Assignment: Integrating Social Media and Calendar Accounts with AI Agents

Table of Contents

1. Introduction

Managing multiple accounts like Gmail, Outlook, and social media platforms can be overwhelming.

This project integrates these accounts into a unified system, providing a combined calendar view and updates from all platforms.

AI agents handle user-specific tasks, such as scheduling repairs or managing marketing campaigns.

2. Objectives

- Integrate Gmail, Outlook, and social media accounts via a single interface.
- Provide a combined calendar view of events from all accounts.
- Use AI agents to automate tasks such as marketing and maintenance.
- Ensure a seamless and secure user experience.

3. System Architecture

3.1. Components

1. User Interface (UI)
   - A dashboard to manage accounts, view calendars, and interact with agents.
2. Backend Services
   - Handles account integration, data aggregation, and AI agent management.
3. Database
   - Stores user data, OAuth tokens, and aggregated data securely.
4. Notification System
   - Sends reminders and task updates.

3.2. Workflow

1. User Authentication
   - Authenticate and integrate accounts via OAuth (e.g., Gmail, Outlook).
2. Data Aggregation
   - Fetch and normalize data from all platforms.
3. AI Agent Management
   - Assign tasks to agents for specific needs (e.g., marketing campaigns).
4. Unified Dashboard
   - Display combined calendar events and updates in the UI.

4. Technical Stack

Frontend

- Framework: React.js or Flutter.
- Features: Account integration, calendar view, task manager.

Backend
- Framework: FastAPI (Python) or Node.js (JavaScript).
- Libraries: LangChain for agents, OAuth libraries for authentication.

Database
- PostgreSQL: Store user data and events.
- Redis: Cache frequently used data (e.g., tokens).

APIs
- Google APIs: Gmail API, Google Calendar API.
- Microsoft APIs: Microsoft Graph API.
- Social Media APIs: Twitter API, Facebook Graph API.

5. Implementation
5.1. User Authentication and Account Integration
- Authenticate users via OAuth for Gmail, Outlook, and other platforms.
- Example OAuth API calls:
- Google Calendar OAuth:
https://accounts.google.com/o/oauth2/auth
- Microsoft Outlook OAuth:
https://login.microsoftonline.com/common/oauth2/v2.0/authorize

5.2. Data Aggregation
- Fetch events from integrated platforms using their APIs.
- Normalize and merge data into a common schema.

5.3. AI Agent Management
- Use LangChain/OpenAI Swarm agents for task automation.
- Examples:
- Calendar Agent: Detect schedule conflicts and suggest meeting slots.
- Social Media Agent: Post updates and track engagement.

5.4. Unified Calendar and Notifications
- Combine calendar events and updates into a single view.
- Use Firebase or WebSockets for real-time notifications.

6. Code Implementation
6.1. Backend API (FastAPI Example)

```
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI()

class CalendarEvent(BaseModel):
      title: str
      start_time: str
      end_time: str
      platform: str

@app.get("/calendar/events", response_model=list[CalendarEvent])
def get_combined_calendar():
      # Placeholder: Fetch and merge events from Gmail, Outlook, etc.
      return [
            {"title": "Meeting with John", "start_time": "10:00 AM",
"end_time": "11:00 AM", "platform": "Google Calendar"},
            {"title": "Team Standup", "start_time": "2:00 PM", "end_time":
"2:30 PM", "platform": "Outlook"},
            ]

if _name_ == "_main_":
      import uvicorn
      uvicorn.run(app, host="0.0.0.0", port=8000)
```

6.2. LangChain AI Agent

```
from langchain.agents import initialize_agent, Tool
from langchain.llms import OpenAI

# Define tools for AI agents
def get_calendar_events():
      return "Fetching events from Google Calendar..."

def get_social_media_updates():
      return "Fetching updates from social media platforms..."

calendar_tool = Tool(
      name="Google Calendar Manager",
      func=get_calendar_events,
      description="Fetch and manage calendar events."
      )

social_media_tool = Tool(
      name="Social Media Manager",
      func=get_social_media_updates,
      description="Fetch and handle social media updates."
      )

# Initialize LangChain agent
llm = OpenAI(model="gpt-4")
agent = initialize_agent(
      tools=[calendar_tool, social_media_tool],
      llm=llm,
      agent="zero-shot-react-description"
      )

# Run the agent with a query
query = "Combine my calendar events and fetch social media notifications."
response = agent.run(query)
print(response)
```

7. Real-World Use Case

Scenario:
A user integrates Gmail, Outlook, and Twitter accounts. The system:

1. Aggregates calendar events.
2. Suggests meeting slots.
3. Uses the Maintenance Agent to book a plumber.

8. Conclusion

This project provides a streamlined way to manage multiple accounts, offering users a unified view of their calendars and social media updates.

AI agents enhance productivity by automating tasks, ensuring a seamless user experience.

Instructions to Run the Code

1. Backend Setup:
- Install dependencies: pip install fastapi uvicorn pydantic.
- Run the backend server: python app.py.
2. Agent Setup:
- Install LangChain: pip install langchain openai.
- Replace placeholder functions with actual implementations.
- Run the script: python agent.py.