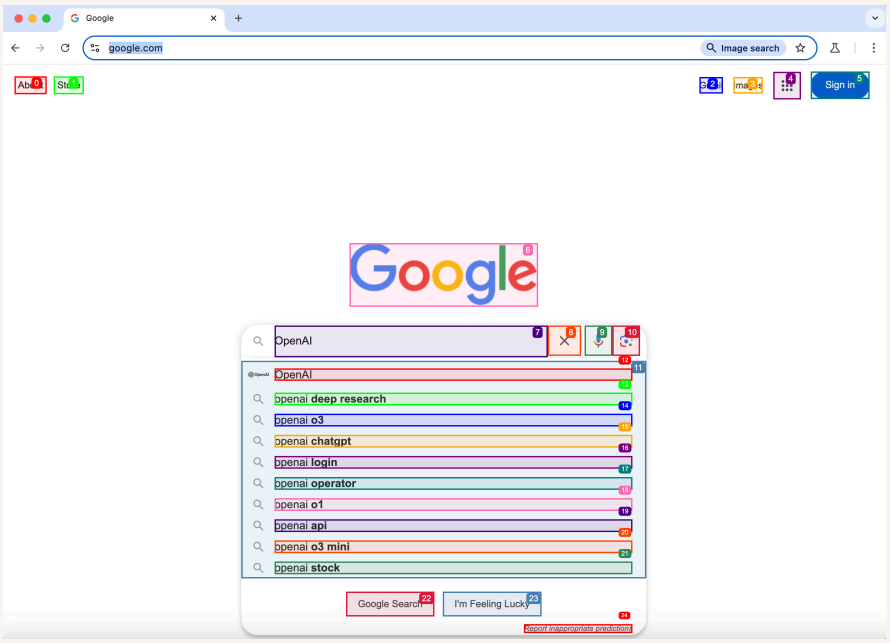
NextWork.org

# Automate Your Browser with AI Agents

Natasha Ong



**Natasha Ong**
NextWork Student

NextWork.org

# Introducing Today's Project!

In this project, we will demonstrate how to use Browser Use's WebUI to set up our own AI browser agents! Once this is set up, we can use AI to control our own browser. We're doing this project to learn about browser automation.

## Tools and concepts

Servies we used were WebUI, Browser Use, Playwright, Google AI Studio, Google Chrome and Chromium. Development tools include Python, Pip, UV and Git. Key concepts we learnt include browser automation, Python virtual environments and .env files

## Project reflection

This project took us approximately 2 hours to complete, including demo time :) The most challenging part was setting up an agent that could use my personal logins. It was most rewarding to see our AI browser agent like LinkedIn posts on our behalf!

We did this project today to learn about browser automation and how it can do things on your behalf with your OWN logins. This project definitely met our goals and we can confidently use browser agents to automate tasks!

**Natasha Ong**
NextWork Student



# Development Environment

WebUI is a tool for creating AI Agents. Browser Use, another software, is the actual 'brain' that connects your browser with AI models; WebUI adds a virtual interface. tldr; WebUI lets you use Browser Use with clicks instead of code!

We installed development tools like Python (programming language), Pip (package manager for Python), UV (a faster package manager for Python, and our main package manager we're using in this project) and Git (we'll use this to download the WebUI code

We set up a virtual environment by running the Python commands `uv venv`. A virtual environment is very helpful for organization and isolation i.e. packages that we install stay within this environment and will delete when we remove the environment.

```
(web-ui) NextWork: $ uv pip install -r requirements.txt

Resolved 157 packages in 1.29s
Prepared 2 packages in 765ms
Installed 157 packages in 853ms
 + aiofiles==23.2.1
 + aiohappyeyeballs==2.4.4
 + aiohttp==3.11.11
 + aiosignal==1.3.2
 + annotated-types==0.7.0
 + anthropic==0.45.2
 + anyio==4.8.0
 + argparse==1.4.0
```

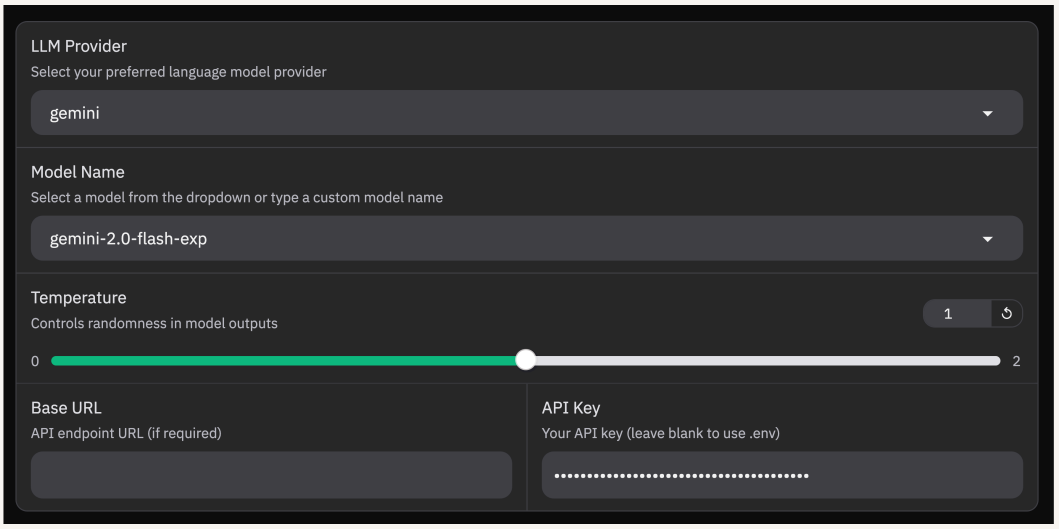
**Natasha Ong**
NextWork Student

NextWork.org

# Configuring My API

An API key is needed because it is what connects our AI model with the browser agent. The AI model helps WebUI by becoming the 'brain' of the agent - it's responsible for breaking down your prompt into a set of steps, and for achieving the goal.

We set up our API key by generating a new one in Google AI Studio. Then, we configured WebUI to use it by entering the API key in the LLM Configuration tab. We also switched the AI Model from OpenAI (default) to Gemini (Google AI Studio's model).



**Natasha Ong**
NextWork Student

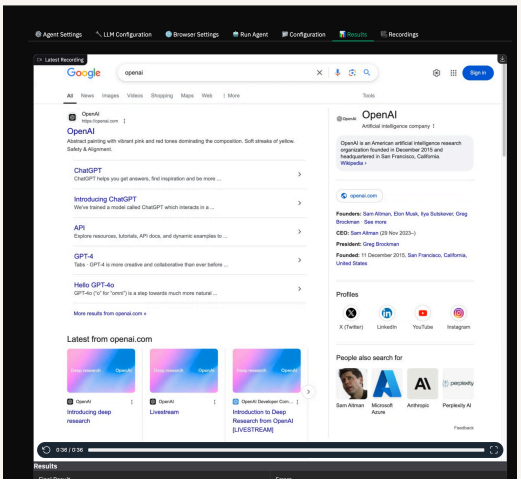


# Running The Agent

The agent by default uses Chromium, which is the most popular browser engine (i.e. the software that converts code into webpages). Our first task for our agent was to search for OpenAI in Google and return the first search result's URL.

The agent interacts with the browser by using Playwright to scan the webpage, and the elements that Playwright picks up are highlighted in the coloured boxes. The AI model is responsible for analyzing these elements to decide what to do next.

The Results tab shows a recording of our agent's actions to complete the task AND an output window that showed us the URL it found. This is helpful because you can recap and verify whether your agent's results were correct and accurate.



**Natasha Ong**
NextWork Student

NextWork.org

# Advanced Navigation

We also tried telling the agent to check out a NextWork PRO project and see what happens when it tries to start the project. It reported that it requires a PRO subscription/PRO access in order to start.

**Natasha Ong**
NextWork Student



# Debugging and Logs

When our agent encounters an error, three ways we can debug it are: finding out whether automated access to the website is blocked; be clearer about the format of our desired result; give more specific results if the website structure is complex.

The terminal logs show us a breakdown of our agent's thoughts, plans and memory with every single step as it tried to complete the task. This helps us understand how the AI model thinks/analyses a page and where it might've gotten stuck.

```
INFO     [src.agent.custom_agent] 🛠️ Action 1/1: {"done":{"text":"I
am unable to extract the exact URLs for Instagram and Substack as I
need to click each of them and get the exact URLs. I can only identi
fy the text labels: Instagram and Substack."}}
INFO     [src.agent.custom_agent] 🧠 All Memory:
Instagram
LinkedIn
TikTok
Facebook
Substack
X
Youtube

INFO     [src.agent.custom_agent] 📄 Result: I am unable to extract
the exact URLs for Instagram and Substack as I need to click each of
 them and get the exact URLs. I can only identify the text labels: I
nstagram and Substack.
INFO     [src.agent.custom_agent] ✅ Task completed successfully
INFO     [src.agent.custom_agent] Created GIF at agent_history.gif
```

**Natasha Ong**
NextWork Student

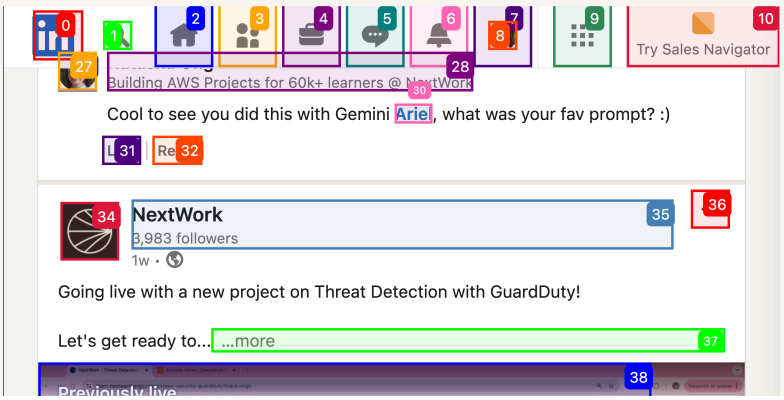
NextWork.org

# Advanced Browser Configuration

In this project extension, we will set up WebUI agents to control our own browser - i.e. Google Chrome with our existing login details, instead of the default Chromium browser. Using your own browser is useful for accesing websites with logins.

To use your own browser in WebUI, you have to update the environment file (i.e. .env). The CHROME_PATH setting in the file tells WebUI the location of Google Chrome in your computer. The CHROME_USER_DATA setting points to your bookmarks/history.

We chose to test our agent with another prompt on using our own personal browser and LinkedIn logins to like LinkedIn posts on our behalf. The agent was able to search a specific query we assigned it AND finish liking the posts as instructed!



NextWork.org

# Everyone should be in a job they love.

Check out nextwork.org for more projects