# Health & Wellness Planner Agent - Project Documentation

## 1. Project Overview

This project is a fully functional, AI-powered Health & Wellness Planner Agent built using the OpenAI Agents SDK. It simulates a digital wellness assistant capable of understanding goals, creating plans, and engaging users in real-time.

## 2. Project Objectives

- Collect and understand health-related goals
- Generate customized dietary and workout plans
- Track and update user progress over time
- Enable real-time interaction using streaming responses
- Provide intelligent handoff to expert agents

## 3. Architecture & Core Features

Key components include:
- Agent + Tool Creation
- Context Management
- Input/Output Guardrails
- Real-Time Streaming
- Specialized Agent Handoff
- Optional Lifecycle Hooks

## 4. Tools Implemented

- GoalAnalyzerTool: Parses user goals
- MealPlannerTool: Provides meal plans
- WorkoutRecommenderTool: Suggests workouts
- CheckinSchedulerTool: Schedules check-ins
- ProgressTrackerTool: Tracks progress

## 5. Specialized Agents

- EscalationAgent: For human coaching
- NutritionExpertAgent: For complex dietary needs

- InjurySupportAgent: For injury-specific recommendations

## 6. 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/
```

## 7. 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]] = []
```

## 8. Guardrails

- Input: Validates goal format and health details
- Output: Ensures JSON or model-based responses

## 9. Real-time Streaming Example

```
async for step in Runner.stream(starting_agent=agent, input="Help me lose weight", context=user_context):
    print(step.pretty_output)
```

## 10. Lifecycle Hooks

Optional logging with:

- on_agent_start, on_tool_start
- on_handoff, on_tool_end

## 11. Example User Journey

User: I want to lose 5kg in 2 months  GoalAnalyzerTool

User: I'm vegetarian  MealPlannerTool

User: I have knee pain  InjurySupportAgent

User: I'm diabetic  NutritionExpertAgent

User: Talk to trainer  EscalationAgent

## 12. Chainlit Integration

Used for real-time UI interaction.

Config:

```
[project]
name = "health-wellness-planner"
```

Code:

```
@cl.on_message
async def main(message):
    async for step in Runner.stream(agent, message.content, context=user_context):
        await cl.Message(content=step.pretty_output).send()
```

# Health & Wellness Planner Agent - Project Documentation

## 13. Setup Instructions

```
pip install openai-agents chainlit
python main.py  # CLI
chainlit run chainlit_app.py  # Chainlit UI
```

## 14. Evaluation Criteria

- Tool Design: 20
- Context: 10
- Guardrails: 15
- Handoff: 15
- Streaming: 15
- Structure: 10
- Multi-turn: 15
- Bonus: 10

## 15. Bonus Features

- Streamlit Dashboard
- PDF Reports
- Database Integration

## Conclusion

The Health & Wellness Planner Agent showcases a modular, intelligent AI system with real-time interaction, contextual understanding, and agent collaboration, enhanced through Chainlit UI integration.