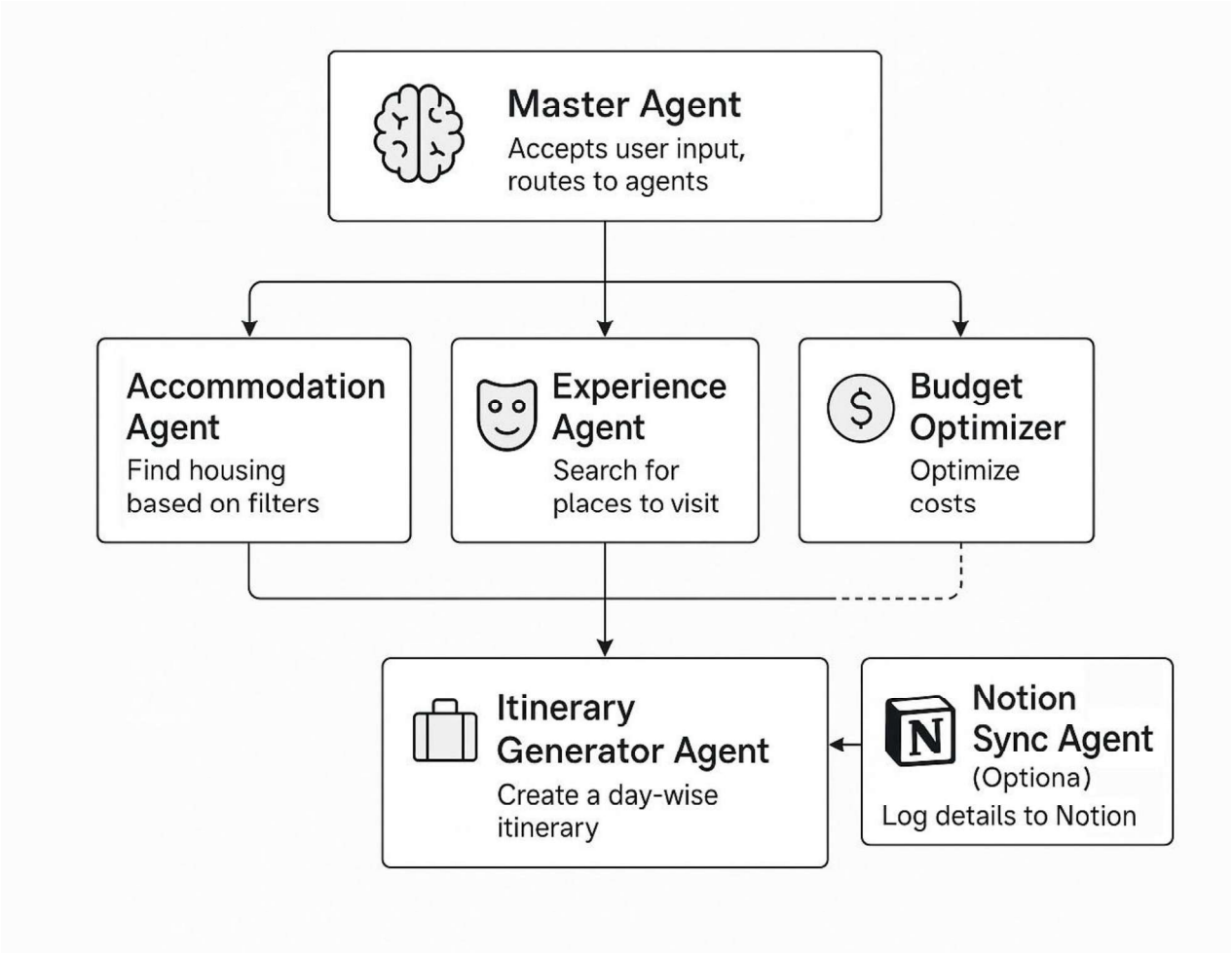
TrailMate Workflow

Step-by-Step Workflow (overview)

1. **Master Agent**
   Takes user input like *"Plan a 5-day budget trip to Paris for 2 art lovers"*, extracts key details, and dispatches tasks to other agents.
2. **Accommodation Agent**
   Finds places to stay based on budget, location, and ratings.
3. **Experience Agent**
   Suggests personalized activities and attractions (food, culture, nature, etc.).
4. **Budget Optimizer Agent**
   Calculates total cost and adjusts plans to fit the user's budget.
5. **Itinerary Generator Agent**
   Builds a daily travel schedule with activities, meals, and accommodations.
6. **Notion Sync Agent (Optional)**
   Saves the final plan in a shareable Notion document.
7. **Cloudera Data Stream Agent (Optional)**
   Adds real-time trends like price spikes or crowd data for smart planning.

Workflow Diagram



Brief explanation of each agent in TrailMate Planning Chatbot Workflow, with tools, APIs, and stacks

1. Master Agent

Role:

- Receives plain-text task (e.g., "Plan a 4-day trip to Rome for 3 people on a medium budget.")
- Extracts **key info**: location, duration, number of travelers, preferences (nature, food, culture), and budget level
- Dispatches to appropriate sub-agents

Tools & Stack:

- **NLP Parsing**: OpenAI GPT-4 / Groq / Hugging Face transformers
- **Intent detection**: LangChain (text chunking & routing)
- **Fast Prototyping**: Python + Streamlit / FastAPI backend

✓ *Why it's useful*: Makes the experience natural for the user (no forms) and intelligent behind the scenes.

---

2. Accommodation Agent

Role:

- Finds hotels, hostels, or Airbnbs based on filters like:
    - Price range
    - Distance to attractions
    - Traveler ratings
    - Amenities

Tools & Stack:

- **Booking API or Hotels API**: RapidAPI, Expedia API, Agoda Affiliate
- **Search automation (if no API)**: BeautifulSoup + Selenium (not preferred for hackathon speed)
- **Fallback**: Mock data or JSON sample hotel lists to simulate

✓ *Why it's useful*: Handles one of the most complex, time-consuming trip decisions.

---

3. Experience Agent

Role:

- Suggests top-rated places to visit based on preferences like:
  - Nature
  - Historical landmarks
  - Food & nightlife
  - Cultural experiences

Tools & Stack:

- **Google Places API**
- **TripAdvisor API** or GetYourGuide API
- **Custom filtering**: Python scripts for keyword-tagged JSON data (for mock/demo)

✅ *Why it's useful*: Helps tailor the trip to user interests (high personalization).

---

4. Budget Optimizer Agent

Role:

- Analyzes the cost of:
  - Hotels + Activities + Daily expenses
- Suggests cost-cutting tips or premium upgrades based on user's budget level

Tools & Stack:

- **Pandas/NumPy**: for cost aggregation
- **OpenAI GPT**: for text-based optimization ("Suggest 2 lower-cost alternatives near Rome's Colosseum")
- **Optional**: Use exchange rate APIs if travel is cross-country

✅ *Why it's useful*: Aligns the trip plan with user affordability, ensuring practicality.

---

5. Itinerary Generator Agent

Role:

- Creates a **day-by-day plan** with:
  - Morning, afternoon, evening activities
  - Rest periods, meals
  - Map-based routing (optional)

Tools & Stack:

- **GPT-4** + prompt template: *"Create a 4-day itinerary in Rome visiting {X} and staying at {Y} within {Z} budget"*
- **DateTime + Jinja2 templates**: for formatting daily schedules
- **Optional**: Map APIs (Google Maps API) to visualize routes

✅ *Why it's useful*: Translates raw data into a structured plan users can follow.

---

6. Notion Sync Agent (Bonus)

Role:

- Exports travel plan and cost summary to a **shared Notion workspace**
- Enables collaborative planning (good for groups)

Tools & Stack:

- **Notion API**: https://developers.notion.com/
- Python library: notion-client or notion-sdk-py
- Use pages and databases to structure trip info

✅ *Why it's useful*: Adds a polished touch — users can view or edit their trip in a familiar workspace.

---

7. Cloudera Data Stream Agent (Bonus)

Role:

- Streams real-time travel insights such as:
  - Price surges
  - Seasonal crowd estimates
  - Event-based fluctuations

Tools & Stack:

- **Cloudera Free Trial** + Kafka / Spark
- Data: use pre-collected datasets (e.g., monthly hotel price trends in a region)
- API or simulated data stream via Pandas or CSV

✅ *Why it's useful*: Adds real-time intelligence to planning decisions, boosting hackathon score (bonus points).

Tech Stack Summary for Fast Hackathon Execution

Layer	Stack / Tools
LLMs	OpenAI GPT-4 / Groq / Hugging Face
APIs	Google Places, Booking, Notion, TripAdvisor (via RapidAPI)
Backend	Python, LangChain, FastAPI
Data & Logic	Pandas, NumPy, Mock JSON
Frontend	Streamlit (easy), or React + Tailwind (if you have time)
Output	Notion, HTML/PDF download (optional)
Bonus Tools	Cloudera, Notion

7-Person Hackathon Role Division

Teammate	Role	Primary Tasks	Key Tools / Stack
1. Master Agent Developer	**LLM Integration & Prompt Parsing**	- Build Master Agent to parse plain-text prompts (location, duration, budget, preferences) - Route tasks to correct sub-agents	GPT-4 / Groq, LangChain, Python
2. Accommodation Agent Developer	**Housing Search API Expert**	- Integrate Booking.com or mock housing API - Filter results by budget, rating, location	RapidAPI (Booking, Expedia), Python Requests, JSON
3. Experience Agent Developer	**Attractions & Activities Planner**	- Use Google Places or mock data - Personalize based on user interests (history, nature, food, etc.)	Google Places API, TripAdvisor, Pandas
4. Budget & Cost Agent Developer	**Budget Optimizer**	- Combine hotel + activity prices - Suggest upgrades or cost-saving alternatives - Connect Cloudera mock stream (optional)	Pandas, NumPy, Cloudera (CSV or mock stream), Python
5. Itinerary Generator Developer	**Itinerary Logic & Output Formatting**	- Structure activities into a clean, day-wise plan - Add time slots & summaries - Format for output	GPT prompt templates, Jinja2, datetime
6. Notion Integration Lead	**Collaborative Output Designer**	- Integrate Notion API - Format final output into a Notion dashboard or page - Sync updates across agents	Notion API, Python `notion-client`, Templates

Teammate	Role	Primary Tasks	Key Tools / Stack
7. UI/UX & Presenter	**Frontend, Demo Video & Slides**	- Build simple UI with Streamlit or React - Create slides and record demo - Explain team roles, workflow, and tools	Streamlit / React, Canva / Google Slides, Loom / OBS

Smart Suggestions:

- **If time allows**, team members can **pair up**:
  - Itinerary + Budget logic = Co-developed
  - Accommodation + Experience agents = Shared codebase
- Focus first on **mock APIs/data** → real API integration can be added after base functionality

Smart Collaboration Plan (Pairing + Focus Strategy)

1. Pair Development: Shared Responsibility for Interconnected Tasks

Some agents in the workflow are closely related — they **depend on similar data structures** or **work on the same section of the user journey**. To maximize productivity and code reuse, these pairs should **collaborate or build off a shared base**:

✅ Pair 1: Itinerary Generator + Budget Optimizer Developers

Why pair them?

- The **Itinerary Generator** relies on **cost breakdowns** to schedule affordable activities and hotels.
- The **Budget Optimizer** needs to access day-wise planning to recommend cheaper alternatives.

Shared elements:

- Travel duration
- Per-day cost calculations
- Data objects for activities, accommodations, meals
- Output formatting (HTML or Notion-ready)

How they can work together:

- Define a **shared data schema** (e.g., `Day 1 = [Hotel, Museum, Lunch, Total Cost]`)
- Build a **cost summary function** used in both agents
- Ensure that budget suggestions feed into the itinerary generator dynamically

---

Pair 2: Accommodation Agent + Experience Agent Developers

Why pair them?

- Both need to **fetch and filter data** based on:
  - Location
  - Budget level
  - Preferences (e.g., near landmarks, "nature", “food”)
- Both return lists of options with similar fields: name, price, rating, location

Shared elements:

- Search filters (location, cost, tags)
- API call templates (or mock JSON file)
- Rating systems or distance calculations
- Output display format

How they can work together:

- Use a **common API handler or mock data parser**
- Create a reusable `filter_results()` function
- Decide on a **shared JSON schema** to unify accommodation and experience formats

---

Focus First on Mock Data / APIs

❗ Why?

API setup (especially authentication, quotas, rate limits) can **slow down early development**. For hackathons, the focus is on **proof-of-concept**, not full production integration.

What to do:

- Create **JSON files** that simulate API responses from:
  - Booking.com (hotels list with prices, ratings)
  - TripAdvisor/Google Places (attractions with tags and prices)
- Load them using `json.load()` or `pandas.read_json()` in the agents
- Build and test logic (filtering, itinerary planning, budgeting) **without internet dependency**

✅ Bonus: Real API Integration as Phase 2

Once the core demo works, swap mock functions with:

```
python
CopyEdit
import requests

response = requests.get(API_ENDPOINT, params=params)
data = response.json()
```

(Or use Python SDKs if provided by the API.)

Summary of Collaboration Plan

Collaboration Area	Teammates	Focus	Shared Component
Itinerary + Budget	Dev 4 + Dev 5	Cost-aware planning	Timeline schema, per-day cost
Accommodation + Experience	Dev 2 + Dev 3	Personalized search	Common search filter logic
Phase Strategy	All	Start with mocks	Later: real API integration
Notion + Presenter	Dev 6 + Dev 7	Output + UI polish	Export templates, visuals

Thank You!