Project Report: AI-Powered Therapist Chatbot and Music Generation System

Made by :
Tanmay Parab(24P0620014)
Ramachandra Kaloji(24P0620010)

Introduction

As part of a Master's degree program in AI, this project develops a web-based application that integrates artificial intelligence (AI) to simulate a therapeutic chatbot and generate personalized music to enhance user mood. The system combines natural language processing (NLP), facial emotion recognition, and generative AI to create an interactive platform where users can engage in therapeutic dialogue, input music preferences, provide facial images for sentiment analysis, and receive AI-generated music tracks playable in the browser. The project demonstrates the potential of AI in mental health support and creative expression, serving as a proof-of-concept for innovative human-computer interaction (HCI) applications. This report outlines the problem addressed, the approach taken, system design, technologies used, results achieved, challenges encountered, future work, and conclusions drawn from the project.

Problem Statement

Mental health support is increasingly in demand, but access to professional therapy is limited by cost, availability, and stigma. While AI cannot replace human therapists, it can provide accessible, simulated therapeutic interactions and mood-enhancing interventions like music. Existing chatbot and music generation systems often lack personalization, multi-modal input (e.g., text and images), or seamless integration. This project addresses the following problems:

- **Lack of Accessible Therapeutic Tools**: Develop an AI-driven system to simulate empathetic dialogue and deliver mood-enhancing music.
- **Limited Personalization**: Incorporate user preferences (text) and emotional states (via facial analysis) to tailor music outputs.
- **Integration Challenges**: Combine NLP, computer vision, and generative AI into a cohesive, user-friendly web application.
- **Technical Constraints**: Ensure efficient, local AI processing and robust file management for a scalable prototype.

The goal is to create a system that leverages AI to provide a therapeutic experience, using multi-modal inputs to generate personalized music, while addressing technical and usability challenges.

Approach

The project adopts a multi-faceted approach to integrate AI technologies into a Flask-based web application:

1. **Conversational AI**:
   - Use the Ollama framework with the `gemma3:4b` model to create a therapist chatbot, initialized with a system prompt (eg:`"act as my therapist"`) for empathetic responses.
   - Save user messages to `user.txt` for potential future analysis.
2. **User Input Collection**:
   - Implement a modal for users to input music preferences (e.g., genres, artists), saved to `final.txt`.
   - Enable facial image uploads for sentiment analysis, with results appended to `final.txt`.
3. **Facial Emotion Recognition**:
   - Use the Facial Expression Recognition (FER) library to detect emotions (e.g., happy, sad) from images, enhancing music personalization.
4. **Music Generation**:
   - Leverage the Ollama `gemma2:2b` model to generate music prompts from `final.txt` content.
   - Use the Beatoven API to create WAV tracks based on these prompts, saved to a `music` folder.
5. **Web Integration**:
   - Develop a responsive web UI with HTML, JavaScript, and Tailwind CSS, featuring chat, input modals, and an audio player.
   - Implement Flask endpoints to manage data flow and serve music files.
6. **File Management**:
   - Clear `user.txt` and `final.txt` after successful music generation to reset the system.

This approach ensures a cohesive system that processes multi-modal inputs, generates personalized outputs, and provides immediate feedback via music playback.

System Design

The system follows a client-server architecture, with AI components integrated for processing and generation:

- **Client**:
    1. A web interface (`index.html`) built with HTML, JavaScript, and Tailwind CSS.
    2. Features:
        - Chat UI for therapist interaction.
        - Modal for music preferences input.
        - File upload for facial images.
        - Audio player for music playback.
    3. JavaScript handles API calls, UI updates, and dynamic content (e.g., `appendAudioMessage` for music).
- **Server**:
    1. A Flask application (`app.py`) with endpoints:
        - `/save_message`: Saves chat messages to `user.txt`.
        - `/save_preferences`: Saves music preferences to `final.txt`.
        - `/analyze_face`: Analyzes facial images and saves sentiments to `final.txt`.
        - `/gen`: Triggers music generation via `generatemusic.py`, clears files, and returns the music filename.
        - `/music/<filename>`: Serves music files from the `music` folder.
- **AI Components**:
    1. **Ollama**:
        - Runs locally (`http://localhost:11434`) with `gemma3:4b` (chat) and `gemma2:2b` (prompts).
        - Processes user messages and generates music prompts.
    2. **FER**:
        - Uses TensorFlow and MTCNN for facial emotion detection.
        - Processes images server-side, outputting emotions (e.g., “happy”).
    3. **Beatoven API**:
        - Generates WAV tracks from text prompts, downloaded to `music/`.
- **Files**:
    1. `user.txt`: Stores timestamped chat messages.
    2. `final.txt`: Stores timestamped preferences and sentiments.
    3. `music/`: Stores generated WAV files.
- **Workflow**:
    1. User interacts via chat, preferences modal, or image upload.
    2. Inputs are saved to `user.txt` (chat) or `final.txt` (preferences, sentiments).
    3. The `/gen` endpoint triggers `generatemusic.py`, which:
        - Reads `final.txt`, generates a prompt via `gemma2:2b`.
        - Sends the prompt to Beatoven for music creation.
        - Saves the track to `music/`.
    4. Files are cleared, and the music file is served via `/music/<filename>` for playback.

The design ensures modularity, with Flask coordinating AI components and file operations, while the UI provides a seamless user experience.

Technologies Used

- **AI Frameworks**:
  - **Ollama**: Runs `gemma3:4b` and `gemma2:2b` for NLP tasks (chat and prompt generation).
  - **FER**: Facial emotion recognition using TensorFlow and MTCNN.
  - **Beatoven API**: AI-powered music generation from text prompts.
- **Backend**:
  - **Flask**: Python web framework for API endpoints and file serving.
  - **OpenCV**: Image processing for FER.
  - **Requests**: HTTP requests for Beatoven and Ollama APIs.
- **Frontend**:
  - **HTML/JavaScript**: Dynamic UI with chat, modals, and audio playback.
  - **Tailwind CSS**: Responsive, modern styling.
- **Dependencies**:
  - Python libraries: `flask`, `fer`, `opencv-python`, `requests`, `tensorflow`.
- **Development Environment**:
  - Python 3.8-3.10, local Ollama server, Beatoven API token.
- **AI Topics**:
  - **NLP**: Conversational AI, prompt engineering.
  - **Computer Vision**: Facial sentiment analysis.
  - **Generative AI**: Music creation from text inputs.

Results

- **Therapist Chatbot**:
  - The `gemma3:4b` model delivers empathetic responses (e.g., “I’m here to help. What’s stressing you out?” for “I’m stressed”).
  - Messages are saved to `user.txt` with timestamps, enabling future analysis.
  - Limitation: Stateless API calls limit context retention.
- **Facial Sentiment Analysis**:
  - FER accurately detects emotions in well-lit, single-face images (e.g., “happy” with 70% confidence).
  - Sentiments are appended to `final.txt` (e.g., “[2025-04-27 10:16:45] Facial Sentiment: happy”).
  - Success rate: ~80% on test images, fails with poor lighting or multiple faces.
- **Music Generation**:
  - The `gemma2:2b` model generates relevant prompts (e.g., “uplifting jazz fusion track” for “Jazz, happy”).

    - Beatoven produces high-quality WAV tracks, saved to `music/` and playable via the UI's `<audio>` player.
    - Example: Input "Classical, sad" → Prompt "soothing classical piano track" → Track played in browser.
- **User Experience**:
    - Responsive UI with intuitive chat, modal inputs, and audio playback.
    - Loading indicators and error messages (e.g., "No face detected") enhance usability.
    - File clearing post-generation ensures a clean state for new interactions.
- **Educational Impact**:
    - Demonstrates integration of NLP, computer vision, and generative AI for a Master's-level project.
    - Showcases prompt engineering, system design, and HCI skills.

Challenges

1. **Model Selection**:
    - **Issue**: Balancing performance and resource usage for Ollama models.
    - **Solution**: Used `gemma3:4b` for chat (higher quality) and `gemma2:2b` for prompts (faster, lighter).
2. **FER Limitations**:
    - **Issue**: Fails with multiple faces, poor lighting, or non-frontal images.
    - **Solution**: Restricted to single-face detection, added error messages (e.g., "No face detected").
3. **Beatoven API Integration**:
    - **Issue**: Polling for task completion and managing API credits.
    - **Solution**: Implemented 5-second polling in `check_task_status`, ensured valid token.
4. **Prompt Engineering**:
    - **Issue**: Broad therapist prompt (`"act as my therapist"`) led to inconsistent responses; `final.txt` parsing was complex due to timestamps.
    - **Solution**: Relied on model pre-training for chat; constrained music prompts to ensure concise outputs.
5. **File Management**:
    - **Issue**: Ensuring files are cleared only on successful generation.
    - **Solution**: Cleared `user.txt` and `final.txt` in `/gen` only after verifying a music file exists.
6. **Audio Playback**:
    - **Issue**: Serving large WAV files with browser compatibility.
    - **Solution**: Used Flask's `send_from_directory` and native `<audio>` element.

Future Work

1. **Chat Context Persistence**:
    - Modify the chatbot to maintain conversation history, improving dialogue coherence by prepending prior messages or the therapist prompt.
2. **Enhanced FER**:
    - Support multiple faces or use advanced models like DeepFace for better accuracy under varied conditions.
3. **Music Customization**:
    - Allow users to specify track length, tempo, or looping in Beatoven API calls.
4. **File Management**:
    - Implement log rotation for `user.txt` and `final.txt` to manage growth.
    - Use unique filenames or timestamps for music files to avoid overwrites.
5. **Professional Validation**:
    - Collaborate with therapists to validate chatbot responses and music's therapeutic impact.
6. **Scalability**:
    - Deploy Ollama on a dedicated server or use cloud-based LLMs for production.
    - Optimize Beatoven API calls for multi-user scenarios.
7. **Prompt Engineering**:
    - Refine the therapist prompt with explicit instructions (e.g., "use empathetic language, ask open-ended questions").
    - Pre-process `final.txt` to remove timestamps for cleaner music prompt generation.

Conclusion

This project successfully developed a Flask-based web application integrating AI technologies—Ollama for NLP, FER for facial emotion recognition, and Beatoven for music generation—to simulate a therapeutic chatbot and generate personalized music. The system addresses the need for accessible, AI-driven mental health tools by combining conversational AI, multi-modal inputs, and generative music in a user-friendly interface. Despite challenges like model limitations and prompt ambiguity, the project achieved robust results, with a functional chatbot, accurate sentiment analysis, and playable music tracks. As a Master's student, this work demonstrates proficiency in AI system design, prompt engineering, and HCI, contributing to the growing field of AI in mental health and creative applications. Future enhancements in context persistence, FER accuracy, and professional validation could elevate the system's impact, paving the way for further research and development.