

## Executive Summary

The goal of this project was to extend the capabilities of a local Large Language Model (LLM) to include geospatial awareness. By leveraging the **Open WebUI Tools API** and **Google Maps API**, the LLM can now interpret location based queries and return embedded maps and navigation routes directly in the chat interface.

---

## Technical Architecture

The system follows a modular "Tool" architecture within Open WebUI.

- **Frontend/Interface:** Open WebUI.
  - **Intelligence:** Local LLM (e.g., Llama 3, qwen3).
  - **Backend Logic:** Python-based Tool Open WebUI.
  - **Geospatial Provider:** Google Maps Static API.
- 

## Installation & Setup

### 1. Google Cloud Configuration

To ensure usage limits and security, follow these steps:

1. Log in to the [Google Cloud Console](#).
2. Create a new project (to utilize the \$200 monthly free credit).
3. **Enable APIs:** Search for and enable the **Maps Static API**.
4. **Credentials:** Create an **API Key**.
  - *Security Best Practice:* Under "API Restrictions," restrict this key to only the "Maps Static API" to prevent unauthorized usage of other services.

### 2. WebUI Installation

1. Run the following command on WSL 2 or Ubuntu:

```
docker run -d -p 3000:8080 --gpus=all -v ollama:/root/.ollama -v  
open-webui:/app/backend/data --name open-webui --restart always
```

`ghcr.io/open-webui/open-webui:ollama`

2. Alternatively, switch to a specific version:

`ghcr.io/open-webui/open-webui:v0.8.0`

`ghcr.io/open-webui/open-webui:v0.8.0-ollama`

`ghcr.io/open-webui/open-webui:v0.8.0-cuda`

### 3. Open WebUI Implementation

3. Open your **Open WebUI** dashboard.
  4. Navigate to **Workspace > Tools > Create a New Tool**.
  5. **Paste the Code:** Copy the provided Python script into the editor.

Tools • Open WebUI

localhost:3000/workspace/tools/edit?id=maptools

New Chat

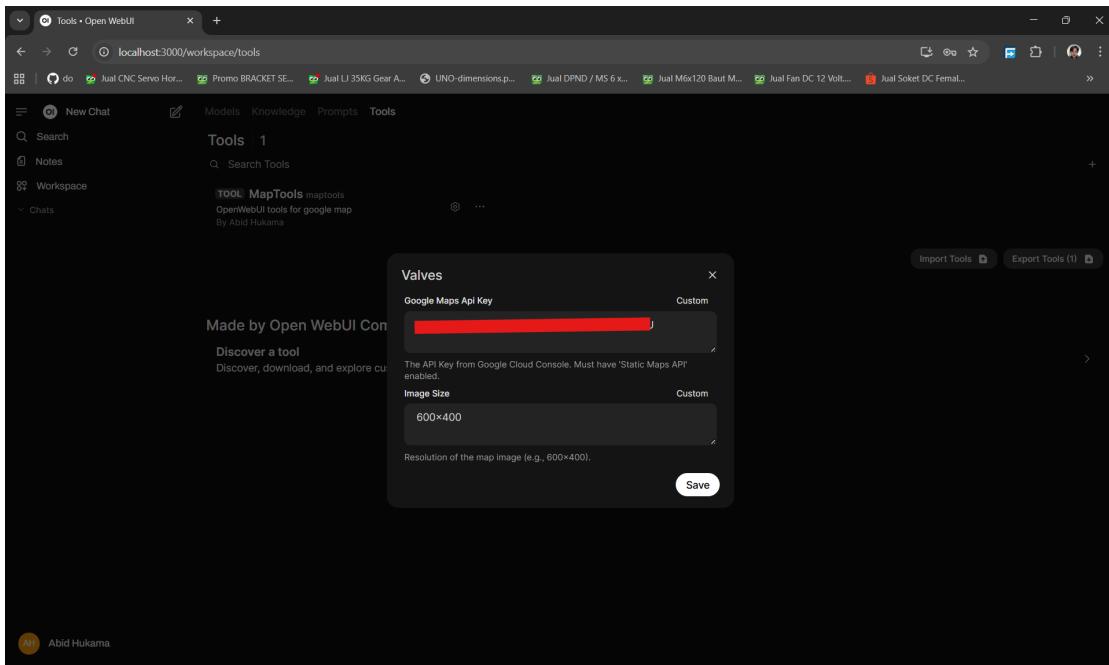
Models Knowledge Prompts Tools

MapTools

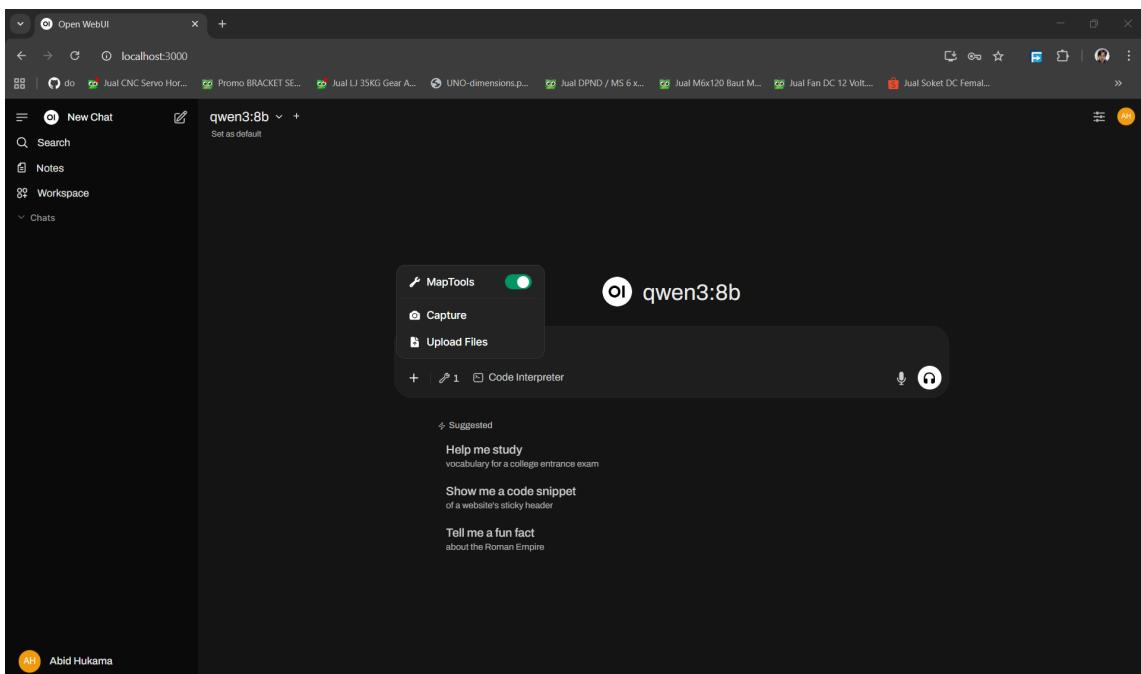
maptools OpenWebUI tools for google map

```
1 import urllib.parse
2 from pydantic import BaseModel, Field
3 from typing import Optional, List
4
5
6 v class Tools:
7     # VALVE CONFIGURATION
8 v     class Valves(BaseModel):
9         GOOGLE_MAPS_API_KEY: str = Field(
10             default="",
11             description="The API Key from Google Cloud Console. Must have 'Static Maps API' enabled."
12         )
13         IMAGE_SIZE: str = Field(
14             default="0x0x400",
15             description="Resolution of the map image (e.g., 600x400)."
16         )
17
18 v     def __init__(self):
19         self.valves = self.Valves()
20         self.base_url = "https://maps.googleapis.com/maps/api/staticmap"
21
22 v     def find_places_and_map(self, query: str) -> str:
23         """
24             Search for a place and return a static image of the location.
25         """
26 v         if not self.valves.GOOGLE_MAPS_API_KEY:
27             return "Error: Google Maps API key is missing."
28
29         safe_query = urllib.parse.quote(query)
30
31         # map_url for a single location with a marker
32         map_url = (
```

6. Configuration (Valves):
    - Save the tool.
    - Click on the **Settings/Gear icon** for the tool.
    - Enter your Maps API KEY in the Valve configuration field.



## 5. Make sure enable Maps Tools Before Chat:





## How It Works

The implementation utilizes a class-based structure that Open WebUI interprets as callable functions for the LLM.

### Logic Flow:

1. **Intent Recognition:** When a user asks "Where is the Eiffel Tower?" the LLM recognizes it needs a map and triggers the `find_places_and_map` function.
2. **Input Sanitization:** The tool uses `urllib.parse.quote(query)` to ensure the user's string is safe for URL transmission.
3. **Dynamic URL Generation:** The script constructs a request to Google's Static Maps API, embedding markers and paths.
4. **Markdown Rendering:** The tool returns a Markdown string containing:
  - o A header with the location name.
  - o An embedded image (the static map).
  - o A deep link to the full Google Maps site for navigation.



## Usage Examples

### Single Location Search

User: "Can you show me where the Space Needle is?"

LLM Output:

Thought for 3 seconds ↴

The Space Needle is located in Seattle, Washington, USA. Here's a map to help you find it:

📍 Map of the Space Needle

🔗 Open in Google Maps for real-time directions and details.

The Space Needle is an iconic landmark in Seattle, best known for its observation deck offering panoramic views of the city and Puget Sound. Enjoy your visit! 🌟

TOOL:maptools/find places and map

Send a Message

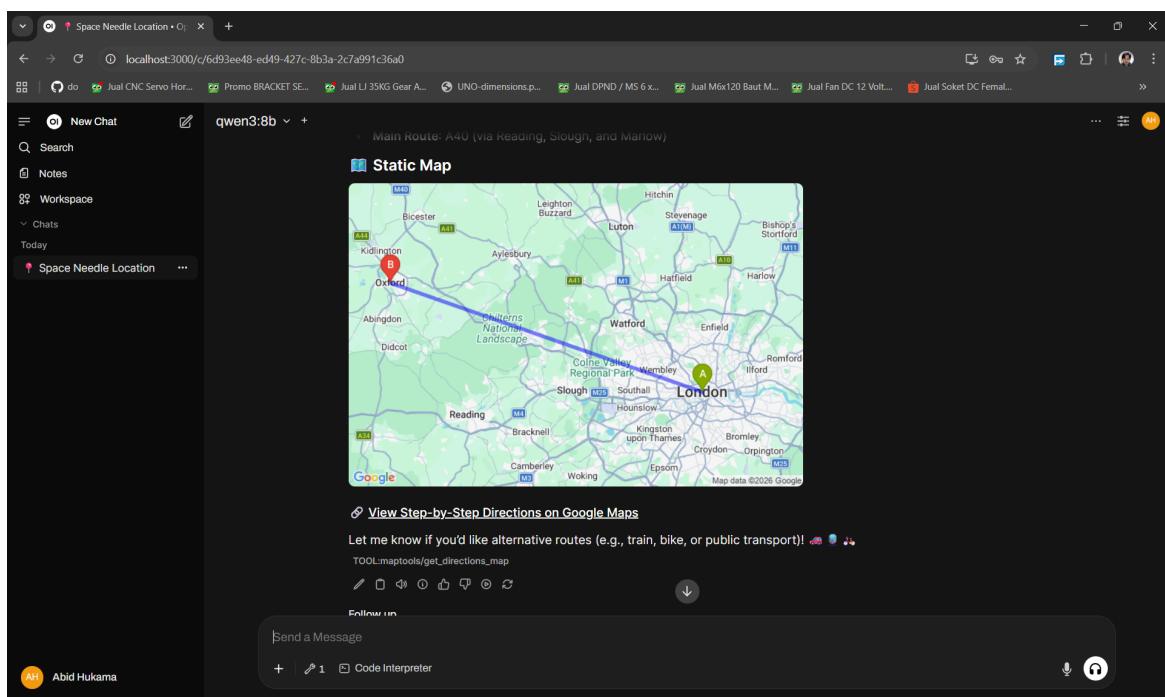
+ ⏪ 1 Code Interpreter

Abid Hukama

## Route & Directions

User: "Show me the way from London to Oxford."

LLM Output:



## Security & Constraints

- **Access Security:** The use of "Valves" ensures that the API key is stored as an environment-level configuration rather than being hardcoded, protecting it from being leaked in the prompt context.
- **Usage Limits:** The Static Maps API is cost-effective; however, it is recommended to set billing alerts in the Google Cloud Console to monitor the free credit usage.
- **Assumptions:** It is assumed the user has an active internet connection to fetch the map images, even if the LLM is running locally.



## Code Maintenance & Best Practices

The current implementation uses `urllib.parse` which is part of the Python standard library, minimizing external dependencies and ensuring a lightweight footprint. For future iterations, consider:

- **Dynamic Zoom:** Allowing the LLM to adjust the zoom level based on the size of the city/area.
- **Cache:** Implementing a local cache for frequently requested maps to reduce API calls.
- **Embedded HTML:** We recommend using an embedded HTML Google Map instead of a static Google Map image. However, Open WebUI currently does not directly support embedded HTML, so a static image is used for this project.
- **Pipelines:** For larger or more complex projects, we recommend using a pipelines approach instead of tools within Open WebUI.