Health & Wellness Planner Agent using OpenAI Agents SDK

create by:

Nabila Bannay Khan

slot: Monday
rollno: 00123375

Project Overview

To develop this Health & Wellness Planner Agent, the goal is to create an intelligent, personalized, and engaging AI-driven wellness assistant capable of understanding user goals, providing real-time feedback, managing context, and integrating specialized agents based on user needs. Below is a **professional approach** to building this system, including detailed architecture, features, and necessary steps.

Goal:

Build a fully functional, AI-powered Health & Wellness Planner that can:

1. Collect and analyze user health goals (fitness and dietary).
2. Generate personalized meal and workout plans.
3. Track progress over time.
4. Provide real-time interaction.
5. Integrate specialized agents (e.g., nutritionist, injury support) when required.

Key Requirements:

User Goal Collection: Use a conversational agent to gather and structure the user's fitness and dietary goals.

Meal Plan Generation: Provide 7-day personalized meal plans, considering dietary preferences (e.g., vegetarian, keto).

Workout Plan Generation: Generate personalized workout routines based on user goals and experience level.

Progress Tracking: Track user progress and schedule regular check-ins.

Real-Time Streaming: Interact with the user in real-time to provide immediate responses.

Agent Handoffs: Automatically escalate to specialized agents (e.g., nutrition expert, injury support) based on the user's needs.

Guardrails: Ensure that all inputs and outputs are validated and structured correctly to avoid errors.

System Architecture

The system architecture involves multiple components working together in a modular structure, with clear separation of concerns.

1. Core Components:

1. **Main Agent** (main.py):
   - Responsible for handling the flow of the user interaction.
   - Coordinates the tools and agents based on user input.
2. **User Session Context** (context.py):
   - Manages the user session state (goal, preferences, progress, handoff logs).
3. **Tools**:
   - **GoalAnalyzerTool**: Validates and structures user goals.
   - **MealPlannerTool**: Generates personalized meal plans based on dietary preferences.
   - **WorkoutRecommenderTool**: Recommends workout routines tailored to user goals.
   - **ProgressTrackerTool**: Tracks user progress and provides feedback.
   - **CheckinSchedulerTool**: Schedules and manages user check-ins.
4. **Specialized Agents**:
   - **NutritionExpertAgent**: Handles complex dietary needs (e.g., diabetes, allergies).
   - **InjurySupportAgent**: Offers workout modifications for physical limitations.
   - **EscalationAgent**: Redirects users to a human coach or trainer if needed.

2. Data Flow:

- **User Inputs** (goals, preferences) → **GoalAnalyzerTool** → Structuring → **MealPlannerTool/WorkoutRecommenderTool** → Real-time Streaming to User.
- **Progress Updates** → **ProgressTrackerTool** → User Feedback.
- **Specialized Needs** (e.g., injury, dietary concerns) → **Agent Handoffs** → Specialized Agents.

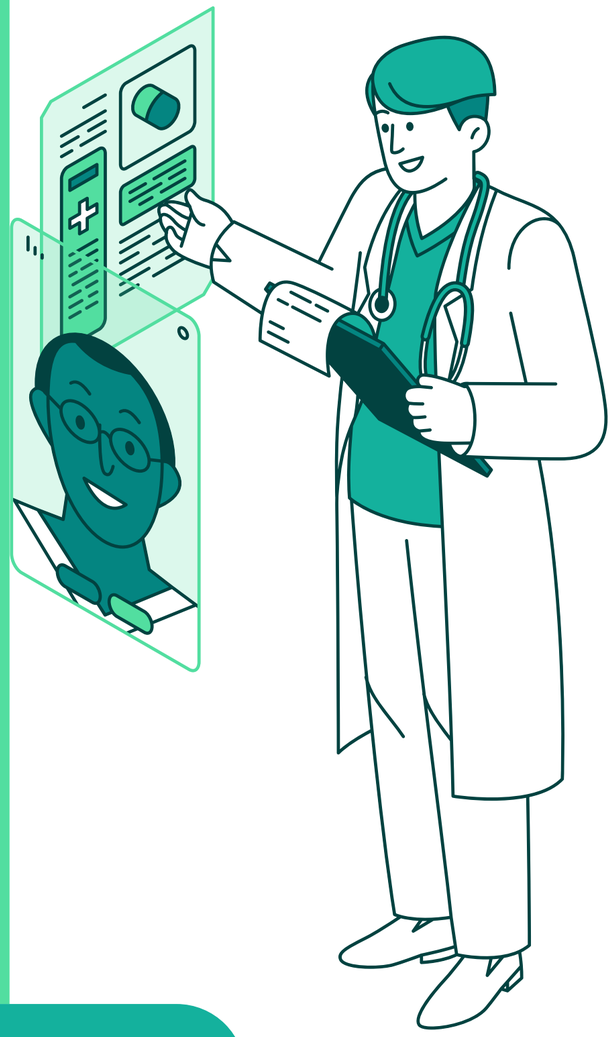
Tool & Agent Breakdown

1. GoalAnalyzerTool:

- **Input**: "I want to lose 5kg in 2 months."
- **Output**: Structured goal in JSON format (e.g., {"goal": "lose weight", "target": "5kg", "duration": "2 months"}).
- **Validation**: Ensures goal input is valid, including numeric values for weight, time, and specific goals (e.g., weight loss, muscle gain).

2. MealPlannerTool:

- **Input**: Dietary preferences, such as "vegetarian," "keto," "gluten-free."
- **Output**: A 7-day meal plan honoring those preferences.
  - **Example Output**: {"meal_plan": ["Day 1: Veggie Salad", "Day 2: Tofu Stir-fry", ...]}
- **Async Processing**: Meal suggestions are generated asynchronously to ensure smooth interaction and timely responses.



3. WorkoutRecommenderTool:

- **Input**: Goal (e.g., "lose weight"), fitness level (beginner, intermediate).
- **Output**: A structured workout plan.
  - **Example Output**: {"workout_plan": [{"day": "Monday", "exercises": ["Push-ups", "Squats", ...]}, ...]}
- **Tailored Plan**: Recommendations are based on the user's goal, experience, and available equipment.

4. ProgressTrackerTool:

- **Input**: Weekly progress updates from the user (e.g., "I lost 1kg this week").
- **Output**: Updated progress log and feedback.
  - **Example Output**: {"progress": {"weight": "4kg lost", "fitness_level": "intermediate"}}
- **Feedback**: Personalized feedback based on progress.

5. Specialized Agents:

- **EscalationAgent**:
  - **Trigger Condition**: User requests to speak to a human coach.
  - **Handoff Logic**: Redirects the user to a human coach or trainer.
- **NutritionExpertAgent**:
  - **Trigger Condition**: Complex dietary needs (e.g., diabetes, allergies).
  - **Functionality**: Provides expert nutritional advice.
- **InjurySupportAgent**:
  - **Trigger Condition**: User mentions injury or physical limitations.
  - **Functionality**: Offers alternative exercises and modifications.

Streaming and Real-Time Interaction

Real-Time Responses:

All tool outputs (e.g., meal plans, workouts) are streamed back to the user in real-time to ensure an engaging, chatbot-like experience.

Why Streaming Matters:

- **Engagement**: Users receive immediate feedback, improving the overall experience.
- **Efficiency**: Enables asynchronous processing of tools (e.g., meal planning), ensuring responsiveness.
- **Smooth Interaction**: Avoids delays, making the conversation feel natural and dynamic.

Guardrails & Validation

Input Guardrails:

1. **Goal Validation**: Ensure inputs like "I want to lose 5kg in 2 months" are in the correct format, with numeric values for weight and duration.
2. **Dietary Input**: Prevent invalid dietary inputs (e.g., conflicting preferences like "vegetarian" and "meat lover").

Output Guardrails:

1. **Structured Output**: All outputs from tools should be in JSON or structured format to ensure consistency and correctness.
2. **Safety & Reliability**: Outputs should respect medical or fitness safety standards, especially when dealing with specialized agents like NutritionExpert or InjurySupport.

Lifecycle Hooks

Use Cases:

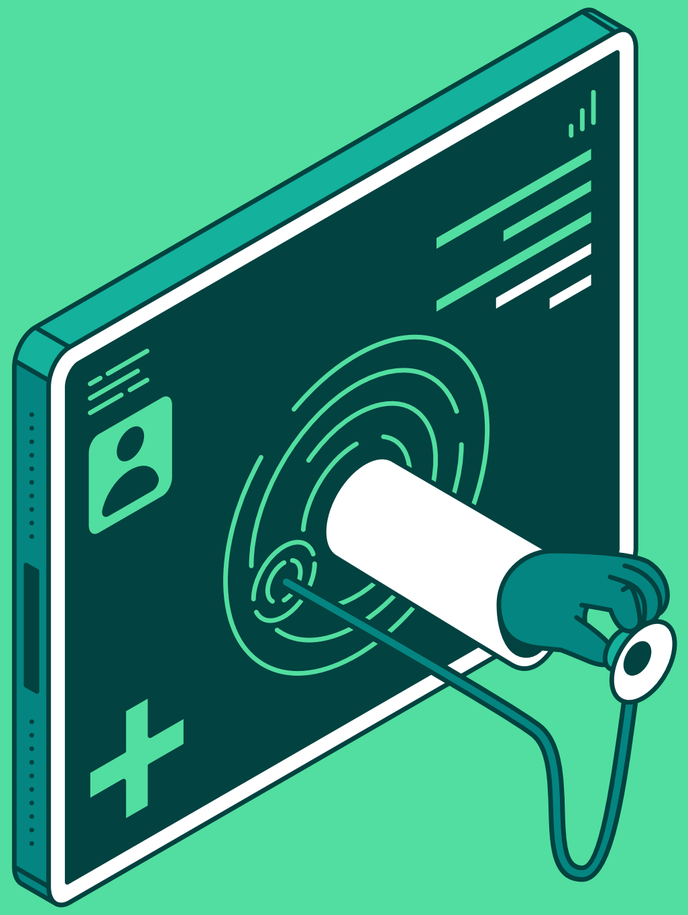
1. **Track User Interactions**: Log the number of interactions to understand user engagement and preferences.
2. **Debugging and Monitoring**: Use hooks to monitor tool invocations, agent handoffs, and responses, aiding debugging and performance optimization.
3. **Session Tracking**: Log progress at each stage (goal setting, meal planning, workout tracking) to enhance user experience.

Streamlit UI:

1. **Collect User Inputs:** Use form fields to collect goals and preferences.
2. **Display Results:** Render meal plans and workout routines in an organized, easy-to-read format.
3. **Real-Time Updates:** Use st.write to display updates as the user progresses, including meal plans and workout logs.

Backend with FastAPI

FastAPI will serve as the backbone of the system, handling requests for goal setting, meal plans, workout recommendations, and more.



FastAPI Setup:

FastAPI will receive requests from the frontend (Streamlit).
It will run the core logic of interacting with the tools and agents, validate inputs, and return structured responses.

Question: Why was this project important for me?

1. **Skill Development**: This project taught me **AI tools**, **FastAPI** backend, and **frontend integration**, enhancing my technical skills.
2. **Practical Exposure**: It provided the opportunity to implement **real-world AI solutions**, which will be helpful in future projects.
3. **Career Growth**: The project strengthened my **GitHub portfolio**, making it valuable for job applications and freelancing opportunities.

Key Technologies Used:

- FastAPI for backend API development
- OpenAI Agents SDK for building intelligent agents
- Streamlit/Next.js for frontend interface (Optional)
- Real-Time Data Streaming with WebSockets

The project aims to personalize fitness and wellness journeys by offering tailored recommendations, while also tracking progress and offering real-time, actionable insights to users.

Conclusion:

The AI-Powered Health & Wellness Planner successfully integrates **cutting-edge AI technologies** with practical health and fitness solutions. By combining **goal analysis**, **personalized meal planning**, **workout recommendations**, and **progress tracking**, this project creates a **dynamic, real-time interactive experience** for users. The integration of **specialized agents** for nutrition and injury support further elevates the platform's capability to provide comprehensive wellness guidance.

Through this project, I have gained valuable experience in **AI systems**, **full-stack development**, and **real-time streaming**, while also contributing to an area with growing demand—personalized health and wellness solutions. The skills developed here have enhanced my technical abilities and prepared me for future opportunities in both **AI-driven applications** and **health tech** industries.

In conclusion, this project serves as a robust, scalable foundation for building **intelligent health assistants**, and provides a solid base for future enhancements like **mobile app integration**, **deeper AI personalization**, and more.

Additional Implementation Details

In this assignment, I implemented the Health & Wellness Planner in both CLI-based and Streamlit-based interfaces:

- ✅ CLI Version: Allows users to interact via command line, useful for quick testing and terminal-based usage.
- ✅ Streamlit Interface: Provides a simple and user-friendly web interface where users can:
  - Set their personal goals
  - Choose dietary preferences
  - View workout or meal recommendations in real-time

Goal Tracking & PDF Export

- Users can set their fitness or dietary goals, and the system allows exporting those goals as a PDF file, saved locally with the current date and time for reference.

Database Integration

- I integrated a database (e.g., SQLite/PostgreSQL) to:
  - Store user goals
  - Track progress history
  - Save meal and workout plans
  - Retrieve data anytime for personalized recommendations

FastAPI Backend

- I used FastAPI to build a clean, efficient API layer between the frontend (Streamlit) and the AI logic.
- This backend:
  - Handles user input
  - Sends it to the AI agent
  - Returns structured responses to the UI
  - Supports future scalability (e.g., connecting with mobile apps or external services)

Thank You!

Thank you for reviewing this project. I hope it has provided insight into my approach towards building **AI-driven systems** and **real-time applications**. If you have any questions or feedback, feel free to reach out. I look forward to applying the knowledge and experience gained through this project to future challenges and opportunities.

Here's a quick **code snippet** showcasing a key part of the project: **Meal Plan Generation** using the AI-powered system.