# The Weekend Trip Planner Agent (Concierge Track Submission)

This project, developed for the Kaggle Agents Intensive Capstone, implements a **Concierge Agent** designed to automate and streamline the often complex process of planning a personalized weekend getaway.

The agent uses a **sequential multi-agent system** powered by the Gemini model to analyze user preferences, retrieve real-time data, and generate a polished, actionable travel plan.

## 🎯 Project Goal & Value

Problem: Planning a trip requires juggling research (destination, activities, weather) and organizing preferences, which is time-consuming and often overwhelming.

Solution: The Weekend Trip Planner Agent provides a single, conversational interface to handle the entire planning workflow, from ideation to final recommendation.

Value: Saves the user time and reduces decision fatigue by providing context-aware, relevant suggestions in a single turn.

## 🔑 Key Concepts Implemented (Required: 3 Concepts)

This agent successfully integrates the following three core concepts from the Agents Intensive course:

### 1. Multi-Agent System: Sequential LLM-Powered Agents

The core logic is structured as a sequential pipeline, where different LLM-powered agents handle distinct responsibilities:

| **Agent Role** | **Function** | **Technology** |
| --- | --- | --- |
| **Planner Agent (Orchestrator)** | Analyzes the session state, user history, and current plan stage. It determines the next required action (e.g., search\_trip\_ideas, refine\_ideas, or finalize\_trip). | LLM (Gemini) with Structured JSON Output |
| **Search Tool** | Executes the necessary external function to retrieve raw, up-to-date data. | Built-in Tool (Mocked WebSearchTool) |
| **Formatter Agent** | Takes the raw, unformatted output from the Search Tool and transforms it into a polite, professional, and user-friendly suggestion. | LLM (Gemini) |

### 2. Tools: Built-in Web Search (Mocked for Execution)

The agent integrates a dedicated tool to gather external, real-world data, which is essential for travel planning.

* **Implementation:** The WebSearchTool class is defined to simulate a live search, demonstrating the architecture required to fetch current information like destination ideas, activity details, or weather forecasts.
* **Usage:** The Planner Agent invokes this tool via the search\_trip\_ideas action, ensuring the recommendations are timely and relevant.

### 3. Sessions & State Management

Conversational commerce and planning requires memory. The agent maintains context across multiple user turns.

* **Implementation:** The TripPlanningSession class manages the state for a single user interaction (identified by a session\_id).
* **Data Stored:**
  + conversation\_history: The full log of user and assistant messages.
  + user\_preferences: Explicitly tracks preferences like "hiking," "historical sites," or "coastal," which inform future planning decisions.
  + current\_plan\_stage: A categorical state (e.g., initial, gathering\_preferences, presenting\_ideas) used by the Planner Agent to avoid asking repetitive questions and guide the conversation flow.

## ⚙️ Setup and Execution

### Prerequisites

* Python 3.8+
* The google-genai library (or similar library for your chosen LLM).
* A Gemini API Key.

### Local Setup

1. **Clone this repository:**  
   git clone [YOUR-REPO-URL]  
   cd [YOUR-REPO-NAME]
2. **Set Environment Variable:**  
   export GEMINI\_API\_KEY="YOUR\_API\_KEY\_HERE"  
     
   *(Remember: Do NOT hardcode this key in the file!)*
3. **Run the Agent:**  
   python agent\_main.py

The if \_\_name\_\_ == "\_\_main\_\_": block runs a three-turn test conversation to demonstrate the sequential planning and state management.