**Approach:**

|  |
| --- |
| **⋮** | **Agent** | **Responsibilities** | **Inputs / Tools** |
| **⋮** | **Orchestrator (Master Agent)** | User intake, orchestration of sub‑tasks, integration and final itinerary packaging | Dialogue flow control, aggregated user preferences |
| **⋮** | **Destination Advisor Agent** | Advice in weather, safety, cultural norms, visa requirements, vaccinations, peak/off‑peak | Weather API, travel advisory API, visa databases |
| **⋮** | **Budget & Cost Agent** | Estimates of flight, hotel, food, local transport, insurance, tours, sustainability options | Flight/hotel pricing tools, cost calculators |
| **⋮** | **Flight Agent** | Compare flights, flexible date options, book tickets | Airline and global distribution system (GDS) API integration |
| **⋮** | **Accommodation Agent** | Hotel/Airbnb/resort options, reviews, location matching, deals | Hotel APIs, reviews database |
| **⋮** | **Transport Agent** | Rental car/taxi/public transport options, transfers, ride scheduling | Car rental APIs, ride‑share APIs |
| **⋮** | **Activities & Excursions Agent** | Suggests activities, tours, day trips, local cultural events | Local tours API, events, attractions database |
| **⋮** | **Health & Safety Agent** | Vaccine requirements, health risk alerts, travel insurance advice | Health API, insurance providers |
| **⋮** | **Logistics Agent** | Packing essentials, SIM card/roaming, local currency, reminders, document storage | Checklist generator, notification scheduler |
| **⋮** | **Accessibility & Special Needs Agent** | Handles mobility, diet, allergies, accessibility‑friendly options | Filter tools, property metadata |
| **⋮** | **Sustainability & Ethics Agent** | Eco‑friendly lodging, local impact options, responsible tours | Sustainability ratings database |

⋮

Data Flow:

1. User says: *“Plan a 10‑day beach trip for two in Bali in November within a mid‑range budget.”*
2. **Orchestrator Agent** captures inputs and splits task:

* Initiates calls to **Destination Advisor** → gets climate, visa, safety, culture info.
* Sends to **Budget Agent** → gets cost breakdown forecasts.
* Invokes **Flight Agent** → recommends flights.
* Invokes **Accommodation Agent**, then **Transport**, then **Activities Agent**, passing previous data as context.

1. Each agent may ask follow‑ups via input-required if necessary (e.g., exact dates, dietary preferences).
2. Intermediate updates and final artifacts are streamed via A2A SSE or messages.
3. Orchestrator compiles results into a finalized travel plan artifact, presented to the user with booking links, alerts, checklists.

✅ Pros of This Approach

* **Modular, scalable architecture**: Each agent handles a distinct domain—easy to maintain and extend.
* **Interoperability**: Agents plug into different systems and can be upgraded independently using A2A
* **Stateful, long-running tasks**: Capable of handling multi‑turn dialogues, clarification, streaming task status.
* **Flexibility to integrate multiple tools**: Agents connect via MCP to tools/APIs without central bottlenecks
* **Adaptability**: Add new agents (e.g., multilingual agent, business‑travel agent) without rearchitecting the system.

Cons & Challenges

* **Complex coordination logic**: Orchestrator must manage dependencies, retries, partial responses and agent routing.
* **Agent discovery and routing**: Requires either registry or routing logic—likely an LLM‑routing component to decide which agent handles what task
* **Latency overhead**: Multiple agent handoffs may increase response time.
* **Data consistency and context tracking**: Must ensure each agent receives full context to avoid misaligned or redundant tasks.
* **Tool integration maintenance**: Each agent must keep APIs (e.g., flight providers, weather, insurance) up-to-date.

**Example of Data Flow:**

User wants to **plan a 10-day beach vacation to Bali in November** for **2 adults**, with **mid-range budget**, including **flights, hotel, transport, and activities**.

💬 **Dialogue Flow with Data Flow between Agents**

**Step 1: User Initiation**

**User:**

I want to plan a 10-day beach vacation in Bali in November for 2 adults. Can you help?

**Step 2: Orchestrator Agent receives the input**

**Tasks parsed:**

* Intent: Plan full vacation
* Entities: Location = Bali, Duration = 10 days, Time = November, Travelers = 2 adults

📤 **Data forwarded to**:  
✅ Destination Advisor Agent  
✅ Budget Agent  
✅ Flight Agent  
✅ Accommodation Agent  
✅ Transport Agent  
✅ Activities Agent

**Step 3: Destination Advisor Agent responds**

Bali in November is warm and humid (avg. 27°C). This is the start of rainy season but popular for surfing.  
Visa-on-arrival is available for Indian passport holders.  
No special vaccinations needed.  
Basic English is spoken in tourist areas.

📤 **Data sent back to Orchestrator**:

{"weather": "Humid, avg 27°C, start of rainy season", "visa": "Visa-on-arrival for Indian passport", "language": "Basic English widely spoken", "safety": "Tourist-friendly" }

**Step 4: Budget Agent responds**

Mid-range estimated budget:

* Flights: ₹35,000 per person
* Hotel: ₹4,000/night
* Transport: ₹800/day
* Food & Misc: ₹1,200/day

📤 **Data sent back to Orchestrator**:

{"flight\_estimate": 70000, "hotel\_estimate": 40000, "transport\_estimate": 8000, "misc": 12000, "total": 130000 }

**Step 5: Flight Agent responds**

Best round-trip flights from Delhi to Bali:

* AirAsia: ₹34,500 (1 stop, 12 hrs)
* Singapore Airlines: ₹40,000 (1 stop, 9 hrs)

📤 **Flight options sent**  
📎 Attach artifact: flight\_quote\_324.json

**Step 6: Accommodation Agent responds**

Mid-range hotels in Kuta & Seminyak:

* Hotel Bali Sun: ₹3,800/night – 8.6 rating
* Seminyak Resort: ₹4,200/night – 9.1 rating

📤 **Hotel options sent**  
📎 Attach artifact: hotel\_quotes\_bali.json

**Step 7: Transport Agent responds**

Rental cars available from ₹750/day.  
Airport pickup can be arranged for ₹1,200.  
Taxis and scooters are common options.

{"rental\_car\_daily": 750, "airport\_pickup": 1200, "alternatives": ["Gojek", "Bluebird taxis", "Scooter rentals"] }

**Step 8: Activities Agent responds**

Recommended activities:

* Surfing in Kuta
* Ubud cultural tour
* Tanah Lot sunset visit
* Scuba diving (₹4,000/person)  
  You can book in advance or on arrival.

📤 Data returned as: JSON

{ "excursions": [ {"name": "Kuta Surfing", "cost": 1000}, {"name": "Ubud Day Tour", "cost": 2500}, {"name": "Tanah Lot Temple", "cost": 800}, {"name": "Scuba Diving", "cost": 4000} ] }

**Step 9: Orchestrator Agent compiles the responses**

🧠 **Itinerary Summary Artifact**

* Dates: 10 days in November
* Destination: Bali
* Budget: ₹1.3L
* Flights: AirAsia @ ₹34.5K
* Hotel: Seminyak Resort @ ₹4.2K/night
* Transport: Airport pickup + scooter
* Activities: Ubud tour, diving, beach

**Step 10: User Response**

**User:**

Great! Can you book the hotel and flights for me?

✅ Orchestrator routes:

* Booking Agent with flight\_quote\_324.json
* Booking Agent with hotel\_quotes\_bali.json

**Step 11: Booking Agent completes transaction**

📤 Returns: JSON

{ "flight\_status": "Booked", "hotel\_status": "Booked", "booking\_refs": { "flight": "AI5678", "hotel": "SEM12345" } }

✅ Summary of Data Flow (Simplified)

graph TD User -->|Intent: Plan Trip| Orchestrator Orchestrator --> DA[Destination Advisor Agent] Orchestrator --> BA[Budget Agent] Orchestrator --> FA[Flight Agent] Orchestrator --> HA[Hotel Agent] Orchestrator --> TA[Transport Agent] Orchestrator --> AA[Activities Agent] DA -->|Visa, Weather| Orchestrator BA -->|Budget Estimate| Orchestrator FA -->|Flight Options| Orchestrator HA -->|Hotel Options| Orchestrator TA -->|Local Transport| Orchestrator AA -->|Things To Do| Orchestrator Orchestrator -->|Final Itinerary| User User -->|Book Now| Orchestrator Orchestrator --> BookingAgent BookingAgent --> Orchestrator --> User

🔍 Highlights

* All agents are **stateless**, but **task-aware**.
* **Artifacts** carry structured outputs (JSON or flat files).
* Orchestrator acts as a **conversation manager and router**.
* Each agent only handles **domain-specific tasks** and tool calls.

**Flowchart**   
  
A[User Initiates: "Plan a Bali trip in November"] --> B[Orchestrator Agent]

B --> C1[Destination Advisor Agent]

B --> C2[Budget & Cost Agent]

B --> C3[Flight Agent]

B --> C4[Accommodation Agent]

B --> C5[Transport Agent]

B --> C6[Activities Agent]

C1 --> D1[Weather, Visa, Culture Info]

C2 --> D2[Budget Estimate]

C3 --> D3[Flight Options]

C4 --> D4[Hotel Options]

C5 --> D5[Local Transport]

C6 --> D6[Excursions & Tours]

D1 --> E[Orchestrator Agent]

D2 --> E[Orchestrator Agent]

D3 --> E[Orchestrator Agent]

D4 --> E[Orchestrator Agent]

D5 --> E[Orchestrator Agent]

D6 --> E[Orchestrator Agent]

E --> F[Itinerary Compilation & Summary to User]

F --> G[User Request: "Book my trip"]

G --> H[Booking Agent (Flights + Hotel)]

H --> I[Booking Confirmations with References]

I --> J[Final Confirmation to User]

**🧠 Agent Implementation Approaches**

**1. 🏠 In-House with Ollama (Fine-tuned Open Models)**

* **How it works**: Fine-tune open-source LLMs (e.g. llama‑3, deepseek‑r1) using LoRA/adapter-style methods and deploy them via Ollama. Agents are trained on travel-specific conversational data (e.g. itinerary planning, booking tasks).
* **Use cases**: Agents with fixed knowledge of your domain—advice, estimation, templated responses (e.g. packing lists, culture tips).
* **Cost/per-user**: After fine‑tuning (one-time), running on CPU or dedicated GPU—cost ~ $0.001–$0.005 per query (electricity, VPS) ● zero per-token API fees [Reddit+2cohorte.co+2Hitesh AI+2](https://www.cohorte.co/blog/run-llms-locally-with-ollama-privacy-first-ai-for-developers-in-2025?utm_source=chatgpt.com).
* **Latency**: ~0.5–3 s depending on hardware; quantization or GPU yields faster (<1 s). On standard hardware, bigger models are still < 5 s [Medium](https://medium.com/%40iitbguha/breaking-the-latency-barrier-a-deep-dive-into-ai-agent-optimization-5cb15cec5dfa?utm_source=chatgpt.com)[cohorte.co](https://www.cohorte.co/blog/run-llms-locally-with-ollama-privacy-first-ai-for-developers-in-2025?utm_source=chatgpt.com).
* **Complexity**: High initial setup: generating synthetic data, training, hosting, update pipelines.

**2. 🌐 Search + Generic LLM (via search APIs + LLM summarizer)**

* **How it works**: Use search APIs (Brave, Exa, Tavily) to retrieve trip-specific dynamic data—weather, visa, local news—and feed into a generic LLM (e.g. GPT‑3.5, Llama) to summarize and respond.
* **Use cases**: Well-suited for Destination Agent, Activities Agent, Safety & Visa, Cultural tips.
* **Cost/per-user**: Search ~ $0.003–$0.007 per query, plus LLM token cost ($0.002–$0.01 per request). Total ≈ $0.01–$0.05 per agent call [Reddit+2Reddit+2Reddit+2](https://www.reddit.com/r/perplexity_ai/comments/1h25hq6?utm_source=chatgpt.com).
* **Latency**: Search latency + model summarization: typically 2–5 s. If streaming summarization, perceived much faster [Reddit+2arXiv+2Reddit+2](https://arxiv.org/abs/2303.16199?utm_source=chatgpt.com).
* **Complexity**: Moderate: implement search integration, caching, tool prompting. Lower training burden but building robust pipelines still non-trivial.

**3. 🏢 Native APIs (OpenAI GPT‑4/GPT‑4o, Claude, Grok)**

* **How it works**: Leverage OpenAI or Claude or Grok endpoints, optionally using browser-enabled or retrieval-enhanced (RAG) flows.
* **Use cases**: Agents requiring deep reasoning, itinerary optimization, conversation routing, booking instructions.
* **Cost/per-user**: GPT‑4o is ~$0.03 per 1K tokens; complex agent sessions might use several thousand tokens → $0.05–$0.15/user. Grok costs similarly or slightly higher; costs unclear but generally higher latency [Reddit](https://www.reddit.com/r/perplexity_ai/comments/1jhzbpj?utm_source=chatgpt.com)[Medium](https://medium.com/%40iitbguha/breaking-the-latency-barrier-a-deep-dive-into-ai-agent-optimization-5cb15cec5dfa?utm_source=chatgpt.com).
* **Latency**: GPT‑4-based APIs average 30–60 s for multi-step agents; Grok may be slower but optimized for search-type tasks [Medium](https://medium.com/%40iitbguha/breaking-the-latency-barrier-a-deep-dive-into-ai-agent-optimization-5cb15cec5dfa?utm_source=chatgpt.com)[Reddit](https://www.reddit.com/r/LLMDevs/comments/1c9l0vy?utm_source=chatgpt.com).
* **Complexity**: Low to medium: uses hosted LLM, minimal infrastructure; need prompt engineering and orchestration logic.

**4. 🔍 Perplexity (Web search + LLM in one)**

* **How it works**: Single API that integrates web search, retrieval, summarization, citations using proprietary Sonar/Llama‑style models.
* **Use cases**: Destination, Safety, Activities, Cultural context—any agent requiring up-to-date real-world info.
* **Cost/per-user**: Expensive—each call may include many citations and charges per retrieval + tokens. Real cost often ~$0.10–$0.15 per simple query, can scale quickly [Reddit+8Reddit+8Reddit+8](https://www.reddit.com/r/perplexity_ai/comments/1h25hq6?utm_source=chatgpt.com)[Reddit+3Reddit+3en.wikipedia.org+3](https://www.reddit.com/r/perplexity_ai/comments/1jbky3f?utm_source=chatgpt.com)[Medium](https://medium.com/%40iitbguha/breaking-the-latency-barrier-a-deep-dive-into-ai-agent-optimization-5cb15cec5dfa?utm_source=chatgpt.com).
* **Latency**: Fast initial latency due to optimization, but overall multistep summaries can take multiple seconds; often higher than search+LLM because of citation overhead [perplexity.ai](https://www.perplexity.ai/hr/hub/blog/introducing-pplx-api?utm_source=chatgpt.com)[Reddit](https://www.reddit.com/r/LLMDevs/comments/1c9l0vy?utm_source=chatgpt.com).
* **Complexity**: Low—simple integration, streaming support; but opaque pricing and reliability concerns exist [Reddit](https://www.reddit.com/r/perplexity_ai/comments/1jma950?utm_source=chatgpt.com).

**🧩 Summary Table**

| **Agent Type** | **In-House (Ollama)** | **Search + Generic LLM** | **OpenAPI GPT/Grok/Claude** | **Perplexity API** |
| --- | --- | --- | --- | --- |
| **Best Suited For** | Domain-specific static tasks | Dynamic real-world data | Complex reasoning, itinerary flow | Info-heavy with citations |
| **Per-user Cost** | ~$0.001–0.005 (electricity) | ~$0.01–0.05 | ~$0.05–0.15 | ~$0.10–0.20+ |
| **Latency (avg)** | 0.5–3 s | 2–5 s | 10–60 s | ~2–6 s |
| **Requires Training** | ✅ High | ⚠️ Moderate | ❌ Minimal | ❌ Minimal |
| **Maintenance Complexity** | ❗ High | ⚠️ Moderate | ⚠️ Prompt engineering | ✅ Integration only but opaque |
| **API Reliability** | ✅ Fully controllable | 🟡 Dependent on provider | 🟡&✅ Most stable | ⚠️ Mixed—some reliability issues |