AI Mind Reset Bot Documentation

Student Name: **Askal Thapa**

Student RollNo: **2315434**

Date: **January 31, 2026**

Link to git project repo: https://github.com/askal-thapa/chat-bot

1 Introduction

The **AI Mind Reset Bot** is an intelligent and empathetic chatbot designed to support users in managing stress, anxiety, and emotional well-being. Unlike standard chatbots, it uses intent classification to distinguish between casual small talk, therapeutic needs, and off-topic queries.

A key feature of this project is the **Mood Engine**, which updates the application's user interface theme in real-time based on the user's detected emotional state. This makes the chatbot experience more immersive, visually engaging, and emotionally responsive.

2 Background

Mental health and emotional well-being applications have increasingly adopted AI-based conversational systems to provide quick and accessible support. Large Language Models (LLMs) such as Gemini can generate human-like responses, but for a mental wellness chatbot it is important to ensure that responses are safe, structured, and aligned with the user's needs.

This project is inspired by modern AI chatbot systems that provide calming conversations, emotional validation, and guided activities such as breathing exercises and grounding techniques. To improve reliability, this system uses structured intent classification and routing, rather than treating every user message the same way.

3 Implementation Details for Interactive Application

3.1 Technology Stack

The application is developed using the following tools and frameworks:

- **Frontend:** Next.js 16 (React 19), TailwindCSS 4, shadcn/ui
- **Backend:** FastAPI (Python)
- **AI Orchestration:** LangGraph (stateful agent workflow)
- **LLM Integration:** LangChain with Google Gemini API

- **Validation:** Pydantic models for structured outputs

3.2 System Architecture

The system follows a decoupled architecture:

- User interacts through the browser UI.
- Next.js frontend sends user messages to the FastAPI backend.
- Backend invokes LangGraph agent logic for intent routing.
- LangChain connects to Gemini to generate responses and mood.
- Frontend updates the UI theme dynamically based on mood.

3.3 LangChain and LangGraph Usage

This project uses **LangChain** and **LangGraph** to create a controlled, stateful chatbot experience.

- **LangChain** is used to connect the backend to the Gemini model and handle prompting.
- **LangGraph** is used to create a graph-based conversation flow where each node has a specific responsibility.
- The graph makes the system modular and easier to debug and extend.

3.4 Intent Classification and Routing

The chatbot uses an intent classification step to detect the type of user message:

- `small_talk` — casual conversation, jokes, fun facts
- `therapy` — emotional support, grounding exercises, calming techniques
- `off_topic` — gently redirecting user back to well-being focus

After intent classification, the request is routed to the correct handler node. This improves relevance and ensures that therapy-style responses are only provided when appropriate.

3.5 Structured Output Format

Instead of returning only plain text, the bot returns structured output which helps the frontend render responses consistently.

Listing 1: Structured Bot Response Model

```
class BotResponse(BaseModel):
    response: str
    mood: str
    suggested_activity: str
    activity_type: str
```

This structured response enables features such as:

- Mood-based UI theming
- Suggested activities (breathing, meditation, jokes)
- Activity-type based UI components

3.6 Dynamic Mood Engine (Frontend)

The Mood Engine is implemented using a **Mood Provider** in the frontend. When the backend returns a mood string, the frontend maps it to a theme palette and updates CSS variables.

Example mood themes:

- **Happy:** warm yellows and greens
- **Anxious:** calming purples and violets
- **Angry:** intense reds and oranges
- **Neutral:** clean sky blues

This provides a smooth and responsive experience with transitions across the UI.

4 Details of Additional Experimentation

During the development of this project, additional experimentation included:

- Testing multiple prompts to improve intent classification accuracy.
- Improving mood detection labels for more stable UI updates.
- Experimenting with different supportive activities such as breathing and grounding prompts.
- Reviewing how to safely redirect off-topic conversations without frustrating the user.

5 Critical Reflection

This project successfully demonstrates how an AI chatbot can be made more emotionally supportive through structured intent routing and mood-aware UI theming. The strongest part of the application is its ability to adapt both the response content and the interface design based on user emotion.

However, there are limitations:

- LLM classification may sometimes mislabel intent.
- The system must be careful about giving mental-health advice and should not replace professional help.
- The project currently does not store session history in a database.

If more time were available, improvements would include streaming responses, voice mode, and persistent session storage.

6 LLM Disclaimer

Large language models were used to assist in generating responses, supporting structured output formatting, and enabling intent classification. All content was reviewed and adapted to ensure it fits the safe and supportive purpose of the project.

7 Bibliography

- Association for Computing Machinery (2018). *ACM Code of Ethics and Professional Conduct.* Available at: https://www.acm.org/code-of-ethics