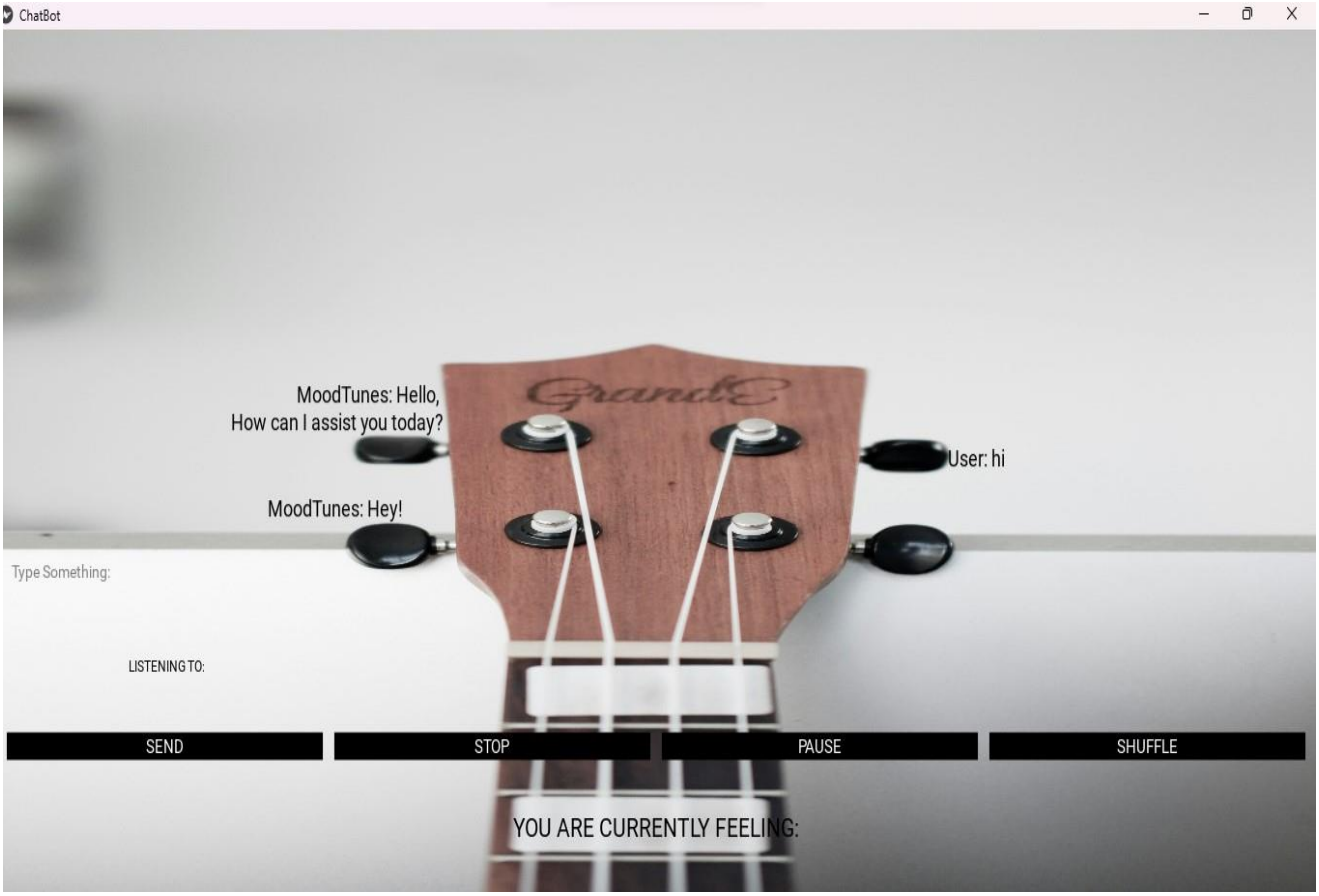
1. Mood Analyzer
2. Song Recommender
3. Chatbot

CHAPTER-9

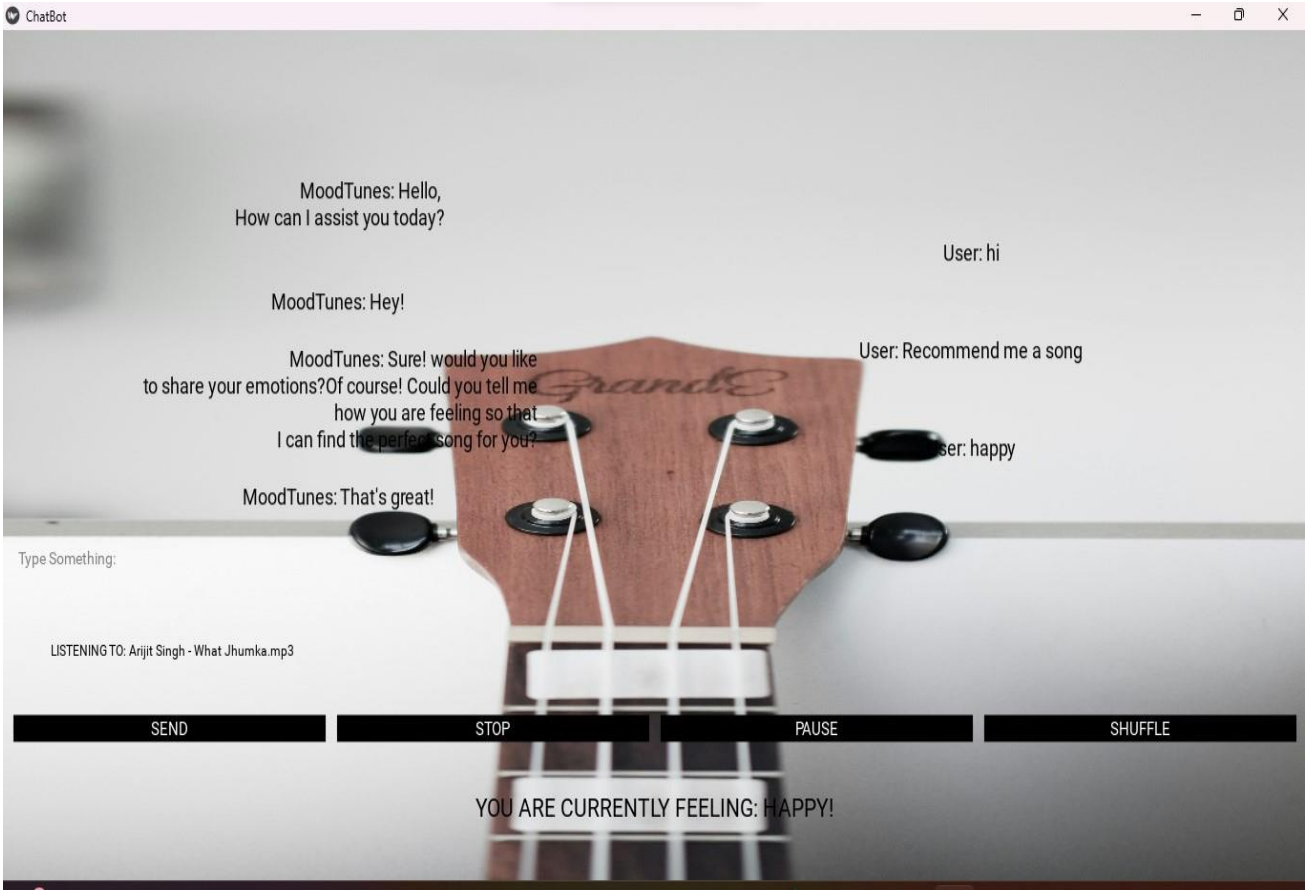
SCREENSHOTS



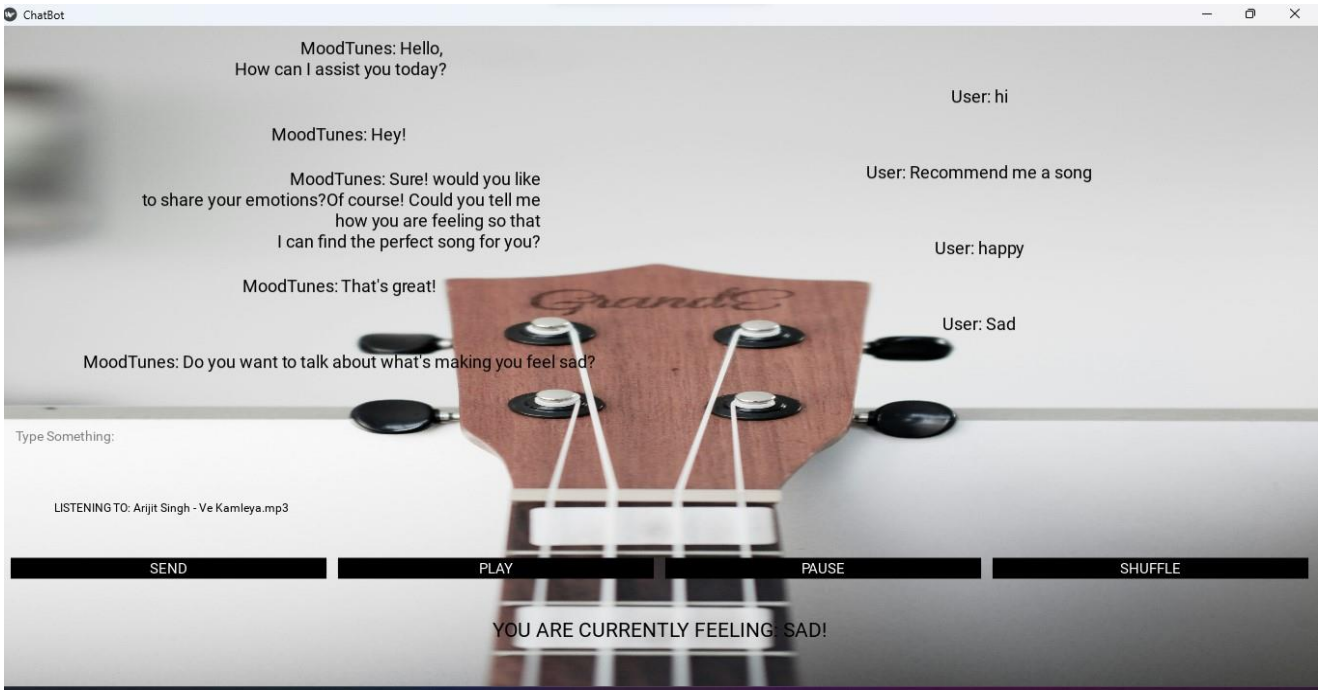
This is the front page that will appear when the chatbot app will open. The chatbot is displaying the text “Hello, How can I assist today?”.

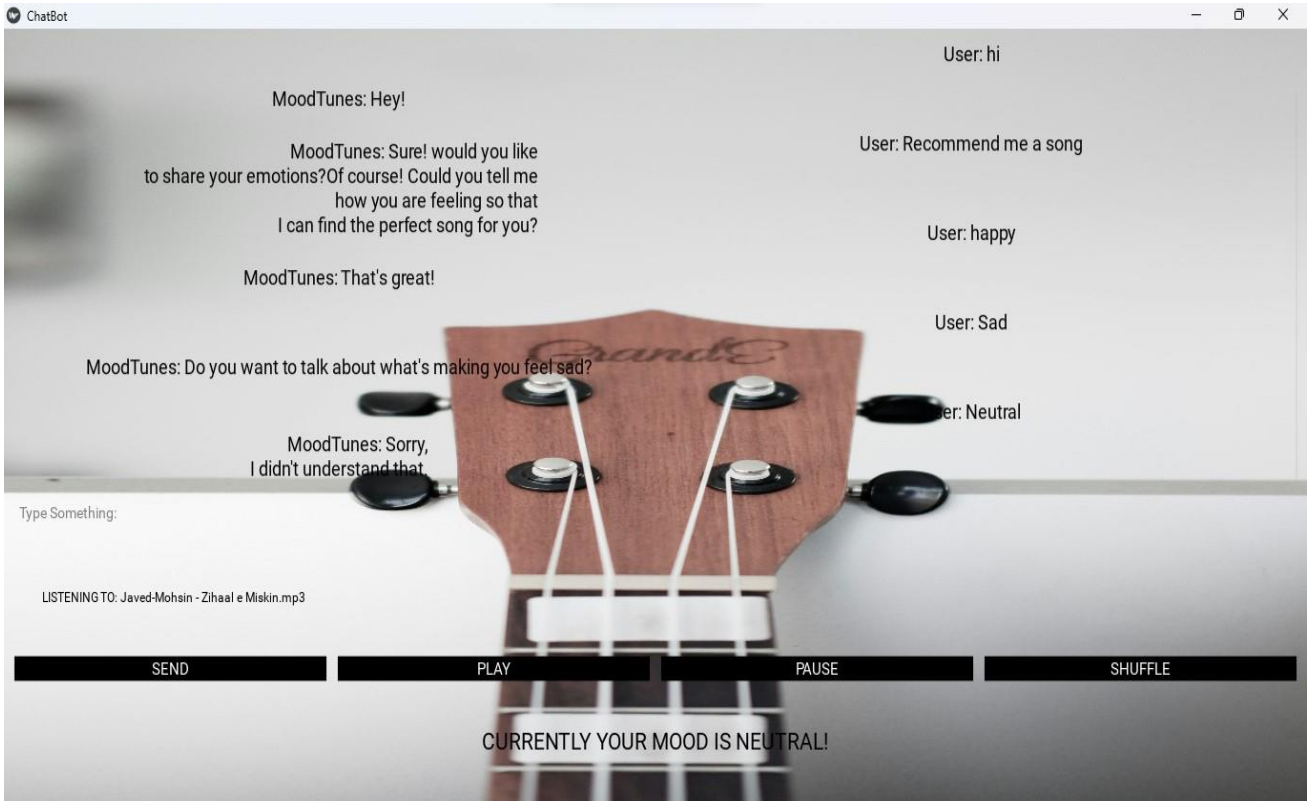


The user sends “hi” to the chatbot. Then chatbot replies back with “Hey” message.



When the user send “Recommend me a song” message . The Chatbot replies back with “Sure! Would you like to share your emotions? Of course! Could you tell me how you are feeling so that I can find the song according to your motions”.


ChatBot
MoodTunes: Hello,
How can I assist you today?
User: hi
MoodTunes: Hey!
MoodTunes: Sure! would you like
to share your emotions?Of course! Could you tell me
how you are feeling so that
I can find the perfect song for you?
User: Recommend me a song
User: happy
MoodTunes: That's great!
User: Sad
MoodTunes: Do you want to talk about what's making you feel sad?
Type Something:
LISTENING TO: Arijit Singh - Ve Kamleya.mp3
SEND
PLAY
PAUSE
SHUFFLE
YOU ARE CURRENTLY FEELING: SAD!


ChatBot
User: hi
MoodTunes: Hey!
MoodTunes: Sure! would you like
to share your emotions?Of course! Could you tell me
how you are feeling so that
I can find the perfect song for you?
User: Recommend me a song
User: happy
MoodTunes: That's great!
User: Sad
MoodTunes: Do you want to talk about what's making you feel sad?
er: Neutral
MoodTunes: Sorry,
I didn't understand that
Type Something:
LISTENING TO: Javed-Mohsin - Zihaal e Miskin.mp3
SEND
PLAY
PAUSE
SHUFFLE
CURRENTLY YOUR MOOD IS NEUTRAL!

CONCLUSION

In conclusion, the Song Recommendation Chatbot project has successfully developed an intelligent system that revolutionizes the way users discover and enjoy music. Through the integration of natural language processing, machine learning algorithms, and personalized recommendation techniques, the chatbot has been able to provide users with accurate and tailored song suggestions based on their musical preferences, moods, and contextual factors. The chatbot's ability to understand user input, recognize intents, and extract relevant information has allowed for dynamic and engaging conversations. Looking ahead, the project sets the foundation for future advancements and enhancements in the realm of music recommendation chatbots. Continued integration with music streaming platforms, expansion of the recommendation algorithms, and incorporation of novel features will further refine the chatbot's capabilities and ensure an exceptional user experience. Ultimately, the Song Recommendation Chatbot project has successfully created a powerful tool that empowers users to explore new music, customize their listening experience, and forge a deeper emotional connection with the songs that touch their hearts.

REFERENCES

- https://www.google.com/
- https://www.javatpoint.com/software-engineering-sdlc-models
- https://www.freecodecamp.org/
- Egencia (2018). What is a Chatbot and How does it work? Retrieved March 9, 2019 from: https://www.youtube.com/watch?v=38sL6pADCog
- Hattie, J. (2012). Visible learning for teachers: Maximizing impact on learning: Routledge. https://chatbotsmagazine.com/a-visual-history-of-chatbots-8bf3b31dbfb2 ● https://www.youtube.com/
- https://stackoverflow.com/
- Wikipedia (2019). Chatbot. Retrieved March 9, 2019 from: http//en.wikipedia.org/wiki/Chatbot