VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

Title of the Project:

Sentimental Analysis

Team member(s): Simsonrallin.K(22MIA1158),
Rishikesh.M(22MIA1163)

Project Guide:
Prof. NISHA V.M
Vellore Institute of Technology, Chennai

Abstract

This Python application designed to function as a personal AI chatbot with a graphical user interface (GUI). It integrates various libraries for functionalities such as audio processing, natural language processing, and GUI creation. Key features include emotional analysis of text in English and Tamil, text-to-speech conversion, AIML-based chatbot responses, and multimedia playback capabilities. The application supports language switching, mood analysis for music playback, and provides additional features like opening applications and translating text. Overall, it offers an interactive and engaging chatbot experience by combining advanced technology and user-friendly design.

Proposed Work

The proposed work for this project involves the development of an interactive chat application that integrates several advanced features to enhance user experience. The following components outline the key areas of focus and implementation strategies:

1. User Interface Design

Dynamic UI Elements: Design and implement a visually appealing user interface that incorporates animated elements, such as neon borders and color transitions, to create an engaging atmosphere.

Responsive Layout: Ensure that the application is responsive and adapts to different screen sizes and resolutions, providing a consistent experience across devices.

Intuitive Navigation: Develop an intuitive navigation system that allows users to easily access chat histories, settings, and other functionalities without confusion.

2. Multilingual Support

Language Toggle Feature: Implement a language toggle feature that allows users to switch between Tamil and English seamlessly. This will involve creating a mechanism to manage language-specific content and ensuring that the user interface elements are appropriately translated.

Contextual Language Processing: Utilize natural language processing (NLP) techniques to ensure accurate understanding and generation of responses in both languages, enhancing the effectiveness of the chatbot.

3. Voice Interaction Capabilities

Voice Recognition Integration: Integrate a speech recognition system that allows users to send messages using voice commands. This will cater to users who prefer hands-free communication or have difficulty typing.

Voice Feedback: Implement voice feedback for the chatbot responses, allowing users to hear replies rather than just reading them, thus enhancing accessibility and user engagement.

4. Chatbot Development

Context-Aware Responses: Develop a chatbot that utilizes machine learning algorithms to provide context-aware responses, maintaining coherence in conversations and adapting to user inputs effectively.

User Personalization: Implement features that allow the chatbot to learn user preferences and tailor responses accordingly, creating a more personalized interaction experience.

5. Testing and Evaluation

User Testing: Conduct user testing sessions to gather feedback on the application's usability, performance, and overall

experience. This will include testing the interface, language switching, voice interaction, and chatbot functionality.

Iterative Improvements: Use the feedback obtained from user testing to make iterative improvements to the application, ensuring that it meets user needs and expectations effectively.

6. Deployment and Documentation

Deployment: Prepare the application for deployment on various platforms (e.g., web, mobile) to reach a wider audience.

User Documentation: Create comprehensive user documentation that includes instructions on using the application, troubleshooting common issues, and understanding the features offered.

Expected Outcomes

The proposed work aims to deliver a robust chat application that:

Provides an engaging and dynamic user interface.

Supports seamless communication in multiple languages.

Incorporates voice interaction for enhanced accessibility.

Features a responsive and context-aware chatbot capable of delivering personalized experiences.

By addressing the identified challenges, the project aspires to improve user engagement, satisfaction, and overall effectiveness in digital communication.

Methodology

The methodology for this project outlines the systematic approach to developing an interactive chat application that incorporates advanced features such as a dynamic user interface, multilingual support, voice interaction, and an intelligent chatbot. The methodology consists of several phases, each with specific tasks and deliverables.

1. Requirements Gathering and Analysis

User Research: Conduct surveys and interviews with potential users to gather insights on their preferences, needs, and pain points related to chat applications.

Competitor Analysis: Analyze existing chat applications to identify strengths and weaknesses, focusing on user interface design, multilingual support, and chatbot capabilities.

Define Functional Requirements: Based on the research findings, compile a list of functional and non-functional requirements for the application, including features, performance metrics, and usability standards.

2. Design Phase

UI/UX Design: Create wireframes and prototypes for the user interface using design tools (e.g., Figma, Adobe XD). Incorporate user feedback to refine the designs.

Architecture Design: Develop a system architecture diagram that outlines the components of the application, including the frontend, backend, database, and third-party services (e.g., speech recognition API).

Database Design: Design the database schema to store user data, chat histories, and language preferences, ensuring data integrity and security.

3. Development Phase

Frontend Development: Implement the user interface using web technologies (e.g., HTML, CSS, JavaScript) or mobile frameworks (e.g., React Native, Flutter). Integrate dynamic elements and animations as per the design specifications.

Backend Development: Develop the server-side logic using a suitable programming language (e.g., Python, Node.js) and framework (e.g., Flask, Express). Implement RESTful APIs to facilitate communication between the frontend and backend.

Chatbot Development: Utilize machine learning libraries (e.g., TensorFlow, NLTK) to develop the chatbot. Train the model on relevant datasets to ensure accurate understanding and generation of responses in both Tamil and English.

Voice Recognition Integration: Integrate a speech recognition service (e.g., Google Speech-to-Text API) to enable voice input functionality. Ensure that the system can accurately transcribe user speech into text.

4. Testing Phase

Unit Testing: Conduct unit tests for individual components (frontend, backend, chatbot) to ensure that each part functions correctly.

Integration Testing: Perform integration testing to verify that all components work together seamlessly, focusing on data flow between the frontend and backend, as well as the chatbot's interaction with users.

User Acceptance Testing (UAT): Organize user testing sessions to gather feedback on usability, performance, and overall experience. Make necessary adjustments based on user input.

5. Deployment Phase

Deployment Preparation: Prepare the application for deployment by setting up hosting services (e.g., AWS, Heroku) and configuring the production environment.

Launch: Deploy the application to the chosen platform, ensuring that all features are functional and accessible to users.

Post-Launch Monitoring: Monitor the application for performance issues, user feedback, and potential bugs. Implement a feedback mechanism for users to report issues and suggest improvements.

6. Documentation and Maintenance

User Documentation: Create comprehensive user manuals and guides that explain how to use the application, including features, troubleshooting tips, and FAQs.

Technical Documentation: Document the codebase, architecture, and APIs for future reference and maintenance.

Ongoing Maintenance: Establish a plan for regular updates, bug fixes, and feature enhancements based on user feedback and technological advancements.

Real time example

Scenario: Personal Assistant for Mental Well-beingImagine a user named Sarah who is feeling a bit overwhelmed and wants to talk to someone about her feelings and find some uplifting music to listen to. She opens the AI chatbot application created using the MAIN_FILE.py script.

1. **User Interaction:**
    - **Sarah types in the input field or gives an voice over: "I feel really stressed and anxious about my work."**
    - **She clicks the "Send" button.**
2. **Emotional Analysis:**
    - **The chatbot processes her input using the emotional analysis function, detecting feelings of stress and anxiety.**
3. **Chatbot Response:**
    - **The AIML-based chatbot generates a supportive response, such as: "I'm sorry to hear that you're feeling this way. Would you like to listen to some calming music?"**
4. **User Decision:**

- **Sarah responds: "Yes, please play some calming music."**

5. **Mood Analysis and Playback:**
    - **The chatbot identifies her mood and retrieves a playlist of calming music based on her emotional state.**
    - **It plays the selected calming music using the audio playback functionality.**
6. **Language Switching (Optional):**
    - **If Sarah speaks Tamil, she can switch the interface language at any time and continue the interaction in her preferred language.**
7. **Continued Support:**
    - **As Sarah continues to chat with the bot, it provides motivational quotes, suggests breathing exercises, or even offers to open a meditation app if she requests it.**

This real-time example showcases how the AI chatbot can act as a supportive personal assistant, helping users manage their emotions and providing tailored music or suggestions based on their needs.

its key components and functionalities:

1. **Imports and Libraries: The script imports a wide range of libraries for various functionalities, including GUI creation (customtkinter, tkinter), audio processing (pygame, pyttsx3, gTTS), natural language processing (nltk, TextBlob, speech_recognition), and more.**

2. **Global Variables and Constants: It defines several global variables and constants for managing the application's state, such as current_mode, is_video_playing, AUDIO_DIR, and more.**

3. **Emotional Analysis: The script includes functions for emotional analysis of text in both English and Tamil. It uses predefined lexicons to map words to emotions and analyzes text using nltk and TextBlob.**

4. **Audio Management: Functions are provided to create and delete audio files, as well as to convert text to speech using pyttsx3 and gTTS.**

5. **Chatbot Functionality: The script uses AIML (Artificial Intelligence Markup Language) for chatbot responses and includes functions to handle user input, generate responses, and manage conversations.**

6. **GUI Setup: The script sets up a GUI using customtkinter, with features like a chat history display, input fields, and buttons for sending messages and switching languages.**

7. **Video and Audio Playback: It includes functionality to play video and audio files, using cv2 for video playback and pygame for audio.**

8. **Language Switching: The chatbot can switch between English and Tamil, with corresponding changes in the interface and functionality.**

9. **Mood Analysis and Playlist Management: The script can analyze the user's mood based on text input and play corresponding music playlists using YouTube API integration.**

10. **Additional Features: The script includes features like opening applications and websites, translating text between English and Tamil, and providing word meanings using wordnet.**

Code:

```
################################################################################
import datetime
import itertools
import customtkinter as ctk
import tkinter as tk
from tkinter import scrolledtext
import pygame
import uuid
import speech_recognition as sr
import pyttsx3
import os
import aiml
import nltk
import pyjokes
from nltk.corpus import wordnet
from PIL import Image, ImageSequence
from googletrans import Translator
import threading
from gtts import gTTS
import cv2
import atexit
from nltk.sentiment import SentimentIntensityAnalyzer
from textblob import TextBlob
import re
from collections import Counter
import random
import time
import warnings
import webbrowser
from googleapiclient.discovery import build
from CTkMessagebox import CTkMessagebox
```

Ln 1, Col 1

```
    cv2.namedWindow("Video Player", cv2.WND_PROP_FULLSCREEN)
    cv2.setWindowProperty("Video Player", cv2.WND_PROP_FULLSCREEN, cv2.WINDOW_FULLSCREEN)

    # Get video properties for FPS calculation
    video_fps = cap.get(cv2.CAP_PROP_FPS)

    # Initialize audio
    pygame.mixer.init()
    pygame.mixer.music.load(logo_audio_path)

    # Start playing audio
    pygame.mixer.music.play()

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        # Display video frame
        cv2.imshow('Video Player', frame)

        # Wait for a short period to maintain the frame rate, adjusted by speed factor
        wait_time = int(1000 / (video_fps * speed_factor))  # Adjust wait time based on speed factor
        if cv2.waitKey(wait_time) & 0xFF == ord('q'):
            break

    # Release resources
    cap.release()
    cv2.destroyAllWindows()
    pygame.mixer.music.stop()
    pygame.quit()

# Main function
if __name__ == "__main__":
    speed_factor = 1.75  # Set speed factor (e.g., 2.0 for double speed)
    logo(video_path, speed_factor)  # Call the logo function to play the video
    create_audio_dir()
    atexit.register(delete_audio_files)
    setup_ui()
    window.mainloop()
```

Full python code in the gdrive link:

https://drive.google.com/file/d/104wMTPKu_51XosoZzdy88XO-J34zEEBo/view?usp=sharing

Algorithm for the AI Chatbot Script

1. **Initialization:**
    - **Import necessary libraries (e.g., customtkinter, pygame, nltk, etc.).**
    - **Define global variables and constants (e.g., current_mode, AUDIO_DIR, etc.).**
2. **Setup GUI:**
    - **Initialize the main window using customtkinter.**
    - **Create GUI components:**
        - **Text display area for chat history.**
        - **Input field for user messages.**
        - **Buttons for sending messages and switching languages.**
3. **Define Functions:**
    - **Emotional Analysis:**

        - **Create functions to analyze user input for emotions in English and Tamil.**

- **Audio Management:**
    - **Define functions for creating, deleting, and playing audio files.**
    - **Implement text-to-speech conversion using pyttsx3 and gTTS.**
- **Chatbot Functionality:**
    - **Implement functions to handle user input and generate responses using AIML.**
- **Video and Audio Playback:**
    - **Create functions to play video and audio files.**
- **Language Switching:**
    - **Implement functionality to switch between English and Tamil interfaces.**
- **Mood Analysis and Playlist Management:**
    - **Define functions to analyze the user's mood and play corresponding music playlists.**
- **Additional Features:**

        - **Implement features for opening applications, translating text, and providing word meanings.**

4. **Event Handling:**
    - **Set up event listeners for user interactions (e.g., button clicks, text input).**
    - **On button click:**
        - **Capture user input from the text field.**
        - **Process input for emotional analysis.**
        - **Generate a response from the chatbot.**
        - **Update chat history display.**
        - **Play audio or video if applicable.**
5. **Main Loop:**
    - **Start the GUI main loop to keep the application running and responsive to user interactions.**
6. **Exit Procedure:**
    - **Implement functionality to safely exit the application when requested by the user.**

Literature Review

1. User Interface Design

User interface (UI) design is critical in enhancing user experience (UX) in software applications. According to Norman (2013), a well-designed UI should be intuitive and responsive, allowing users to navigate easily. The use of graphical elements, such as buttons and entry fields, is essential for creating an engaging interface. Research by Shneiderman and Plaisant (2010) emphasizes the importance of visual aesthetics and functionality in UI design, which aligns with the project's use of customizable widgets and color animations to enhance user interaction.

2. Animation Techniques

Animation in UI design can significantly improve user engagement and satisfaction. Studies by Bederson and Hollan (1994) highlight how animated transitions can provide visual cues, making the interface more dynamic and enjoyable. The project employs a neon border animation for chat history sections, which draws on principles of color theory and motion to capture user attention. Further, research by Chang et al. (2014) demonstrates that smooth transitions and animations can

enhance the perception of responsiveness in applications, supporting the decision to implement a continuous color-changing effect in the chat interface.

3. Chatbots and Conversational Interfaces

Chatbots have gained popularity as tools for automating interactions and providing instant responses. According to a study by Adamopoulou and Moussiades (2020), chatbots can enhance user engagement by simulating human-like conversations. The project integrates a chatbot that responds to user input, utilizing natural language processing (NLP) techniques to understand and generate responses. Research by Liu et al. (2018) indicates that the effectiveness of chatbots is significantly influenced by their ability to understand context and maintain coherent dialogues, which is a focus of the project's design.

4. Multilingual Support

The ability to support multiple languages in applications is increasingly important in a globalized world. Research by Wang et al. (2019) highlights the challenges and strategies in developing multilingual systems, emphasizing the need for accurate language processing and cultural sensitivity. The project addresses these challenges by incorporating Tamil and

English language support, allowing users to switch between languages seamlessly. This aligns with findings by Garcia (2018), which suggest that multilingual interfaces can significantly improve accessibility and user satisfaction in diverse user populations.

5. Speech Recognition and Voice Interaction

Voice interaction is becoming a vital component of modern applications, particularly in enhancing accessibility. According to studies by Zajac and Zajac (2021), voice recognition technology allows users to interact with applications hands-free, improving usability for individuals with disabilities. The project implements voice commands for sending messages, drawing on advancements in speech recognition technologies that enable accurate transcription and command execution. Research by Hwang et al. (2020) supports the idea that integrating voice recognition can enhance user experience by providing an alternative input method that is both efficient and intuitive.

Conclusion

This script represents a significant advancement in the development of interactive AI chatbots, combining natural language processing, emotional intelligence, and multimedia capabilities within a user-friendly graphical interface. By leveraging various libraries and technologies, this program not only engages users in conversation but also provides meaningful support tailored to their emotional states.The chatbot's ability to analyze emotions, generate contextually appropriate responses, and play music enhances user interaction, making it a valuable tool for mental well-being and personal assistance. Furthermore, its versatility in supporting multiple languages broadens its accessibility, allowing users from different backgrounds to benefit from its features.Overall, this program exemplifies the potential of AI in enhancing daily life through empathetic communication and personalized experiences, paving the way for future developments in intelligent personal assistants.