How to build Multi Agent Supervisor System with LangGraph, Qwen & Streamlit

CodeTops
1K subscribers

Subscribe

32 | Share | Clip | Save

1,228 views Apr 27, 2025 #LangChain #LangGraph #AI

This video is a tutorial on how to build a Multi-Agent Supervisor System with LangGraph, Qwen, and Streamlit.

In this step-by-step tutorial, we'll dive into building a powerful multi-agent system where a Supervisor Agent manages multiple agents, all using LangGraph, Qwen, and Streamlit.

🔍 What You'll Learn:
- Design a Supervisor-Agent architecture
- Set up multiple agents with different tasks
- Integrate Qwen as LLM
- Build an interactive frontend with Streamlit

https://github.com/Mercytopsy/AI-Health-Supervisor-Agent

Mercytopsy / AI-Health-Supervisor-Agent

Type / to search

Code | Issues | Pull requests | Actions | Projects | Security | Insights

AI-Health-Supervisor-Agent Public

Watch 2 | Fork 6 | Star 7

main | 1 Branch | 0 Tags | Go to file | Add file | Code

Mercytopsy Update tools.py 590b2b7 · 3 months ago 22 Commits

Architectural_Diagram.png	add architectural digram	3 months ago
Health_Assistant_AI_Agent.ipynb	update jupyter file	3 months ago
README.md	Update README.md	3 months ago
agents.py	add all files	3 months ago
main.py	add all files	3 months ago
requirements.txt	add all files	3 months ago
tools.py	Update tools.py	3 months ago
utils.py	add all files	3 months ago

About

A multi-agent AI system designed to provide personalized health guidance across fitness, diet, and mental wellness

Readme
Activity
7 stars
2 watching
6 forks
Report repository

Releases

No releases published

Packages

No packages published

Languages

Jupyter Notebook 80.3% Python 19.7%

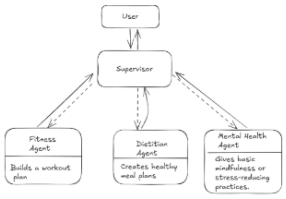
README

AI-Health-Supervisor-Agent

A multi-agent AI system designed to provide personalized health guidance across fitness, diet, and mental wellness, coordinated by a central **Supervisor Agent**.

This AI assistant helps users improve their well-being by generating:

- 🏋️ Customized workout plans
- 🥗 Healthy meal suggestions
- 🧘 Mindfulness and stress-reducing tips



🧰 Tech Stack

- LangGraph – for building the multi-agent state machine
- LangChain – for Tool orchestration
- Qwen2.5:14b – for agent reasoning and responses
- Streamlit – for frontend
- Python – main programming language

Installation & Setup

1. Clone the repository

```
git clone https://github.com/Mercytopsy/AI-Health-Supervisor-Agent.git
cd AI-Health-Supervisor-Agent
```

2. Install dependencies

```
pip install -r requirements.txt
```

3. Set up Qwen

```
ollama pull qwen2.5:14b
```

4. Add Fitness & Dietitian API Key

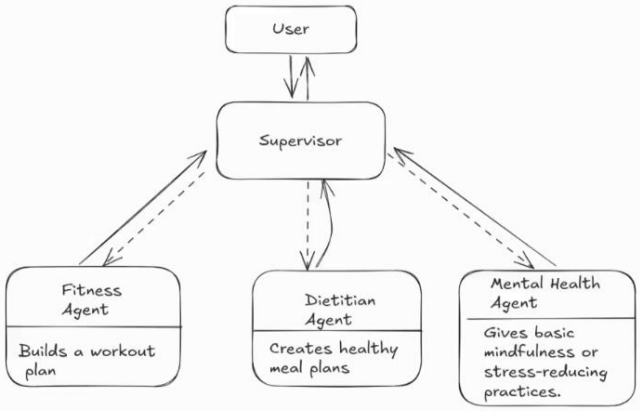
```
Create a .env file:
```

5. Run the App

```
streamlit run main.py
```

Multi-Agent Supervisor System

AI Health Assistant



LLM

ollama pull qwen2.5:14b

Qwen

Tools

Dietitian Agent- Spoonacular API

Fitness Agent - API Ninjas

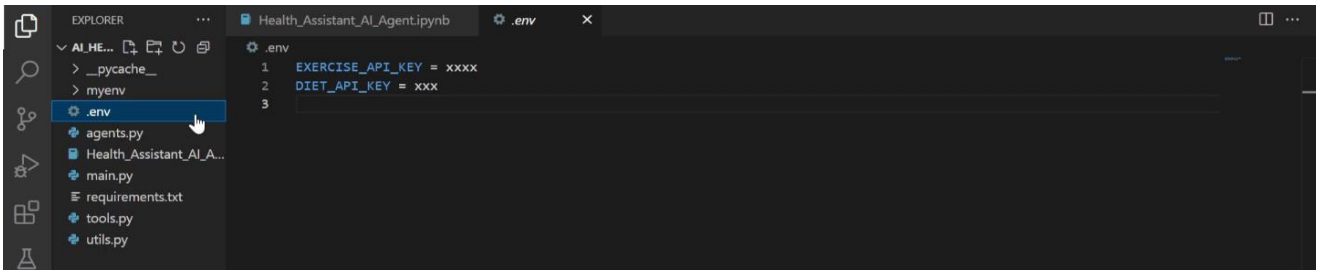
Requirements.txt

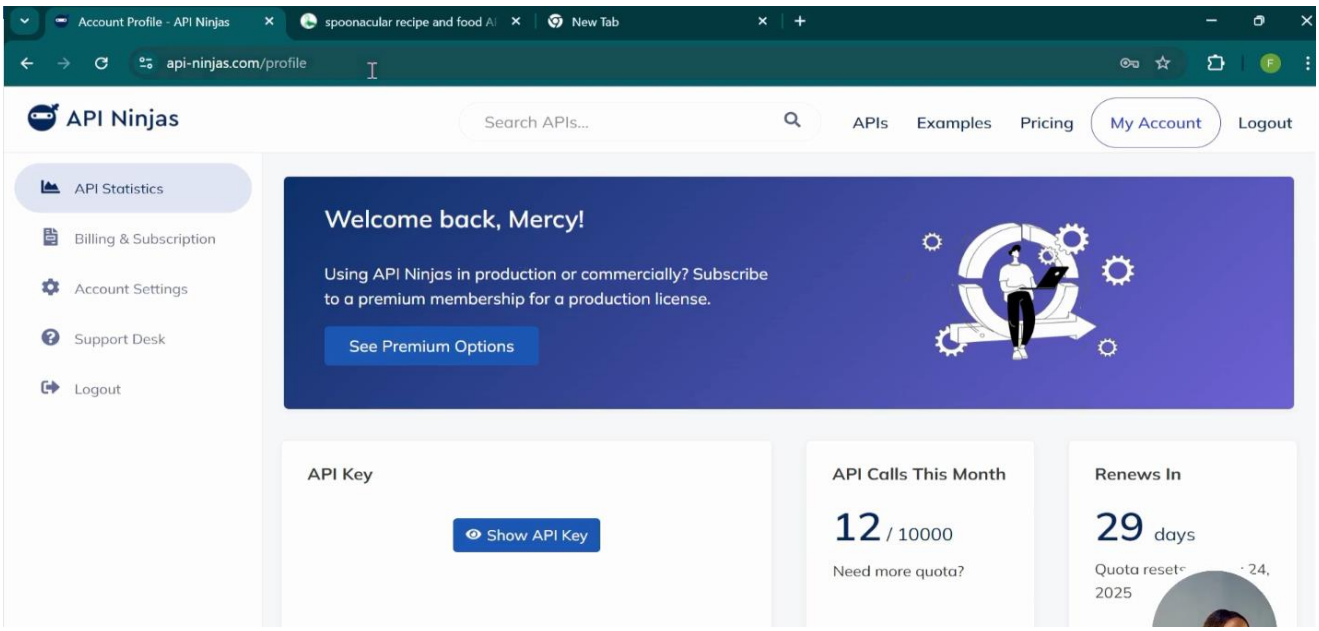
langgraph

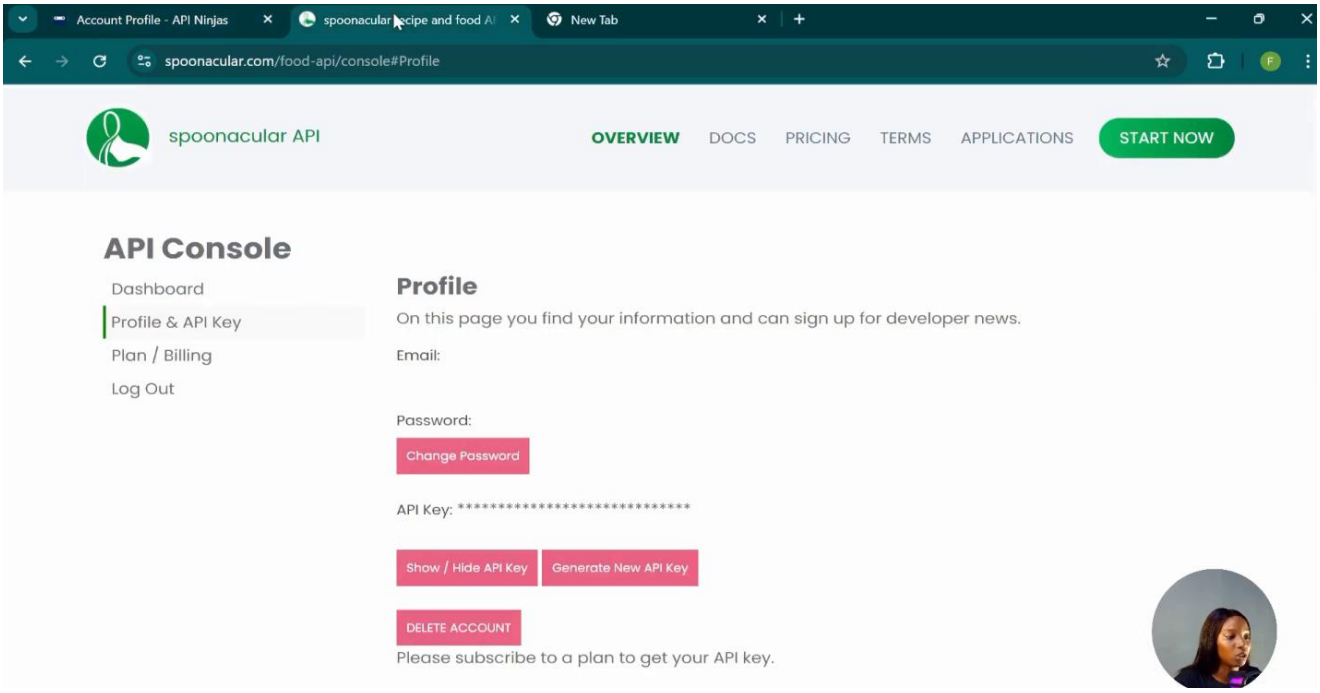
langchain

langchain_ollama

streamlit







EXPLORER

AI_HEALTH_ASSISTANT

- __pycache__
- myenv
- .env
- agents.py
- Health_Assistant_AI_A...
- main.py
- requirements.txt
- tools.py
- utils.py

Health_Assistant_AI_Agent.ipynb × .env

Health_Assistant_AI_Agent.ipynb > AI Health Assistant - Supervisor Agent > from langchain_core.messages import HumanMessage,AIMessage

Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ··· myenv (Python 3.11.11)

Empty markdown cell, double-click or press enter to edit.

```python
from langchain_core.messages import HumanMessage,AIMessage
from langgraph.prebuilt import create_react_agent
from langgraph.graph import StateGraph, MessagesState, START, END
from langgraph.checkpoint.memory import MemorySaver
from langchain.prompts import PromptTemplate
from IPython.display import display, Image
from typing import Annotated, Literal
from langchain_ollama import ChatOllama

from typing_extensions import TypedDict
from langchain.tools import tool
from langgraph.types import Command
import requests
import random
import uuid
import os
```

[1] Python

main.py
requirements.txt
tools.py
utils.py

```python
fitness_api_key = os.getenv("EXERCISE_API_KEY")
diet_api_key = os.getenv("DIET_API_KEY")
```

[2] ✓ 0.0s Python

Define State

```python
class State(MessagesState):
    next: str
```

[46] Python

Health_Assistant_AI_A...
main.py
requirements.txt
tools.py
utils.py

Define Tools

Create Fitness Tool

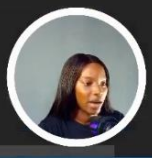
```python
class FitnessData:

    def __init__(self):
        self.base_url = "https://api.api-ninjas.com/v1/exercises"
        self.api_key = fitness_api_key


    def get_muscle_groups_and_types(self):

        muscle_targets = {
```

.env
agents.py
Health_Assistant_AI_A...
main.py
requirements.txt
tools.py
utils.py

```python
        muscle_targets = {
                'full_body': ["abdominals", "biceps", "calves", "chest", "forearms", "glutes",
                    "hamstrings", "lower_back", "middle_back", "quadriceps",
                    "traps", "triceps", "adductors"
                    ],
                'upper_body': ["biceps", "chest", "forearms", "lats", "lower_back", "middle_back", "neck", "traps", "triceps"
                'lower_body': ["adductors", "calves", "glutes", "hamstrings", "quadriceps"]
            }
        exercise_types = {'types':["powerlifting","strength", "stretching", "strongman"]}

        return muscle_targets, exercise_types


    def fetch_exercises(self, type, muscle, difficulty):
        headers = {
            'X-Api-Key':self.api_key
        }
        params= {
            'type': type,
            'muscle': muscle,
            'difficulty': difficulty
            }
        try:
            response = requests.get(self.base_url, headers=headers,params=params)
            result = response.json()
            if not result:
                print(f"No exercises found for {muscle}")
            return result
        except requests.RequestException as e:
            print(f"Request failed: {e}")
            return []
```

[50]

OUTLINE
TIMELINE



```python
def generate_workout_plan(self, query='full_body', difficulty='intermediate'):
    output=[]
    muscle_targets, exercise_types = self.get_muscle_groups_and_types()
    muscle = random.choice(muscle_targets.get(query))
    type = random.choice(exercise_types.get('types'))
    result = self.fetch_exercises('stretching', muscle, difficulty)
    print(result)
    limit_plan = result[:3]
    for i, data in enumerate(limit_plan):
        if data not in output:
            output.append(f"Exercise {i+1}: {data['name']}")
            output.append(f"Muscle: {data['muscle']}")
            output.append(f"Instructions: {data['instructions']}")

    return output
```

```python
fitness = FitnessData()
fitness.generate_workout_plan("full_body")
```

[6] ✓ 2.2s Python

```
[{'name': 'Tricep Side Stretch', 'type': 'stretching', 'muscle': 'triceps', 'equipment': 'body_only', 'difficulty': 'intermediate
```

```
['Exercise 1: Tricep Side Stretch',
 'Muscle: triceps',
 'Instructions: Bring right arm across your body and over your left shoulder, holding your elbow with your left hand, until you f
 'Exercise 2: Triceps Stretch',
 'Muscle: triceps',
 'Instructions: Reach your hand behind your head, grasp your elbow and gently pull. Hold for 10 to 20 seconds, then switch sides.
```

myenv
.env
agents.py
Health_Assistant_AI_A...
main.py
requirements.txt
tools.py
utils.py

Dietitian Tool

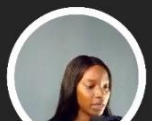
```python
class Dietitian:

    def __init__(self):
        self.base_url = "https://api.spoonacular.com"
        self.api_key = diet_api_key

    def fetch_meal(self, time_frame="day", diet="None"):

        url = f"{self.base_url}/mealplanner/generate"
        params = {
            "timeFrame":time_frame,
            "diet": diet,
            "apiKey":self.api_key
        }

        response = requests.get(url, params=params)
        if not response:
            print('Meal Plan not found')
        return response.json()

    def get_recipe_information(self, recipe_id):
```

```python
def get_recipe_information(self, recipe_id):

    url = f"{self.base_url}/recipes/{recipe_id}/information"
    params = {"apiKey": self.api_key}
    response = requests.get(url, params=params)
    if not response:
        print("Recipe not found")
    return response.json()


def generate_meal_plan(self, query):
```

agents.py
Health_Assistant_AI_A...
main.py
requirements.txt
tools.py
utils.py

```python
def generate_meal_plan(self, query):
    meals_processed = []
    meal_plan = self.fetch_meal(query)
    print(meal_plan)

    meals = meal_plan.get('meals')
    nutrients = meal_plan.get('nutrients')

    for i, meal in enumerate(meals):
        recipe_info = self.get_recipe_information(meal.get('id'))
        ingredients = [ingredient['original'] for ingredient in recipe_info.get('extendedIngredients')]

        meals_processed.append(f"🍽 Meal {i+1}: {meal.get('title')}")
        meals_processed.append(f"Prep Time: {meal.get('readyInMinutes')}")
        meals_processed.append(f"Servings: {meal.get('servings')}")


        meals_processed.append("📄 Ingredients:\n" + "\n".join(ingredients))
        meals_processed.append(f"📋 Instructions:\n {recipe_info.get('instructions')}")



    meals_processed.append(
```

```python
meals_processed.append(
"\n🍴 Daily Nutrients:\n"
f"Protein: {nutrients.get('protein', 'N/A')} kcal\n"
f"Fat: {nutrients.get('fat', 'N/A')} g\n"
f"Carbohydrates: {nutrients.get('carbohydrates', 'N/A')} g"
)


return meals_processed
```



myenv
.env
agents.py
Health_Assistant_AI_A...
main.py
requirements.txt
tools.py
utils.py

```python
@tool
def diet_tool(query: Annotated[str, "This input will either be None, vegetarian, and vegan"]):
    """use this tool to get diet plan for the user.
    The diet type provided serves as your input  \
                                                 """
    dietitian_tool = Dietitian()
    result = dietitian_tool.generate_meal_plan(query)

    return result
```

[34] Python

```python
diet = Dietitian()
diet.generate_meal_plan("vegetarian")
```

[10] 1.9s Python

```
{'meals': [{'id': 716276, 'image': 'doughnuts-716276.jpg', 'imageType': 'jpg', 'title': 'Doughnuts', 'readyInMinutes': 45, 'servi
```

myenv
.env
agents.py
Health_Assistant_AI_A...
main.py
requirements.txt
tools.py
utils.py

```python
diet = Dietitian()
diet.generate_meal_plan("vegetarian")
```

[10] ✓ 2.8s Python

```
{'meals': [{'id': 716276, 'image': 'doughnuts-716276.jpg', 'imageType': 'jpg', 'title': 'Doughnuts', 'readyInMinutes': 45, 'servi
```

```
['🍽 Meal 1: Doughnuts',
 'Prep Time: 45',
 'Servings: 2',
 '📝 Ingredients:\n1.5 cups of flour\n30 ml honey\n1 tablespoon of powdered milk\n1/2 teaspoon salt\n150 ml warm water\n1 teaspoo
 '📋 Instructions:\n In a bowl mix the water with the yeast and honey, whisk and allow to sit for 15 minutes or until the mixture
 '🍽 Meal 2: Linguine Alla Carbonara',
 'Prep Time: 30',
 'Servings: 4',
 '📝 Ingredients:\n4 strips thick cut bacon, diced\n1/4 teaspoon Dried Chili Flakes\n2 Fresh Eggs, room temperature\n2 Garlic Clo
 '📋 Instructions:\n <ol><li>In a large bowl, whisk together eggs and parmesan cheese. Once combined, mix in parsley. Set bowl as
 '🍽 Meal 3: Easy Tabouleh',
 'Prep Time: 45',
 'Servings: 1',
 '📝 Ingredients:\n1/2 cup bulgur\n2 smalls cucumbers\n1 bunch of flat leaf parsley\n1/2 lemon juice from a lemon\n3 tablespoons
 '📋 Instructions:\n <ol><li>Chop the vegetables and parsley finely. The restaurant version has the veggies diced fairly small ar
 '\n🍴 Daily Nutrients:\nProtein: 57.88 kcal\nFat: 71.28 g\nCarbohydrates: 258.02 g']
```

myenv
.env
agents.py
Health_Assistant_AI_A...
main.py
requirements.txt
tools.py
utils.py

Define LLM and System Prompt

```python
llm = ChatOllama(model="qwen2.5:14b")
memory = MemorySaver()
```

[13] ✓ 0.0s Python

Creating the Agent

Build Fitness Agent

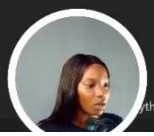
```python
fitness_agent_prompt = """
```

.env
agents.py
Health_Assistant_AI_A...
main.py
requirements.txt
tools.py
utils.py

Build Fitness Agent

```python
fitness_agent_prompt = """
You can only answer queries related to workout.
"""


fitness_agent = create_react_agent(
    llm,
    tools = [fitness_data_tool],
    prompt = fitness_agent_prompt)


def fitness_node(state: State) -> Command[Literal["supervisor"]]:
    result = fitness_agent.invoke(state)
    return Command(
        update={
            "messages": [
                AIMessage(content=result["messages"][-1].content, name="fitness")
            ]
        },
        goto="supervisor",
    )
```

[36]



.env
agents.py
Health_Assistant_AI_A...
main.py
requirements.txt
tools.py
utils.py

Build Dietitian Agent

```python
dietitian_system_prompt = """
You can only answer queries related to diet and meal plans.  .
"""
dietitian_agent = create_react_agent(
    llm,
    tools = [diet_tool],
    prompt = dietitian_system_prompt)


def dietitian_node(state: State) -> Command[Literal["supervisor"]]:
    result = dietitian_agent.invoke(state)
    return Command(
        update={
            "messages": [
                AIMessage(content=result["messages"][-1].content, name="dietitian")
            ]
        },
        goto="supervisor",
    )
```

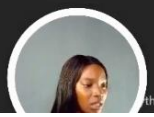
[37]

requirements.txt
tools.py
utils.py

Mental Health Agent

```python
def mental_health_node(state: State)-> Command[Literal["supervisor"]]:
    prompt = PromptTemplate.from_template(
        """You are a supportive mental wellness coach.
        Your task is to:
        - Give a unique mental wellness tip or stress-reducing practice.
        - Make it simple, kind, and useful. Avoid repeating tips."""
    )

    chain = prompt | llm
    response = chain.invoke(state)
    return Command(
        update={
            "messages": [
                AIMessage(content=f"Here's your wellness tip: {response.content}", name="wellness")
            ]
        },
        goto="supervisor",
    )
```

[38]



Supervisor Agent

```python
members = ["fitness", "dietitian", "wellness"]
options = members + ["FINISH"]


system_prompt = (
    "You are a supervisor tasked with managing a conversation between the"
    f" following workers: {members}. Given the following user request,"
    " respond with the worker to act next. Each worker will perform a"
    " task and respond with their results and status. When finished,"
    " respond with FINISH."


    "Guidelines:\n"
    "1. Always check the last message in the conversation to determine if the task has been completed.\n"
    "2. If you already have the final answer or outcome, return 'FINISH'.\n"

)

class Router(TypedDict):
    """Worker to route to next. If no workers needed, route to FINISH."""

    next: Literal[*options]
```

```python
{next["Dietitian"]}
```

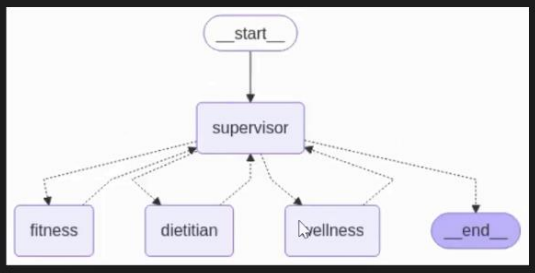
The expected response from the Supervisor agent is like above

```python
def supervisor_node(state: State)-> Command[Literal[*members, "__end__"]]:
    messages = [
        {"role": "system", "content": system_prompt},
    ] + state["messages"]
    response = llm.with_structured_output(Router).invoke(messages)
    goto = response["next"]
    if goto == "FINISH":
        goto = END

    return Command(goto=goto, update={"next": goto})
```

Build Supervisor Multi-Agent Graph

```python
builder = StateGraph(State)
builder.add_edge(START, "supervisor")
builder.add_node("supervisor", supervisor_node)
builder.add_node("fitness", fitness_node)
builder.add_node("dietitian", dietitian_node)
builder.add_node("wellness", mental_health_node)
graph = builder.compile(checkpointer=memory)
```

```python
graph
```



utils.py

Calling the Agents

```python
def parse_langgraph_output(stream):
    results = []
    for key, value in stream.items():
        if key == "supervisor":
            continue
        messages = value.get("messages", [])
        for msg in messages:
            if isinstance(msg, str):
                results.append((key, msg))
            elif isinstance(msg, AIMessage):
                results.append((key, msg.content))
    return results
```

[21] Python

.gitignore U
agents.py
Architectural_Diagra...
Health_Assistant... M
main.py
README.md
requirements.txt
tools.py
utils.py

```python
# "Can you give me a daily routine that includes meals, workouts, and stress relief?"
```

[ ] Python

```python
# Get the final step in the stream
final_event = None
config = {"configurable": {"thread_id": "1", "recursion_limit": 10}}
inputs = {
            "messages": [
                HumanMessage(
                    content="Give me wellness tips for the month?"
                )
            ],
        }


for step in graph.stream(inputs, config=config):
    final_event = step  # Keep updating to the latest step
    print(final_event)

    response_message=parse_langgraph_output(final_event)
    for agent, content in response_message:
        print(f"**Agent :** `{agent}`\n\n{content}")
        print("="*50)
```

[ ] Python

OUTLINE

```python
# Can you build a weekly workout routine for beginners?
```

```
{'supervisor': {'next': 'wellness'}}
{'wellness': {'messages': [AIMessage(content='Here\'s your wellness tip: **Tip:** Practice "Mindful Moments" throughout the day.\
**Agent :** `wellness`

Here's your wellness tip: **Tip:** Practice "Mindful Moments" throughout the day.

Instead of trying to find large chunks of time for mindfulness, integrate short moments into your daily routine. Take a minute wh
==================================================
{'supervisor': {'next': 'dietitian'}}
```

Let us see how this will look using **Streamlit** as below

.env
agents.py
Health_Assistant_AI_A...
main.py
requirements.txt
tools.py
utils.py

AI Health Assistant - Supervisor Agent

Vision

A personalized digital wellness coach that feels like a team of specialized experts working together to improve your fitness, nutrition, and mental health.

Generate + Code + Markdown

- **Goal**: Non-medical health Q&A assistant with agents for nutrition, fitness, and mental health.
- **Agents**:
  - `Fitness Agent`: Builds a workout plan.
  - `Dietician Agent`: Creates healthy meal plans.
  - `Mental Health Agent`: Gives basic mindfulness or stress-reducing practices.

Empty markdown cell, double-click or press enter to edit.

```python
from langchain_core.messages import HumanMessage,AIMessage
from langgraph.prebuilt import create_react_agent
from langgraph.graph import StateGraph, MessagesState, START, END
```

EXPLORER

AI_HE...
- __pycache__
- myenv
- .env
- agents.py
- Health_Assistant_AI_A...
- main.py
- requirements.txt
- tools.py
- utils.py

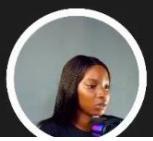
Health_Assistant_AI_Agent.ipynb | agents.py

agents.py > ...

```python
from langchain_core.messages import HumanMessage,AIMessage
from langgraph.prebuilt import create_react_agent
from langgraph.graph import StateGraph, MessagesState,START, END
from langgraph.checkpoint.memory import MemorySaver
from langchain.prompts import PromptTemplate
from typing import Annotated, Literal
from langchain_ollama import ChatOllama

from typing_extensions import TypedDict
from langgraph.types import Command
from langchain_openai import ChatOpenAI

from tools import (fitness_data_tool, diet_tool)
import streamlit as st


# Configuration
llm = ChatOllama(model="qwen2.5:14b")
```


.env
agents.py
Health_Assistant_AI_A...
main.py
requirements.txt
tools.py
utils.py

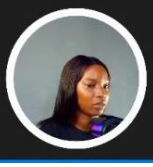

```python
# Configuration
llm = ChatOllama(model="qwen2.5:14b")


memory = MemorySaver()
members = ["fitness", "dietitian", "wellness"]
options = members + ["FINISH"]

fitness_agent_prompt = """You can only answer queries related to workout. """
dietitian_system_prompt = """You can only answer queries related to diet and meal plans."""
system_prompt = (
    "You are a supervisor tasked with managing a conversation between the"
    f" following workers: {members}. Given the following user request,"
    " respond with the worker to act next. Each worker will perform a"
    " task and respond with their results and status. When finished,"
    " respond with FINISH."


    "Guidelines:\n"
    "1. Always check the last message in the conversation to determine if the task has been completed.\n"
    "2. If you already have the final answer or outcome, return 'FINISH'.\n"

)

fitness_agent = create_react_agent(llm, tools = [fitness_data_tool], prompt = fitness_agent_prompt)


dietitian_agent = create_react_agent(llm, tools = [diet_tool], prompt = dietitian_system_prompt)
```



OUTLINE

__pycache__
myenv
.env
agents.py
Health_Assistant_AI_A...
main.py
requirements.txt
tools.py
utils.py

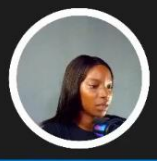

```python
# class State(TypedDict):
#     messages: Annotated[list, add_messages]
#     next: str | None

class State(MessagesState):
    next: str


class Router(TypedDict):
    """Worker to route to next. If no workers needed, route to FINISH."""

    # next: Literal[*options]
    next: Literal[*options]



def fitness_node(state: State) -> Command[Literal["supervisor"]]:
    result = fitness_agent.invoke(state)
    return Command(
        update={
            "messages": [
                AIMessage(content=result["messages"][-1].content, name="fitness")
            ]
        },
        goto="supervisor",
    )



def dietitian_node(state: State) -> Command[Literal["supervisor"]]:
    result = dietitian_agent.invoke(state)
```



OUTLINE

TIMELINE

```python
def dietitian_node(state: State) -> Command[Literal["supervisor"]]:
    result = dietitian_agent.invoke(state)
    return Command(
        update={
            "messages": [
                AIMessage(content=result["messages"][-1].content, name="dietitian")
            ]
        },
        goto="supervisor",
    )



def mental_health_node(state: State):
    prompt = PromptTemplate.from_template(
        """You are a supportive mental wellness coach.
        Your task is to:
        - Give a unique mental wellness tip or stress-reducing practice.
        - Make it simple, kind, and useful. Avoid repeating tips."""
    )

    chain = prompt | llm
    response = chain.invoke(state)
    return Command(
        update={
            "messages": [
                AIMessage(content=f"Here's your wellness tip: {response.content}", name="wellness")
            ]
        },
        goto="supervisor",
    )
```



```python



def supervisor_node(state: State) -> Command[Literal[*members, "__end__"]]:
    messages = [
        {"role": "system", "content": system_prompt},
    ] + state["messages"]
    response = llm.with_structured_output(Router).invoke(messages)
    goto = response["next"]
    if goto == "FINISH":
        goto = END

    return Command(goto=goto, update={"next": goto})
```

This completes the agent.py file that contains all the setup

```python
from langchain_core.messages import HumanMessage,AIMessage

def parse_langgraph_output(stream):
    results = []
    for key, value in stream.items():
        if key == "supervisor":
            continue
        messages = value.get("messages", [])
        for msg in messages:
            if isinstance(msg, str):
                results.append((key, msg))
            elif isinstance(msg, AIMessage):
                results.append((key, msg.content))
    return results
```

```python
from dotenv import load_dotenv
import requests
import random
import os

load_dotenv()

fitness_api_key = os.getenv("EXERCISE_API_KEY")
diet_api_key = os.getenv("DIET_API_KEY")


class FitnessData:

    def __init__(self):
        self.base_url = "https://api.api-ninjas.com/v1/exercises"
        self.api_key = fitness_api_key


    def get_muscle_groups_and_types(self):
```

```
.env
agents.py
Health_Assistant_AI_A...
main.py
requirements.txt
tools.py
utils.py
```

```python
    def get_muscle_groups_and_types(self):

        muscle_targets = {
                'full_body': ["abdominals", "biceps", "calves", "chest", "forearms", "glutes",
                    "hamstrings", "lower_back", "middle_back", "quadriceps",
                    "traps", "triceps", "adductors"
                    ],
                'upper_body': ["biceps", "chest", "forearms", "lats", "lower_back", "middle_back", "neck", "traps", "
                'lower_body': ["adductors", "calves", "glutes", "hamstrings", "quadriceps"]
            }
        exercise_types = {'types':["powerlifting","strength", "stretching", "strongman"]}

        return muscle_targets, exercise_types


    def fetch_exercises(self, type, muscle, difficulty):
        headers = {
            'X-Api-Key':self.api_key
        }
        params= {
```

```
.env
agents.py
Health_Assistant_AI_A...
main.py
requirements.txt
tools.py
utils.py
```

```python
        params= {
            'type': type,
            'muscle': muscle,
            'difficulty': difficulty
            }
        try:
            response = requests.get(self.base_url, headers=headers,params=params)
            result = response.json()
            if not result:
                print(f"No exercises found for {muscle}")
            return result
        except requests.RequestException as e:
            print(f"Request failed: {e}")
            return []



    def generate_workout_plan(self, query='full_body', difficulty='intermediate'):
```

```
.env
agents.py
Health_Assistant_AI_A...
main.py
requirements.txt
tools.py
utils.py
```

```python
    def generate_workout_plan(self, query='full_body', difficulty='intermediate'):
        output=[]
        muscle_targets, exercise_types = self.get_muscle_groups_and_types()
        muscle = random.choice(muscle_targets.get(query))
        type = random.choice(exercise_types.get('types'))
        result = self.fetch_exercises('stretching', muscle, difficulty)
        print(result)
        limit_plan = result[:3]
        for i, data in enumerate(limit_plan):
            if data not in output:
                output.append(f"Exercise {i+1}: {data['name']}")
                output.append(f"Muscle: {data['muscle']}")
                output.append(f"Instructions: {data['instructions']}")

        return output



class Dietitian:

    def __init__(self):
        self.base_url = "https://api.spoonacular.com"
        self.api_key = diet_api_key

    def fetch_meal(self, time_frame="day", diet="None"):
```

```
Health_Assistant_AI_A...
main.py
requirements.txt
tools.py
utils.py
```

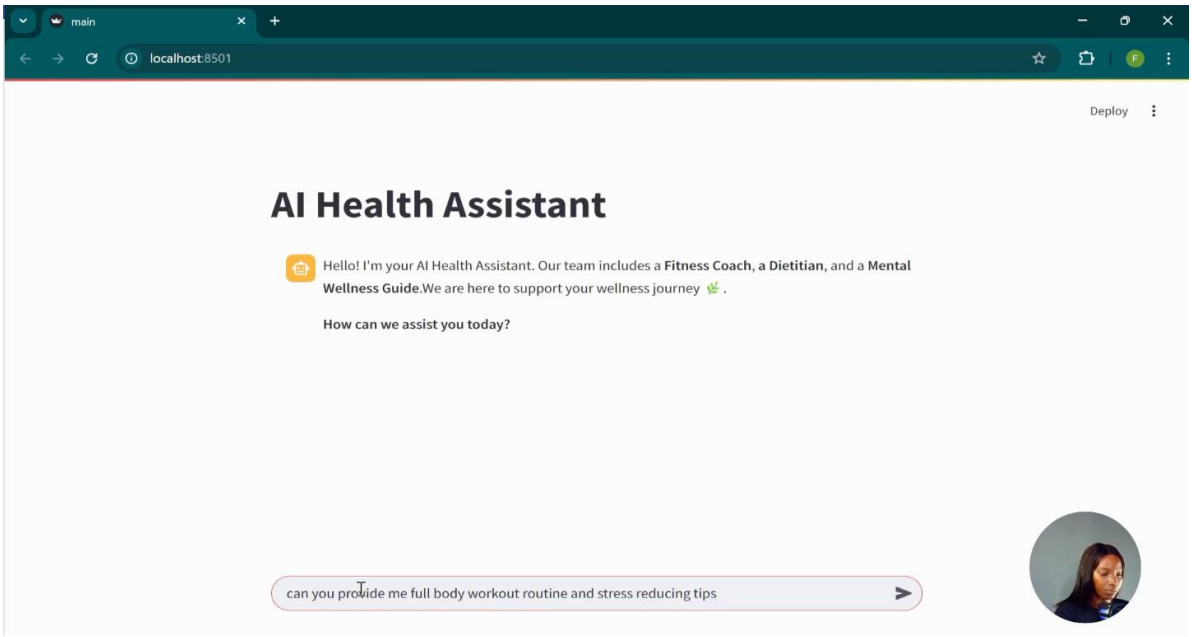
```python
    def fetch_meal(self, time_frame="day", diet="None"):

        url = f"{self.base_url}/mealplanner/generate"
        params = {
            "timeFrame":time_frame,
            "diet": diet,
            "apiKey":self.api_key
        }

        response = requests.get(url, params=params)
        if not response:
            print('Meal Plan not found')
        return response.json()

    def get_recipe_information(self, recipe_id):
```

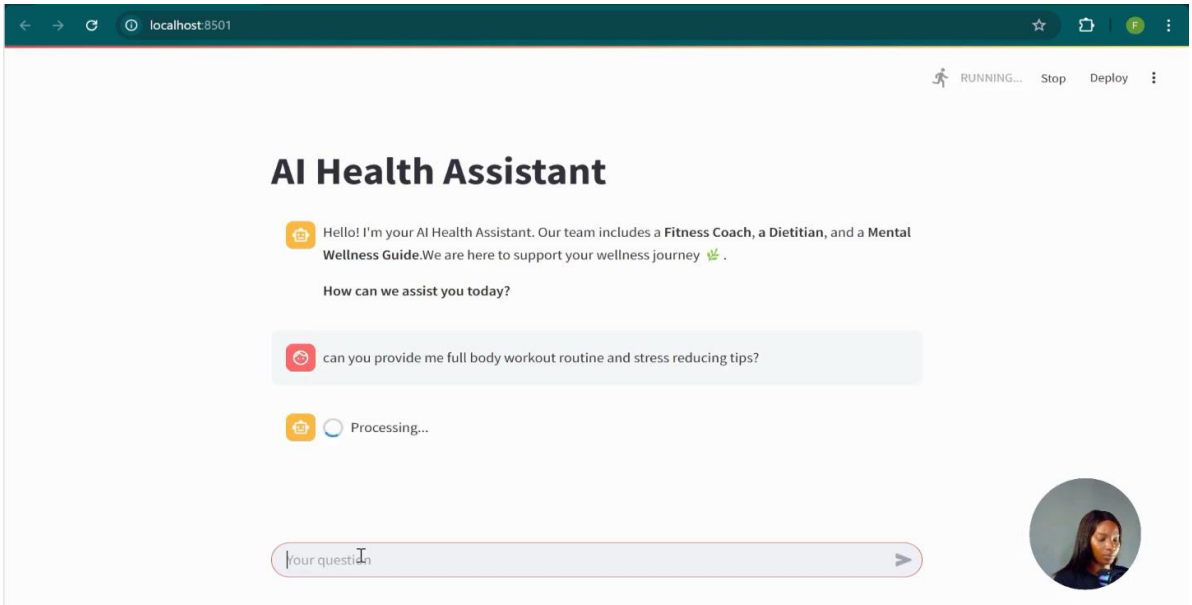
```python
def get_recipe_information(self, recipe_id):

    url = f"{self.base_url}/recipes/{recipe_id}/information"
    params = {"apiKey": self.api_key}
    response = requests.get(url, params=params)
    if not response:
        print("Recipe not found")
    return response.json()


def generate_meal_plan(self, query):
    meals_processed = []
    meal_plan = self.fetch_meal(query)
    print(meal_plan)

    meals = meal_plan.get('meals')
    nutrients = meal_plan.get('nutrients')

    for i, meal in enumerate(meals):
        recipe_info = self.get_recipe_information(meal.get('id'))
        ingredients = [ingredient['original'] for ingredient in recipe_info.get('extendedIngredients')]

        meals_processed.append(f"🍽️ Meal {i+1}: {meal.get('title')}")
        meals_processed.append(f"Prep Time: {meal.get('readyInMinutes')}")
        meals_processed.append(f"Servings: {meal.get('servings')}")


        meals_processed.append("📝 Ingredients:\n" + "\n".join(ingredients))
        meals_processed.append(f"📋 Instructions:\n {recipe_info.get('instructions')}")
```

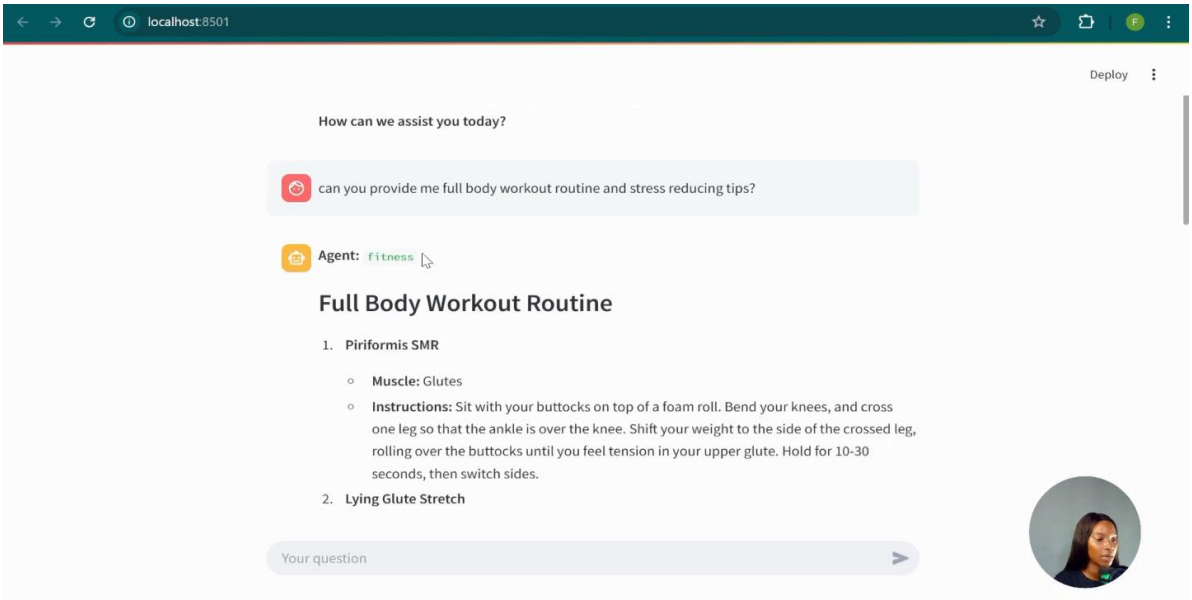
```python
            )
        ],
    }
    config = {"configurable": {"thread_id": "2", "recursion_limit": 15}}
    # Get the final step in the stream
    final_event = None
    for step in graph.stream(inputs, config=config):
        final_event = step  # Keep updating to the latest step
        print(final_event)

        response_message=parse_langgraph_output(final_event)

        for agent, content in response_message:

            assistant_reply = f"**Agent:** `{agent}`\n\n{content}"
            st.session_state.messages.append({"role": "assistant", "content": assistant_reply})
            st.markdown(assistant_reply)
            st.markdown("---")
```

```
Microsoft Windows [Version 10.0.26100.3775]
(c) Microsoft Corporation. All rights reserved.

(C:\Users\ODUNAYO\Documents\Machine Learning Projects\LLM Project\AI_Health_Assistant\myenv) C:\Users\ODUNAYO\Documents\Machine Learning Projects\LLM Project\AI_Health_Assistant>conda activate myenv

(myenv) C:\Users\ODUNAYO\Documents\Machine Learning Projects\LLM Project\AI_Health_Assistant>streamlit run main.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.0.239:8501
```

We can then run the main.py file to start the Streamlit FE


main
localhost:8501
Deploy
AI Health Assistant
Hello! I'm your AI Health Assistant. Our team includes a Fitness Coach, a Dietitian, and a Mental Wellness Guide.We are here to support your wellness journey .
How can we assist you today?
can you provide me full body workout routine and stress reducing tips


localhost:8501
RUNNING...
Stop
Deploy
AI Health Assistant
Hello! I'm your AI Health Assistant. Our team includes a Fitness Coach, a Dietitian, and a Mental Wellness Guide.We are here to support your wellness journey .
How can we assist you today?
can you provide me full body workout routine and stress reducing tips?
Processing...
Your question


localhost:8501
Deploy
How can we assist you today?
can you provide me full body workout routine and stress reducing tips?
Agent: fitness
Full Body Workout Routine
1. Piriformis SMR
Muscle: Glutes
Instructions: Sit with your buttocks on top of a foam roll. Bend your knees, and cross one leg so that the ankle is over the knee. Shift your weight to the side of the crossed leg, rolling over the buttocks until you feel tension in your upper glute. Hold for 10-30 seconds, then switch sides.
2. Lying Glute Stretch
Your question

localhost:8501

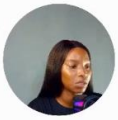
Deploy

4. **Leg-Up Hamstring Stretch**
   - **Muscle:** Hamstrings
   - **Instructions:** Lie flat on your back, bend one knee, and put that foot flat on the floor. Extend the other leg in the air. Pull the leg toward your nose. Switch sides.
5. **Standing Toe Touches**
   - **Muscle:** Hamstrings
   - **Instructions:** Stand straight, bend at the waist with legs straight, and let your upper body hang down. Hold for 10-20 seconds.
6. **Lying Groin Stretch with Band**
   - **Muscle:** Hamstrings
   - **Instructions:** Lie on your back with legs extended. Loop a band around one foot and swing that leg to the side. Hold for 10-20 seconds, then switch sides.

Stress Reducing Tips

- **Deep Breathing:** Practice deep breathing exercises to calm your mind and reduce stress.
- **Mindfulness Meditation:** Spend a few minutes each day focusing on the present moment to

Your question



Deploy

- **Adequate Sleep:** Ensure you get enough sleep each night to help your body recover and manage stress.
- **Healthy Diet:** Maintain a balanced diet rich in fruits, vegetables, and whole grains to support overall well-being.
- **Social Connections:** Spend time with friends and family to foster supportive relationships.

Feel free to ask if you need more information or specific exercises!

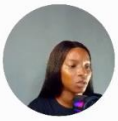
Agent: `wellness`

Here's your wellness tip: **Tip: The 5-4-3-2-1 Grounding Exercise**

When you feel overwhelmed or anxious, try this simple grounding technique to bring your focus back to the present moment:

1. **5 things you can see**: Look around you and notice five things you hadn't paid attention to before. It could be a picture on the wall, a plant, or the way light reflects off a surface.
2. **4 things you can touch**: Focus on four things you can physically feel. This could be the

Your question



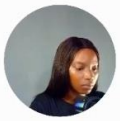
localhost:8501

Deploy

1. **5 things you can see**: Look around you and notice five things you hadn't paid attention to before. It could be a picture on the wall, a plant, or the way light reflects off a surface.
2. **4 things you can touch**: Focus on four things you can physically feel. This could be the texture of your clothing, the surface of a table, or the feeling of your feet on the ground.
3. **3 things you can hear**: Listen carefully and identify three sounds. It might be the hum of a computer, birds chirping outside, or distant conversations.
4. **2 things you can smell**: Take a moment to notice two scents. If you can't smell anything, think of your two favorite scents instead.
5. **1 thing you can taste**: Focus on one thing you can taste. It could be the lingering flavor of a meal, a sip of water, or even just the taste of your mouth.

This exercise helps anchor you in the present and can reduce feelings of anxiety or stress. Remember, it's okay to take a moment for yourself. You deserve it!

I'm a vegetarian, can you give me a customized plan



RUNNING... Stop Deploy

think of your two favorite scents instead.

5. **1 thing you can taste**: Focus on one thing you can taste. It could be the lingering flavor of a meal, a sip of water, or even just the taste of your mouth.

This exercise helps anchor you in the present and can reduce feelings of anxiety or stress. Remember, it's okay to take a moment for yourself. You deserve it!

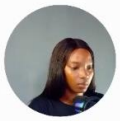
Agent: `dietitian`

I'm here to help with diet and meal plans. If you have any questions or need assistance related to that, feel free to ask!

I'm a vegetarian, can you give me a customized plan

Processing...

Your question



Deploy

Agent: `dietitian`

Here's a customized vegetarian meal plan for you:

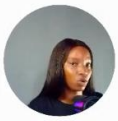
Meal 1: Gluten Free Quinoa and Corn Flour Crepes

- **Prep Time:** 60 minutes
- **Servings:** 4

Ingredients:

- Virgin coconut oil for cooking
- 1/2 cup corn flour (different from cornmeal)
- 1 tablespoon ground flax seeds (plus 3 tablespoons boiling water, whisked)
- 1 tablespoon pure maple syrup
- 1/2 cup quinoa flour
- 2 1/2 cups hemp, soy, almond, or rice milk
- 1/4 teaspoon sea salt

Your question



localhost:8501

Deploy

- 1/4 teaspoon sea salt
- 1 cup tapioca flour
- 1/2 teaspoon xanthan gum

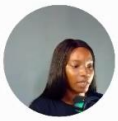
Instructions:

1. Whisk the dry ingredients together in a bowl. In a separate bowl, whisk 2 1/2 cups hemp milk, flax seed mixture, maple syrup, and melted coconut oil together. Add the wet ingredients to the dry ingredients and mix gently. If the batter is too thick, add more hemp milk to make it pourable. Let it rest in the refrigerator for at least 30 minutes.
2. Heat a nonstick skillet over medium heat and add a small dab of coconut oil. Pour about 1/2 cup batter into the skillet, rotating to cover the bottom. Cook until the top is dry, then flip and cook the other side for 30-60 seconds. Repeat until all batter is used.

Meal 2: Pasta With Feta Cheese And Asparagus

- **Prep Time:** 20 minutes

Your question



localhost:8501

Deploy

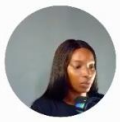
Meal 2: Pasta With Feta Cheese And Asparagus

- **Prep Time:** 20 minutes
- **Servings:** 2

Ingredients:

- 8 oz linguine (or spaghetti)
- 1/2 lb asparagus, cleaned and cut into 2" pieces
- 2 Tbsp extra-virgin olive oil
- 1/4 tsp kosher salt
- 1/4 tsp freshly cracked black pepper
- 1 Tbsp chopped garlic
- 1 tsp dried thyme
- 1 tsp dried oregano
- 1/2 tsp lemon zest
- 1/4 tsp red pepper flakes
- 6 oz feta cheese, crumbled

Your question



localhost:8501

Deploy

- 1/4 c reserved pasta water
- 2 Tbsp chopped flat-leaf Italian parsley (optional)

Instructions:

1. Boil water and add salt. Cook the pasta until al dente.
2. In a pan, heat olive oil and add asparagus, salt, and pepper. Cook for 5-10 minutes.
3. Add garlic, thyme, oregano, lemon zest, and red pepper flakes a minute before asparagus is done.
4. Add feta cheese and cooked pasta, along with reserved pasta water. Stir to combine and serve with parsley.

Meal 3: Quinoa and Chickpea Salad with Sun-Dried Tomatoes and Dried Cherries

- **Prep Time:** 45 minutes
- **Servings:** 6

Your question

localhost:8501

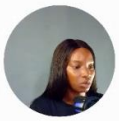
Deploy

- **Servings:** 6

Ingredients:

- 1/3 cup raw cashews, chopped
- 1/3 cup dried cherries
- 1/2 cup dried chickpeas (or 1 14 oz can)
- 1/2 teaspoon dried thyme
- 1/2 teaspoon ground cumin
- 1 1/2 tablespoons honey
- Juice of 1 medium orange
- 2 tablespoons olive oil
- 1 cup dried quinoa (2 cups cooked)
- 2 teaspoons red wine vinegar
- Sea salt and black pepper to taste
- 1/2 cup sun-dried tomatoes
- 1/2 teaspoon turmeric

Your question



localhost:8501

Deploy

Instructions:

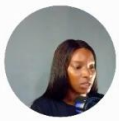
1. Rinse and soak quinoa and chickpeas overnight. Cook chickpeas until soft.
2. Soak sun-dried tomatoes in hot water for 30 minutes, then chop.
3. Cook quinoa until water is absorbed. Toast cashews in a skillet.
4. Combine chickpeas, quinoa, sun-dried tomatoes, cashews, and cherries in a bowl. Whisk dressing ingredients and pour over salad. Serve at room temperature or chilled.

Daily Nutrients:

- **Protein:** 55.07 g
- **Fat:** 66.1 g
- **Carbohydrates:** 245.83 g

Enjoy your meals!

Your question



localhost:8501

Deploy

- **Fat:** 66.1 g
- **Carbohydrates:** 245.83 g

Enjoy your meals!

Agent: `fitness`

I can provide you with a customized workout plan. Would you like a full-body, upper-body, or lower-body exercise plan?

Agent: `wellness`

Here's your wellness tip: **Tip: The 5-4-3-2-1 Grounding Exercise**

When you feel overwhelmed, take a moment to ground yourself using your senses. This simple exercise can help bring you back to the present and reduce stress.

Your question

localhost:8501

Deploy

When you feel overwhelmed, take a moment to ground yourself using your senses. This simple exercise can help bring you back to the present and reduce stress.

1. **5 things you can see**: Look around you and notice five things. It could be a picture on the wall, a plant, or the color of the floor.
2. **4 things you can touch**: Focus on four things you can physically feel. This could be the texture of your clothing, the surface of a table, or the feeling of your feet on the ground.
3. **3 things you can hear**: Listen carefully and identify three sounds. It might be the hum of a computer, birds outside, or distant chatter.
4. **2 things you can smell**: Take a deep breath and notice two scents. If you can't smell anything, think of your two favorite scents.
5. **1 thing you can taste**: Focus on one thing you can taste. It could be the aftertaste of a meal, a sip of water, or even just the air in your mouth.

This exercise helps to anchor you in the moment and can be done anywhere, anytime. Remember to be gentle with yourself as you practice!

Your question