GAYATRI VIDYA PARISHAD

COLLEGE FOR DEGREE AND PG COURSES (A)

(Affiliated to Andhra University)

Rushikonda, Visakhapatnam

Department of Computer Applications

VISION

"Creating human excellence for a better society"

MISSION

"Unfold into a world class organization with strong academic and research base, producing responsible citizens to cater to the changing needs of the society."

MUSIC AND GAME RECOMMENDATION BASED ON EMOTION IN VOICE

A project report submitted in partial fulfilment of
the requirements for the award of the Degree of

Master of Computer Applications

Submitted by

K. Manoj Krishna

(Regd. No: PG222302032)

Under the Guidance of

Sri A. V. Prabhakar

Assistant Professor

Department of Computer Applications

GAYATRI VIDYA PARISHAD

COLLEGE FOR DEGREE AND PG COURSES(A)

(Affiliated to Andhra University)

Rushikonda, Visakhapatnam.

2022-2024

GAYATRI VIDYA PARISHAD
COLLEGE FOR DEGREE AND PG COURSES (A)
(Affiliated to Andhra University)
Rushikonda, Visakhapatnam
Department of Computer Applications

CERTIFICATE

This is to certify that the project report titled "**MUSIC AND GAME RECOMMENDATION BASED ON EMOTION IN VOICE"** is the bonafide record of project work carried out by **Mr. K. Manoj Krishna (**Regd. No. PG-222302032**)** as a student of this college, during the academic year 2022-2024, in partial fulfillment of the requirement for the award of the degree of Master of Computer Applications.

Project Guide
Sri A. V. Prabhakar

Head of Department
Prof. I. S. Pallavi

External Examiner

DECLARATION

I, Mr. K. Manoj Krishna hereby declares that the project report titled "**MUSIC AND GAME RECOMMENDATION BASED ON EMOTION IN VOICE",** is an original work done at **Gayatri Vidya Parishad College for Degree and PG Courses (Autonomous), Visakhapatnam,** submitted in partial fulfillment of the requirements for the award of Master of Computer Science, Gayatri Vidya Parishad College for Degree and PG Courses (A), affiliated to Andhra University. I assure that this project is not submitted in any other University or college.

K. MANOJ KRISHNA
(PG222302032)

ACKNOWLEGMENT

I consider it as a privilege to thank all those who helped me a lot for successful completion of the project "**MUSIC AND GAME RECOMMENDATION BASED ON EMOTION IN VOICE".**

I would like to thank **Prof. K.S,Bose, I/C Principal of Gayatri Vidya Parishad College for Degree and PG Courses(Autonomous)** who has provided full-fledged lab and infrastructure for successful completion of my project.

I would like to thank **Prof. I. S. Pallavi, Director of MCA**, who has obliged in responding to every request though she is busy with her hectic schedule of administration and teaching and suggested me with several changes a lot in completion of this project.

I would like to thank our ever-accommodating my project guide **Sri A.V. Prabhakar, Assistant Professor**, Department of Computer Applications,who has been very obliged in responding to every request though he is busy with his hectic schedule of teaching.

I thank all the **Teaching and Non-teaching staff** who has been aconstant source of support and encouragement during the study tenure.

(K. MANOJ KRISHNA)

ABSTRACT

ABSTRACT

Emotion plays a crucial role in human communication, significantly impacting our interactions and decision-making processes. This project explores the innovative integration of Speech Emotion Recognition (SER) technology with personalized entertainment recommendations. By analyzing vocal inputs, the system identifies the speaker's emotional state using advanced machine learning algorithms. Once the emotion is detected, the system recommends tailored music playlists and games that align with the user's current mood. This fusion of SER with personalized entertainment aims to enhance user experiences by providing emotionally congruent content, fostering emotional well-being, and potentially offering therapeutic benefits. Our approach leverages a robust dataset for training the SER model and employs collaborative filtering techniques for recommendation, ensuring relevance and personalization. This project not only showcases the potential of emotion-aware technologies but also paves the way for more empathetic and responsive digital environments. By extracting and classifying audio features, the system identifies emotions such as happiness, sadness, anger, fear, and neutrality. This project combines audio preprocessing, feature extraction, deep learning models, and adaptive algorithms to create a robust and responsive user experience. The applications of this technology are diverse, ranging from entertainment and gaming to emotional well-being support, showcasing the transformative potential of AI and ML in understanding and responding to human emotions. Moreover, this system can be integrated into various platforms, from mobile apps to virtual assistants, further expanding its usability. Future developments could explore real-time emotion recognition and multi-modal inputs, incorporating facial expressions and body language to enhance accuracy and personalization even further.

CONTENTS

INTRODUCTION

1. INTRODUCTION

In the real world, we are surrounded by humans who can learn everything from their experiences with their learning capability, and we have computers or machines which work on our instructions.A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.

The need for machine learning is increasing day by day. The reason behind the need for machine learning is that it is capable of doing tasks that are too complex for a person to implement directly. As a human, we have some limitations as we cannot access the huge amount of data manually, so for this, we need some computer systems and here comes the machine learning to make things easy for us.

We can train machine learning algorithms by providing them the huge amount of data and let them explore the data, construct the models, and predict the required output automatically. The performance of the machine learning algorithm depends on the amount of data, and it can be determined by the cost function. With the help of machine learning, we can save both time and money.

The importance of machine learning can be easily understood by its use's cases, Currently. machine learning is used in self-driving cars, cyber fraud detection, face recognition and friend suggestion by Facebook, etc. Various top companies such as Netflix and Amazon have built machine learning models that are using a vast amount of data to analyse the user interest and recommend product accordingly.

1.1 Introduction to Machine Learning

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models to unable parameters

that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

1.2 Categories of Machine Leaning

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.Supervised learning involves somehow modelling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modelling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

As its name suggests, Supervised machine learning is based on supervision. It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output. Here, the labelled data specifies that some of the inputs are already mapped to the output. More preciously, we can say; first, we train the machine with the input and corresponding output, and then we ask the machine to predict the output using the test dataset.

Let's understand supervised learning with an example. Suppose we have an input dataset of cats and dog images. So, first, we will provide the training to the machine to understand the images, such as the shape & size of the tail of cat and dog. Shape of eyes, colour, height (dogs are taller, cats are smaller), etc. After completion of training, we input the picture of a cat and ask the machine to identify the object and predict the output. Now, the machine is well trained, so it will check all the features of the object, such as height, shape, color, eyes, ears, tail, etc.,

and find that it's a cat. So, it will put it in the Cat category. This is the process of how the machine identifies the objects in Supervised Learning

1.3 Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, Al is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn. The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. These data-driven decisions can be used, instead of using programming logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

1.4 Focuses in Machines Learning

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of Al as whole still has a long way to go. The reason behind is that ML. has not been able to overcome number of challenges. The challenges that ML is facing currently are-

Quality of Data - Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming Task - Another challenge faced by ML. models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of Specialist Persons - As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No Clear Objective for Formulating Business Problems - Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of Overfitting & Underfitting - If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of Dimensionality - Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in Deployment - Complexity of the ML model makes it quite difficult to be deployed in real life.

1.5 Applications of Machines Learning

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of Al and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

1.6 Machine Learning Techniques

Artificial Intelligence (AI) and Machine Learning (ML) encompass a wide range of techniques and methodologies that enable machines to perform tasks that typically require human intelligence. These techniques are at the heart of many modern applications, from voice assistants and chatbots to advanced medical diagnostics and autonomous vehicles.

One of the fundamental techniques in AI is the use of algorithms to process and analyze data. These algorithms can be broadly categorized into traditional AI algorithms and those used in machine learning. Traditional AI algorithms include search algorithms, which are used to navigate through data to find specific information or solutions to problems. Examples include the A* search algorithm, which is used in pathfinding and graph traversal, and the minimax algorithm, which is used in decision-making processes, particularly in game theory and competitive environments.

Machine learning algorithms, on the other hand, are designed to enable systems to learn from data and improve their performance over time. These algorithms can be divided into three

main types: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves training a model on a labeled dataset, meaning that each training example is paired with an output label. The model learns to map inputs to the correct output based on this training data. Common algorithms used in supervised learning include linear regression, logistic regression, support vector machines (SVM), and decision trees. Neural networks, a more advanced form of supervised learning, are particularly powerful for tasks involving complex and high-dimensional data, such as image and speech recognition.

Unsupervised learning deals with unlabeled data, meaning the algorithm tries to find patterns and relationships within the data without any explicit instructions on what to look for. Clustering algorithms, such as k-means and hierarchical clustering, are used to group similar data points together. Dimensionality reduction techniques, like principal component analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE), are used to reduce the number of variables under consideration and to visualize high-dimensional data. Unsupervised learning is particularly useful for exploratory data analysis and for finding hidden patterns or intrinsic structures in the data.

Reinforcement learning is a type of machine learning where an agent learns to make decisions by performing actions in an environment to maximize some notion of cumulative reward. Unlike supervised learning, where the correct answer is provided during training, reinforcement learning relies on a reward signal to evaluate the actions taken by the agent. The agent receives feedback in the form of rewards or penalties and uses this feedback to learn the best strategy or policy to achieve its goals. This approach is highly effective in dynamic environments where the optimal actions are not always apparent and must be discovered through trial and error. Reinforcement learning has been successfully applied to a wide range of problems, including robotics, game playing, and autonomous driving.

Another critical technique in AI and ML is the use of neural networks, which are designed to simulate the way the human brain processes information. Neural networks consist of layers of interconnected nodes, or neurons, that process data in a hierarchical manner. The most basic form of a neural network is the feedforward neural network, where information flows in one direction from the input layer to the output layer. More complex architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have been developed to handle specific types of data and tasks. CNNs are particularly effective for image processing tasks due to their ability to capture spatial hierarchies in images, while RNNs are suited for sequential data, such as time series or natural language, because they can maintain information about previous inputs through their recurrent connections.

Deep learning, a subset of machine learning, refers to the use of neural networks with many layers (hence "deep") to model complex patterns in data. The increased depth of these networks allows them to learn more abstract and high-level features from raw data, which has led to significant advancements in fields such as computer vision, natural language processing, and speech recognition. Training deep learning models typically requires large amounts of data and computational resources, but the results have been groundbreaking, enabling the development of AI systems that can outperform humans in certain tasks.

Overall, the techniques in AI and ML are diverse and continually evolving, driven by advances in computational power, availability of data, and innovative research. These techniques form the backbone of intelligent systems that are transforming industries and shaping the future of technology.

1.7 Future Directions

Artificial Intelligence (AI) and Machine Learning (ML) have profoundly impacted numerous sectors, revolutionizing how we approach problem-solving and decision-making. Their influence extends across industries such as healthcare, finance, transportation, and entertainment, driving significant advancements and efficiencies. In healthcare, AI and ML are transforming diagnostic and treatment processes. AI-driven diagnostic tools can analyze medical images with remarkable accuracy, often outperforming human radiologists in detecting diseases like cancer. Machine learning models predict patient outcomes and recommend personalized treatment plans based on vast amounts of historical data. These technologies are also accelerating drug discovery by identifying potential drug candidates and predicting their efficacy, significantly reducing the time and cost involved in bringing new medications to market. Moreover, AI-powered wearable devices continuously monitor patients' vital signs, enabling early detection of health issues and timely medical interventions.

The financial sector has also witnessed substantial changes due to AI and ML. These technologies enhance fraud detection by identifying unusual patterns in transaction data, thereby preventing fraudulent activities. In trading, machine learning algorithms analyze market data to forecast stock prices and inform investment strategies, often executing trades at high speeds and with greater precision than human traders. Furthermore, AI-driven chatbots and virtual assistants provide personalized customer service, handling routine inquiries and transactions, which frees up human agents for more complex tasks.

Transportation is another field greatly impacted by AI and ML. Autonomous vehicles, which rely on sophisticated machine learning models to navigate and make real-time decisions, promise to reduce accidents caused by human error and improve traffic flow. AI algorithms

optimize logistics and supply chain management by predicting demand, managing inventory, and routing deliveries efficiently. These advancements lead to cost savings and enhanced customer satisfaction through faster and more reliable services.

In the entertainment industry, AI and ML are used to create personalized user experiences. Streaming services like Netflix and Spotify employ machine learning algorithms to analyze users' viewing and listening habits, recommending content tailored to individual preferences. AI-driven tools are also used in content creation, such as generating realistic graphics in video games or producing original music and scripts. Additionally, sentiment analysis tools gauge audience reactions to movies, shows, and advertisements, helping creators and marketers refine their content to better meet audience expectations.

The impact of AI and ML extends beyond these sectors, influencing areas such as education, agriculture, and environmental conservation. In education, adaptive learning platforms use machine learning to tailor educational content to students' individual learning styles and paces, improving learning outcomes. In agriculture, AI-powered systems monitor crop health, optimize irrigation, and predict yields, enhancing productivity and sustainability. Environmental conservation efforts benefit from AI's ability to analyze data from sensors and satellite images, tracking wildlife populations and detecting illegal activities like poaching and deforestation.

Looking to the future, AI and ML are poised to drive further innovations and societal changes. One key area of development is the advancement of explainable AI (XAI), which aims to make AI systems more transparent and understandable to humans. As AI systems become more complex, ensuring that their decision-making processes are interpretable and trustworthy is crucial, particularly in high-stakes domains like healthcare and finance. Another promising direction is the integration of AI with the Internet of Things (IoT). IoT devices generate vast amounts of data, and AI can analyze this data to derive insights and make intelligent decisions in real-time. This synergy has applications in smart cities, where AI-powered systems manage energy consumption, traffic flow, and public safety, creating more efficient and livable urban environments.

AI and ML will also play a significant role in addressing global challenges such as climate change and pandemics. Machine learning models can predict climate patterns, optimize renewable energy sources, and improve disaster response strategies. In public health, AI can assist in monitoring disease outbreaks, developing vaccines, and managing healthcare resources more effectively. Ethical considerations and regulatory frameworks will be critical as AI and ML continue to evolve. Ensuring that these technologies are developed and deployed

responsibly, with attention to issues such as bias, privacy, and job displacement, will be essential to maximizing their benefits while mitigating potential risks. Collaborative efforts between governments, industry, and academia will be necessary to create policies and standards that promote ethical AI development and use.

1.8 Challenges and Limitations of Machine Learning

Artificial Intelligence (AI) and Machine Learning (ML) have made significant strides in recent years, but they still face several challenges and limitations. Understanding these issues is crucial for developing more robust and ethical AI systems.

1. Technical Challenges

- Overfitting and Underfitting: One of the key challenges in ML is achieving a balance between overfitting and underfitting. Overfitting occurs when a model learns the noise in the training data rather than the underlying pattern, resulting in poor performance on new, unseen data. Underfitting happens when a model is too simple to capture the complexity of the data. Finding the right model complexity and regularization techniques is essential for optimal performance.
- Scalability: As datasets grow larger and more complex, scaling ML algorithms becomes challenging. Training models on massive datasets requires substantial computational resources and time. Ensuring that algorithms can efficiently handle large-scale data while maintaining performance is a significant hurdle.
- Data Quality and Quantity: The effectiveness of ML models depends heavily on the quality and quantity of the data. Inadequate or biased data can lead to inaccurate or skewed predictions. Data preprocessing, cleaning, and augmentation are critical to ensure that models are trained on high-quality data that accurately represents the problem domain.

2. Data Privacy and Security

- Privacy Concerns: AI systems often require access to large amounts of personal data, raising concerns about data privacy. Ensuring that sensitive information is protected and used responsibly is a major challenge. Techniques like anonymization and federated learning are being developed to address privacy concerns, but balancing privacy with model effectiveness remains a complex issue.
- Data Security: AI systems are vulnerable to security threats such as adversarial attacks, where malicious inputs are designed to fool the model into making incorrect predictions.

Ensuring the robustness of AI systems against such attacks and protecting them from data breaches is crucial for maintaining trust and security.

3. Ethical and Bias Issues

- Algorithmic Bias: AI and ML systems can inadvertently perpetuate or even exacerbate existing biases present in the training data. For instance, biased training data can lead to biased outcomes in areas such as hiring, law enforcement, and credit scoring. Identifying and mitigating biases in AI models is essential for promoting fairness and equity.
- Ethical Considerations: The deployment of AI systems raises various ethical concerns, including the potential for misuse, impacts on employment, and decision-making transparency. Ensuring that AI systems are designed and used ethically involves addressing questions about accountability, transparency, and the broader societal impact of AI technologies.

4. Interpretability and Explainability

- Black-Box Nature: Many advanced ML models, particularly deep learning models, operate as "black boxes," meaning their internal decision-making processes are not easily interpretable. This lack of transparency can hinder trust and make it difficult to understand how decisions are made. Developing techniques for model interpretability and explainability is essential for ensuring that AI systems are transparent and their decisions can be understood and justified.
- Model Interpretability: For AI systems to be widely accepted and trusted, it is crucial that their predictions and decision-making processes are interpretable by humans. Efforts to enhance model interpretability involve creating methods and tools that provide insights into how models arrive at their conclusions, which is especially important in high-stakes domains like healthcare and finance.

5. Societal Impact and Public Perception

- Job Displacement: The automation of tasks through AI and ML can lead to job displacement, as machines and algorithms increasingly perform tasks previously done by humans. Addressing the impact on employment and developing strategies for workforce retraining and support are important for mitigating the negative effects of automation.
- Public Perception: Public perception of AI and ML can vary, with concerns about the potential misuse of technology and its impact on daily life. Building public trust involves transparent communication about how AI systems work, their benefits, and their limitations.

LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 Introduction

Integrating artificial intelligence (AI) and machine learning (ML) into personalized content recommendation systems has led to innovative applications such as music and game recommendations based on the emotion detected in a user's voice. Emotion recognition from voice involves analyzing vocal features like tone, pitch, and intensity to classify emotions such as happiness, sadness, anger, and calmness using advanced algorithms trained on datasets like CREMA and RAVDESS. Once a user's emotional state is identified, the system can suggest music or games that either match or alter their mood, enhancing the user's experience. For example, happy emotions may prompt upbeat music or fast-paced games, while sadness might lead to comforting songs or story-driven games. This approach not only increases user satisfaction by providing a personalized and engaging experience but also demonstrates the potential of adaptive technologies in understanding and responding to individual emotional needs.

2.2 Existing System

- The current system for emotion recognition in voice focuses on accurately identifying emotional states such as happiness, sadness, anger, and calmness by analyzing vocal features like tone, pitch, speed, and intensity.
- Utilizing advanced AI and ML models trained on comprehensive datasets like CREMA and RAVDESS, this system achieves high precision in emotion detection. However, it does not extend its functionality to recommend personalized content like songs or games based on the detected emotions.
- This limitation means that while the system effectively recognizes and categorizes emotions, it stops short of leveraging this information to enhance user experiences through tailored content recommendations.
- Without integrating recommendation capabilities, the system misses the opportunity to create a more interactive and emotionally responsive user interface that could improve user engagement and satisfaction.
- Consequently, there remains significant potential for future development to bridge this gap and fully harness the power of emotion recognition technology. By incorporating personalized content suggestions, the system could provide a more holistic and dynamic user experience that responds to individual emotional states.

2.3 Problem Statement

- The current emotion recognition system accurately identifies emotional states from voice inputs but fails to utilize this information to enhance user experiences through personalized content recommendations.
- This limitation restricts the system's potential to create a more engaging and emotionally responsive user interface.

2.4 Proposed System

- The proposed system builds upon the existing emotion recognition capabilities by incorporating personalized content recommendations for songs and games based on the detected emotional states.
- Utilizing the same advanced AI and ML models trained on datasets like CREMA and RAVDESS for accurate emotion detection, this enhanced system analyzes vocal features to classify emotions such as happiness, sadness, anger, and calmness.
- Upon identifying a user's emotional state, the system leverages a recommendation engine to suggest music tracks and games tailored to the user's current mood.
- For instance, when a happy emotion is detected, the system might recommend upbeat and energetic songs or fast-paced, fun games.
- Conversely, if sadness is identified, the system could suggest comforting and empathetic music or story-driven, immersive games. This integration aims to create a more personalized and engaging user experience by aligning entertainment options with the user's emotional needs.
- By providing responsive and adaptive content recommendations, the proposed system enhances user satisfaction and engagement, offering a holistic and emotionally attuned interaction that significantly improves upon the limitations of the current system.

2.5 Objectives

- Achieve high accuracy in detecting emotions from voice inputs.
- Provide personalized music recommendations based on detected emotions.
- Provide personalized game recommendations based on detected emotions.
- Measure user satisfaction with recommendations.
- Ensure real-time processing and response for recommendations.
- Maintain a user-friendly interface for input and feedback.
- Integrate feedback mechanisms to continuously improve recommendation quality.

2.6 About Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

2.6.1 Advantages of Python

Python is a high-level, interpreted, interactive and object-oriented scripting lan-guage. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages. Python is a must for students and working professionals to become a great Soft-ware Engineer especially when they are working in Web Development Domain. I will list down some of the key advantages of learning Python:

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

2.6.2 Characteristics of Python

- Following are important characteristics of Python Programming −
- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

2.6.3 New Approach for Building window Software

The Python Framework simplifies Windows development. It provides developers with a single approach to build both desktop applications sometimes called smart client ap-plications

and Web-Based applications. It also developers to use the same tools and skills to develop software for a verity of system ranging from handled smart phones to large server installations.

2.6.4 Applications of Python

As mentioned before, Python is one of the most widely used language over the web. I'm going to list few of them here:

- Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- Easy-to-read − Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain − Python's source code is fairly easy-to-maintain.
- A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Interactive Mode − Python has support for an interactive mode which allows inter-active testing and debugging of snippets of code.
- Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Extendable − You can add low-level modules to the Python interpreter. These mod-ules enable programmers to add to or customize their tools to be more efficient.
- Databases − Python provides interfaces to all major commercial databases.
- GUI Programming − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- Scalable − Python provides a better structure and support for large programs than shell scripting.

2.6.5 Python – GUI (programming)

Python provides various options for developing graphical user interfaces (GUIs). Most important are listed below.

- Tkinter − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.
- wxPython − This is an open-source Python interface for wxWin-dows http://wxpython.org.
- JPython − JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine.

2.6.6 Python Libraries

Python libraries are pre-written code packages that provide various functionalities, making programming more efficient and reducing the need to write code from scratch. They encompass a wide range of applications, from data analysis and machine learning to web development and scientific computing.

- **filedialog**: Provides dialogs for file selection and saving.
- **messagebox**: Displays pop-up messages and dialogs in Tkinter applications.
- **ttk**: Offers themed widgets in Tkinter for a modern UI look.
- **Image**: Part of the Pillow library, used for image processing.
- **ImageTk**: Integrates PIL images with Tkinter for GUI display.
- **pygame**: A library for creating games and multimedia applications.
- **spotipy**: A lightweight Python library for accessing the Spotify Web API.
- **webbrowser**: Allows opening URLs in a web browser from Python scripts.
- **os**: Provides a way to interact with the operating system, including file and directory manipulation.
- **requests**: Simplifies making HTTP requests to web services.
- **librosa**: A library for audio and music analysis.
- **numpy as np**: Provides support for large, multi-dimensional arrays and matrices, along with mathematical functions.
- **threading**: Facilitates concurrent execution of code using threads.

2.6.7 Tkinter Programming

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.

2.6.8 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 −

- Button : The Button widget is used to display buttons in your application.
- Canvas :The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.
- Check button :The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.
- Entry :The Entry widget is used to display a single-line text field for accepting val-ues from a user.
- Frame : The Frame widget is used as a container widget to organize other widgets.
- Label : The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
- Listbox : The Listbox widget is used to provide a list of options to a user.
- Menubutton : The Menubutton widget is used to display menus in your application.
- Menu : The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.
- Message : The Message widget is used to display multiline text fields for accepting values from a user.
- Radiobutton : The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.
- Scale : The Scale widget is used to provide a slider widget.
- Scrollbar: The Scrollbar widget is used to add scrolling capability to various widg-ets, such as list boxes.
- Text: The Text widget is used to display text in multiple lines.
- Toplevel: The Toplevel widget is used to provide a separate window container.
- 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.
- PanedWindow: A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.
- LabelFrame: A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
- tkMessageBox:This module is used to display message boxes in your applications.

2.7 Functional Requirements

The emotion-based music and game recommendation system must allow users to upload audio files for emotion prediction and generate corresponding music and game

recommendations. It should integrate with APIs like Spotify for playlist suggestions and handle various audio qualities and languages. The system needs an intuitive user interface for audio uploads, viewing recommendations, and submitting feedback.

Input:

- Audio File Input:

  Format: Accept common audio formats like WAV, and also accept dynamic recording.

  Length: Handle varying lengths, from short clips to longer recordings.

Output:

- Predicted Emotion:

  Emotion Classes: Output a predefined emotion category (e.g., Happy, Sad, Angry, Neutral).

  Confidence Scores: Provide a confidence score indicating the certainty of the prediction.

- Recommendations Based on Predicted Emotion:

  Playlist Recommendation: Suggest music playlists tailored to the detected emotion using Spotify API.

  Game Recommendation: Recommend games that match the predicted emotion for enhanced engagement.

Software Requirements:

- Operating System: Windows
- Programming Languages: Python
- Libraries and Frameworks: TensorFlow, Librosa, Tkinter, Numpy, Pandas
- Development Tools: Python IDLE (Python 3.12 64-bit).

Hardware Requirements:

- Processor: Multi-core processor (Intel i5/i7 or AMD Ryzen 5/7).
- RAM: Minimum 8 GB, recommended 16 GB or more for model training.
- Storage: SSD with at least 256 GB free space.

2.8 Non-Functional Requirements

The system must be reliable, scalable, and perform efficiently to handle varying loads. It should offer a user-friendly interface and ensure data security through robust authentication and authorization. Additionally, it must comply with privacy standards and provide minimal latency in recommendations.

- Reliability: The system should operate continuously without unexpected failures.
- Scalability: The system must handle an increasing number of users and data smoothly.

- Performance Efficiency: The system should provide fast response times with minimal latency.
- Usability: The interface should be intuitive and easy for users to navigate.
- Security: The system must protect user data with strong authentication and authorization measures.

UML MODELLING

3. UML MODELLING

3.1 Introduction to UML

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software system. UML was created by the Object Management Group (OMG) and UML 1.0 specification drift was proposed to the OMG in January 1997.OMG is continuously making efforts to create a truly industry standard.

- UML stands for Unified Modelling Language.
- UML is different from the other common programming language such as C++, java, COBOL
- UML is a pictorial language used to make software blueprints.
- UML can be described as a general-purpose visual modelling language to visualize, specify, construct and document software system.
- Although UML is generally used to model software system, it is not limited within this boundary. It is generally used to model software system as well. For example, the process flows in a manufacturing unit.

UML is not a programming language, but tools can be used to generate code in various language using UMI diagrams. UML has a direct relation with object-oriented analysis and design. After some standardization, UML has become an OMG standard.

3.2 Goals of UML

- A picture is worth a thousand words, this idiom absolutely fits describing UML. Object-oriented concepts were introduced much earlier than UML.
- At that point of time, there were no standard methodologies to organize and consolidate the Object-oriented development. It was then that UML came into picture.
- There are number of goals for developing UML but the most important is to define some general-purpose modelling language.
- In which all models can use and it also need to be made simple to understand and use.

3.3 UML Standard Diagrams

The elements are like components which can be associated in diverse ways to make a complete UML picture, which is known as diagram. Thus, it is very important to understand the different diagrams to implement the knowledge in real-life system. Any complex system is best understood by making some kind of diagrams or pictures. These diagrams have a better impact on our understanding. If we look around, we will realize that the diagram is not a new

concept. But it is used widely in different forms in different industries. We prepare UML diagram to understand the system in a better and simple way. A single diagram is not enough to cover all the aspects of the system. UMI. Defines various kinds of diagrams to cover most of the aspects of a system. You can also create your own set of diagrams to meet your requirements. Diagrams are generally made in an incremental and iterative way. There are two broad categories of diagram and they are again divided into subcategories:

Structural Diagrams

Behavioural Diagrams

3.3.1 Structural Diagrams

The structural diagram represents the static aspect of the system. These static aspects represent those parts of a diagram, which forms the main structure and are therefore stable. These static parts are represented by classes, interfaces, object, components, and nodes. The four structural diagrams are:

- Class diagram
- Object diagram
- Component diagram
- Deployment diagram

3.3.2 Behavioural Diagrams

Any system can have two aspects, static and dynamic. So, a model is considered as complete when both the aspects are fully covered. Behavioural diagram captures the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts of a system. UML has the following five types of behavioural diagrams:

- Use case diagram
- Sequence diagram
- Activity diagram

3.4 UML DIAGRAMS

3.4.1 Introduction to Use Case Diagram

Use cases are used during requirement elicitation and analysis to represent the functionality of the system. Use cases focus on the behaviour of the system from an external point of view. A use case describes a function provided by the system that yields a visible result for an actor. An actor describes any entity that interacts with the system.

3.4.1.1 Actors

Actors represent external entities that interact with the system. An actor can be human or an external system. During this activity, developers identify the actors involved in this system. In this project, user is :

User: Any person

3.4.1.2 Use Cases

The identification of actors and use cases results in the definition of the boundary of the system, which is, in differentiating the tasks accomplished by the system and the tasks accomplished by its environment. The actors are outside the boundary of the system, whereas the use cases are inside the boundary of the system.

Actors are external entities that interact with the system. Use cases describe the behaviour of the system as seen from an actor's point of view. Actors initiate a use case to access the system functionality. The use case then initiates other use cases and gathers more information from the actors. When actors and use cases exchange information, they are said to be communicate.To describe a use case, we use a template composed of six fields

Use Case Name	: The name of the use case
Participating Actors	: The actors participating in the particular use case
Flow of events	: Sequence of steps describing the function of use case
Exit Condition	: Condition for terminating the use case

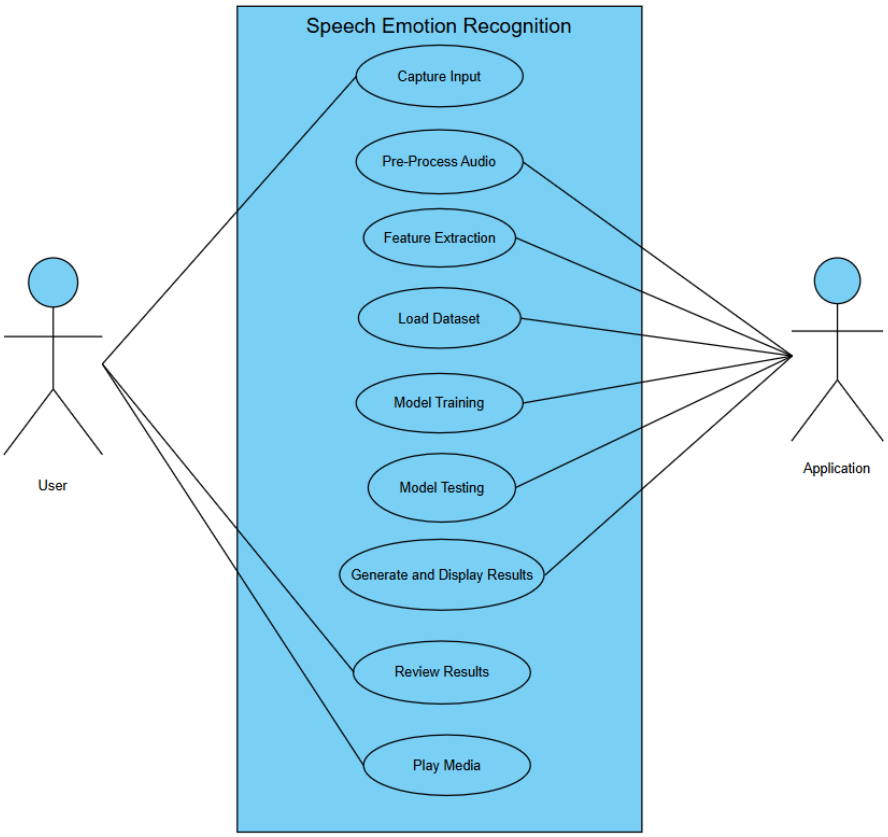


Fig. 3.4.1.2 Use Case Diagram

Description:

User: The user will upload the dataset and selects the model for classification and training.

System: System will check the dataset and it preprocess the data therefore it splits into training and testing and predict the results

Use case for Capture Input :

Use Case ID	UC001
Use case Name	Capture Input
Participating Actors	User
Entry Condition	User initiates the emotion recognition process by providing an audio input.
Flow of Events	▪ User opens the application. ▪ User records or uploads an audio file.
Exit Condition	Audio input is successfully received and ready for pre-processing.
Quality Requirements	Audio input should be clear and of sufficient length to capture the user's emotion.

Table 3.4.1.2(a): Capture Input

Use case for Pre-process Audio :

Use Case ID	UC002
Use case Name	Pre-process Audio
Participating Actors	Application
Entry Condition	Audio input is received
Flow of Events	▪ Application processes the audio to remove noise. ▪ Audio is normalized and segmented if necessary.
Exit Condition	Pre-processed audio data is ready for feature extraction.
Quality Requirements	Pre-processing should retain the integrity of the original audio's emotional content.

Table 3.4.1.2(b): Pre-process Audio

Use case for Feature Extraction :

Use Case ID	UC003
Use case Name	Feature Extraction
Participating Actors	Application
Entry Condition	Pre-processed audio is available.
Flow of Events	▪ Application extracts relevant features from the audio (e.g., pitch, tone, rhythm). ▪ Features are formatted for model input.
Exit Condition	Features are successfully extracted and stored.
Quality Requirements	Feature extraction should be accurate and relevant to emotion detection.

Table 3.4.1.2naïve: Feature Extraction

Use case for Load Dataset :

Use Case ID	UC004
Use case Name	Load Dataset
Participating Actors	Application
Entry Condition	Features are ready for comparison or model training.
Flow of Events	▪ Application loads the dataset for model training or comparison. ▪ Dataset is prepared for model consumption.
Exit Condition	Dataset is successfully loaded.
Quality Requirements	Dataset should be comprehensive and representative of different emotions.

Table 3.4.1.2(d): Load Dataset

Use case for Model Training :

Use Case ID	UC005
Use case Name	Model Training
Participating Actors	Application
Entry Condition	Dataset is loaded.
Flow of Events	▪ Application initializes the model training process. ▪ Model is trained using the loaded dataset. ▪ Model parameters are optimized.
Exit Condition	Model is successfully trained.
Quality Requirements	Training should be efficient and yield a high accuracy rate for emotion recognition.

Table 3.4.1.2(e): Model Training

Use case for Model Testing :

Use Case ID	UC006
Use case Name	Model Testing
Participating Actors	Application
Entry Condition	Model is trained.
Flow of Events	▪ Application tests the model using test data. ▪ Model performance is evaluated.
Exit Condition	Model testing is complete with performance metrics.
Quality Requirements	Testing should accurately reflect real-world performance.

Table 3.4.1.2(f): Model Testing

Use case for Generate and Display Results :

Use Case ID	UC007
Use case Name	Generate and Display Results
Participating Actors	Application
Entry Condition	Model testing is complete.
Flow of Events	▪ Application processes the test results. ▪ Emotion predictions are generated and displayed ▪ Recommends games and playlists from spotify app
Exit Condition	User views the predicted emotion.
Quality Requirements	Output generation should be quick and accurate and display should be user-friendly and clear. Recommendations should be relevant to the user's emotional state

Table 3.4.1.2(g): Generate and Display Results

Use case for Review Results :

Use Case ID	UC008
Use case Name	Review Results
Participating Actors	User
Entry Condition	Results are displayed.
Flow of Events	▪ User views the results. ▪ User can interact with the results for further analysis.
Exit Condition	User has reviewed the results.
Quality Requirements	Results should be easy to interpret and detailed.

Table 3.4.1.2(h): Review Results

Use case for Play Media :

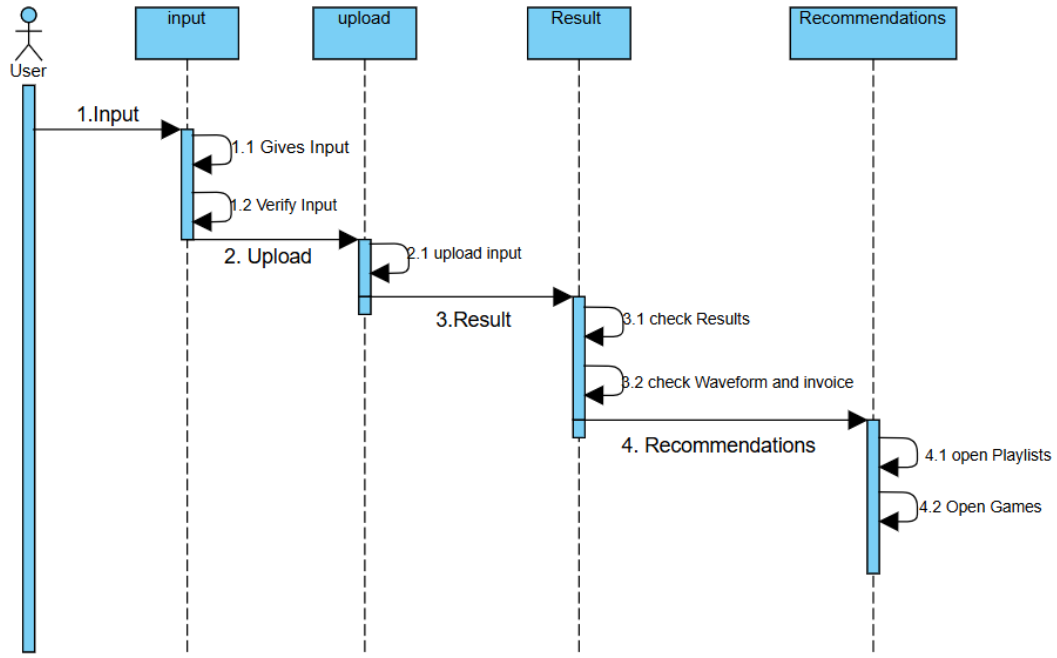
Use Case ID	UC009
Use case Name	Play Media
Participating Actors	User
Entry Condition	User views the results
Flow of Events	▪ User selects the option to play songs or games. ▪ Application opens spotify & play songs or games based on the predicted emotion.
Exit Condition	Songs or games are played based on the user's selection.
Quality Requirements	Provide a good user experience.

Table 3.4.1.2(i): Play Media

3.4.2 Sequence Diagram

A sequence diagram is the most commonly used interaction diagram. It simply depicts interaction between objects in a sequential order ie. The order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems. Sequence diagrams can be useful references for businesses and other organizations. Purpose of sequence diagrams are:

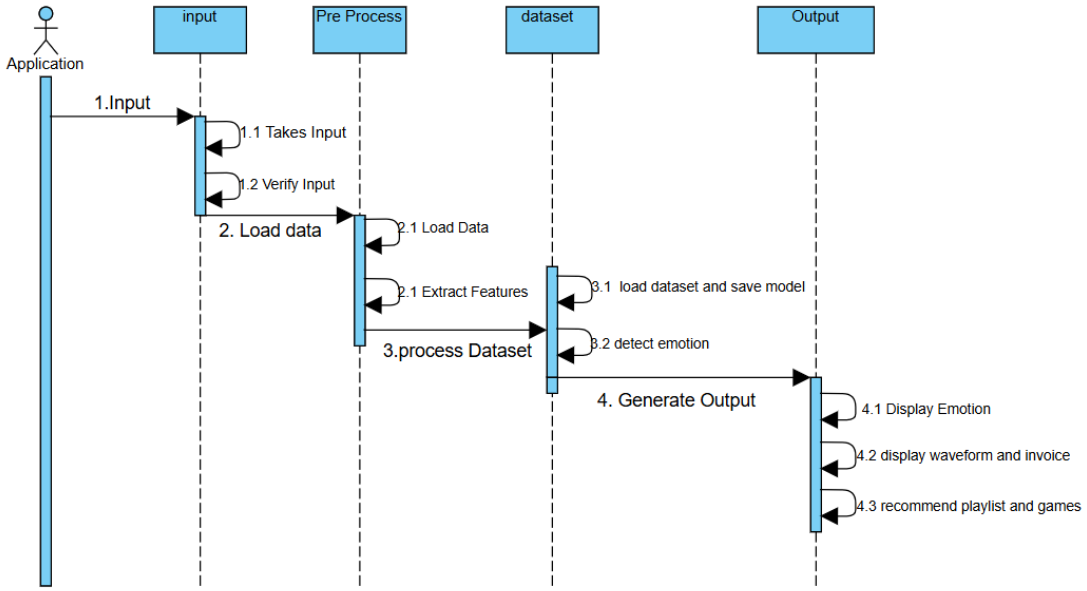
- Represent the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario.



Fig, 3.4.2(a) Sequence Diagram for User

Description:

The sequence diagram describes the flow of events in which the user participates in the process of the execution of application. The user gives input by uploading a audio file or by recording and then after predicting the results he reviews and verify it and interact further.



Fig, 3.4.2(b) Sequence Diagram for System

Description:

The sequence diagram describes the flow of events in which the system participates in the process of the execution of application. The system takes input uploaded by user and then predicts the results by comparing features extracted from the input taken with the features in the model trained and saved. Then the system further go on with the user queries.

3.4.3 Activity Diagram

An activity diagram is a type of UML (Unified Modelling Language) diagram that represents the flow of activities within a system or process. Activity diagrams are valuable tools for visualizing and understanding workflows, making them essential for process modelling and software development. It is used to model the dynamic aspects of a system by showing the sequence of actions or steps and their flow of control.

Key Components of an Activity Diagram:

- Activities: Represent tasks or actions taken in the system. They are depicted as rounded rectangles.
- Transitions: Indicate the flow from one activity to another. They are shown as arrows connecting the activities.
- Start (Initial) Node: Denotes the beginning of the workflow. It is shown as a filled black circle.

- End (Final) Node: Marks the end of the workflow. It is depicted as a filled black circle inside a larger unfilled circle.
- Decision Nodes: Represent points where the workflow can branch based on a condition. They are shown as diamonds with outgoing arrows labeled with conditions.
- Merge Nodes: Combine multiple flows into a single flow. They are also shown as diamonds but with multiple incoming arrows and one outgoing arrow.
- Fork Nodes: Split a flow into multiple concurrent flows. They are depicted as thick horizontal or vertical bars with one incoming arrow and multiple outgoing arrows.
- Join Nodes: Synchronize multiple concurrent flows into a single flow. They are also shown as thick bars with multiple incoming arrows and one outgoing arrow.
- Swim lanes: Divide the diagram into columns or rows to represent different actors, departments, or systems that perform the activities. Each swim lane contains the activities performed by the corresponding entity.

Steps to Create an Activity Diagram:

- Identify the Process: Define the process or system you want to model.
- Define the Activities: List out all the activities or tasks involved in the process.
- Determine the Flow: Establish the sequence of activities and how they flow from one to another.
- Add Decision Points: Identify any points where decisions are made, leading to different branches in the workflow.
- Incorporate Forks and Joins: Include any concurrent activities and how they synchronize.
- Use Swim lanes: If applicable, divide the diagram using swim lanes to represent different actors or systems.
- Validate the Diagram: Ensure that the diagram accurately represents the process and that the flow is logical.

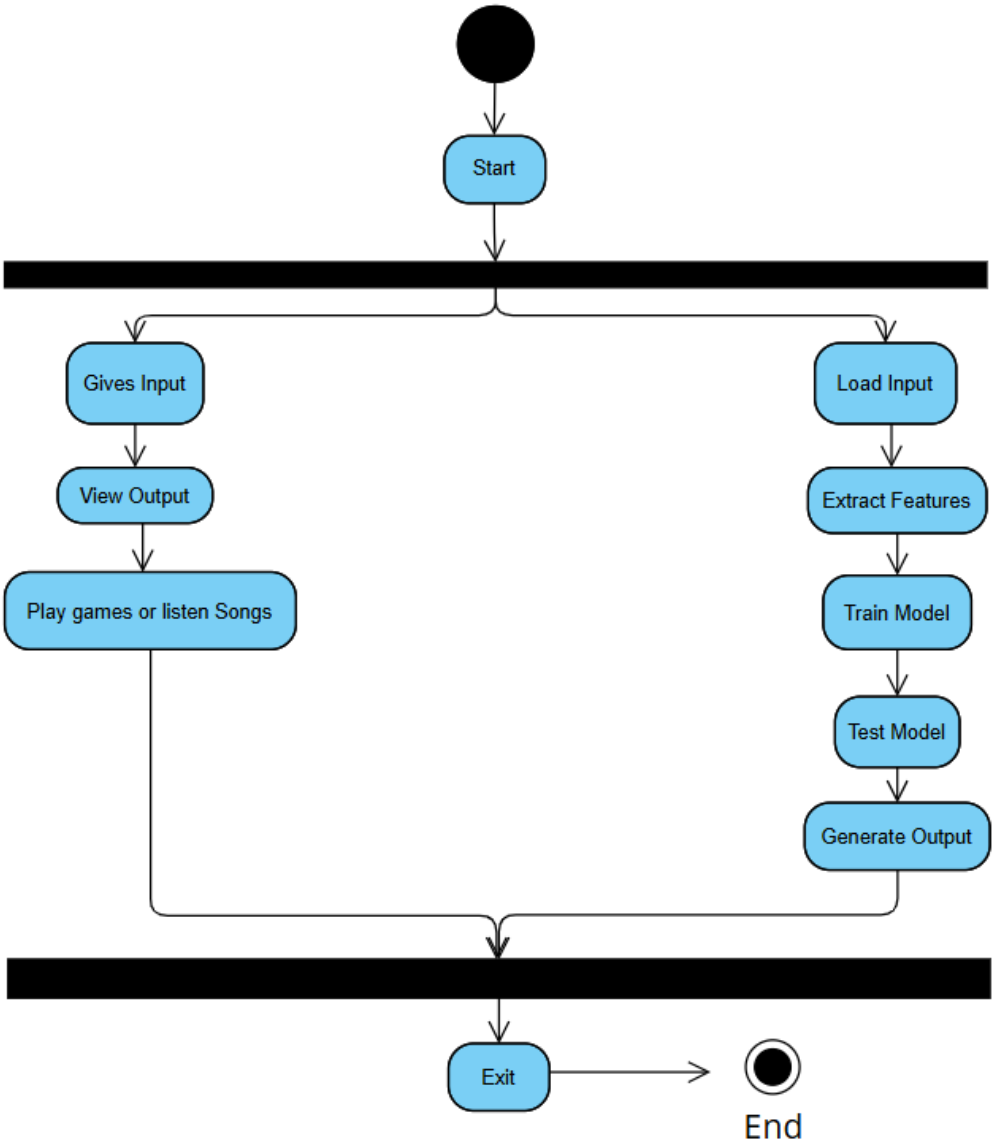


Fig. 3.4.3 Activity Diagram

Description:

The activity diagram for the speech emotion recognition project starts with the user inputting an audio file, which undergoes preprocessing (noise reduction and feature extraction). The preprocessed audio features are fed into a trained model to predict the emotion, and based on this prediction, suitable playlists and games are recommended to the user. Finally, the results are displayed to the user.

DESIGN

4. DESIGN

System Design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. In System design, developers:

- Define design goals of the project
- Decompose the system into smaller sub systems
- Design hardware/software strategies
- Design persistent data management strategies
- Design global control flow strategies
- Design access control policies and
- Design strategies for handling boundary conditions.
- System design is not algorithmic. It is decomposed of several activities. They are:
- Identify Design Goals
- Design the initial subsystem decomposition
- Refine the subsystem decomposition to address the design goals.

System Design is the transform of analysis model into a system design model. Developers define the design goals of the project and decompose the system into smaller subsystems that can be realized by individual teams. Developers also select strategies for building the system, such as the hardware/software platform on which the system will run, the persistent data management strategy, the goal control flow the access control policy and the handling of boundary conditions. The result of the system design is model that includes a clear description of each of these strategies, subsystem decomposition, and a UML deployment diagram representing the hardware/software mapping of the system.

4.1 Design Goals

Design goals are the qualities that the system should focus on. Many design goals can be inferred from the non-functional requirements or from the application domain.

User friendly: The system is user friendly because it is easy to use and understand.

Reliability: Proper checks are there for any failure in the system if they exist.

4.2 System Architecture

As the complexity of systems increases, the specification of the system decomposition is critical. Moreover, subsystem decomposition is constantly revised whenever new issues are addressed. Subsystems are merged into alone subsystem, a complex subsystem is split into parts, and some subsystems are added to take care of new functionality. The first iterations over the subsystem decomposition can introduce drastic changes in the system design model.

4.3 System Design

The overall project aim is to Predict the personality through Machine Learning. With the advancement of artificial intelligence and machine learning (ML), autism can be predicted at quite early stage. The main aim of this project is to analyses various Machine learning algorithms, used by various researcher like Naïve Bayes and SVM and compare the result based on their accuracy and efficiency.

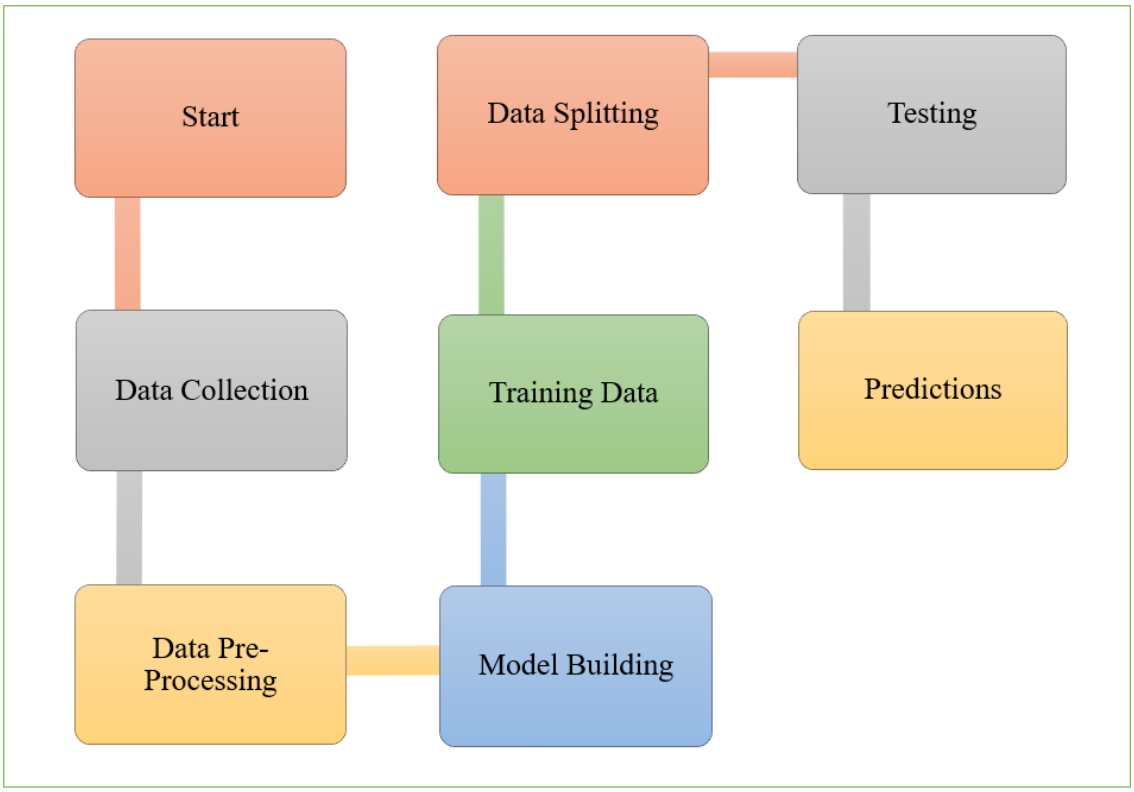


Fig. 4.3 System's Block Design

4.4 Implementation of Project

Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?

- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

Objectives

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow

Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to ther system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's mlationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner, the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively.

2. Create document, report, or other formats that contain information produced by the system. The output form of an information system should accomplish one or more of the following objectives.

Convey information about past activities, current status or projections of the Future.

Signal important events, opportunities, problems, or warnings.

- Trigger an action.
- Confirm an action.

4.5 Algorithms

CNN [Convolutional neural network] Algorithm :

Convolutional neural networks are a type of deep learning technology that has grown dominant in a number of tasks related to computer vision and is gaining interest in a wide range of areas, including radiology. Convolutional neural networks are made up of numerous building pieces, such as convolution layers, max - pooling, and fully connected layers, and are meant to learn spatial hierarchies of information automatically and adaptively via a backpropagation method. Understanding the ideas, benefits, and limits of convolutional neural networks is vital for capitalizing on their promise to enhance radiologist performance and, ultimately, patient care.

Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM):

RNNs are designed to process sequential data by maintaining a hidden state that captures information from previous time steps. However, traditional RNNs struggle with long-term dependencies due to issues like vanishing and exploding gradients.LSTMs address these challenges by introducing a specialized architecture with memory cells and gating mechanisms. LSTMs have three main components:

Forget Gate: Decides which information from the cell state should be discarded.

Input Gate: Determines which new information should be added to the cell state.

Output Gate: Controls how much of the cell state should be outputted to the next layer.

This architecture allows LSTMs to capture long-term dependencies and manage temporal relationships in data more effectively, making them well-suited for tasks like emotion recognition from voice, where understanding the context over time is crucial

CODING

5. CODING

5.1 Coding Approach

The objective of the coding or programming phase is to translate the design of the system produced during the design phase into code in a given programming language, which can be executed by a computer and that performs the computation specified by the design. The coding phase affects both testing and maintenance. The goal of coding is not to reduce the implementation cost, but the goal should be to reduce the cost of later phases.

There are two major approaches for coding any software system. They are Top-Down approach and bottom-up approach.

Bottom-up Approach can best suit for developing the object-oriented systems. During system design phase, we decompose the system into an appropriate number of subsystems, for which objects can be modelled independently. These objects exhibit the way the subsystems perform their operations.

Once objects have been modelled, they are implemented by means of coding. Even though related to the same system as the objects are implemented of each other, the Bottom-Up approach is more suitable for coding these objects. In this approach, we first do the coding of objects independently and then we integrate these modules into one system to which they belong.

This code will first allow the admin to login with a valid user id and password. After this, admin will select the admission or academic module and enter all the students' details branch wise. TPO will select placement module and enter the details based on academic performance. Reports are generated accordingly and can be viewed by teaching staff and students.

5.2 Verification and Validation

Verification is the process of checking the product built is right. Validation is the process of checking whether the right product is built. During the Development of the system, Coding for the object has been thoroughly verified from different aspects regarding their design, in the way they are integrated etc. The various techniques that have been followed for validation discussed in testing the current system.

Validations applied to the entire system at two levels:

Form level Validation: Validations of all the inputs given to the system at various points in the forms are validated while navigating to the next form. System raises appropriate custom and pre-defined exceptions to alert the user about the errors occurred or likely to occur.

Field level Validation: Validations at the level of individual controls are also applied wherever necessary. System pops up appropriate and sensuous dialogs wherever necessary. In this project, validations are performed on each individual control. If any one of text field is not filled or any wrong, click occurs then system will generate appropriate exceptions.

5.3 Source Code

```
import tkinter as tk
from tkinter import *
from tkinter import filedialog, messagebox, ttk
from PIL import Image, ImageTk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import tensorflow as tf, customtkinter as ctk, matplotlib.pyplot as plt, sounddevice as sd,
wavio, time, pygame, spotipy, webbrowser, os, requests, librosa, numpy as np, threading
from spotipy.oauth2 import SpotifyClientCredentials
from datetime import datetime
from tkinter import font
pygame.mixer.init()
model = tf.keras.models.load_model('emotion_recognition_model.h5')
EMOTIONS = ['angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'surprise']
IMAGE_DIR = 'C:/Users/annap/project/static/'
# Spotify API credentials
client_id = 'aaaa058c985d4a0a8c4748f3e2a141ad'
client_secret = '011e082ed763413ab497194745df9a10'
redirect_uri = 'http://localhost:8888/callback'  # Set your redirect URI
sp =
spotipy.Spotify(client_credentials_manager=SpotifyClientCredentials(client_id=client_id,
client_secret=client_secret))
current_audio_file = None
recorded_audio_file = 'recording.wav'
image_label = None
emotion_text_label = None
invoice_text = None
waveform_canvas = None
```

```
invoice_text_widget = None  # Define invoice_text_widget as a global variable
language_entry = None
playlists = []
show_waveform_button = None
show_invoice_button = None
menu_fg_color = "#000000"  # Black color for menubar and menu items
men_fg_color = "#ffffff"
content_frame=None
predicted_emotion=None
upload_frame=None
emotion_game_map = {
   "happy": [("Snake", "http://slither.com/io"),
             ("fighting", "https://krunker.io/")],
   "sad": [("Guess the word", "https://skribbl.io/"),
           ("Tetris Effect",
"https://store.steampowered.com/app/1003590/Tetris_Effect_Connected/")],
   "angry": [("shellshock", "https://shellshock.io/"),
             ("DOOM", "https://store.steampowered.com/app/379720/DOOM/")],
   "disgust": [("Minecraft", "https://www.minecraft.net/en-us"),
           ("Overcooked! 2", "https://store.steampowered.com/app/728880/Overcooked_2/")],
   "surprise": [("Minecraft", "https://www.minecraft.net/en-us"),
           ("Guess the word", "https://skribbl.io/"),],
   "fear": [("Snake", "http://slither.com/io"),
           ("Overcooked! 2", "https://store.steampowered.com/app/728880/Overcooked_2/")],
   "neutral": [("Minecraft", "https://www.minecraft.net/en-us"),
           ("Overcooked! 2", "https://store.steampowered.com/app/728880/Overcooked_2/")]
}
def open_game_link(url):
   webbrowser.open(url)
def show_info(message):
   messagebox.showinfo("Information", message
def show_error(message):
messagebox.showerror("Error", message)
def display_game_recommendations(emotion):
   recommendations = emotion_game_map.get(emotion, [])
```

```
    for widget in game_recommendation_frame.winfo_children():
        widget.destroy()
    if recommendations:
        for game, url in recommendations:
            link = tk.Label(game_recommendation_frame, text=game, fg="blue", cursor="hand2",
width=38, heigh=4)
            link.pack(pady=5)
            link.bind("<Button-1>", lambda e, url=url: open_game_link(url))
    else:
        no_recommendation = tk.Label(game_recommendation_frame, text="No game
recommendations available.")
        no_recommendation.pack()
def show_waveform():
    global current_audio_file, recorded_audio_file
    file_path = current_audio_file if current_audio_file else recorded_audio_file
    feature, audio, sample_rate = extract_features(file_path)
    display_waveform_window(audio, sample_rate)
def display_waveform_window(audio, sample_rate):
    global waveform_canvas
    try:
        waveform_window = tk.Toplevel()
        waveform_window.title("Waveform Display")
        waveform_window.geometry("800x600")
        fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(8, 6))
        ax1.plot(audio)
        ax1.set_title('Raw Waveform')
        ax1.set_xlabel('Sample')
        ax1.set_ylabel('Amplitude')
        mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)
        img = librosa.display.specshow(mfccs, x_axis='time', ax=ax2)
        fig.colorbar(img, ax=ax2)
        ax2.set_title('MFCCs')
        ax2.set_xlabel('Time')
        ax2.set_ylabel('MFCC Coefficient')
        waveform_canvas = FigureCanvasTkAgg(fig, master=waveform_window)
```

```
        waveform_canvas.draw()
        waveform_canvas.get_tk_widget().pack(side=tk.BOTTOM, fill=tk.BOTH,
expand=True)
        close_button = ctk.CTkButton(waveform_window, text="Close Waveform",
command=waveform_window.destroy)
        close_button.place(x=10, y=10)
    except Exception as e:
        show_error(f"Failed to display waveform: {e}")
def close_waveform():
    global waveform_canvas
    if waveform_canvas:
        waveform_canvas.get_tk_widget().destroy()
        waveform_canvas = None
    else:
        show_info("Waveform window is not currently open.")
def show_invoice():
    global current_audio_file, recorded_audio_file
    file_path = current_audio_file if current_audio_file else recorded_audio_file
    feature, audio, sample_rate = extract_features(file_path)
    predicted_emotion = EMOTIONS[np.argmax(model.predict(np.expand_dims(feature,
axis=0)))]
    generate_invoice_text(file_path, audio, sample_rate, predicted_emotion)
def generate_invoice_text(file_path, audio, sample_rate, emotion):
    global content_frame
    unique_id = str(int(time.time()))
    current_date = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    amplitude = np.max(audio)
    pitch = np.mean(librosa.core.pitch_tuning(librosa.core.piptrack(y=audio,
sr=sample_rate)[0]))
    duration = librosa.core.get_duration(y=audio, sr=sample_rate)
    global invoice_text_widget
    if invoice_text_widget:
        invoice_text_widget.destroy()
    invoice_frame = tk.Frame(content_frame, bd=10, relief=tk.GROOVE)
    invoice_frame.place(x=1300, y=200)
```

```
    header_text = f"""
    Invoice

    Invoice Number: {unique_id}
    Date and Time: {current_date}

    Description of Services:
    ------------------------------------------------
    Item: Audio Emotion Detection
    Description: Processing and predicting emotion
    ------------------------------------------------
    """
    header_label = tk.Label(invoice_frame, text=header_text, anchor="w", justify=tk.LEFT,
font=("Arial", 14, "bold"))
    header_label.grid(row=0, column=0, pady=10, sticky="w")
    details = [
        ("Amplitude:", f"{amplitude:.2f}"),
        ("Pitch:", f"{pitch:.2f}"),
        ("Frequency:", f"{duration:.2f} seconds"),
        ("Predicted Emotion:", emotion.capitalize())
    ]
    for i, (label_text, value_text) in enumerate(details):
        label = tk.Label(invoice_frame, text=label_text, anchor="w", justify=tk.LEFT)
        label.grid(row=i+1, column=0, sticky="w")
        value_label = tk.Label(invoice_frame, text=value_text, anchor="w", justify=tk.LEFT)
        value_label.grid(row=i+1, column=1, sticky="w")
    overlay_button = ctk.CTkButton(invoice_frame, text="close", command=close_invoice)
    overlay_button.grid(row=len(details)+2, column=0, pady=10, sticky="w")  # Adjust
position as     needed
    invoice_text_widget = invoice_frame
def close_invoice():
    global invoice_text_widget
    if invoice_text_widget:
        invoice_text_widget.destroy()
        invoice_text_widget = None
```

```
    else:
        show_info("Invoice display is not currently open.")
def display_image(emotion):
    global image_label, emotion_text_label, show_waveform_button, show_invoice_button,
content_frame, playlist_listbox, language_combobox, upload_frame
    image_path = IMAGE_DIR + emotion + ".jpeg"
    image = Image.open(image_path)
    image = image.resize((300, 300), Image.LANCZOS)
    photo = ImageTk.PhotoImage(image)
    if image_label is None:
        image_label = tk.Label(upload_frame, image=photo)
        image_label.image = photo
        image_label.place(x=700, y=210)
    else:
        image_label.config(image=photo)
        image_label.image = photo
    if emotion_text_label is None:
        emotion_text_label = ctk.CTkLabel(upload_frame, text=f"Predicted Emotion:
{emotion.capitalize()}", font=("Arial", 16), bg_color=men_fg_color,
text_color=menu_fg_color)
        emotion_text_label.place(x=580, y=130)
    else:
        emotion_text_label.configure(text=f"Predicted Emotion: {emotion.capitalize()}")
    if show_waveform_button is None:
        show_waveform_button = ctk.CTkButton(upload_frame, text="Show Waveform",
command=show_waveform, font=("Arial", 12))
    show_waveform_button.place(x=690, y=420)  # Adjust the x, y coordinates as needed
    if show_invoice_button is None:
        show_invoice_button = ctk.CTkButton(upload_frame, text="Show Invoice",
command=show_invoice, font=("Arial", 12))
    show_invoice_button.place(x=540, y=420)  # Adjust the x, y coordinates as needed
    playlist_label = ctk.CTkLabel(upload_frame, text="Recommended Playlists:",
font=("Arial", 14, "bold"))
    playlist_label.place(x=120, y=180)
```

```
    language_label = ctk.CTkLabel(upload_frame, text="Select Language:", font=("Arial", 14,
"bold"))
    language_label.place(x=120, y=150)
    language_combobox = ctk.CTkComboBox(upload_frame, values=["Telugu", "Hindi",
"English", "Spanish", "French", "German", "Italian"], font=("Arial", 12))
    language_combobox.place(x=250, y=150)
    language_combobox.set("Telugu")
    scrollbar = tk.Scrollbar(upload_frame, orient=tk.VERTICAL)
    playlist_listbox = tk.Listbox(upload_frame, yscrollcommand=scrollbar.set, font=("Arial",
12), width=38, height=10)
    scrollbar.configure(command=playlist_listbox.yview)
    for playlist in playlists:
        playlist_listbox.insert(tk.END, playlist['name'])
    playlist_listbox.place(x=150, y=270)
    scrollbar.place(x=500, y=270)
    playlist_listbox.bind("<Double-1>", open_playlist_url)
    i=Image.open("C:/Users/annap/project/static/refresh.jpg")
    ctk_image = ctk.CTkImage(light_image=i, size=(30, 30))
    refresh_button = ctk.CTkButton(upload_frame, image=ctk_image, text="",
command=refresh_playlists, font=("Arial", 12), width=30, height=30, fg_color="white")
    refresh_button.place(x=340, y=220)
    show_waveform_button.configure(state=tk.NORMAL)
    show_invoice_button.configure(state=tk.NORMAL)
def get_spotify_playlists(emotion):
    try:
        selected_language = language_combobox.get()
        query = f"{emotion} {selected_language}"
        results = sp.search(q=query, type='playlist', limit=5)
        playlists = []
        for playlist in results['playlists']['items']:
            playlists.append({
                'name': playlist['name'],
                'url': playlist['external_urls']['spotify']
            })
        return playlists
```

```
    except Exception as e:
        print(f"Error retrieving playlists: {str(e)}")
        return []
def extract_features(file_name):
    try:
        audio, sample_rate = librosa.load(file_name, res_type='kaiser_fast')
        mfccs = np.mean(librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40).T, axis=0)
        return mfccs, audio, sample_rate
    except Exception as e:
        show_error(f"Error encountered while parsing file: {file_name}, error: {e}")
        return None, None, None
def predict_emotion():
    global invoice_text, playlists, playlist_listbox, predicted_emotion
    try:
        file_path = current_audio_file if current_audio_file else recorded_audio_file
        feature, audio, sample_rate = extract_features(file_path)
        if feature is not None:
            feature = np.expand_dims(feature, axis=0)
            prediction = model.predict(feature)
            predicted_emotion = EMOTIONS[np.argmax(prediction)]
            display_image(predicted_emotion)
            refresh_playlists()
            display_game_recommendations(predicted_emotion)
            show_waveform_button.configure(state=tk.NORMAL)  # Enable show waveform
button
            show_invoice_button.configure(state=tk.NORMAL)  # Enable show invoice button
            return predicted_emotion  # Return the predicted emotion
        else:
            show_error("Failed to extract features from the selected file.")
            return None
    except Exception as e:
        show_error(f"Failed to predict emotion: {e}")
        return None
def refresh_playlists():
    global playlists, playlist_listbox, predicted_emotion
```

```
    #predicted_emotion = predict_emotion()
    if predicted_emotion:
        playlists = get_spotify_playlists(predicted_emotion)
        playlist_listbox.delete(0, tk.END)
        for playlist in playlists:
            playlist_listbox.insert(tk.END, playlist['name'])
def open_playlist_url(event):
    global playlists, current_audio_file, recorded_audio_file
    selection_index = playlist_listbox.curselection()
    if selection_index:
        url = playlists[selection_index[0]]['url']
        webbrowser.open_new(url)
def select_file():
    global current_audio_file, audio_file_label
    file_path = filedialog.askopenfilename(filetypes=[("WAV files", "*.wav")])
    if file_path:
        current_audio_file = file_path
        audio_file_label.configure(text=f"{file_path.split('/')[-1]}")
def play_audio():
    global current_audio_file, recorded_audio_file
    file_path = current_audio_file if current_audio_file else recorded_audio_file
    try:
        pygame.mixer.music.load(file_path)
        pygame.mixer.music.play()
    except Exception as e:
        show_error(f"Error playing audio: {e}")
def record_audio():
    global recorded_audio_file
    duration = 5
    samplerate = 44100
    show_info("Recording will start now. Please speak into the microphone.")
    recording = sd.rec(int(samplerate * duration), samplerate=samplerate, channels=2)
    sd.wait()
    wavio.write(recorded_audio_file, recording, samplerate, sampwidth=2)
    show_info("Recording finished.")
```

```
    audio_file_label.configure(text="Recorded File: recording.wav")
    current_audio_file = None
    refresh_playlists()
def open_main_gui():
    global root, middle_frame, main_frame, upload_frame,  audio_file_label
    middle_frame.pack_forget()  # Hide the middle frame
    main_frame.pack(fill="both", expand=True)  # Show the main frame
    root.title("Emotion Recognition")
    root.geometry(f"{root.winfo_screenwidth()}x{root.winfo_screenheight()}+0+0")
    # Load images once
    menu_items_top = [
        ("Home", "C:/Users/annap/project/static/home.jpeg"),
        ("Abstract", "C:/Users/annap/project/static/abstract.jpeg"),
        ("About Us", "C:/Users/annap/project/static/About.jpeg"),
        ("Project", "C:/Users/annap/project/static/project.jpg"),
        ("Help", "C:/Users/annap/project/static/settings.jpeg"),
    ]
    menu_items_bottom = [
        ("Conclusion", "C:/Users/annap/project/static/conclusion.jpeg"),
        ("Exit", "C:/Users/annap/project/static/exit.jpg"),
    ]
    frame = ctk.CTkFrame(main_frame, fg_color=menu_fg_color)
    frame.pack(fill="y", side="left")
    profile_image = ctk.CTkLabel(frame, text=" 🔵 ", font=("Arial", 40),
fg_color=menu_fg_color, text_color="white")
    profile_image.pack(pady=20)
    app_name = ctk.CTkLabel(frame, text="EmotiVoice", font=("Arial", 20),
fg_color=menu_fg_color, text_color="white")
    app_name.pack(pady=10)
    labels = []
    active_text = ctk.StringVar()
    content_frame = ctk.CTkFrame(main_frame, fg_color=men_fg_color)
    content_frame.pack(fill="both", expand=True, padx=20, pady=20)
    def on_menu_click(item):
        for label in labels:
```

```
            label[1].configure(fg_color=menu_fg_color, text_color="white")
        item[1].configure(fg_color="grey", text_color="black", width=70)
        active_text.set(item[1].cget("text"))
        page_name = item[1].cget("text")
        for widget in content_frame.winfo_children():
            widget.destroy()
        if page_name == "Home":
            label = ctk.CTkLabel(content_frame, text="Home", justify="left", font=("Times New
Roman", 28, "bold"))
            label.place(x=10, y=40)
            custom_font = ctk.CTkFont(family="Times New Roman", size=18)
            label = ctk.CTkLabel(content_frame, text="", font=custom_font, justify="left",
anchor="nw")
            label.pack(pady=80, padx=40, fill='both', expand=True)
            tk_font = font.Font(family="Times New Roman", size=20)
            def auto_type(text, idx=0, current_text=""):
                try:
                    if idx < len(text):
                        new_char = text[idx]
                        current_line = current_text.split('\n')[-1]
                        current_line_width = tk_font.measure(current_line + new_char)
                        if current_line_width > label.winfo_width():
                            current_text += '\n'
                        current_text += new_char
                        label.configure(text=current_text)
                        idx += 1
                        content_frame.after(100, auto_type, text, idx, current_text)
                except tk.TclError:
                    pass
            auto_type("""Welcome to EmotiVoice!
                    This innovative project harnesses the power of emotion recognition from voice to
EmotiVoice uniquely attuned to your feelings.""")
        elif page_name == "Abstract":
            ctk.CTkLabel(content_frame, text="Abstract", font=("Times New Roman", 28,
"bold")).pack(pady=40)
```

```
        abstract_text = """
        Emotion plays a crucial role in human communication, significantly impacting our
        """
        text_widget = ctk.CTkLabel(content_frame, text=abstract_text.strip(),
wraplength=800, justify="left", font=("Times New Roman", 20))
        text_widget.pack(pady=20, padx=10)
        image_path = "C:/Users/annap/project/static/ab.jpg"
        original_image = Image.open(image_path)
        resized_image = original_image.resize((400, 300), Image.LANCZOS)
        photo = ImageTk.PhotoImage(resized_image)
        label = ctk.CTkLabel(content_frame, image=photo, text="")
        label.image = photo
        label.pack()
    elif page_name == "Project":
        global current_audio_file, game_recommendation_frame, upload_frame
        ctk.CTkLabel(content_frame, text="Project", font=("Arial", 20)).pack(pady=20)
        upload_frame = ctk.CTkFrame(content_frame, width=1250, height=650,
fg_color="white", border_color="black", border_width=2)
        upload_frame.pack(pady=20, padx=20)
        upload_frame.pack_propagate(False)
        j = Image.open("C:/Users/annap/project/static/upload.jpg")
        ctk_image1 = ctk.CTkImage(light_image=j, size=(30, 30))
        upload_button = ctk.CTkButton(upload_frame, text="", image=ctk_image1,
command=select_file, fg_color="white", width=20)
        upload_button.place(x=720, y=10)
        predict_button = ctk.CTkButton(upload_frame, text="predict",
command=predict_emotion)
        predict_button.place(x=550, y=70)
        k = Image.open("C:/Users/annap/project/static/play.png")
        ctk_image2 = ctk.CTkImage(light_image=k, size=(30, 30))
        play_button = ctk.CTkButton(upload_frame, text="", image=ctk_image2,
command=play_audio, fg_color="white", width=20)
        play_button.place(y=60, x=770)
        l = Image.open("C:/Users/annap/project/static/record.png")
        ctk_image3 = ctk.CTkImage(light_image=l, size=(30, 30))
```

```
        button_record = ctk.CTkButton(upload_frame, text="", image=ctk_image3,
command=record_audio, fg_color="white", width=20)
        button_record.place(y=10, x=770)
        audio_file_label = ctk.CTkLabel(upload_frame, text="No files selected",
text_color="black")
        audio_file_label.pack(pady=20)
        game_recommendation_frame = tk.Frame(content_frame, bd=10)
        game_recommendation_frame.place(x=1250, y=650)
    elif page_name == "Conclusion":
        ctk.CTkLabel(content_frame, text="Conclusion", font=("Times New Roman", 28,
"bold")).pack(pady=40)
        conclusion_text = """
        The integration of Speech Emotion Recognition (SER) technology with personalized
playlist
        """
        text_widget = ctk.CTkLabel(content_frame, text=conclusion_text.strip(),
wraplength=800, justify="left", font=("Times New Roman", 20))
        text_widget.pack(pady=20, padx=10)
        image_path = "C:/Users/annap/project/static/refresh.jpg"
        original_image = Image.open(image_path)
        resized_image = original_image.resize((400, 300), Image.LANCZOS)
        photo = ImageTk.PhotoImage(resized_image)
        label = ctk.CTkLabel(content_frame, image=photo, text="")
        label.image = photo
        label.pack()
    elif page_name == "About Us":
        ctk.CTkLabel(content_frame, text="About Us", font=("Times New Roman", 28,
"bold")).pack(pady=40)
        about_us_text = """
        At EmotiVoice, we are passionate about creating technology that understands and
responds
        """
        text_widget = ctk.CTkLabel(content_frame, text=about_us_text.strip(),
wraplength=800, justify="left", font=("Times New Roman", 20))
        text_widget.pack(pady=20, padx=10)
```

```
        image_path = "C:/Users/annap/project/static/refresh.jpg"
        original_image = Image.open(image_path)
        resized_image = original_image.resize((400, 300), Image.LANCZOS)
        photo = ImageTk.PhotoImage(resized_image)
        label = ctk.CTkLabel(content_frame, image=photo, text="")
        label.image = photo
        label.pack()
    elif page_name == "Help":
        ctk.CTkLabel(content_frame, text="Help", font=("Arial", 20)).pack(pady=20)
        bg_color = "#000080"
        card_color = "#FFFFFF"
        form_color = "#99D0F0"
        form_frame = ctk.CTkFrame(content_frame, fg_color=bg_color, width=600,
height=300)
        form_frame.place(x=500, y=200)
        ctk.CTkLabel(form_frame, text="Contact Us", fg_color=bg_color,
text_color="white", font=("Arial", 28)).pack(pady=30)
        name_entry = ctk.CTkEntry(form_frame, placeholder_text="Enter your Name",
width=300)
        name_entry.pack(padx=20,pady=5)
        email_entry = ctk.CTkEntry(form_frame, placeholder_text="Enter a valid email
address", width=300)
        email_entry.pack(padx=70,pady=5)
        message_entry = ctk.CTkTextbox(form_frame, height=150, width=300)
        message_entry.pack(padx=50,pady=15)
        def clear_form():
            name_entry.delete(0, ctk.END)
            email_entry.delete(0, ctk.END)
            message_entry.delete("1.0", ctk.END)
            submitted_label.pack(pady=5)  # Show "Submitted" message
        submit_button = ctk.CTkButton(form_frame, text="SUBMIT",
command=clear_form)
        submit_button.pack(pady=10)
        submitted_label = ctk.CTkLabel(form_frame, text="Submitted", fg_color=bg_color,
font=("Arial", 16), text_color="white")
```

```
        frame_office = ctk.CTkFrame(content_frame, fg_color=form_color, width=150,
height=120)
        frame_phone = ctk.CTkFrame(content_frame, fg_color=form_color, width=150,
height=120)
        frame_fax = ctk.CTkFrame(content_frame, fg_color=form_color, width=150,
height=120)
        frame_email = ctk.CTkFrame(content_frame, fg_color=form_color, width=150,
height=120)
        frame_office.place(x=380, y=100)
        frame_phone.place(x=550, y=100)
        frame_fax.place(x=730, y=100)
        frame_email.place(x=910, y=100)
        def load_image(file_path, width, height):
            image = Image.open(file_path)
            image = image.resize((width, height), Image.LANCZOS)
            return ImageTk.PhotoImage(image)
        office_image_path = "C:/Users/annap/OneDrive/Desktop/location.jpeg"
        phone_image_path = "C:/Users/annap/OneDrive/Desktop/contact.jpeg"
        fax_image_path = "C:/Users/annap/OneDrive/Desktop/fax.jpeg"
        email_image_path = "C:/Users/annap/OneDrive/Desktop/mail.png"
        office_image = load_image(office_image_path, 50, 50)
        phone_image = load_image(phone_image_path, 50,50)
        fax_image = load_image(fax_image_path, 50, 50)
        email_image = load_image(email_image_path, 50, 50)
        ctk.CTkLabel(frame_office, image=office_image,text="", justify="center",
fg_color=form_color).place(x=50, y=5)
        ctk.CTkLabel(frame_phone, image=phone_image,text="", justify="center",
fg_color=form_color).place(x=60, y=5)
        ctk.CTkLabel(frame_fax, image=fax_image,text="", justify="center",
fg_color=form_color).place(x=55, y=20)
        ctk.CTkLabel(frame_email, image=email_image,text="", justify="center",
fg_color=form_color).place(x=58, y=20)
        ctk.CTkLabel(frame_office, text="OUR MAIN OFFICE", justify="center",
fg_color=form_color, font=("Arial", 14, "bold")).place(x=10,y=50)
```

```
            ctk.CTkLabel(frame_office, text="SoHo 94 Broadway St\nNew York, NY 1001",
justify="center", fg_color=form_color).place(x=10,y=70)
            ctk.CTkLabel(frame_phone, text="PHONE NUMBER", justify="center",
fg_color=form_color, font=("Arial", 14, "bold")).place(x=20,y=45)
            ctk.CTkLabel(frame_phone, text="234-9876-5400\n888-0123-4567 \n(Toll Free)",
justify="center", fg_color=form_color).place(x=30,y=70)
            ctk.CTkLabel(frame_fax, text="FAX", justify="center", fg_color=form_color,
font=("Arial", 14, "bold")).place(x=60,y=60)
            ctk.CTkLabel(frame_fax, text="1-234-567-8900", justify="center",
fg_color=form_color).place(x=30,y=80)
            ctk.CTkLabel(frame_email, text="EMAIL", justify="center", fg_color=form_color,
font=("Arial", 14, "bold")).place(x=60,y=60)
            ctk.CTkLabel(frame_email, text="hello@theme.com", justify="center",
fg_color=form_color).place(x=30,y=80)
        elif page_name == "Exit":
            root.destroy()
    def on_label_enter(event, label):
        if label.cget("text") != active_text.get():
            label.configure(fg_color=menu_fg_color, text_color="yellow")
    def on_label_leave(event, label):
        if label.cget("text") != active_text.get():
            label.configure(fg_color=menu_fg_color, text_color="white")
    def create_menu_item(menu_item):
        label = ctk.CTkLabel(frame, text=menu_item[0], fg_color=menu_fg_color,
text_color="white", font=("Arial", 20), width=140, height=60,
image=ctk.CTkImage(Image.open(menu_item[1]), size=(20, 20)), compound="left",
padx=10, anchor="w", corner_radius=0)
        label.pack(fill="both", pady=5)
        label.bind("<Button-1>", lambda e: on_menu_click((menu_item, label)))
        label.bind("<Enter>", lambda e: on_label_enter(e, label))
        label.bind("<Leave>", lambda e: on_label_leave(e, label))
        return (menu_item, label)
    for item in menu_items_top + menu_items_bottom:
        labels.append(create_menu_item(item))
    on_menu_click(labels[0])
```

```
def switch_to_middle_frame():
    main_frame.pack_forget()  # Hide the main frame
    middle_frame.pack(fill="both", expand=True)  # Show the middle frame
root = ctk.CTk()
root.title("Emotion Recognition")
root.geometry(f"{root.winfo_screenwidth()}x{root.winfo_screenheight()}+0+0")
middle_frame = ctk.CTkFrame(root)
main_frame = ctk.CTkFrame(root)
middle_frame.pack(fill="both", expand=True)
def close():
    root.destroy()
image_path = "C:/Users/annap/project/static/bg.png"
img = Image.open(image_path)
photo = ctk.CTkImage(light_image=img, dark_image=img, size=(root.winfo_screenwidth(),
root.winfo_screenheight()))
label = ctk.CTkLabel(middle_frame, image=photo, text="")
label.image = photo
label.place(x=0, y=0, relwidth=1, relheight=1)
middle_button = ctk.CTkButton(middle_frame, text="Proceed", command=open_main_gui)
middle_button.place(x=620,y=600)
exit_button = ctk.CTkButton(middle_frame, text="exit", command=close)
exit_button.place(x=770,y=600)
middle_frame.pack(fill="both", expand=True)
root.mainloop()
```

TESTING

6. TESTING

Testing is the process of finding differences between the expected behavior specified by system models and the observed behavior of the system. Testing is a critical role in quality assurance and ensuring the reliability of development and these errors will be reflected in the code, so the application should be thoroughly tested and validated.

Unit testing finds the differences between the object design model and its corresponding components. Structural testing finds differences between the system design model and a subset of integrated subsystems. Functional testing finds differences between the use case model and the system. Finally, performance testing, finds differences between non-functional requirements and actual system performance. From modelling point of view, testing is the attempt of falsification of the system with respect to the system models. The goal of testing is to design tests that exercise defects in the system and to reveal problems.

6.1 Testing Activities

Testing a large system is a complex activity and like any complex activity. It has to be broke into smaller activities. Thus, incremental testing was performed on the project ie, components and subsystems of the system were tested separately before integrating them to form the subsystem for system testing

6.2 Testing Types

Unit Testing

Unit testing is a crucial aspect of developing the emotion-based music and game recommendation system. It involves testing individual components of the application in isolation to ensure that each part functions correctly. For this project, unit tests will be created for the emotion recognition algorithm, verifying that it accurately processes audio inputs and predicts emotions. Additionally, the recommendation engines for both music and games will be tested to confirm they return appropriate results based on the predicted emotions. Testing will also cover edge cases, such as handling invalid or noisy audio inputs, to ensure the system behaves predictably under all conditions. By systematically testing each unit of the application, we can identify and fix bugs early in the development process, resulting in a more robust and reliable system

Equivalence testing

It is a black box testing technique that minimizes the number of test cases. The possible inputs are partitioned into equivalence classes and a test case is selected for each class.

Boundary testing

It is a special case of equivalence testing and focuses on the conditions at the boundary of the equivalence classes. Boundary testing requires that the elements be selected from the edges of the equivalence classes.

Path testing

It is a white box testing technique that identifies faults in the implementation of the component the assumption here is that exercising all possible paths through the code at least once. Most faults will trigger failure. This acquires knowledge of source code.

Integrating Testing

Integration testing for the emotion-based music and game recommendation system focuses on verifying that the individual components work together seamlessly. This includes ensuring the emotion recognition module accurately passes predicted emotions to the recommendation engines for both music and games. The interaction between the frontend, which handles user inputs and displays recommendations, and the backend, which processes audio and generates suggestions, will be rigorously tested. API integrations, such as those with Spotify for playlist recommendations, will be scrutinized to ensure they respond correctly and provide relevant results. By methodically testing these interactions, we can identify and address issues arising from component integration, thereby ensuring that the system operates smoothly as a cohesive whole.

Validation Testing

Validation testing aims to confirm that the emotion-based recommendation system meets its intended use and fulfills user requirements. This involves end-to-end testing from audio input to the delivery of music and game recommendations, ensuring the system performs accurately under real-world conditions. User acceptance testing (UAT) will be conducted to gather feedback from actual users, verifying that the system's recommendations align with their expectations and emotional states. Additionally, validation testing will include performance checks to ensure the system remains responsive and efficient under various load conditions. This comprehensive approach guarantees that the system is not only functional but also provides value and satisfaction to its users.

System Testing

System testing encompasses the complete and integrated emotion-based recommendation system, verifying that it meets all specified requirements. This phase involves

thorough testing of the entire system, including the user interface, emotion recognition accuracy, recommendation quality, and system performance under different scenarios. Security testing will also be part of this phase to ensure user data is protected throughout the process. System testing aims to simulate real-world usage as closely as possible, identifying any discrepancies or issues that might affect the user experience. By rigorously testing the system as a whole, we can ensure it delivers reliable, accurate, and user-friendly recommendations, providing a robust solution for emotion-based music and game recommendations.

White-Box Testing

White box testing for the music and games recommendation system based on emotion recognition from voice involves analyzing the internal algorithms that process audio data and generate recommendations. It ensures that all code paths for emotion detection and recommendation logic are thoroughly tested to verify accuracy and performance under different scenarios.

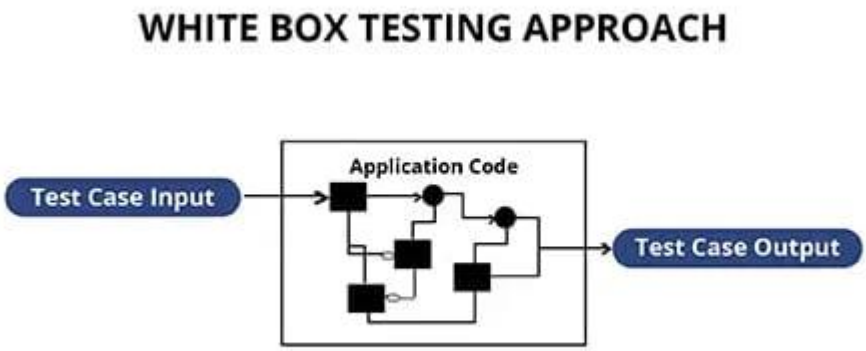


Fig. 6.2(a) White-Box Testing Diagram

Black Box Testing

Black box testing for the music and games recommendation system involves evaluating the application's functionality by testing its inputs and outputs without examining the internal code. It ensures that the system correctly interprets emotional data from voice inputs and provides appropriate music and game recommendations based on predefined criteria and user interactions. This method attempts to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structures or external database access

- Behaviour or performance errors
- Initialization and termination errors

In the Table creation module, the tables are created with user specified fields and user can create marry tables at a time. They may specify conditions, constraints and calculations in creation of tables.

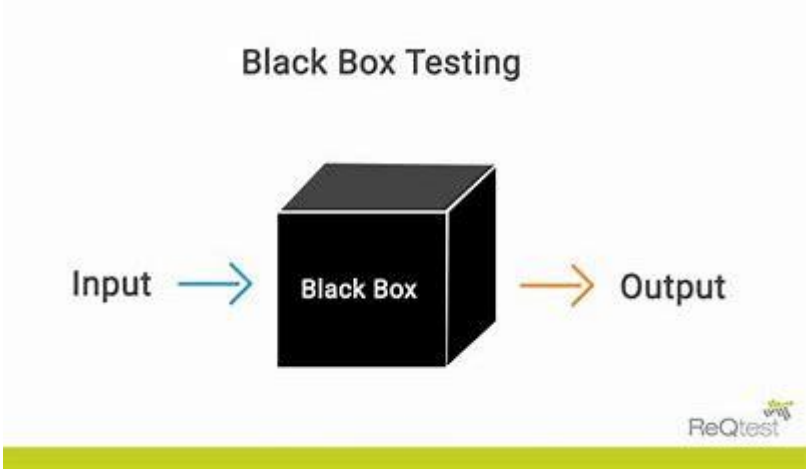


Fig. 6.2(b) Black Box Testing Diagram

6.3 Test Plan:

The test plan for the emotion-based music and game recommendation system involves several key steps to ensure robust functionality and user satisfaction. Initially, we will test the system's ability to handle voice input by uploading valid audio files and verifying that they are correctly recognized and processed. Next, we will evaluate the emotion prediction model's accuracy in identifying emotions from the voice clips. Following this, we will assess the system's capability to generate appropriate playlist recommendations using the Spotify API, ensuring the playlists align with the predicted emotions. Similarly, we will test the game's recommendation component to confirm that it suggests games matching the user's emotional state. The plan also includes handling edge cases, such as invalid or low-quality audio inputs, to ensure the system gracefully manages errors and provides meaningful feedback to users. Additionally, we will test the system's performance under load by processing multiple audio files simultaneously to ensure consistent performance. Integration tests with the Spotify API and the game recommendation algorithm will be conducted to validate seamless communication and data exchange between systems. Finally, user feedback mechanisms will be tested to verify that feedback is accurately recorded and utilized for improving the recommendation system. This comprehensive test plan aims to ensure the system is reliable, accurate, and user-friendly

6.4 Test Cases:

Test case for voice input handling

Test case ID	TC001
Pre-requisites	Valid audio file with a voice clip with .wav extension
Action	Upload the audio file to the system
Expected Result	Audio file is successfully uploaded and recognized by system
Test Result	Pass

Table 6.4(a): Test case for voice input handling

Test case for Emotion Prediction from audio

Test case ID	TC002
Pre-requisites	Uploaded Audio file
Action	Process the audio file emotion recognition model
Expected Result	Predicts the emotion from the audio
Test Result	Pass

Table 6.4(b): Test case for Emotion Prediction from audio

Test case for playlist recommendation from predicted emotion

Test case ID	TC003
Pre-requisites	Predicted the emotion from the audio
Action	Generate playlist from spotify app to recommend
Expected Result	System provides the playlist that matches the emotion
Test Result	Pass

Table 6.4(c): Test case for playlist recommendation from predicted emotion

Test case for game recommendation from predicted emotion

Test case ID	TC004
Pre-requisites	Predicted the emotion from the audio
Action	Generate a games list to recommend
Expected Result	System suggests list of games that aligns with the emotion
Test Result	Pass

Table 6.4(d): Test case for game recommendation from predicted emotion

Test case for Integration with Spotify API

Test case ID	TC005
Pre-requisites	Spotify API credentials
Action	Request a playlist from spotify app to recommend based on predicted emotion
Expected Result	Successfully communicates with Spotify API and retrieves a relevant playlist
Test Result	Pass

Table 6.4(e): Test case for Integration with Spotify API

Test case for Integration with Game recommendation system

Test case ID	TC006
Pre-requisites	Game recommendation directory or dictionary
Action	Request a game recommendation based on predicted emotion
Expected Result	Successfully retrieves and displays a relevant list of games
Test Result	Pass

Table 6.4(f): Test case for Integration with Game recommendation system

SCREENS

7. SCREENS

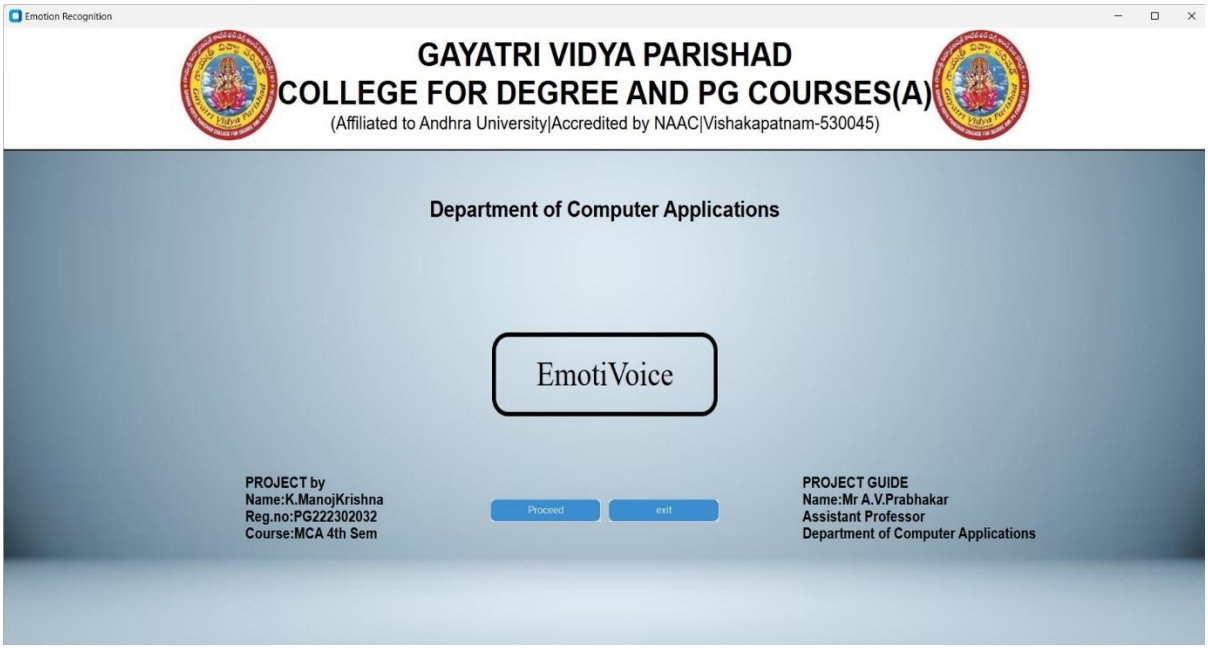


Fig. 7.1 Start Page

Project Execution starts from the welcome page having the title of the project. If the user click on “Proceed” button then it will take user into the application.

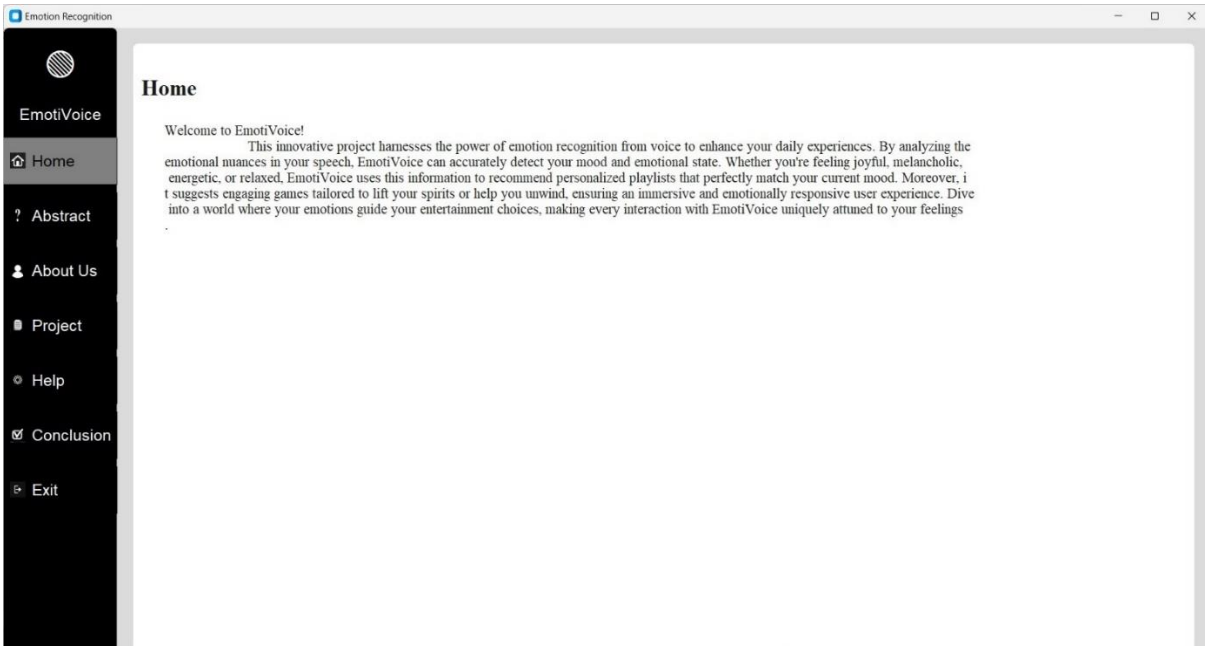


Fig. 7.2 Home Page

The Home page is initially displayed which gives an intoduction about the project. The above screen will be displayed when the user clicks proceed butoon in welcome page

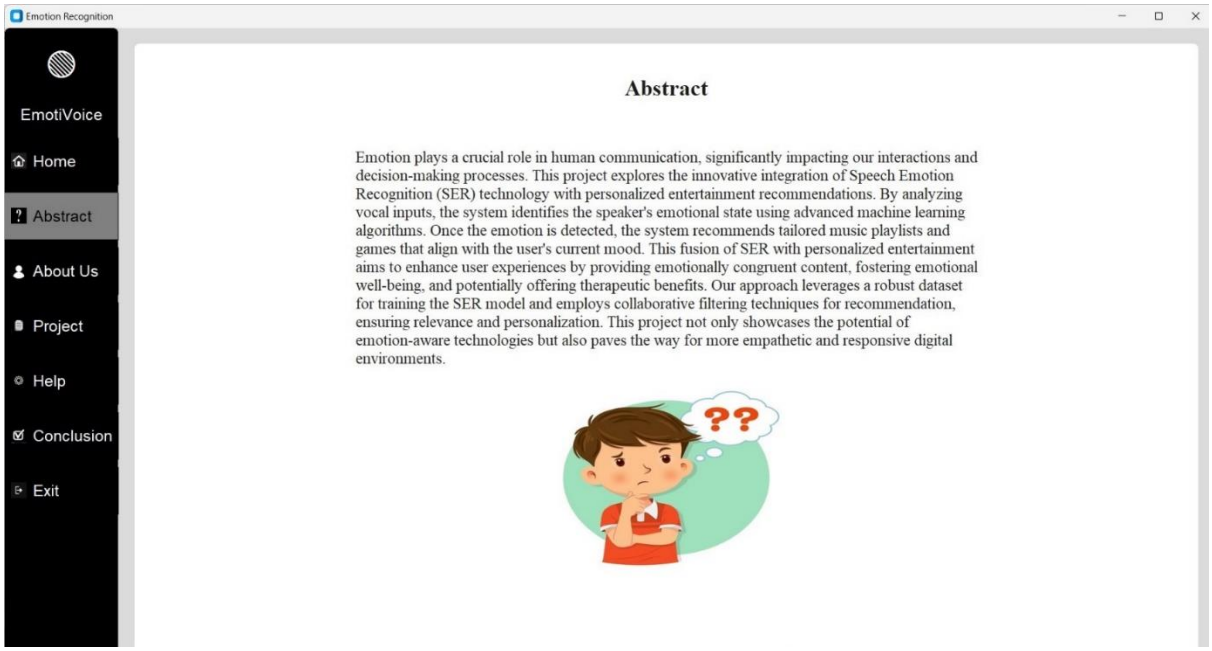


Fig. 7.3 Abstract Page

On Clicking “Abstract” in the left menu bar the abstract of the project is displayed which describes about the project in detail

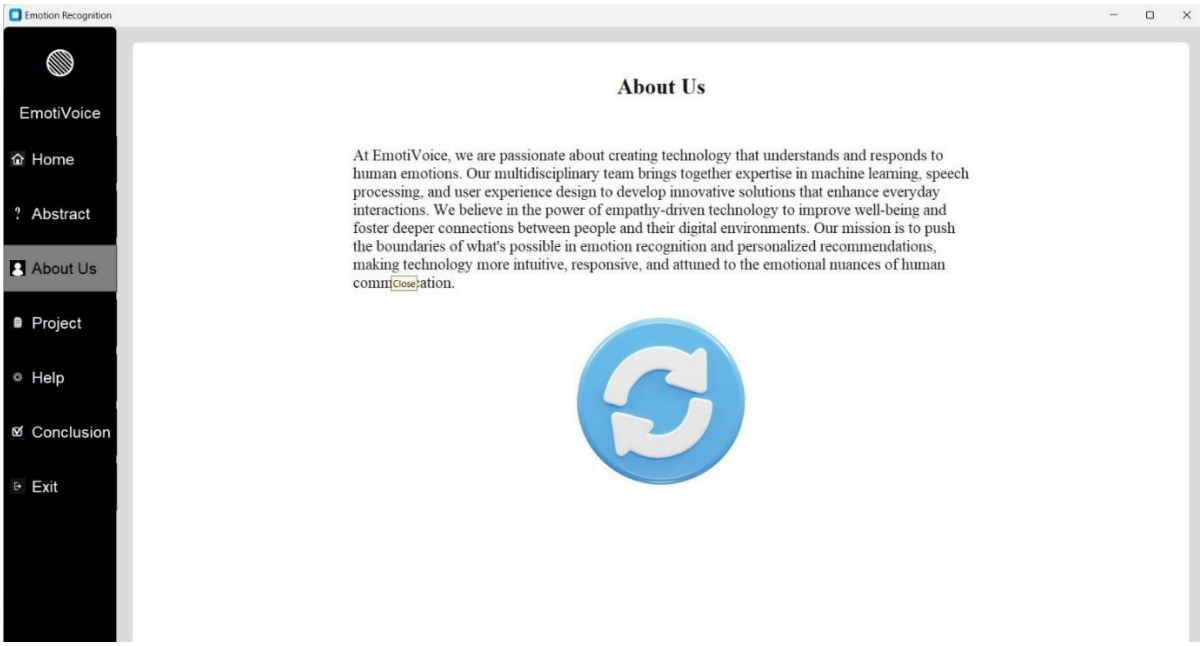


Fig. 7.4 AboutUs Page

The user can go to about us page on clicking “About Us” on the same menu bar where he can fully know about the algorithms and so on used in this project

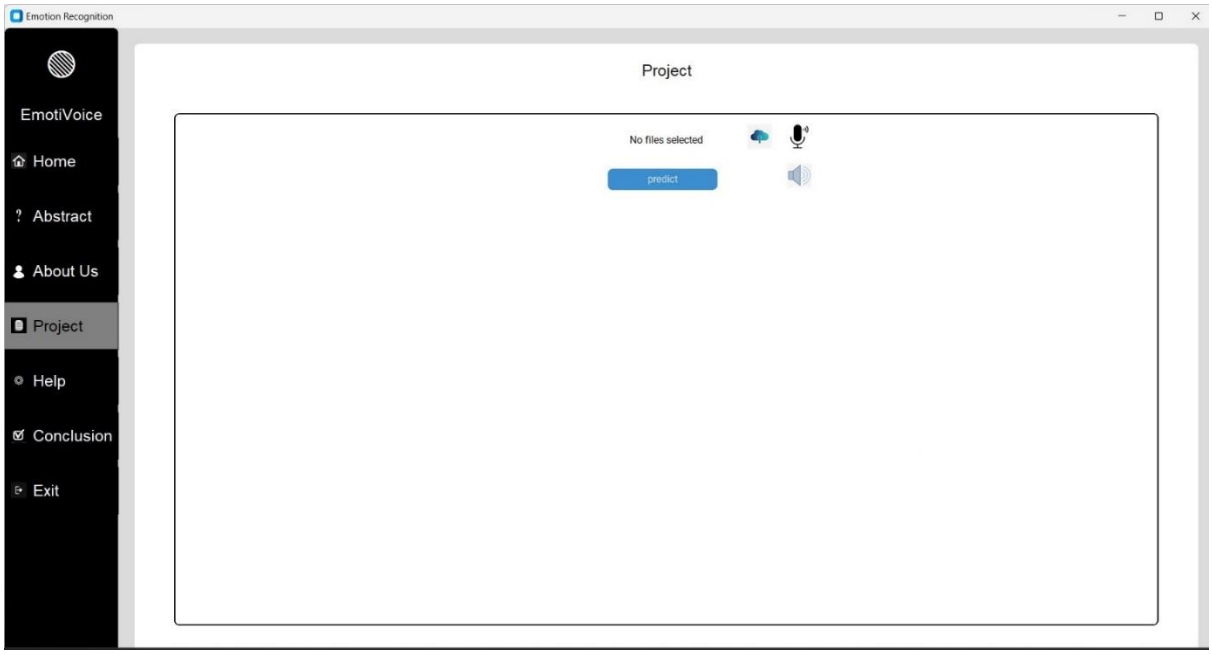


Fig. 7.5(a) Project Page

The project option in the menu bar will take the user to acutual project execution where he can prediction the emotion from the voice

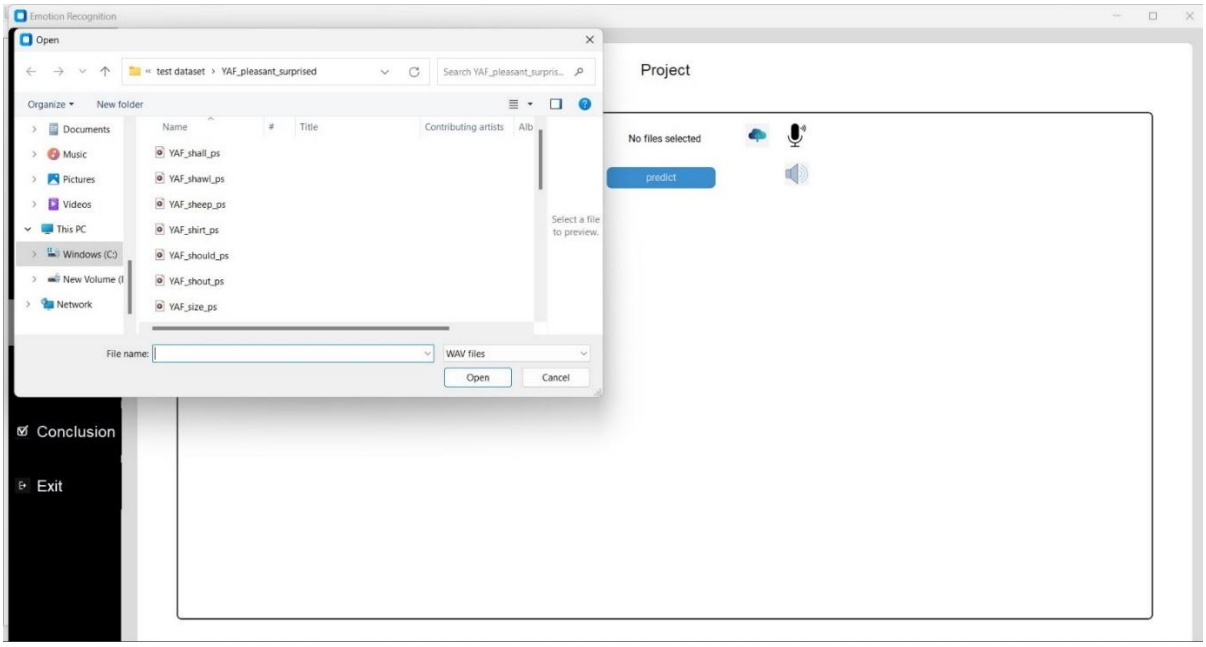


Fig. 7.5(b) Project Page

The user can upload a audio file on clicking “Upload symbol” in the project page and he can also record the audio dynamically by clicking on “Mic symbol”

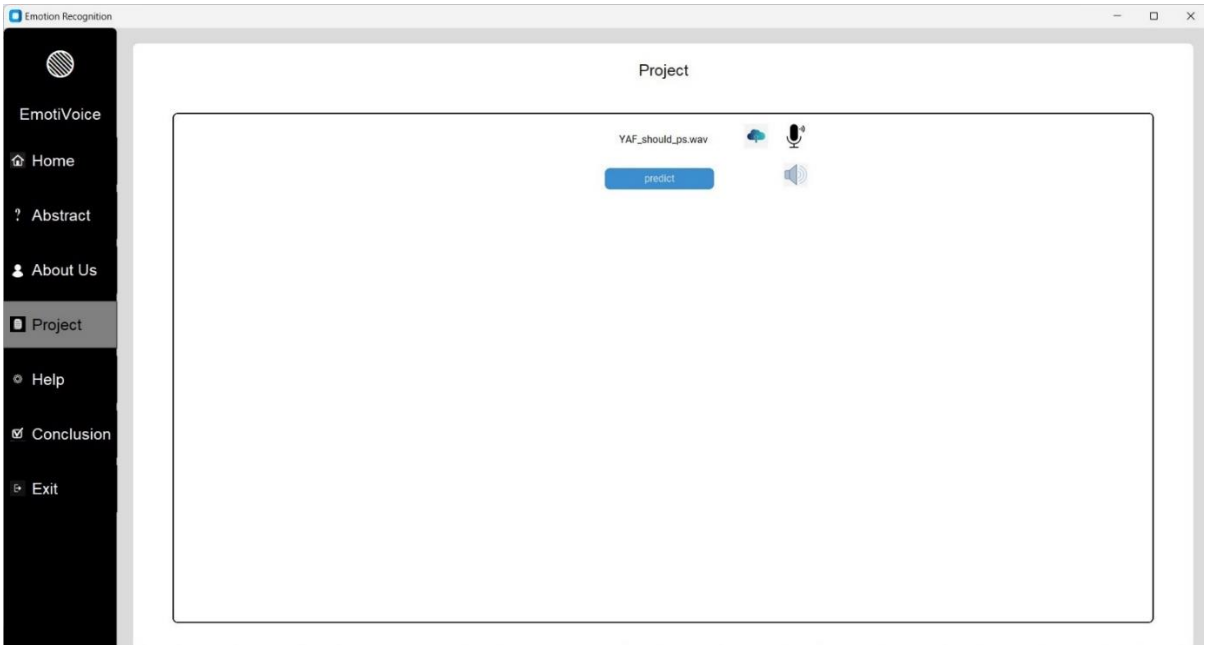


Fig. 7.5(c) Project Page

After uploading file the user can experience a screen like shown above with the file name he uploaded, then after clicking "Predict" button, he can proceed further.

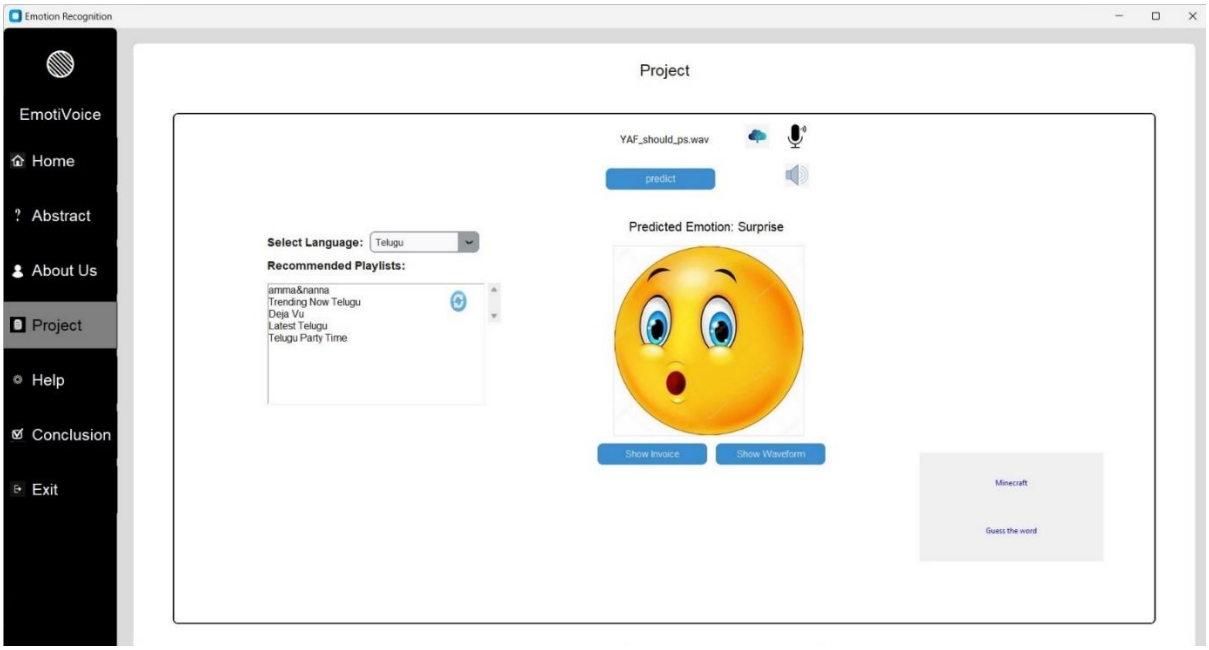


Fig. 7.5(d) Project Page

After the predict button is activated the emotion of the audio file uploaded is predicted and displays the emotionand an image related to the predicted emotion and also recommends playlists and games for the emotion predicted which are used to control the emotion of the user

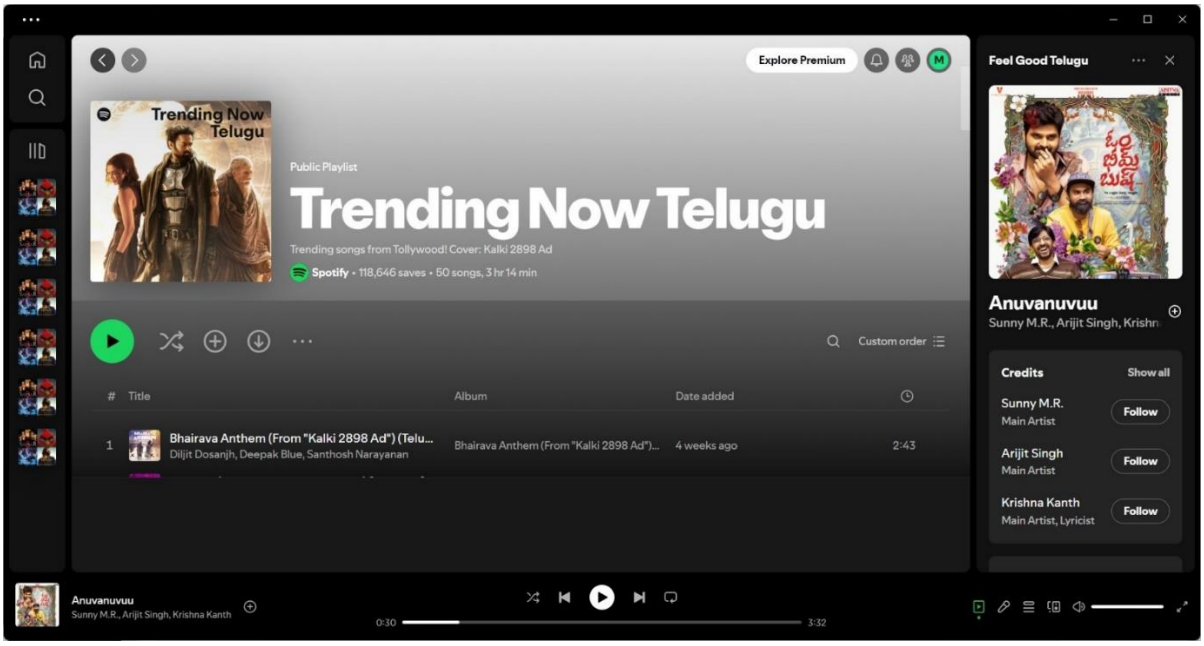


Fig. 7.5(e) Project Page

On doing “Double click” on the playlist recommended the project will be redirected to playlist in the spotify app and the user enjoy the songs.

.

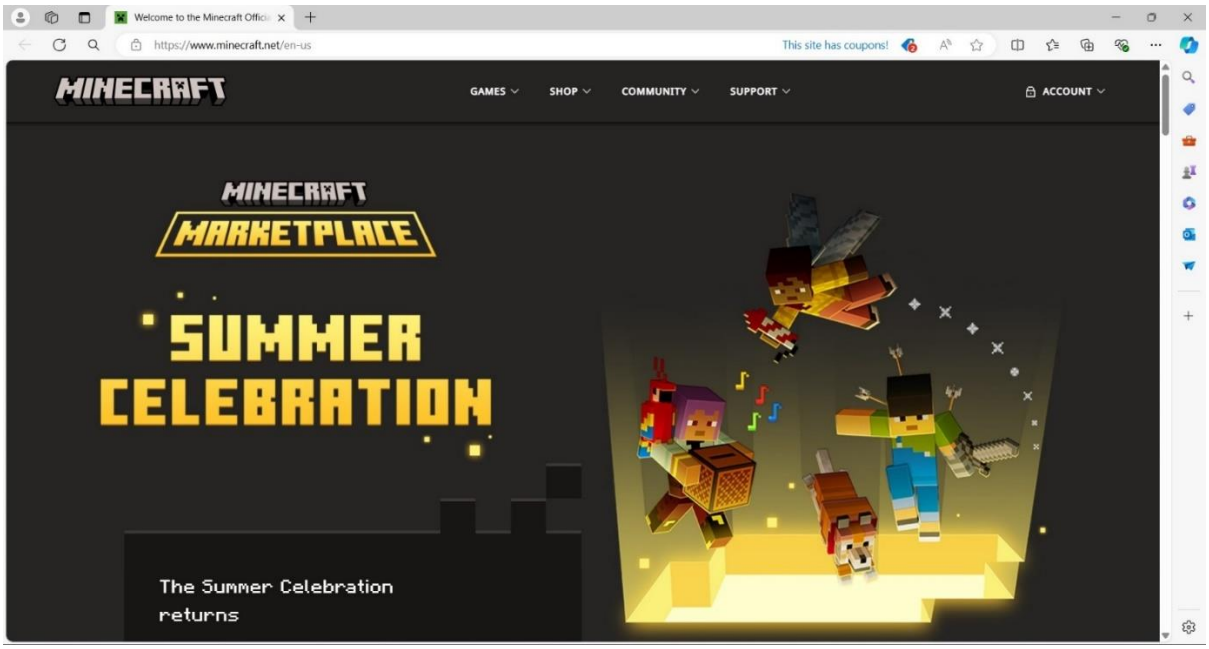


Fig. 7.5(f) Project Page

The user on clicking on the game recommended he can experience the page of the game which user can play a game

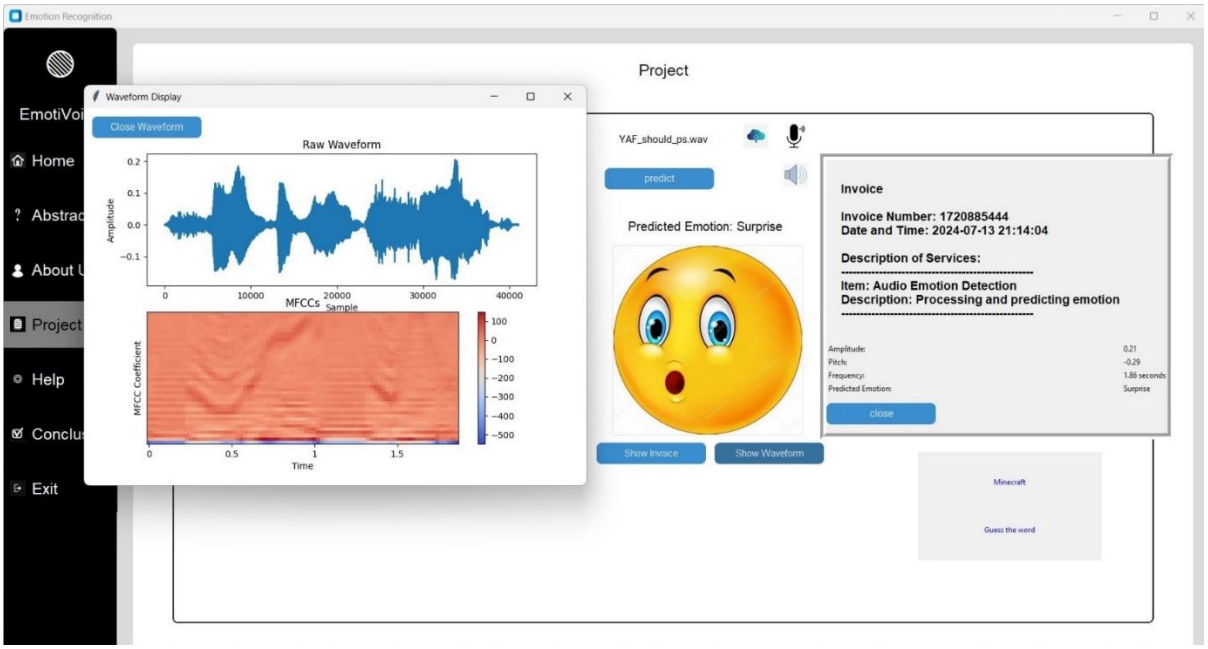


Fig. 7.5(g) Project Page

The user can check the invoice and waveform of the audio file on clicking the "show waveform" and "show invoice" button displayed in the project page

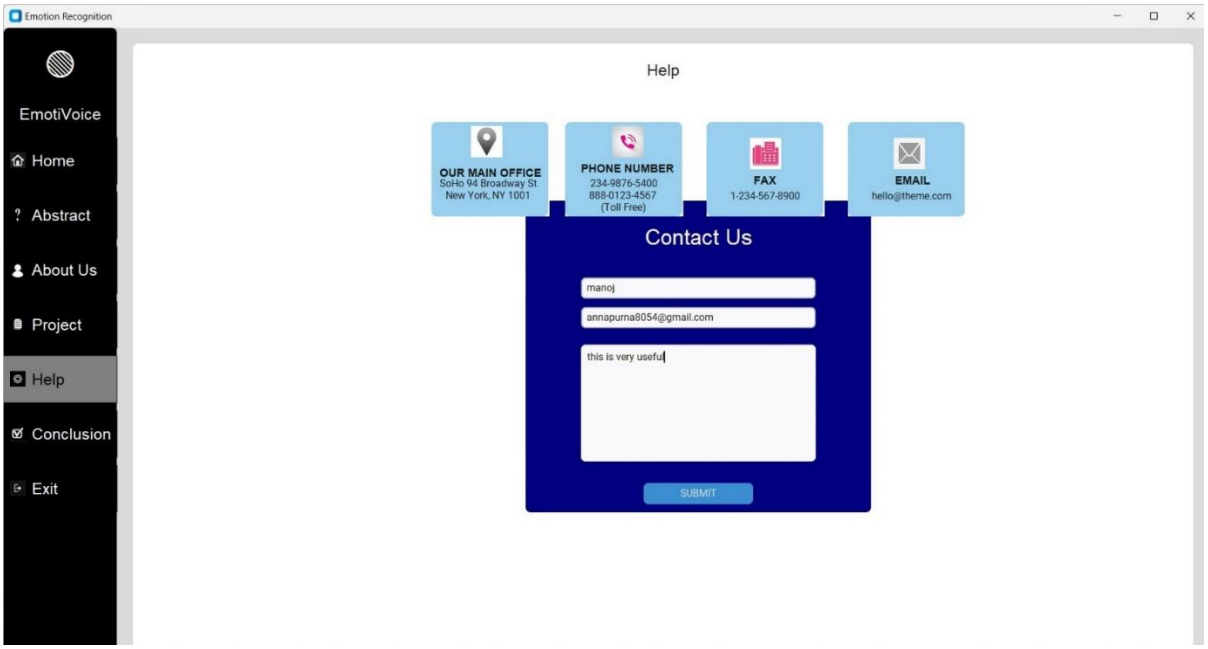


Fig. 7.6(a) Help Page

The user can shoot their queries through the form seen in the above image on selecting "Help" option in menu bar also he can contact with the number provided in the form

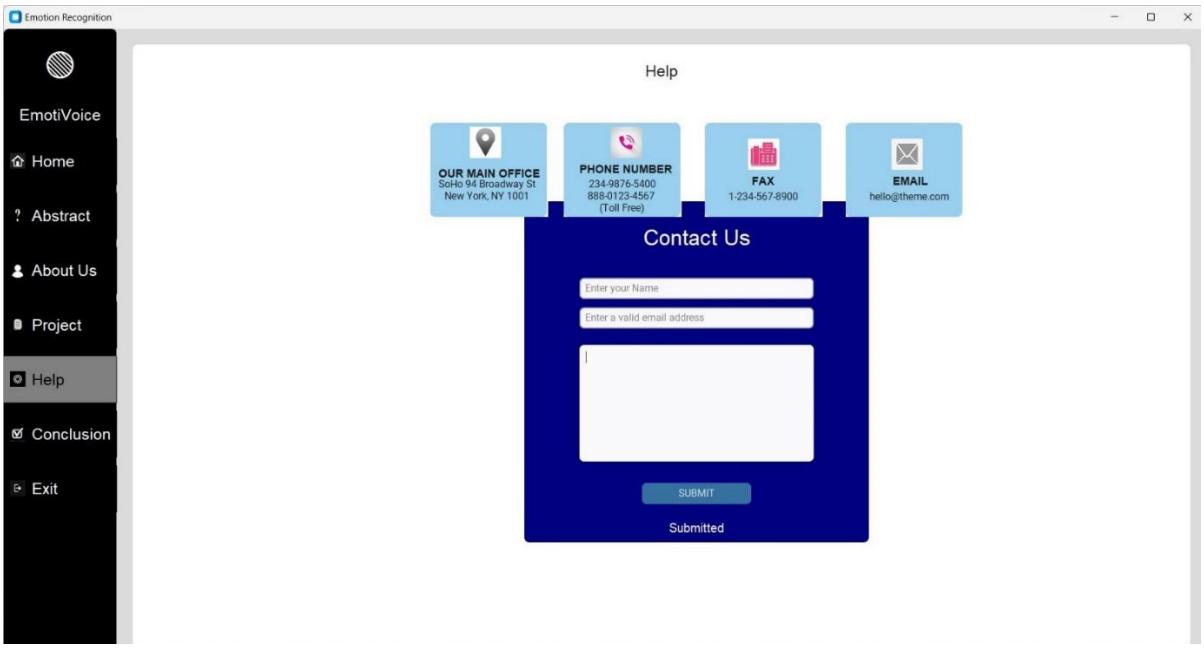


Fig. 7.6(b) Help Page

The user can experience the an acknowledgmeent for the form he want to submit as like above image

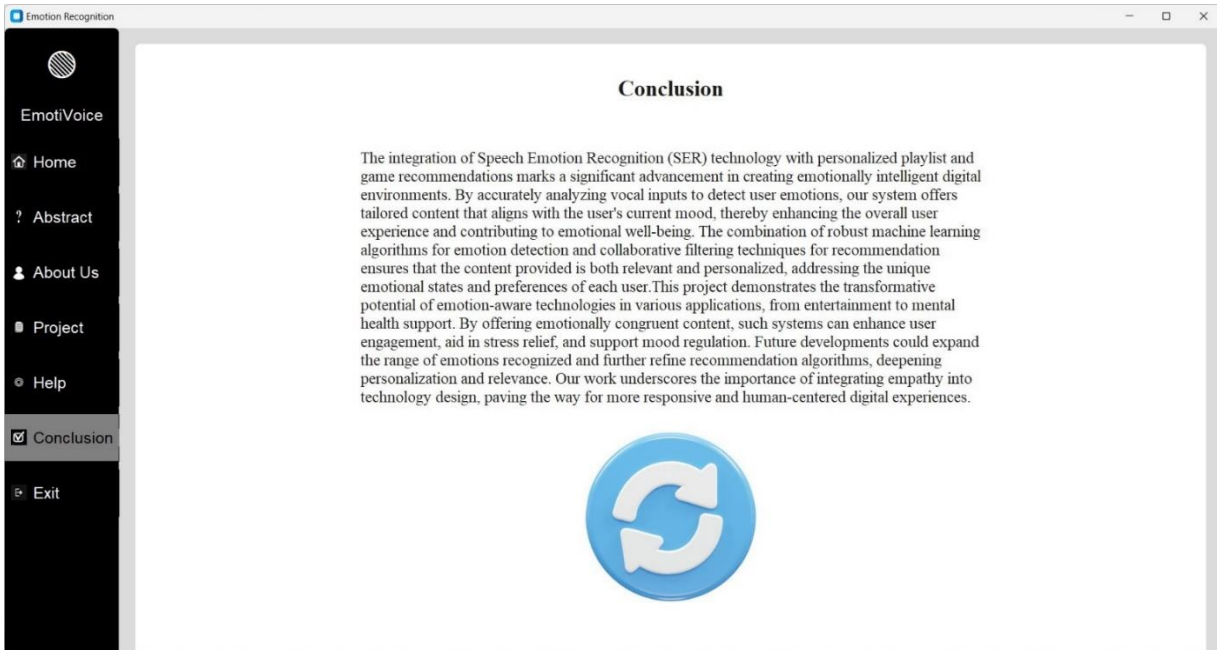


Fig. 7.7 Conclusion Page

The user can know the summary of the project in the conclusion option in menu bar and can exit the application by selecting the exit option in menu bar

CONCLUSION

8. CONCLUSION

The integration of Speech Emotion Recognition (SER) technology with personalized playlist and game recommendations marks a significant advancement in creating emotionally intelligent digital environments. By accurately analyzing vocal inputs to detect user emotions, our system offers tailored content that aligns with the user's current mood, thereby enhancing the overall user experience and contributing to emotional well-being. The combination of robust machine learning algorithms for emotion detection and collaborative filtering techniques for recommendation ensures that the content provided is both relevant and personalized, addressing the unique emotional states and preferences of each user.This project demonstrates the transformative potential of emotion-aware technologies in various applications, from entertainment to mental health support. By offering emotionally congruent content, such systems can enhance user engagement, aid in stress relief, and support mood regulation. Future developments could expand the range of emotions recognized and further refine recommendation algorithms, deepening personalization and relevance. Our work underscores the importance of integrating empathy into technology design, paving the way for more responsive and human-centered digital experiences.

FUTURE SCOPE

9. FUTURE SCOPE

The future scope of the emotion-based music and game recommendation system is vast and promising, encompassing several areas for enhancement and expansion. One key avenue is the integration of more advanced and nuanced emotion recognition algorithms, potentially incorporating multi-modal inputs such as facial expressions and physiological signals to improve accuracy and reliability. Additionally, expanding the system to support multiple languages and diverse accents can make it more inclusive and globally applicable. The recommendation engine itself can be refined to include more personalized suggestions by leveraging machine learning techniques to analyze user preferences and feedback over time. Another exciting prospect is the incorporation of real-time emotion tracking, allowing dynamic adaptation of music and game recommendations as the user's emotional state evolves. Collaboration with wearable technology companies to monitor users' emotional states throughout their daily activities can further enhance the system's responsiveness. Furthermore, the system can be extended to other domains such as mental health, providing therapeutic content and support based on real-time emotional assessments. Finally, continuous user testing and iterative improvements will ensure that the system remains user-friendly, effective, and aligned with the evolving needs and expectations of its users.

REFERENCES

10. REFERENCES

10.1 Academic Papers and Journals

- Ververidis, D., & Kotropoulos, C. (2006). Emotion recognition from speech: A survey. International Journal of Signal Processing, 11(1), 1-21.
- El Ayadi, M., Kamel, M. S., & Karray, F. (2011). Survey on speech emotion recognition: Features, classification schemes, and databases. Pattern Recognition, 44(3), 572-587.
- Zhao, S., Dong, M., Wu, F., & Li, Y. (2019). Speech emotion recognition using deep convolutional neural network and discriminant temporal pyramid matching. IEEE Transactions on Multimedia, 21(3), 753-766.
- Fayek, H. M., Lech, M., & Cavedon, L. (2017). A comprehensive review on emotion recognition based on audio signal. Speech Communication, 89, 93-110.
- Lotfian, R., & Busso, C. (2019). Deep learning for emotion recognition on small datasets using transfer learning. Proceedings of the Interspeech 2019, 1636-1640.

10.2 Web References

- https://towardsdatascience.com/using-machine-learning-for-emotion-recognition-from-speech-59a6e7e7b1a6
- https://www.analyticsvidhya.com/blog/2021/02/building-a-speech-emotion-recognition-system-using-machine-learning/
- https://realpython.com/python-speech-recognition/
- https://developer.spotify.com/documentation/web-api/
- https://data-flair.training/blogs/emotion-recognition-deep-learning-project/

APPENDIX

11. APPENDIX

11.1 List of Figures:

11.1 List of Figures: