Project: TrailMate - Multi-Agent AI Travel Planning Chatbot

Overview

TrailMate is a travel planning chatbot powered by a multi-agent AI system. It helps individuals or groups plan personalized trips by finding suitable accommodations, recommending activities, and managing budgets. The system takes a plain-text input (e.g., "Plan a 3-day trip to Dubai for 4 friends under $800 with beach activities") and outputs a full itinerary.

Core Stack

Component	**Technology**
Backend	Python (FastAPI / LangChain)
LLM	OpenAI GPT-4 API / Claude / Gemini
Agents	LangChain Agents or custom Python classes
Frontend	Streamlit or React (preferred: Streamlit for quick prototyping)
APIs	Booking.com, TripAdvisor, Yelp, Skyscanner, Google Maps
Data Storage (optional)	ChromaDB / Pinecone / PostgreSQL
Hosting	HuggingFace Spaces / Azure / Vercel
Scraping	BeautifulSoup + Requests (for fallback if no API)

Agent Architecture

Master Agent (Coordinator)

- **Input**: User text query
- **Task**:
    - Parses query using LLM to extract structured info (location, budget, group size, preferences)
    - Delegates subtasks to specialized agents
    - Collects results and formats final plan

Specialized Agents:

1. Housing Agent

- **Input**: Location, date, budget, preferences

- **Task**:
    - Use Booking.com or Airbnb API
    - Fallback: scrape using BeautifulSoup (e.g., Airbnb public listings)
- **Output**: Top 3 accommodation options

2. Activity Agent

- **Input**: Location, preferences
- **Task**:
    - Use TripAdvisor API (RapidAPI) or Yelp API
    - Filters based on type (e.g., beach, food, museums)
- **Output**: Top 5 activities

3. Budget Agent

- **Input**: Accommodation and activity data
- **Task**:
    - Sum cost of hotel + activities
    - Suggest alternate options if cost exceeds budget
- **Output**: Final plan within budget

4. Group Coordinator Agent

- **Input**: Group preference forms
- **Task**:
    - Collects group member preferences via Streamlit/Google Forms
    - Embeds responses (OpenAI embeddings or sentence-transformers)
    - Clusters or ranks to find majority preferences

5. Personalization Agent

- **Input**: User preferences (e.g., solo trip, comfort level)
- **Task**:
    - Adds humanized descriptions
    - Suggests tips like best time to visit, hidden gems

Agent Orchestration

Agent orchestration is managed by the Master Agent, which acts as the central coordinator of the workflow. The orchestration follows these steps:

1. **Input Reception**:
    - The Master Agent receives a natural language input from the user (via UI or API).
2. **Intent Parsing**:
    - The Master Agent uses an LLM to extract structured details such as destination, budget, travel dates, group size, and preferences.
3. **Task Delegation**:
    - Based on extracted entities, subtasks are dispatched to respective agents:
        - Housing Agent for accommodation
        - Activity Agent for attractions
        - Budget Agent for cost optimization
        - Group Coordinator Agent (if applicable)
        - Personalization Agent for customizing tone and suggestions
4. **Execution and Parallelization**:
    - Each agent performs its task independently. This can be done sequentially or in parallel for faster performance using task queues or async calls.
5. **Result Aggregation**:
    - Once each agent returns its results, the Master Agent compiles the full travel itinerary including accommodations, activities, and a cost summary.
6. **Final Output Generation**:
    - The Master Agent formats the result into a user-friendly output (structured itinerary or conversational response) and sends it back to the frontend.
7. **Optional Refinement**:
    - If the output exceeds budget or lacks preferences, the Master Agent can loop back and re-query agents with adjusted parameters.

Full Implementation Flow (Step-by-Step)

Step 1: Setup

- Create GitHub repo with folders:
    - /agents, /backend, /frontend, /data, /utils

- Setup virtual environment and install dependencies

Step 2: Master Agent

- Parses plain text using OpenAI API
- Extracts structured trip data and delegates to specialized agents

Step 3: Agent APIs or Scraping Setup

- Register for necessary APIs:
    - Yelp API
    - TripAdvisor via RapidAPI
    - Booking.com scraping via public search pages

Step 4: Agent Logic

- Implement agents as Python classes or LangChain tools
- Ensure each agent performs its designated subtask effectively

Step 5: Frontend (Streamlit)

- Create a form to collect trip requests
- Display the planned itinerary using an interactive interface

Step 6: Workflow Integration

- MasterAgent orchestrates calls to:
    - Housing Agent
    - Activity Agent
    - Budget Agent
    - Group Coordinator Agent (if group)
    - Personalization Agent
- Finalize and format trip plan for output

Step 7: Testing

- Use diverse trip prompts to evaluate the chatbot's performance
- Validate correctness of extracted information and plan recommendations

Step 8: Deployment

- Deploy Streamlit app to Hugging Face Spaces or Vercel
- Upload GitHub repo with proper documentation and instructions

---

Sample Prompt and Output

Prompt: "Plan a 4-day trip to Dubai for 2 people under $600. We like beaches and cheap food."
Output:

- Accommodation: XYZ Hotel ($300 total)
- Day 1: Visit Jumeirah Beach, local food tour
- Day 2: Desert Safari, Camel Ride
- Total: $570

Optional Features

- Save trips to user's account (add SQLite or Firebase)
- Email itinerary with SendGrid
- Map view with Google Maps API
- PDF download of trip