

Cycling Trip Planner Agent

Overview

Build an AI agent that helps a cyclist plan a multi-day bike trip through conversation.

Example user input:

"I want to cycle from Amsterdam to Copenhagen. I can do around 100km a day, prefer camping but want a hostel every 4th night. Traveling in June."

What the agent should do:

1. Understand the request
2. Ask clarifying questions if needed
3. Use tools to get route, accommodation, and weather data
4. Break the trip into daily segments
5. Present a day-by-day plan
6. Adjust if the user changes preferences

API Provided

Anthropic Claude API key (provided separately)

Required Tools

You must implement the following tools. Mock data is fine — we're testing architecture, not API integration.

Tool	Purpose
get_route	Get cycling route between two points — distance, estimated days, waypoints
find_accommodation	Find places to stay near a location — camping, hostels, hotels
get_weather	Get typical weather for a location and month
get_elevation_profile	Get terrain difficulty — elevation gain, difficulty rating

Optional tools (bonus):

- get_points_of_interest
- check_visa_requirements
- estimate_budget

Technical Requirements

Stack:

Requirement	Details
Language	Python 3.10+
Framework	FastAPI
LLM	Anthropic Claude
Data validation	Pydantic

What to build:

1. **Chat API endpoint** (POST /chat) with conversation state
2. **Tool system** — properly defined and callable by agent
3. **Agent orchestration** — decides which tools to call and when

Project structure:

```
/src
/agent - agent logic, prompts, orchestration
/tools - tool definitions
/api - FastAPI routes
/tests - at least basic tests for tools
README.md - setup instructions, design decisions
```

Deliverables

1. **Public GitHub repository**
2. **README** with:
 - How to run locally
 - Architecture decisions (1 page max)
 - What you would develop with more time
3. **Screen recording** showing a full conversation from start to finished trip plan

Evaluation Criteria

Criteria	Weight
Agent architecture — Clean separation of agent logic, tools, and API. Easy to understand and extend.	25%
Tool design — Tools are well-defined, properly typed, reusable. Agent calls them correctly.	20%
Multi-step reasoning — Agent plans across steps, not just single tool call and done.	20%
Conversation handling — Agent maintains context, asks clarifying questions, adapts when user changes preferences.	15%
Code quality — Readable, typed, proper error handling, follows Python best practices.	10%
Product thinking — Would a real cyclist find this useful? Does the output make sense?	10%

Submission

What to include in your email:

- GitHub repo link
- Screen recording link (in repo, Google Drive, or Loom)

Good luck! We're excited to see what you build.