**Event\_finder\_tool.py**

**Purpose**: This tool is designed to find local events in a specific location within a given date range. It can also filter events by type (e.g., concert, sports, festival). Its main goal is to help an agent discover what's happening in a destination for travel planning purposes.

* **Input**:
  + location (string): The city or area to search for events.
  + start\_date (string): The start date of the search period in 'YYYY-MM-DD' format.
  + end\_date (string): The end date of the search period in 'YYYY-MM-DD' format.
  + event\_type (string, optional): A category to filter the events, such as 'concert', 'sports', or 'festival'.
* **Output**:
  + A **JSON string** containing a list of the top 20 events. Each event in the list is a dictionary with the following keys:
    - title: The name of the event.
    - date: The date of the event.
    - venue: The name of the venue.
    - link: A URL for more information.
  + If an error occurs, the output will be a string describing the error.

**destination\_info\_tool.py**

**Purpose**: This tool performs a two-step web search and extraction process to gather detailed information about a destination. It first searches for relevant web pages and then scrapes the most important content from them. This is useful for getting a comprehensive overview of a place, including things to do, attractions, and general facts.

* **Input**:
  + query (string): The search query to find information about the destination (e.g., 'best things to do in Paris').
  + num\_results (integer, optional): The number of top results to extract content from. The default is 5.
* **Output**:
  + A **JSON string** that is an array of objects. Each object represents a web page and includes:
    - url: The URL of the page.
    - content: The full, extracted text content of the page.
  + If no relevant URLs are found, it returns a JSON with {"status": "no\_relevant\_urls\_found"}.
  + If an error occurs, it returns a JSON with an {"error": ...} message.

**Flight\_prices\_tool.py**

**Purpose**: This tool finds flight prices and itineraries for round-trip flights between specified source and destination airports. It provides key details like price, flight duration, and a human-readable summary, which helps in planning travel budgets.

* **Input**:
  + source (string): The departure city or airport code (e.g., 'City:warsaw\_pl').
  + destination (string): The arrival city or airport code (e.g., 'City:dubrovnik\_hr').
  + adults (integer): The number of adult passengers.
  + currency (string): The preferred currency for the prices (e.g., 'USD', 'EUR').
  + outboundDepartureDateStart (string): The earliest departure date for the outbound flight in 'YYYY-MM-DDTHH:MM:SS' format.
  + outboundDepartureDateEnd (string): The latest departure date for the outbound flight in 'YYYY-MM-DDTHH:MM:SS' format.
  + inboundDepartureDateStart (string): The earliest departure date for the inbound flight in 'YYYY-MM-DDTHH:MM:SS' format.
  + inboundDepartureDateEnd (string): The latest departure date for the inbound flight in 'YYYY-MM-DDTHH:MM:SS' format.
* **Output**:
  + A **JSON object** with a list of flight itineraries. Each itinerary includes:
    - priceUSD: The price in US dollars.
    - priceEUR: The price in euros.
    - durationOutbound: The duration of the outbound flight.
    - durationInbound: The duration of the inbound flight.
    - lastAvailableSeats: The number of seats left.
    - outbound: A detailed summary of the outbound flight.
    - inbound: A detailed summary of the inbound flight.
    - bookingUrl: A direct link to book the flight.
    - human\_readable\_summary: A brief, easy-to-read summary of the flight and its cost.
  + If an error occurs, the output will be a dictionary with an {"error": ...} message.

**Hotel\_prices\_tool.py**

**Purpose**: This tool is designed to search for hotels in a specified city or region using Google Hotels. It provides detailed information about available hotels, including pricing, ratings, location, and a human-readable summary. This is crucial for agents helping a user plan accommodation for their trip.

* **Input**:
  + query (string): The city, region, or specific hotel name to search for.
  + check\_in\_date (string): The check-in date in 'YYYY-MM-DD' format.
  + check\_out\_date (string): The check-out date in 'YYYY-MM-DD' format.
  + adults (integer, optional): The number of adults for the booking. Defaults to 2.
  + children (integer, optional): The number of children. Defaults to 0.
  + sort\_by (string, optional): A parameter to sort the results (e.g., 'PRICE\_ASC').
* **Output**:
  + A **JSON object** containing search results. The object includes:
    - query: The original search query.
    - check\_in: The check-in date.
    - check\_out: The check-out date.
    - total\_found: The total number of hotels found.
    - hotels: A list of simplified hotel objects. Each hotel object contains:
      * name: The hotel's name.
      * description: A brief description.
      * rating: The overall rating.
      * hotel\_class: A star rating (e.g., 5-star).
      * price: A nested object with per-night and total price details.
      * address: A nested object with latitude and longitude.
      * summary: A human-readable summary of the hotel, its rating, and price.
  + If an error occurs, it returns a JSON with an {"error": ...} message.

**Weather\_Forecast\_tool.py**

**Purpose**: This tool retrieves the daily weather forecast for a given location for a specified number of days. It provides key weather metrics and a summary, which is vital for an agent helping a user prepare for their trip by checking the local weather.

* **Input**:
  + location (string): The city or location to get the weather forecast for.
  + days (integer, optional): The number of days for the forecast. Defaults to 5.
  + units (string, optional): The unit system for temperature. 'metric' (Celsius) or 'imperial' (Fahrenheit). Defaults to 'metric'.
* **Output**:
  + A **JSON object** with a list of daily forecasts and a human-readable summary. The object includes:
    - forecasts: A list of dictionaries, where each dictionary contains the following for a specific day:
      * date: The date of the forecast.
      * condition: A detailed description of the weather (e.g., 'light rain', 'clear sky').
      * temp\_high: The highest temperature.
      * temp\_low: The lowest temperature.
      * wind\_speed: The wind speed.
      * humidity: The humidity percentage.
      * precipitation: The amount of precipitation.
    - human\_readable\_summary: A combined text summary of the forecast for all the days.
  + If an error occurs, the output will be a string describing the error.

**Make\_quotation\_tool.py**

**Purpose**: This tool calculates a total estimated vacation cost by combining several factors: the average cost of hotels, the average cost of flights, and an estimated daily spend on food and attractions. It determines the daily spend by querying an LLM (Large Language Model) internally. The final output includes a breakdown of these costs.

* **Input**:
  + hotel\_prices (List

float

): A list of hotel prices per night.

* + flight\_prices (List

float

): A list of round-trip flight prices.

* + start\_date (string): The start date of the vacation in 'YYYY-MM-DD' format.
  + end\_date (string): The end date of the vacation in 'YYYY-MM-DD' format.
  + destination (string): The destination city or location of the vacation.
* **Output**:
  + A **dictionary** containing a detailed cost breakdown. The dictionary includes:
    - days: The total number of days of the vacation.
    - hotel\_total: The total estimated cost of hotels for the duration of the trip.
    - flight\_avg: The average price of the flights.
    - daily\_cost\_estimate: The estimated daily cost for a tourist's food and attractions, provided by the LLM.
    - daily\_total: The total estimated cost for food and attractions for the entire trip.
    - total\_estimate: The final estimated total cost of the vacation (sum of hotel\_total, flight\_avg, and daily\_total).

ok now lets edit the prompts for the agents

manager:

      llm: gpt-4o-mini

      prompt\_config:

        role: Vacation Manager & Orchestrator

        instruction: |

          You are the central manager of the VacayMate system.

          Your job is to:

          1. Receive user input (current location, vacation destination, date range).

          2. Validate the input and extract structured details.

          3. Route tasks to the appropriate agents (Researcher, Calculator, Planner, Summarizer).

          4. Collect results and ensure workflow completes in the correct sequence.

          ⚠️ Never perform calculations or planning yourself — always delegate.

        output\_constraints:

          - Return structured, validated input with keys: current\_location, destination, date\_range

          - Only delegate tasks, do not attempt to complete them

        goal: Orchestrate the full vacation planning workflow

    researcher:

      llm: gpt-4o-mini

      tools:

        - google\_search\_api

        - travel\_price\_checker

        - local\_event\_finder

        - hotel\_booking\_api

      prompt\_config:

        role: Data Researcher for vacation planning

        instruction: |

          Collect raw data for the given destination and date range:

          - Flight options & prices

          - Hotel availability & pricing

          - Local activities and attractions

          - Local events & restaurants

          Return results in structured JSON format.

        output\_constraints:

          - Organize data into flights, hotels, activities, and events

          - Ensure at least 3 options per category (if available)

        goal: Gather all raw data necessary for planning and cost estimation

    calculator:

      llm: gpt-4o-mini

      tools:

        - simple\_calculator

      prompt\_config:

        role: Financial Calculator

        instruction: |

          Using the cost data provided (flights, hotels, activities), calculate:

          - Total estimated trip cost

          - Per-person breakdown

          - Daily budget allocation

          Include assumptions if necessary.

        output\_constraints:

          - Provide total, per-person, and daily costs in structured format

          - Ensure math accuracy using calculator tool

        goal: Generate a precise vacation quotation

    planner:

      llm: gpt-4o-mini

      tools:

        - weather\_api

      prompt\_config:

        role: Itinerary Planner

        instruction: |

          Design a day-by-day itinerary for the vacation using the activity and event data provided.

          Consider:

          - Location proximity (avoid unnecessary travel)

          - Weather forecasts

          - Logical grouping of activities

          - Variety (mix of cultural, leisure, dining, and events)

        output\_constraints:

          - Provide a structured day-by-day itinerary

          - Each day must include at least one main activity and optional extras

        goal: Produce a realistic, enjoyable day-by-day vacation plan

    summarizer:

      llm: gpt-4o-mini

      prompt\_config:

        role: Vacation Summarizer & Presenter

        instruction: |

          Combine the quotation (from Calculator) and itinerary (from Planner) into one polished output.

          The output must include:

          - A cost summary section

          - A detailed daily itinerary section

          - A friendly conclusion

        output\_constraints:

          - Output should be user-friendly and well-formatted (Markdown or rich text)

          - Do not lose details from the quotation or itinerary

        goal: Deliver the final vacation plan in a clear and engaging way

the researcher uses 3 tools

1. destination\_info\_tool.py

2. Flight\_prices\_tool.py

3. Hotel\_prices\_tool.py

the planner uses 2 tools

1. Weather\_Forecast\_tool.py

2. Event\_finder\_tool.py

the calculator uses 1 tool

1. Make\_quotation\_tool.py

the manager manages all the data transfer and tasks between the agents