Multi-Agent Fitness App Documentary (Personalized Fitness App)

Introduction

A system composed of multiple autonomous agents. Each agent can interact with others and the environment to achieve specific goals. Importance in Today's World Enhanced problem-solving capabilities. Scalability and efficiency in various applications, including healthcare and fitness.

Research:

Application in Fitness Agents can track user activities, dietary preferences, and health metrics.

Market Standards in Fitness Applications:

Current Trends in Fitness Apps Personalization:

Tailoring recommendations based on user data. Social Integration: Connecting users for motivation and support.Adopting Multi-Agent Systems Ensures adaptability and responsiveness to user needs. Examples include virtual personal trainers and dietary assistants.

Use Case Generation for Fitness Apps:

Personalized Workout Plans: Agents analyze user performance data to create customized workout schedules. Nutrition Guidance: Agents assess dietary habits and suggest meal plans based on personal health goals. Community Support: Agents facilitate connections between users for sharing progress and tips.
Resource Asset Collection in Fitness Apps:
From the GitHub: https://github.com/Sven-Bo/streamlit-multipage-app-example

Overview

Personalized fitness app is designed to help users manage their fitness and nutritional needs. It includes features for tracking workouts, meal planning, and social interaction among users, aiming to provide a comprehensive tool for fitness enthusiasts.

Key Features

1. **User Interface**:
   Streamlit Framework: The app uses Streamlit, a powerful framework for building interactive web applications in Python, allowing for a smooth and user-friendly experience. Responsive Design: The layout is designed to be intuitive, making it easy for users to navigate between different functionalities.
2. **Fitness Tracking:**
   Workout Log: Users can log their workouts, including exercises, durations, and intensity levels. Progress Monitoring: The app may include graphs or statistics to help users visualize their progress over time, such as weight changes, workout frequency, or strength improvements.
3. **Nutrition & Meal Planning**:
   Meal Suggestions: The app provides meal suggestions based on user preferences, dietary restrictions, or fitness goals (e.g., weight loss, muscle gain).
   Nutritional Information: Users can view nutritional information for meals to make informed dietary choices. Custom Meal Plans: Users can create and save custom meal plans tailored to their dietary needs.

4. **Social Features:**
   Community Interaction: The app may include features for users to connect, share their progress, and motivate each other through social feeds or forums.
   Challenges and Competitions: Users can participate in challenges to encourage engagement and commitment to fitness goals.
5. **User Profile Management:**
   Personal Profiles: Users can create profiles that store their fitness goals, preferences, and dietary restrictions. Customizable Settings: Users can customize notifications, reminders for workouts or meal prep, and track their goals.

Technical Details

1. Dependencies:
   The app may utilize libraries like Pandas for data manipulation, NumPy for numerical operations, Matplotlib or Plotly for data visualization, and Streamlit for building the app interface. Ensure that your `requirements.txt` file is up to date with all the libraries used in your application.
2. Data Management:
   User data might be stored in a database (e.g., SQLite, PostgreSQL) or managed using files (e.g., CSV) for persistence. The app should include functionalities for data input and output to ensure users can easily access and update their information.
3. Security Measures:
   User data privacy and security are essential. Implement measures to protect user information, such as encrypting sensitive data and using secure authentication methods.
4. Deployment:
   The app is hosted on Streamlit sharing platform, making it accessible for users without needing local installation. Regular updates and maintenance should be planned to fix bugs and improve functionalities based on user feedback.

Main.py

```
import streamlit as st


# Page setup
st.set_page_config(page_title="Personal Fitness App", page_icon=":muscle:")


# Navigation setup
pages = {
    "Client Details":"views/Client_form.py",
    "Nutrition & Meal Planning": "views/Nutrition&MealPlanning.py",
    "Social Features": "views/Social_Features.py",
    "Workout Planning": "views/workout.py",
    "Menstrual Cycle": "views/menstrual_cycle.py",
    "Contact Us":"forms/contact.py"
}


# Sidebar navigation
st.sidebar.title("Explore")  # Add title to the sidebar
selected_page = st.sidebar.selectbox("Select a Page", options=list(pages.keys()))



# Load the selected page
if selected_page:
    exec(open(pages[selected_page]).read())  # Dynamically load the selected page
```

Client.py

```
import streamlit as st


# Mock functions to replace Langflow-based functionality for illustration
def ask_ai(profile, question):
    # Extract relevant information from the profile
    name = profile["general"]["name"]
    weight = profile["general"]["weight"]
    height = profile["general"]["height"]
    gender = profile["general"]["gender"]
    activity_level = profile["general"]["activity_level"]
    goals = profile["goals"]

    # Basic logic to provide a more useful response based on the question
    if "what should I eat" in question.lower() and "muscle gain" in question.lower():
        # Example advice for muscle gain
        response = (
            f"{name}, to support muscle gain, you should focus on high-protein foods. "
            f"Here are some recommendations:\n"
            f"- **Protein Sources:** Include chicken breast, lean beef, fish, eggs, dairy products, and legumes.\n"
            f"- **Carbohydrates:** Don't skip on complex carbs such as whole grains (brown rice, oats) and plenty of vegetables.\n"
            f"- **Healthy Fats:** Avocados, nuts, seeds, and olive oil are great sources of healthy fats.\n"
            f"- **Hydration:** Drink plenty of water throughout the day.\n"
            f"- **Meal Frequency:** Consider eating 5-6 smaller meals throughout the day to fuel your muscles consistently."
        )
        return response

    # Placeholder for other types of questions
    elif "how can I lose weight" in question.lower():
```

```
        response = (
            f"{name}, to lose weight, focus on a caloric deficit, combining regular exercise with a balanced diet. "
            f"Prioritize whole foods, limit processed sugars, and consider portion control."
        )
        return response

    return "I'm not sure how to answer that. Could you ask something else?"

def get_macros(general_info, goals):
    return {"calories": 2000, "protein": 150, "fat": 70, "carbs": 250}

def create_profile(profile_id):
    profile = {
        "general": {"name": "", "age": 0, "weight": 0.0, "height": 0.0, "gender": "", "activity_level": ""},
        "goals": [],
        "nutrition": {}
    }
    return profile_id, profile

def get_notes(profile_id):
    return []

def get_profile(profile_id):
    return {
        "general": {"name": "John", "age": 25, "weight": 70.0, "height": 175.0, "gender": "Male", "activity_level": "Moderately Active"},
        "goals": ["Muscle Gain"],
        "nutrition": {"calories": 2500, "protein": 180, "fat": 80, "carbs": 300}
    }

def update_personal_info(profile, section, **kwargs):
```

```
    profile[section].update(kwargs)
    return profile

def add_note(note_text, profile_id):
    return {"_id": len(st.session_state.notes) + 1, "text": note_text}

def delete_note(note_id):
    pass

st.title("Personal Fitness App")

def personal_data_form():
    with st.form("personal_data"):
        st.header("Personal Data")
        profile = st.session_state.profile

        name = st.text_input("Name", value=profile["general"]["name"])
        age = st.number_input("Age", min_value=1, max_value=120, step=1,
value=profile["general"]["age"])
        weight = st.number_input("Weight (kg)", min_value=0.0, max_value=300.0, step=0.1,
value=float(profile["general"]["weight"]))
        height = st.number_input("Height (cm)", min_value=0.0, max_value=250.0, step=0.1,
value=float(profile["general"]["height"]))
        genders = ["Male", "Female", "Other"]
        gender = st.radio("Gender", genders, index=genders.index(profile["general"].get("gender",
"Male")))
        activities = ["Sedentary", "Lightly Active", "Moderately Active", "Very Active", "Super Active"]
        activity_level = st.selectbox("Activity Level", activities,
index=activities.index(profile["general"].get("activity_level", "Sedentary")))

        if st.form_submit_button("Save"):
            if all([name, age, weight, height, gender, activity_level]):
                with st.spinner("Saving information..."):
```

```
            st.session_state.profile = update_personal_info(
                profile, "general", name=name, age=age, weight=weight, height=height, gender=gender, activity_level=activity_level
            )
            st.success("Information saved.")
        else:
            st.warning("Please fill in all of the data!")


def macros():
    profile = st.session_state.profile
    st.header("Macros")
    if st.button("Generate Macros with AI"):
        result = get_macros(profile.get("general"), profile.get("goals"))
        profile["nutrition"] = result
        st.success("AI-generated macros")

    with st.form("nutrition_form"):
        calories = st.number_input("Calories", min_value=0, step=1, value=profile["nutrition"].get("calories", 0))
        protein = st.number_input("Protein", min_value=0, step=1, value=profile["nutrition"].get("protein", 0))
        fat = st.number_input("Fat", min_value=0, step=1, value=profile["nutrition"].get("fat", 0))
        carbs = st.number_input("Carbs", min_value=0, step=1, value=profile["nutrition"].get("carbs", 0))

        if st.form_submit_button("Save"):
            with st.spinner("Saving nutrition information..."):
                st.session_state.profile = update_personal_info(profile, "nutrition", calories=calories, protein=protein, fat=fat, carbs=carbs)
                st.success("Nutrition information saved")

def notes():
```

```
    st.subheader("Notes")
    for i, note in enumerate(st.session_state.notes):
        cols = st.columns([5, 1])
        with cols[0]:
            st.text(note.get("text"))
        with cols[1]:
            if st.button("Delete", key=i):
                delete_note(note.get("_id"))
                st.session_state.notes.pop(i)
                st.rerun()

    new_note = st.text_input("Add a new note")
    if st.button("Add Note"):
        if new_note:
            note = add_note(new_note, st.session_state.profile_id)
            st.session_state.notes.append(note)
            st.rerun()



def forms():
    if "profile" not in st.session_state:
        profile_id = 1
        profile = get_profile(profile_id)
        if not profile:
            profile_id, profile = create_profile(profile_id)

        st.session_state.profile = profile
        st.session_state.profile_id = profile_id

    if "notes" not in st.session_state:
```

```
        st.session_state.notes = get_notes(st.session_state.profile_id)


    personal_data_form()
    macros()
    notes()



if __name__ == "__main__":
    forms()
```

client-form end

Nutrition and meals planning

```
import streamlit as st


# Title for the app
st.title("Nutrition and Meal Tracker")


# Define food categories, their respective foods, calorie information, and YouTube links
food_categories = {
    "Vegan": {
        "foods": [
            {"name": "Quinoa", "calories": 222, "benefits": "High in protein and fiber, gluten-free.",
             "recipe_link": "https://www.youtube.com/watch?v=vfUDxET4yVY"},
            {"name": "Chickpeas", "calories": 164, "benefits": "Rich in protein and fiber; great for salads.",
             "recipe_link": "https://www.youtube.com/shorts/L4OXzlw4mps"},
            {"name": "Lentils", "calories": 230, "benefits": "Excellent source of protein, fiber, and iron.",
             "recipe_link": "https://www.youtube.com/watch?v=UaEogCDFV7c"},
            {"name": "Tofu", "calories": 144, "benefits": "High in protein, low in calories; versatile for many dishes.",
             "recipe_link": "https://www.youtube.com/watch?v=-zkj_8bOd58"},
            {"name": "Nuts (Almonds, Walnuts)", "calories": 576, "benefits": "Healthy fats, protein, and omega-3s.",
             "recipe_link": "https://www.youtube.com/watch?v=xcz2LIvpMwI"},
            {"name": "Green Peas", "calories": 62, "benefits": "Good source of protein, vitamins A, C, K.",
             "recipe_link": "https://www.youtube.com/watch?v=6J7nK7ul6UM"},
            {"name": "Brown Rice", "calories": 215, "benefits": "Whole grain, good source of magnesium and fiber.",
             "recipe_link": "https://www.youtube.com/watch?v=EDkTRENbQxs"},
            {"name": "Hemp Seeds", "calories": 170, "benefits": "High in protein and omega-3 fatty acids.",
             "recipe_link": "https://www.youtube.com/watch?v=AGaXInOghq0"},
            {"name": "Avocado", "calories": 160, "benefits": "Healthy fats, great for heart health.",
             "recipe_link": "https://www.youtube.com/watch?v=FMGArdg4-pY"},
```

```
        {"name": "Sweet Potatoes", "calories": 86, "benefits": "Rich in vitamins A and C, and fiber.",
         "recipe_link": "https://www.youtube.com/shorts/j3fJBXxRvGI"}
      ]
   },
   "Vegetarian": {
      "foods": [
        {"name": "Greek Yogurt", "calories": 100, "benefits": "High in protein and probiotics.",
         "recipe_link": "https://www.youtube.com/watch?v=g6BHi9chkbk"},
        {"name": "Eggs", "calories": 68, "benefits": "Complete protein source, high in vitamins.",
         "recipe_link": "https://www.youtube.com/watch?v=ZSGCk7Q1JcA"},
        {"name": "Cottage Cheese", "calories": 206, "benefits": "Rich in protein and calcium.",
         "recipe_link": "https://www.youtube.com/watch?v=JslwCzHnA_8"},
        {"name": "Spinach", "calories": 23, "benefits": "High in iron, calcium, and antioxidants.",
         "recipe_link": "https://www.youtube.com/watch?v=FvM5mcBa5Zc"},
        {"name": "Mushrooms", "calories": 22, "benefits": "Low calorie, rich in B vitamins.",
         "recipe_link": "https://www.youtube.com/watch?v=d_Pc_buMnjE"},
        {"name": "Bell Peppers", "calories": 31, "benefits": "High in vitamin C and antioxidants.",
         "recipe_link": "https://www.youtube.com/watch?v=77JHtM_OjI4"},
        {"name": "Broccoli", "calories": 55, "benefits": "Rich in vitamins K and C, fiber.",
         "recipe_link": "https://www.youtube.com/watch?v=LDFbvgzaE04"},
        {"name": "Cauliflower", "calories": 25, "benefits": "Low-calorie, high in fiber and vitamin C.",
         "recipe_link": "https://www.youtube.com/watch?v=LDFbvgzaE04"},
        {"name": "Chia Seeds", "calories": 138, "benefits": "High in fiber and omega-3 fatty acids.",
         "recipe_link": "https://www.youtube.com/watch?v=LDFbvgzaE04"},
        {"name": "Oats", "calories": 389, "benefits": "Good source of fiber and complex
carbohydrates.",
         "recipe_link": "https://www.youtube.com/watch?v=LDFbvgzaE04"}
      ]
   },
   "Low Carb High Protein": {
      "foods": [
```

```
        {"name": "Chicken Breast", "calories": 165, "benefits": "Lean protein, low in fat.",
         "recipe_link": "https://www.youtube.com/watch?v=jrDjMhQRwDM",
         "recipe_link": "https://www.youtube.com/watch?v=jrDjMhQRwDM"},
        {"name": "Egg Whites", "calories": 17, "benefits": "High protein, low in calories and fat.",
         "recipe_link": "https://www.youtube.com/watch?v=5bAa1CwK5Tc"},
        {"name": "Greek Yogurt", "calories": 100, "benefits": "High in protein, low in carbs.",
         "recipe_link": "https://www.youtube.com/watch?v=3gG7tKD_KPk"},
        {"name": "Protein Powder", "calories": 120, "benefits": "Quick protein source for smoothies or
shakes.",
         "recipe_link": "https://www.youtube.com/watch?v=jrDjMhQRwDM"},
        {"name": "Turkey Breast", "calories": 135, "benefits": "Low fat, high protein.",
         "recipe_link": "https://www.youtube.com/watch?v=jrDjMhQRwDM"},
        {"name": "Pork Tenderloin", "calories": 143, "benefits": "Lean cut, great source of protein.",
         "recipe_link": "https://www.youtube.com/watch?v=jrDjMhQRwDM"},
        {"name": "Beef Jerky", "calories": 116, "benefits": "High in protein, convenient snack.",
         "recipe_link": "https://www.youtube.com/watch?v=jrDjMhQRwDM"},
        {"name": "Seitan", "calories": 120, "benefits": "High protein meat substitute.",
         "recipe_link": "https://www.youtube.com/watch?v=jrDjMhQRwDM"},
        {"name": "Cottage Cheese", "calories": 206, "benefits": "Rich in protein and calcium.",
         "recipe_link": "https://www.youtube.com/watch?v=fVDmsTtZgWc"}
    ]
  }
}

# Initialize session state for meal tracking if it doesn't exist
if "meal_log" not in st.session_state:
  st.session_state.meal_log = []

# User selection for dietary preference
diet_choice = st.selectbox("Select Your Dietary Preference:", list(food_categories.keys()))
```

```
# Use an expander to hide/show the Nutrition-Rich Foods section
with st.expander(f"Nutrition-Rich Foods for {diet_choice} Diet", expanded=False):
    for item in food_categories[diet_choice]["foods"]:
        st.write(f"**{item['name']}**: {item['benefits']} (Calories: {item['calories']})")
        st.markdown(f"[Recipe Video]({item['recipe_link']})")

# Meal tracking section
st.subheader("Meal Tracking")
food_selected = st.multiselect("Select Food Items:", [item["name"] for item in
food_categories[diet_choice]["foods"]])
quantities = {}

# Create a number input for each selected food item
for food in food_selected:
    quantity = st.number_input(f"Quantity of {food} (in servings):", min_value=1, value=1, key=food)
    quantities[food] = quantity

if st.button("Log Meals"):
    # Calculate total calories for the logged meals
    for food in food_selected:
        calories_per_serving = next(item["calories"] for item in food_categories[diet_choice]["foods"] if
item["name"] == food)
        total_calories = calories_per_serving * quantities[food]
        st.session_state.meal_log.append
```

nutrition planning end

Social feature

```
import streamlit as st


# Initialize session state for user profiles and community features
if "users" not in st.session_state:
    st.session_state.users = {}
if "activity_feed" not in st.session_state:
    st.session_state.activity_feed = []
if "recipes" not in st.session_state:
    st.session_state.recipes = []


# Sidebar for navigation
st.sidebar.title("Navigation")
page = st.sidebar.selectbox("Choose a section:", ["User Profile", "Activity Feed", "Challenges",
"Recipe Sharing"])


# 1. User Profile
if page == "User Profile":
    st.title("User Profile")
    username = st.text_input("Enter your username")
    if st.button("Create/Update Profile"):
        if username:
            st.session_state.users[username] = {"workouts": [], "achievements": []}
            st.success(f"Profile for {username} created/updated!")
        else:
            st.error("Please enter a username.")


# 2. Activity Feed
elif page == "Activity Feed":
    st.title("Activity Feed")
    username = st.text_input("Enter your username to post an update")
```

```
    activity_update = st.text_area("Share your workout or meal update")
    if st.button("Post Update"):
        if username in st.session_state.users and activity_update:
            st.session_state.activity_feed.append({"user": username, "update": activity_update})
            st.success("Update posted!")
        else:
            st.error("Please enter a valid username and an update.")

    # Display activity feed
    st.subheader("Updates:")
    for activity in st.session_state.activity_feed:
        st.write(f"{activity['user']}: {activity['update']}")

# 3. Challenges
elif page == "Challenges":
    st.title("Challenges")
    st.write("Join our monthly fitness challenges!")
    st.subheader("Current Challenges:")
    st.write("- 30-Day Plank Challenge")
    st.write("- 10,000 Steps a Day Challenge")
    st.write("- Weekly Yoga Challenge")

# 4. Recipe Sharing
elif page == "Recipe Sharing":
    st.title("Recipe Sharing")
    recipe_name = st.text_input("Recipe Name")
    recipe_details = st.text_area("Recipe Details")
    if st.button("Share Recipe"):
        if recipe_name and recipe_details:
            st.session_state.recipes.append({"name": recipe_name, "details": recipe_details})
            st.success("Recipe shared!")
```

```
    else:
        st.error("Please fill in both fields.")

# Display shared recipes
st.subheader("Shared Recipes:")
for recipe in st.session_state.recipes:
    st.write(f"**{recipe['name']}**: {recipe['details']}")
```

social feature end

Contact us.py

```
import streamlit as st
import pandas as pd
from datetime import datetime

# Function to save appointment data
def save_appointment(data):
    # Here you can implement saving to a database or CSV file
    # For example, appending to a CSV file
    df = pd.DataFrame([data])
    df.to_csv('appointments.csv', mode='a', header=False, index=False)

# Streamlit app
st.title("Contact Us")

# Appointment form
with st.form(key='appointment_form'):
    st.header("Request an Appointment with a Nutritionist")

    name = st.text_input("Your Name", "")
    email = st.text_input("Your Email", "")
    appointment_date = st.date_input("Preferred Appointment Date", datetime.today())
    appointment_time = st.time_input("Preferred Appointment Time", datetime.now().time())

    submit_button = st.form_submit_button(label='Submit')

    if submit_button:
        if name and email:
            appointment_data = {
                'Name': name,
```

```
            'Email': email,
            'Date': appointment_date,
            'Time': appointment_time
        }
        save_appointment(appointment_data)
        st.success("Your appointment request has been submitted successfully!")
    else:
        st.error("Please fill in all fields.")

# Display previous appointments (optional)
if st.checkbox("Show Previous Appointments"):
    try:
        appointments = pd.read_csv('appointments.csv', names=['Name', 'Email', 'Date', 'Time'])
        st.dataframe(appointments)
    except FileNotFoundError:
        st.error("No previous appointments found.")
```

contact us end

Menstrual cycle

```
import streamlit as st
from datetime import datetime, timedelta

# Initialize session state for menstrual cycle tracking
if "cycle_data" not in st.session_state:
    st.session_state.cycle_data = []

# 1. Menstrual Cycle Tracker
st.title("Menstrual Cycle Tracker")
st.subheader("Track Your Menstrual Cycle")

# Input for cycle start date and length
start_date = st.date_input("Start Date of Last Period", datetime.today() - timedelta(days=28))
cycle_length = st.number_input("Cycle Length (in days)", min_value=21, max_value=35, value=28)
symptoms = st.text_area("Symptoms (e.g., cramps, headaches)")
mood = st.selectbox("Mood", ["Happy", "Sad", "Irritable", "Anxious", "Neutral"])

if st.button("Save Cycle Data"):
    cycle_entry = {
        "start_date": start_date,
        "cycle_length": cycle_length,
        "symptoms": symptoms,
        "mood": mood,
        "next_period": start_date + timedelta(days=cycle_length)
    }
    st.session_state.cycle_data.append(cycle_entry)
    st.success("Cycle data saved!")

# 2. Cycle Summary
st.subheader("Cycle Summary")
```

```
if st.session_state.cycle_data:
    st.write("Previous Cycle Entries:")
    for entry in st.session_state.cycle_data:
        st.write(f"**Start Date:** {entry['start_date']}")
        st.write(f"**Cycle Length:** {entry['cycle_length']} days")
        st.write(f"**Next Expected Period:** {entry['next_period']}")
        st.write(f"**Symptoms:** {entry['symptoms']}")
        st.write(f"**Mood:** {entry['mood']}")
        st.write("---")
else:
    st.write("No cycle data available. Please add your cycle data.")

# 3. Dietary Recommendations
st.subheader("Dietary Recommendations for Menstrual Health")

phases = {
    "Menstrual Phase": {
        "duration": "Days 1-5",
        "foods": [
            "Leafy greens",
            "Bananas",
            "Dark chocolate",
            "Nuts",
            "Whole grains",
            "Quinoa",
            "Lentils",
            "Pumpkin seeds",
            "Fish (like salmon)",
            "Chickpeas"
        ],
```

```
    "tips": "Focus on iron-rich foods to replenish what you lose. Stay hydrated and opt for light meals.",
    "explanation": "During menstruation, the body loses blood, leading to a drop in iron levels. Consuming iron-rich foods can help combat fatigue and maintain energy levels. Foods high in magnesium can also help reduce cramps.",
    "youtube_links": [
      "https://www.youtube.com/watch?v=E-8gvJlkY8c",
      "https://www.youtube.com/@theyogainstituteofficial"
    ]
  },
  "Follicular Phase": {
    "duration": "Days 6-14",
    "foods": [
      "Lean proteins",
      "Fruits",
      "Vegetables",
      "Eggs",
      "Seeds",
      "Whole grains",
      "Greek yogurt",
      "Fish (like cod or tuna)",
      "Berries",
      "Avocados"
    ],
    "tips": "Increase protein intake and healthy fats to support muscle growth and energy levels. Focus on antioxidants from fruits and vegetables.",
    "explanation": "In this phase, estrogen levels rise, enhancing energy and mood. Eating nutrient-dense foods helps support your body during this revitalizing phase.",
    "youtube_links": [
      "https://www.youtube.com/watch?v=E-8gvJlkY8c",
      "https://www.youtube.com/@theyogainstituteofficial"
    ]
  },
```

```
  "Ovulation Phase": {
    "duration": "Days 15-17",
    "foods": [
      "Berries",
      "Avocados",
      "Cruciferous vegetables",
      "Fish (like mackerel)",
      "Legumes",
      "Citrus fruits",
      "Nuts",
      "Sweet potatoes",
      "Chia seeds",
      "Pomegranate"
    ],
    "tips": "Focus on antioxidant-rich foods to support hormonal balance and boost mood. Healthy fats are important for hormone production.",
    "explanation": "Ovulation is when the body is at its peak fertility and energy. Consuming foods rich in vitamins and minerals can enhance overall well-being and mood.",
    "youtube_links": [
      "https://www.youtube.com/watch?v=E-8gvJlkY8c",
      "https://www.youtube.com/@theyogainstituteofficial"
    ]
  },
  "Luteal Phase": {
    "duration": "Days 18-28",
    "foods": [
      "Complex carbohydrates",
      "Magnesium-rich foods",
      "Fermented foods",
      "Oily fish",
      "Dark leafy greens",
      "Chickpeas",
```

```
            "Brown rice",
            "Sweet potatoes",
            "Nuts",
            "Dark chocolate"
        ],
        "tips": "Manage cravings with balanced meals and ensure sufficient magnesium to reduce PMS symptoms. Incorporate fiber to help with digestion.",
        "explanation": "This phase often comes with PMS symptoms. Focus on balancing blood sugar and incorporating foods that soothe cramps and help stabilize mood.",
        "youtube_links": [
            "https://www.youtube.com/watch?v=E-8gvJlkY8c",
            "https://www.youtube.com/@theyogainstituteofficial"
        ]
    }
}

for phase, details in phases.items():
    with st.expander(phase, expanded=False):
        st.write(f"**Duration:** {details['duration']}")
        st.write("**Recommended Foods:**")
        for food in details['foods']:
            st.write(f"- {food}")
        st.write(f"**Tips:** {details['tips']}")
        st.write(f"**Explanation:** {details['explanation']}")
        st.write("**YouTube Links:**")
        for link in details['youtube_links']:
            st.write(f"- [Watch Here]({link})")
        st.write("---")

# Footer
st.write(f"Total Entries: {len(st.session_state.cycle_data)}")
```

menstrual cycle end

```
import streamlit as st
from datetime import datetime, timedelta

# Initialize session state for user profiles, workouts, and exercises
if "users" not in st.session_state:
    st.session_state.users = {}
if "workouts" not in st.session_state:
    st.session_state.workouts = []
if "exercises" not in st.session_state:
    st.session_state.exercises = {
        "Strength": ["Squat", "Deadlift", "Bench Press", "Lunges"],
        "Cardio": ["Running", "Cycling", "Jump Rope", "Swimming"],
        "Flexibility": ["Yoga", "Pilates", "Stretching"]
    }

# Sidebar for navigation
st.sidebar.title("Navigation")
page = st.sidebar.selectbox("Choose a section:", ["User Profile", "Workout Logging", "Exercise Library", "Monthly Goal Setting", "Monthly Schedule"])

# 1. User Profile
if page == "User Profile":
    st.title("User Profile")
    username = st.text_input("Enter your username")
    if st.button("Create/Update Profile"):
        if username:
            if username not in st.session_state.users:
                st.session_state.users[username] = {"workouts": [], "goals": []}
            st.success(f"Profile for '{username}' created/updated!")
        else:
            st.error("Please enter a username.")
```

```
# 2. Workout Logging
elif page == "Workout Logging":
    st.title("Workout Logging")
    username = st.text_input("Enter your username to log workout")  # Prompt for username
    if username not in st.session_state.users:
        st.error("Please create a user profile first.")
    else:
        workout_name = st.selectbox("Select Workout Type", list(st.session_state.exercises.keys()))
        exercise = st.selectbox("Select Exercise", st.session_state.exercises[workout_name])
        sets = st.number_input("Sets", min_value=1, value=3)
        reps = st.number_input("Reps", min_value=1, value=10)
        weight = st.number_input("Weight (kg)", min_value=0.0, value=0.0, format="%.2f")

        if st.button("Log Workout"):
            workout_entry = {
                "user": username,
                "exercise": exercise,
                "sets": sets,
                "reps": reps,
                "weight": weight,
                "date": datetime.now().strftime("%Y-%m-%d")
            }
            st.session_state.workouts.append(workout_entry)
            st.session_state.users[username]["workouts"].append(workout_entry)
            st.success("Workout logged!")

        # Display logged workouts
        st.subheader("Logged Workouts:")
        if username in st.session_state.users:
            for workout in st.session_state.users[username]["workouts"]:
```

```
            st.write(f"{workout['user']} logged {workout['exercise']} on {workout['date']} - "
                     f"{workout['sets']} sets of {workout['reps']} reps at {workout['weight']} kg")

# 3. Exercise Library
elif page == "Exercise Library":
    st.title("Exercise Library")
    exercise_type = st.selectbox("Select Exercise Type", list(st.session_state.exercises.keys()))
    st.subheader(f"{exercise_type} Exercises")
    for exercise in st.session_state.exercises[exercise_type]:
        st.write(f"- {exercise}")

# 4. Monthly Goal Setting
elif page == "Monthly Goal Setting":
    st.title("Monthly Goal Setting")
    username = st.text_input("Enter your username to set goals")  # Prompt for username
    if username not in st.session_state.users:
        st.error("Please create a user profile first.")
    else:
        goal = st.text_input("Enter your monthly goal (e.g., 'Complete 10 workouts', 'Run 50 km')")
        if st.button("Save Goal"):
            if goal:
                st.session_state.users[username]["goals"].append(goal)
                st.success("Goal saved!")
            else:
                st.error("Please enter a goal.")

        # Display goals
        st.subheader("Your Monthly Goals:")
        if username in st.session_state.users and st.session_state.users[username]["goals"]:
            for goal in st.session_state.users[username]["goals"]:
                st.write(f"- {goal}")
```

```
        else:
            st.write("No goals set yet.")

# 5. Monthly Schedule
elif page == "Monthly Schedule":
    st.title("Monthly Workout Schedule")
    username = st.text_input("Enter your username to generate schedule")  # Prompt for username
    if username not in st.session_state.users:
        st.error("Please create a user profile first.")
    else:
        workout_frequency = st.number_input("Enter workout frequency (days per week):",
min_value=1, max_value=7, value=3)

        # Generate a monthly schedule
        if st.button("Generate Schedule"):
            start_date = datetime.now()
            end_date = start_date + timedelta(days=30)
            current_date = start_date

            schedule = {}
            workout_count = 0

            while current_date < end_date:
                if current_date.weekday() < workout_frequency:  # Check if it's a workout day
                    if st.session_state.workouts:
                        workout = st.session_state.workouts[workout_count % len(st.session_state.workouts)]
                        schedule[current_date.strftime("%Y-%m-%d")] = workout
                        workout_count += 1
                current_date += timedelta(days=1)

            st.subheader("Your Monthly Schedule:")
```

```
        for date, workout in schedule.items():
            st.write(f"{date}: {workout['exercise']} - {workout['sets']} sets of {workout['reps']} reps at {workout['weight']} kg")
    else:
        st.write("Click 'Generate Schedule' to create your monthly workout plan.")


# Footer
st.sidebar.title("Profile Summary")
if 'users' in st.session_state:
    for user in st.session_state.users.keys():
        st.sidebar.write(f"**{user}**: {len(st.session_state.users[user]['workouts'])} workouts logged, "
                         f"{len(st.session_state.users[user]['goals'])} goals set.")
```

workout end