**Assignment Document: Health & Wellness Planner Agent using OpenAI Agents SDK**

## 🔹 Overview

This assignment challenges you to build a fully functional AI-powered **Health & Wellness Planner Agent** using the OpenAI Agents SDK. The goal is to simulate a digital wellness assistant that can interact with users in natural language, understand their goals, and provide personalized suggestions and feedback.

The planner agent should:

* **Collect user fitness and dietary goals** through multi-turn natural language conversation.
* **Analyze those goals** and generate structured health plans (e.g., a 7-day vegetarian meal plan or a weekly strength training workout plan).
* **Use context and state** to remember past conversations and progress.
* **Stream responses** to users in real time for an engaging, chatbot-like experience.
* **Apply input and output guardrails** to ensure user input is valid and tool output is structured and trustworthy.
* **Handle handoffs** to other specialized agents such as a Nutrition Expert or Injury Support Assistant based on user needs.
* **(Optionally) Use lifecycle hooks** to track tool usage, logging, and handoff activities.

This assignment is designed to mimic a real-world, user-facing AI system that must manage dynamic user inputs, multi-step workflows, and structured decision-making while maintaining smooth, real-time interaction.

## 💪 Project Objective

* Understand user health goals
* Generate personalized meal and workout plans
* Track progress and schedule reminders
* Provide real-time interaction via streaming
* Delegate to specialized agents when needed

## ✅ SDK Features Overview

| Feature | Requirement |
| --- | --- |
| Agent + Tool Creation | ✅ Required |
| State Management | ✅ Required |
| Guardrails (Input/Output) | ✅ Required |
| Real-Time Streaming | ✅ Required |
| Handoff to Another Agent | ✅ Required |
| Lifecycle Hooks | ✅ Optional |

## 🔧 Tools

| Tool Name | Purpose |
| --- | --- |
| GoalAnalyzerTool | Converts user goals into structured format using input/output guardrails |
| MealPlannerTool | Async tool to suggest 7-day meal plan honoring dietary preferences |
| WorkoutRecommenderTool | Suggests workout plan based on parsed goals and experience |
| CheckinSchedulerTool | Schedules recurring weekly progress checks |
| ProgressTrackerTool | Accepts updates, tracks user progress, modifies session context |

## 🤝 Handoffs (Specialized Agents)

Specialized agents receive control through handoff() based on user input.

| Agent Name | Trigger Condition |
| --- | --- |
| EscalationAgent | User wants to speak to a human coach |
| NutritionExpertAgent | Complex dietary needs like diabetes or allergies |
| InjurySupportAgent | Physical limitations or injury-specific workouts |

Each agent should:

* Be declared and passed in the handoffs parameter of the main agent
* Optionally implement on\_handoff() for logging or initialization

## 📦 Context Management

Define a shared context class:

**class** UserSessionContext(BaseModel):  
 name: str  
 uid: int  
 goal: Optional[dict] = None  
 diet\_preferences: Optional[str] = None  
 workout\_plan: Optional[dict] = None  
 meal\_plan: Optional[List[str]] = None  
 injury\_notes: Optional[str] = None  
 handoff\_logs: List[str] = []  
 progress\_logs: List[Dict[str, str]] = []

Used by all tools, hooks, and agents as RunContextWrapper[UserSessionContext].

## 🔒 Guardrails

### Input Guardrails

* Validate goal input format: quantity, metric, duration (e.g. “lose 5kg in 2 months”)
* Ensure valid dietary or injury-related inputs
* Block unsupported or incomplete entries

### Output Guardrails

* Ensure tools return structured JSON or Pydantic models
* Useful for validating and parsing agent responses

## 🔄 Streaming

Use Runner.stream(...) to stream real-time responses.

**async** **for** step **in** Runner.stream(starting\_agent=agent, input="Help me lose weight", context=user\_context):  
 print(step.pretty\_output)

Stream full conversation flow including tool calls and tool responses.

## 🔁 Optional Lifecycle Hooks

Use RunHooks or AgentHooks to log or trigger behaviors:

**RunHooks** (global events):

* on\_agent\_start, on\_agent\_end
* on\_tool\_start, on\_tool\_end
* on\_handoff

**AgentHooks** (agent-specific):

* on\_start, on\_end
* on\_tool\_start, on\_tool\_end
* on\_handoff

Use cases:

* Logging tool invocations
* Tracking number of user interactions
* Debugging handoff behavior

## 🧭 User Journey (Example Flow)

User: I want to lose 5kg in 2 months  
-> GoalAnalyzerTool extracts structured goal  
  
User: I’m vegetarian  
-> MealPlannerTool provides meal plan (streamed)  
  
User: I have knee pain  
-> Handoff to InjurySupportAgent  
  
User: I’m also diabetic  
-> Handoff to NutritionExpertAgent  
  
User: I want to talk to a real trainer  
-> EscalationAgent handoff is triggered

## 📁 Suggested Folder Structure

health\_wellness\_agent/  
├── main.py  
├── agent.py  
├── context.py  
├── guardrails.py  
├── hooks.py  
├── tools/  
│ ├── goal\_analyzer.py  
│ ├── meal\_planner.py  
│ ├── workout\_recommender.py  
│ ├── scheduler.py  
│ ├── tracker.py  
├── agents/  
│ ├── escalation\_agent.py  
│ ├── nutrition\_expert\_agent.py  
│ └── injury\_support\_agent.py  
├── utils/  
│ └── streaming.py  
└── README.md

## 📥 Submission Requirements

* Functional agent with all tools
* Use of context, handoffs, and guardrails
* Real-time streaming with Runner.stream()
* Modularized code with proper structure
* CLI or frontend UI (e.g., Streamlit) (**Optional**)

## 📊 Evaluation Criteria (100 Points)

| Category | Points |
| --- | --- |
| Tool Design + Async Integration | 20 |
| Context & State Management | 10 |
| Input/Output Guardrails | 15 |
| Handoff Logic | 15 |
| Real-time Streaming | 15 |
| Code Structure & Logging | 10 |
| Multi-turn Interaction | 15 |
| (Optional) Lifecycle Hook Usage | +10 |

## 💡 Bonus Ideas

* Streamlit dashboard
* User progress PDF report
* Integration with a database or file storage

## 🚀 Getting Started

1. Install SDK: pip install openai-agents
2. Start building from main.py
3. Use [docs](https://openai.github.io/openai-agents-python) as reference

**End of Assignment Document**