Mindful Moment Agent: Engineering Report

Final Project Submission
December 2025

1. Introduction

For this project, the goal was to build a "Mindful Moment Agent"—a simple AI companion that helps users manage stress and emotions. The key requirement was privacy. Most AI apps send data to the cloud, but for something personal like mental health, we wanted everything to run locally on the user's computer.

The system uses a local Large Language Model (Llama 3.2) to analyze how a user is feeling and then suggests a quick "micro-action" (like a breathing exercise) to help them feel better.

2. System Design

The application is built using Python and FastAPI for the backend, with a simple HTML/CSS frontend. We chose a micro-service approach where different "agents" handle different parts of the job.

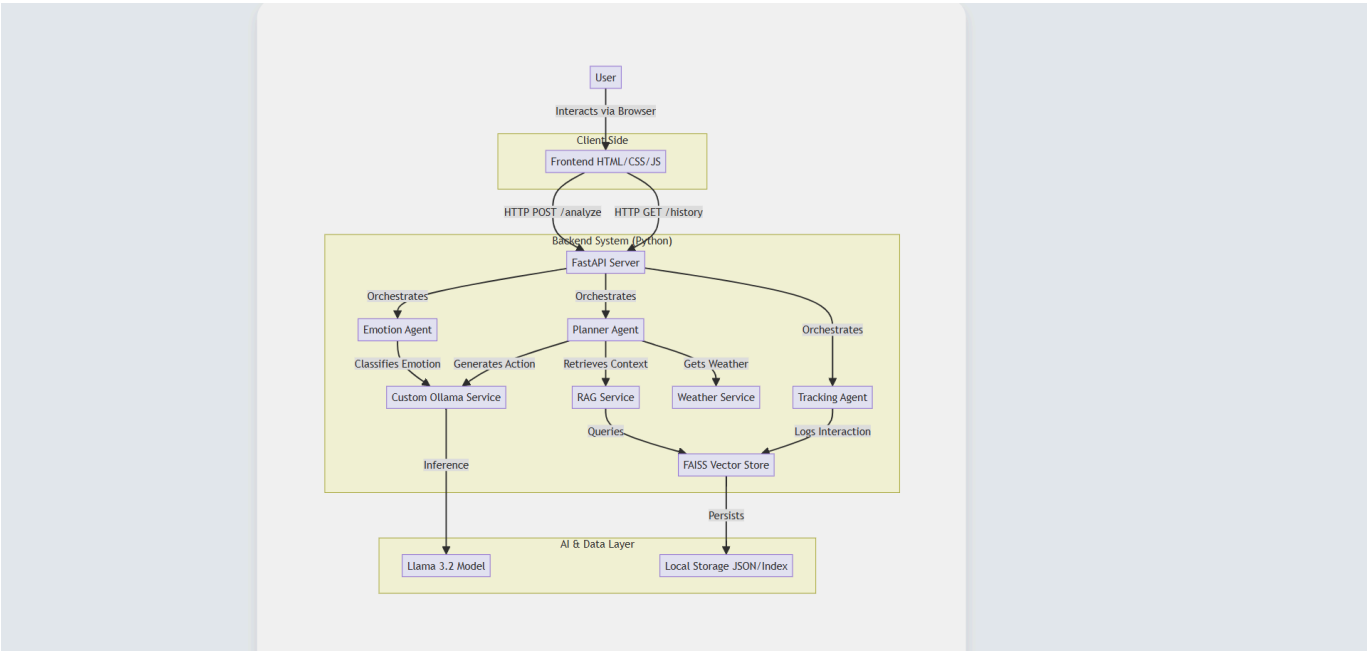

Figure 1: Overview of the system components.

Key Components

- **Frontend:** A clean web interface where users type their thoughts. We used a "Glassmorphism" style to make it look modern and calming.
- **Backend (FastAPI):** This acts as the main controller. It receives the text from the frontend and passes it to the agents.
- **Emotion Agent:** This part of the code is responsible for figuring out the user's mood. It sends a prompt to the Llama 3 model asking it to classify the text into categories like "Stressed," "Happy," or "Anxious."
- **RAG Service:** We didn't want the AI to just hallucinate advice. So, we built a database of over 500 verified wellness tips. The system searches this database for tips that match the user's current emotion.

3. Implementation Details

Working with Local LLMs

One of the biggest hurdles was getting the local AI to work smoothly with our Python code. We initially tried using standard libraries like LangChain, but we kept running into compatibility errors with our specific Python version.

To fix this, we ended up writing a custom script to talk to the Ollama API directly. It was a bit more work upfront, but it actually made the system faster and gave us more control over exactly what we sent to the model.

Data Generation

When we started, we only had about 10 wellness tips, which got repetitive very quickly. To solve this, we wrote a script to generate a larger dataset. We created templates for different emotions and used them to generate around 570 unique actions. This makes the app feel much more "alive" because it rarely repeats the same advice twice.

4. How It Works (Workflow)

Here is what happens when a user types something like "I'm really tired of work":

1. The text is sent to the backend API.
2. The **Emotion Agent** analyzes it and decides the user is "Tired" (Confidence: 90%).
3. The **RAG System** looks up "Tired" in our database and finds actions like "Hydration Boost" or "Quick Stretch."
4. The **Planner Agent** takes the best suggestion and formats it into a nice message for the user.
5. Finally, the interaction is saved to a local history file so the user can see it later.

5. Conclusion

This project successfully demonstrates that you don't need a massive cloud server to build useful AI applications. By using efficient tools like FastAPI and local models like Llama 3, we built a privacy-focused wellness agent that is responsive and helpful.

Future improvements could include adding voice support or connecting it to a smartwatch to detect stress automatically, but the current version fulfills all the core requirements set out at the beginning.