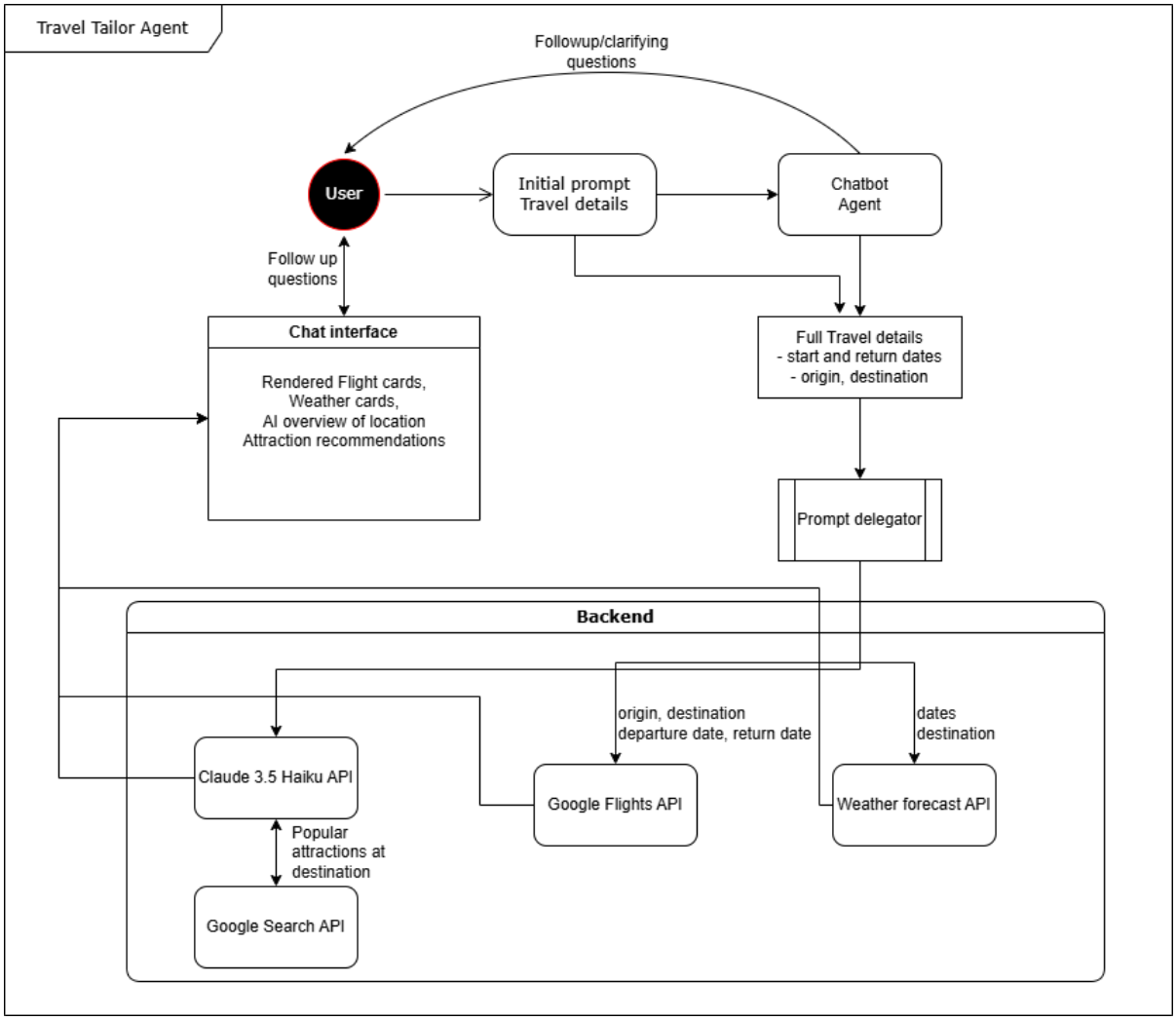
## 5.1 Introduction

Travel Tailor Agent is a personalized travel itinerary agent designed to help users plan trips from idea to execution within a unified interface. Users interact with the agent via a chat interface, starting with an initial prompt including destination, departure city, and dates. The chatbot parses the natural language query, asking clarifying questions until all travel details are obtained. This data is sent to the backend, where tools like Google Flights, OpenMeteo weather API, and Google Search API are utilized. The results are displayed in the chat interface, featuring 5 options for both departing and returning flights, the weather forecast for the travel period, an AI overview of typical weather patterns for that time of year, and a list of popular local attractions. Users can then ask follow-up questions and continue the conversation as they would with an experienced travel agent.




## 5.2 Functional Requirements:

The functional requirements of this project include what the system is capable of doing from a user's perspective and the internal processes required to support these interactions.

5.2.1 Natural Language Understanding and Query Parsing

This functionality covers how the system receives user input, interprets it, and extracts the necessary travel details, prompting for clarification when needed.

- The system shall provide a text-based chat interface for the user to input queries relating to their travels, and parse the natural language user input to identify key pieces of information like: destination, origin, and travel dates.
- The system shall identify missing required travel details from the initial user query and ask clarifying questions to get these pieces of information.
- The system shall update the internal representation of the travel details as the user provides more information.

5.2.2 Tool Integration and Data Retrieval

This functionality describes the system's ability to connect with external services to fetch relevant travel information based on the parsed user request.

- Flights
    - The system shall utilize the Google Flights API to search for available flight options based on the user's specified departure city, destination, and dates.
- Weather Forecast and AI overview
    - The system shall utilize the OpenMeteo weather API to retrieve the weather forecast for the destination during the specified travel dates.
    - The system shall utilize the Google Search API along with the Claude 3.5 Haiku model to generate an AI overview of the weather at the location during the dates travelled.
- Attractions
    - The system shall utilize the Google Search API to find information about popular attractions in the destination city.
- The system shall handle successful data retrieval from all integrated APIs.
- The system shall handle errors and exceptions that may occur during API calls, such as network issues or invalid requests.

5.2.3 Personalized Itinerary Generation and Display

This functionality defines how the retrieved information and generated content are presented to the user within the chat interface in a clear and organized manner.

- The system shall display the retrieved departing & returning flight options within the chat interface in flight cards, including key details such as airline, departure/arrival times, layover(s), duration, and price.
- The system shall display the weather forecast for the specified travel dates in the destination.
- The system shall display an AI-generated overview of typical weather patterns for the destination around the specified time of year.
- The system shall display a list of popular attractions in the destination city.
- The system shall format the displayed information clearly and legibly within the chat interface.

5.2.4 Conversational Follow up and Information Retrieval

This functionality describes the agent's ability to maintain context and respond to subsequent user queries about the destination or presented information, simulating a real travel agent.

- The system shall maintain the context of the user's travel request and previous interactions throughout the conversation.
- The system shall allow the user to ask follow-up questions about the destination or the information presented.
- The system shall generate responses to follow-up questions based on the gathered information or by making additional tool calls if necessary.
- The system shall simulate the conversational style of a helpful and knowledgeable travel agent.

## 5.3 Performance Requirements

This section specifies the constraints on the speed, efficiency, and responsiveness of Travel Tailor Agent to ensure a satisfactory user experience.

5.3.1 Response Time for Queries to Chatbot

This requirement specifies the maximum time allowed for the system to process user queries and present the first set of consolidated travel information (flights, weather, attractions).

- The system shall process the initial user query and display the initial set of travel information within 5 seconds under normal operating conditions.
- The system shall respond to follow up questions in >1 second if tool use is not required, and up to 2 or 3 seconds if one is required.

5.3.2 API Call Latency

This requirement specifies the system's ability to handle variations in the response times of integrated external APIs without causing excessive delays for the user.

- The system shall manage concurrent API calls efficiently to minimize the overall user waiting time, even if individual API calls experience moderate latency (up to 5 seconds per call).

## 5.4 Environment Requirements

This section lists the necessary hardware, software, and other resources required for the development and deployment of Travel Tailor Agent.

5.4.1 Development Environment Requirements

This subsection details the environment needed by a developer to build, test, and maintain the system.

- Programming languages: Python 3.11, Typescript
- Frameworks/Libraries: FastAPI, uvicorn, Pydantic, Claude, NextJS, shadcn/ui, TailwindCSS
- Exernal API Keys: Anthropic API key, SerpAPI key (for Google Flights and Search API), OpenMeteo API key

5.4.2 Execution Environment Requirements

This section details the environment where the developed system will be deployed and run to be accessible to users.

- Hosting: A cloud platform like AWS, Google Cloud, or Azure, or a virtual private server
- Operating System: Server-grade Linux distribution
- Web Server: uvicorn and FastAPI based server can run on a VPS