Multi-Agent Healthcare Assistant – Project Documentation

Overview

This project implements an **AI-powered multi-agent healthcare assistant** using the **AGNO framework**, **FastAPI**, and a **Next.js frontend**. The assistant performs various tasks such as user greeting, mood tracking, CGM (glucose) monitoring, food logging, and adaptive meal planning. It supports both a visual dashboard and a conversational AI interface.

Backend Setup

1. Environment Setup

- Installed **UV** as the package manager.
- Installed required Python libraries:
  - `agno`
  - `ag-ui-protocol`
  - `sqlalchemy`
  - `faker`
  - `fastapi`

2. Sample Data Generation (`generate_data.py`)

Created a fake **SQLite health database** (`users.db`) with the following data:

- **100 fake users** using Faker:
  - Name, city, dietary preference, medical conditions, physical limitations
- For users with **Type 2 Diabetes**:
  - Logged **7 days of random CGM readings**
- For **all users**:

- Logged **5 random mood entries** with timestamps

This simulates realistic test data for the assistant without needing real health records.

Agent System Architecture

Created a folder `agents/` and defined **six specialized agents** using the AGNO framework:

1. Greeting Agent

- **Tool**: `get_user_by_id`
- Fetches user info from the database by user ID.
- Greets user using their name and city.
- Handles invalid IDs gracefully.

2. Mood Tracker Agent

- **Tools**:
  - `log_mood_entry`: Logs user mood with a timestamp.
  - `get_mood_history`: Retrieves user's past moods.
- Stores data in SQLite and uses `Memory` for conversational context.
- Provides mood summaries and trends.

3. CGM (Glucose Monitor) Agent

- **Tool**: `log_glucose_reading`
- Logs blood glucose levels.
- Analyzes if readings are:
  - Too low (<80 mg/dL)
  - Normal (80–300 mg/dL)
  - Too high (>300 mg/dL)
- Suggests appropriate actions.

4. Meal Planner Agent

- **Tools**:
  - `get_user_by_id`
  - `get_cgm_context`: Fetches latest CGM data
- Uses mood, health, and glucose data to generate a **3-meal adaptive plan**.
- Acts like a virtual nutritionist using memory + agentic context.

5. Food Log Agent

- **No DB/tool yet** – powered by OpenAI LLM only.
- Takes a meal description and breaks it down into **macronutrients** (carbs, protein, fat).
- Useful for nutrition awareness.

6. General Interrupt Agent

- Handles **general questions** and out-of-context queries.
- Keeps the conversation flowing and brings the user back to the task.

Team Setup: `healthcare_team`

- Uses **AGNO Team** in `coordinate` mode.
- Team members:
  - Greeting Agent
  - Mood Tracker Agent
  - CGM Agent
  - Food Intake Agent
  - Meal Planner Agent
  - Interrupt Agent
- Shares `Memory` and user context across agents.
- Uses `ReasoningTools` for intelligent responses.
- Enables **multi-agent decision making** with persistent state.

AG-UI Web Integration (FastAPI)

Wrapped the multi-agent team into a **FastAPI app** using `AGUIApp`:

- Route: `http://localhost:8000/agui`
- Registers all agents and memory
- Adds custom `api_routes` (like greeting, CGM history, mood summary)
- Runs server at **port 8000**

Frontend Setup

Starter Project

- Cloned CopilotKit + AGNO Starter
- Built with:
  - `Next.js` + `React`
  - `TailwindCSS` for styling
  - `Recharts` for graphs
  - `ReactMarkdown` for displaying structured responses

Features Implemented

Welcome Section

- Fetches user name using API
- Displays greeting on dashboard

CGM Chart

- Fetches last 7 glucose readings
- Visualizes as **line chart**

Mood Summary

- Counts mood entries from API
- Displays as **bar chart**

Food Logging

- User enters meal text (e.g. "dal & rice")
- Sends to agent
- Shows nutrient breakdown (markdown format)

Adaptive Meal Planning

- On button click, fetches a **3-meal plan**
- Tailored using health, mood, and CGM context

Chat Interface

- Chatbot interface powered by **CopilotChat**
- Uses same agent system as backend
- Allows natural conversation for all tasks

CopilotKit Runtime Setup (`routes.ts`)

This file enables **frontend-backend agent communication**:

- Sets up **OpenAIAdapter** for LLM reasoning
- Connects `agno_agent` hosted at `/agui` as a **multi-agent runtime**
- Creates a POST API route: `/api/copilotkit`

Used by `CopilotChat` to relay messages to backend agents

Tech Stack

Layer	Tech Used
Language	Python, TypeScript
Backend	FastAPI, SQLite, AGNO
Frontend	Next.js, React, TailwindCSS
Charts	Recharts
AI/LLM	OpenAI Chat API
State	AGNO Memory, SqliteStorage
Agent UI	AGUI (via CopilotKit)

`deploy/` Folder – Deployment Setup

This folder contains everything needed to run the full project using Docker.

`Dockerfile.backend`

- Builds the FastAPI + AGNO backend
- Installs Python and required packages
- Copies backend code (e.g., agents/)
- Starts the app with `uv run agents/main.py`

`Dockerfile.frontend`

- Builds the AG-UI (CopilotKit) Next.js frontend
- Installs Node.js and dependencies
- Copies frontend code
- Runs `npm run dev` to serve the app

`docker-compose.yml`

- Orchestrates both backend and frontend containers
- Maps ports `8000` (backend) and `3000` (frontend)
- Mounts the SQLite database from `backend/data/`
- Loads environment variables from `.env`

`.env`

- Stores sensitive keys like `OPENAI_API_KEY`
- Injects the API key securely into the backend container

Final Outcome

This project demonstrates a **complete end-to-end AI-powered healthcare assistant** capable of:

- Conversational and visual interaction
- Logging mood, meals, glucose

- Personalized meal recommendations
- Multi-agent task coordination
- Smooth UI with context-aware chat

It is modular, extensible, and works well for real-time health tracking and assistant systems.