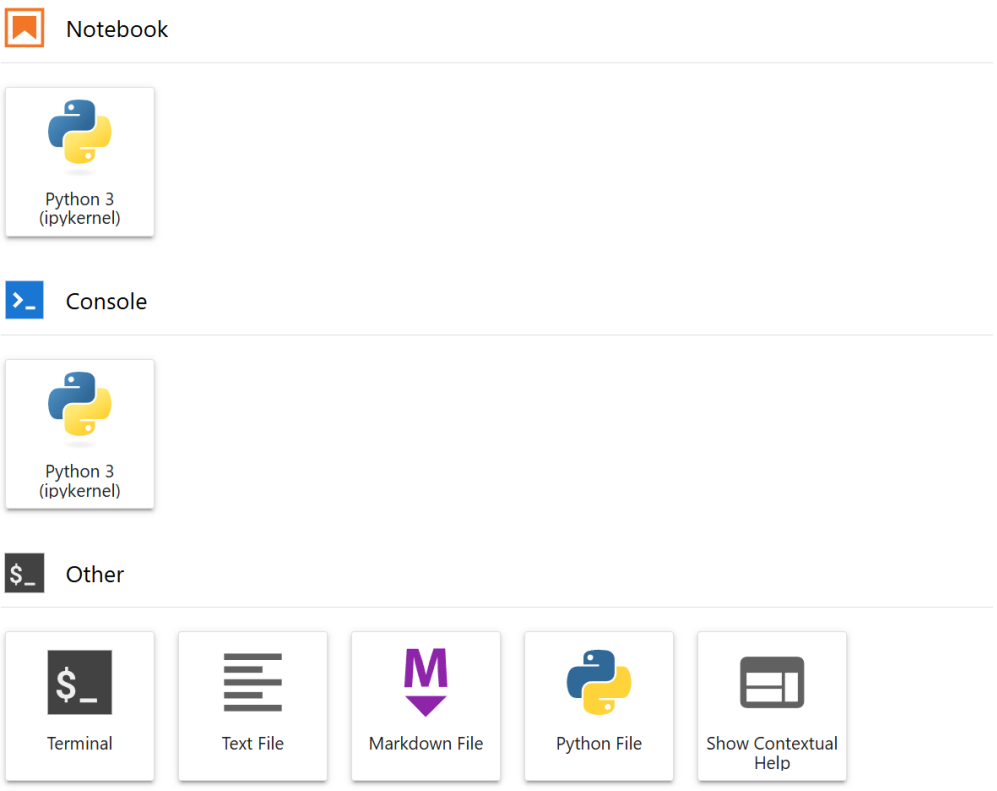
Deepseek-R1 Chatbot on AMD MI300x GPU

Welcome to this hands-on workshop! In this section, we will set up a Deepseek-R1 Chatbot demo based on 1 Mi300x GPU. Through this workshop, you will learn how to leverage vLLM to fully utilize MI300x strong AI computing performance and set up a personal Deepseek-R1 Chatbot on your laptop.

Step 1 Access MI300x GPU

In this workshop, we will use the MI300x cloud instance from Hot Isle servers. Please access your GPU instance link, which has been provided by this workshop. For myself, this instance been assigned to me http://llmcluster.embeddedllm.com/user/1
. After clicking it, you will find the below snapshot.



In this workshop, I will use Terminal directly. After clicking Terminal, you need to use the command of “rocm-smi” to check GPU status, like below:

```
root@ENC1-CLS01-SVR06:/workspace# rocm-smi

============================================ ROCm System Management Interface ============================================
================================================== Concise Info ==================================================
Device  Node  IDs              Temp        Power     Partitions          SCLK     MCLK    Fan  Perf  PwrCap  VRAM%  GPU%
              (DID,     GUID)  (Junction)  (Socket)  (Mem, Compute, ID)
==================================================================================================================
0       3     0x74a1,   41632  34.0°C      163.0W    NPS1, SPX, 0        2100Mhz  900Mhz  0%   auto  750.0W  77%    0%
==================================================================================================================
```

The command “rocminfo” can list the information of MI300x GPU, like below:

```
*******
Agent 3
*******
  Name:                    gfx942
  Uuid:                    GPU-ba05ac0db51ef16f
  Marketing Name:          AMD Instinct MI300X
  Vendor Name:             AMD
  Feature:                 KERNEL_DISPATCH
  Profile:                 BASE_PROFILE
  Float Round Mode:        NEAR
  Max Queue Number:        128(0x80)
  Queue Min Size:          64(0x40)
  Queue Max Size:          131072(0x20000)
  Queue Type:              MULTI
  Node:                    2
  Device Type:             GPU
  Cache Info:
    L1:                      32(0x20) KB
    L2:                      4096(0x1000) KB
    L3:                      262144(0x40000) KB
  Chip ID:                 29857(0x74a1)
  ASIC Revision:           1(0x1)
  Cacheline Size:          64(0x40)
  Max Clock Freq. (MHz):   2100
  BDFID:                   15616
  Internal Node ID:        2
  Compute Unit:            304
  SIMDs per CU:            4
  Shader Engines:          32
  Shader Arrs. per Eng.:   1
  WatchPts on Addr. Ranges:4
  Coherent Host Access:    FALSE
  Memory Properties:
  Features:                KERNEL_DISPATCH
  Fast F16 Operation:      TRUE
  Wavefront Size:          64(0x40)
  Workgroup Max Size:      1024(0x400)
  Workgroup Max Size per Dimension:
    x                        1024(0x400)
    y                        1024(0x400)
    z                        1024(0x400)
  Max Waves Per CU:        32(0x20)
  Max Work-item Per CU:    2048(0x800)
  Grid Max Size:           4294967295(0xffffffff)
  Grid Max Size per Dimension:
    x                        4294967295(0xffffffff)
    y                        4294967295(0xffffffff)
    z                        4294967295(0xffffffff)
  Max fbarriers/Workgrp:   32
  Packet Processor uCode:: 166
  SDMA engine uCode::      22
  IOMMU Support::          None
```

Now you can enjoy MI300x GPU.

Step2: Launch vLLM server on MI300x GPU

This cloud instance has pre-installed AMD optimized vLLM docker (https://hub.docker.com/r/rocm/vllm/tags), which brings the out-of-box experience for AMD vLLM users.

Launching a vLLM server is also very easy, just one command, like blow:

***vllm serve deepseek-ai/DeepSeek-R1-Distill-Qwen-32B ***
***--host 0.0.0.0 ***
***--port 8000 ***
***--api-key abc-123 ***
***--trust-remote-code ***
--seed 42

Here, we need to configure port number as the current network environment settings. For Hot Isle servers, 8000 is open to customers to launch vLLM server.
Api-key is also defined by yourself. In this example, we define it as “abc-123”.

Once you see the below information, that means your vLLM sever has been launched successfully.

```
INFO 03-28 04:07:14 [launcher.py:34] Route: /pooling, Methods: POST
INFO 03-28 04:07:14 [launcher.py:34] Route: /score, Methods: POST
INFO 03-28 04:07:14 [launcher.py:34] Route: /v1/score, Methods: POST
INFO 03-28 04:07:14 [launcher.py:34] Route: /v1/audio/transcriptions, Methods: POST
INFO 03-28 04:07:14 [launcher.py:34] Route: /rerank, Methods: POST
INFO 03-28 04:07:14 [launcher.py:34] Route: /v1/rerank, Methods: POST
INFO 03-28 04:07:14 [launcher.py:34] Route: /v2/rerank, Methods: POST
INFO 03-28 04:07:14 [launcher.py:34] Route: /invocations, Methods: POST
INFO:     Started server process [1044]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

You can test the vLLM sever work from laptop.

In your laptop, you need to launch your command prompt , and input the command with the obtained IP address:
curl http://llmcluster.embeddedllm.com/serve/1/v1/models -H "Authorization: Bearer abc-123"

if you see the below output, that means your vLLM server on MI300x has been set up successfully.

```
C:\Users\ningzhan>curl http://llmcluster.embeddedllm.com/serve/1/v1/models -H "Authorization: Bearer abc-123"
{"object":"list","data":[{"id":"deepseek-ai/DeepSeek-R1-Distill-Qwen-32B","object":"model","created":1743577234,"owned_b
y":"vllm","root":"deepseek-ai/DeepSeek-R1-Distill-Qwen-32B","parent":null,"max_model_len":131072,"permission":[{"id":"mo
delperm-5c7432752a1c44fab980ca0f4266aabf","object":"model_permission","created":1743577234,"allow_create_engine":false,"
allow_sampling":true,"allow_logprobs":true,"allow_search_indices":false,"allow_view":true,"allow_fine_tuning":false,"org
anization":"*","group":null,"is_blocking":false}]}]}
```

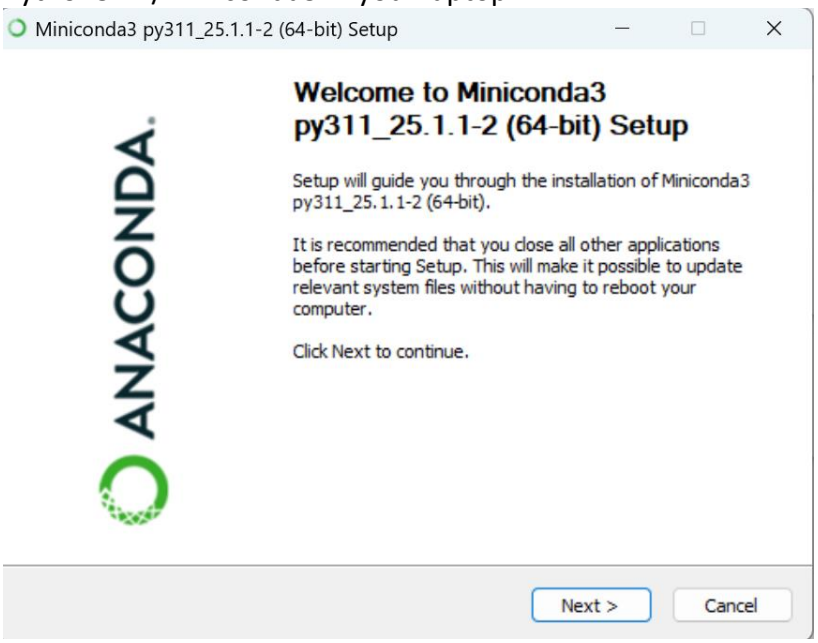
Step3: Setup Open-WebUI

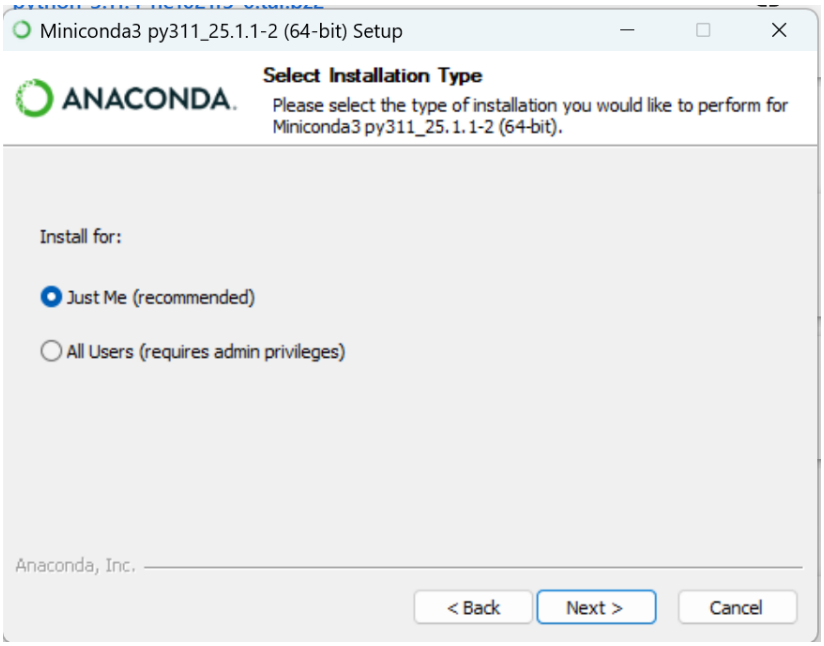
Open WebUI is an extensible, feature-rich, and user-friendly self-hosted AI platform designed to operate entirely offline. It supports various LLM inference framework, like vLLM and OpenAI-compatible APIs, making it a powerful AI deployment solution. In this workshop, we use it to set up our own Chatbot.
You can install Open-WebUI server in your laptop.
According to Open-WebUI document, the latest Open-WebUI depends on Python3.11, and the related modules in Python3.12 are not fully compatible with the version required by Open WebUI. So, it is strongly recommended to use Python3.11 environment in our laptop.

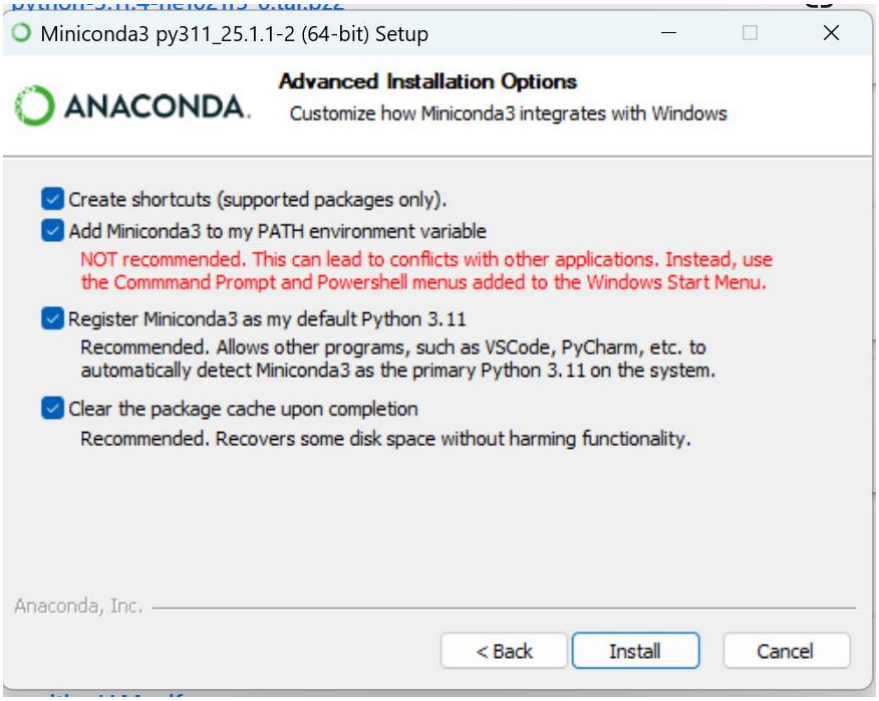
Python3.11/Miniconda3 can be downloaded in these links:

1) Windows: https://repo.anaconda.com/miniconda/Miniconda3-py312_25.1.1-2-Windows-x86_64.exe
2) MacOS_x86: https://repo.anaconda.com/miniconda/Miniconda3-py312_25.1.1-2-MacOSX-x86_64.sh
3) MacOS-ARM: https://repo.anaconda.com/miniconda/Miniconda3-py312_25.1.1-2-MacOSX-arm64.sh

We can take windows OS as example. By clicking the downloaded file, you begin to install Python3.11/Miniconda3 in your laptop.







Once python3.11 environment on your laptop is ready, you can open Anaconda PowerShell prompt and run below commands to install Open-WebUI.

curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
python get-pip.py
pip install open-webui

```
(base) PS C:\Users\ningzhan> python --version
Python 3.11.11
(base) PS C:\Users\ningzhan> curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
(base) PS C:\Users\ningzhan> python get-pip.py
Collecting pip
  Downloading pip-25.0.1-py3-none-any.whl.metadata (3.7 kB)
Downloading pip-25.0.1-py3-none-any.whl (1.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 224.1 kB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 25.0
    Uninstalling pip-25.0:
      Successfully uninstalled pip-25.0
Successfully installed pip-25.0.1
(base) PS C:\Users\ningzhan> pip install open-webui
Collecting open-webui
  Downloading open_webui-0.5.20-py3-none-any.whl.metadata (19 kB)
Collecting aiocache (from open-webui)
  Downloading aiocache-0.12.3-py2.py3-none-any.whl.metadata (8.3 kB)
Collecting aiofiles (from open-webui)
  Downloading aiofiles-24.1.0-py3-none-any.whl.metadata (10 kB)
Collecting aiohttp==3.11.11 (from open-webui)
  Downloading aiohttp-3.11.11-cp311-cp311-win_amd64.whl.metadata (8.0 kB)
Collecting alembic==1.14.0 (from open-webui)
  Downloading alembic-1.14.0-py3-none-any.whl.metadata (7.4 kB)
Collecting anthropic (from open-webui)
  Downloading anthropic-0.49.0-py3-none-any.whl.metadata (24 kB)
```

The whole procedure to install Open-WebUI may need some time, and you can take a break now.

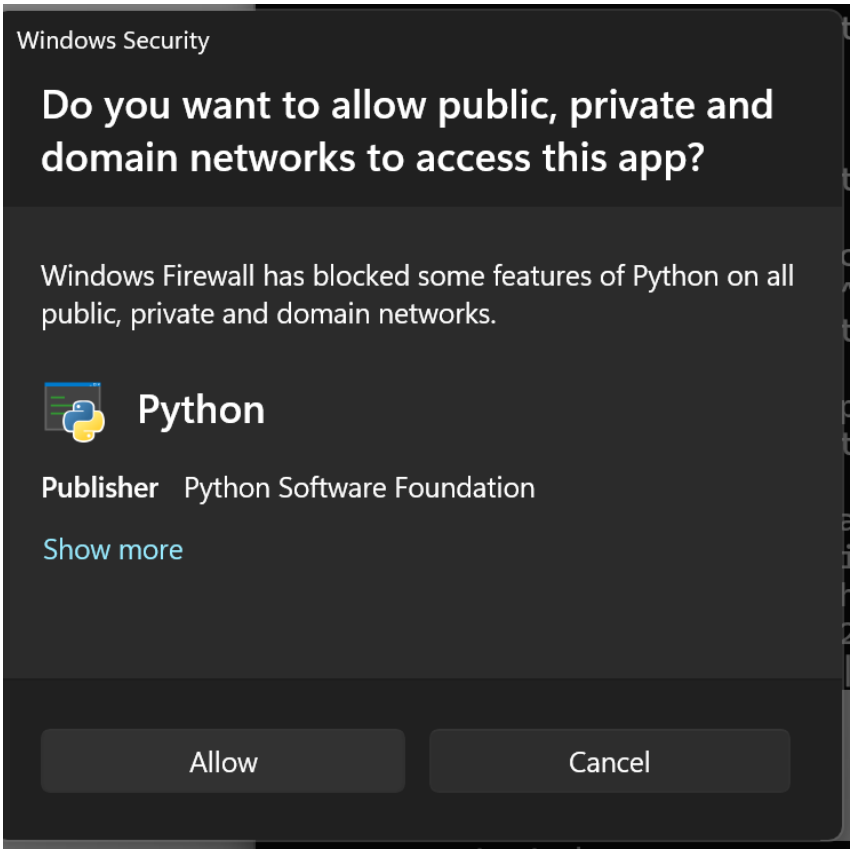
Once it is done, you can launch Open-WebUI server as below. on your laptop, port number is determined by yourself. Open-WebUI default port is 8080, you can choose other value.

open-webui serve --port {PORT_NUM}

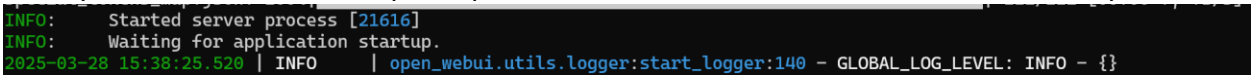
In my test, I launched it like below snapshot,

```
(base) PS C:\Users\ningzhan> open-webui serve --port 5009
Loading WEBUI_SECRET_KEY from file, not provided as an environment variable.
Generating a new secret key and saving it to C:\Users\ningzhan\.webui_secret_key
Loading WEBUI_SECRET_KEY from C:\Users\ningzhan\.webui_secret_key
C:\Users\ningzhan\AppData\Local\miniconda3\Lib\site-packages\open_webui
C:\Users\ningzhan\AppData\Local\miniconda3\Lib\site-packages
C:\Users\ningzhan\AppData\Local\miniconda3\Lib
INFO  [alembic.runtime.migration] Context impl SQLiteImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
INFO  [alembic.runtime.migration] Running upgrade  -> 7e5b5dc7342b, init
INFO  [alembic.runtime.migration] Running upgrade 7e5b5dc7342b -> ca81bd47c050, Add config table
INFO  [alembic.runtime.migration] Running upgrade ca81bd47c050 -> c0fbf31ca0db, Update file table
INFO  [alembic.runtime.migration] Running upgrade c0fbf31ca0db -> 6a39f3d8e55c, Add knowledge table
Creating knowledge table
Migrating data from document table to knowledge table
INFO  [alembic.runtime.migration] Running upgrade 6a39f3d8e55c -> 242a2047eae0, Update chat table
Converting 'chat' column to JSON
Renaming 'chat' column to 'old_chat'
Adding new 'chat' column of type JSON
Dropping 'old_chat' column
```

If you meet the below warning, please accept it to allow network connection for Open-WebUI server.



Once you meet the below output, Open-WebUI server has been launched successfully.

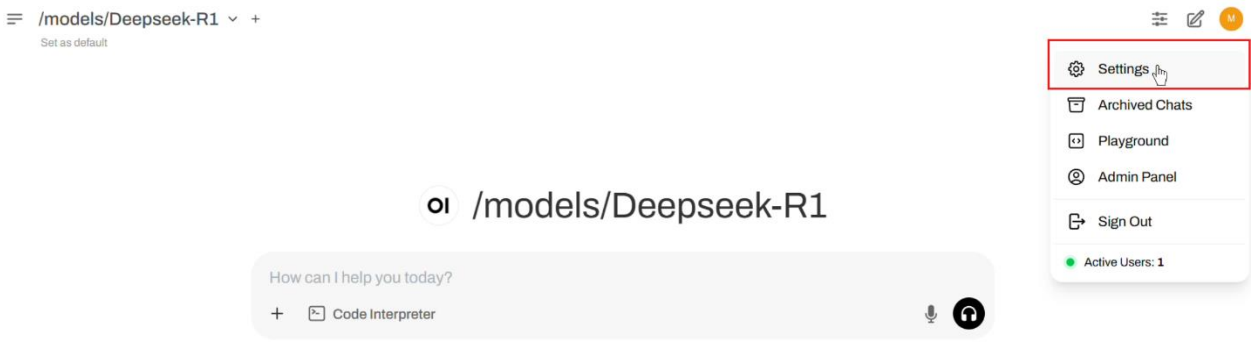


Step 4 Configure Open-WebUI client and run Chatbot

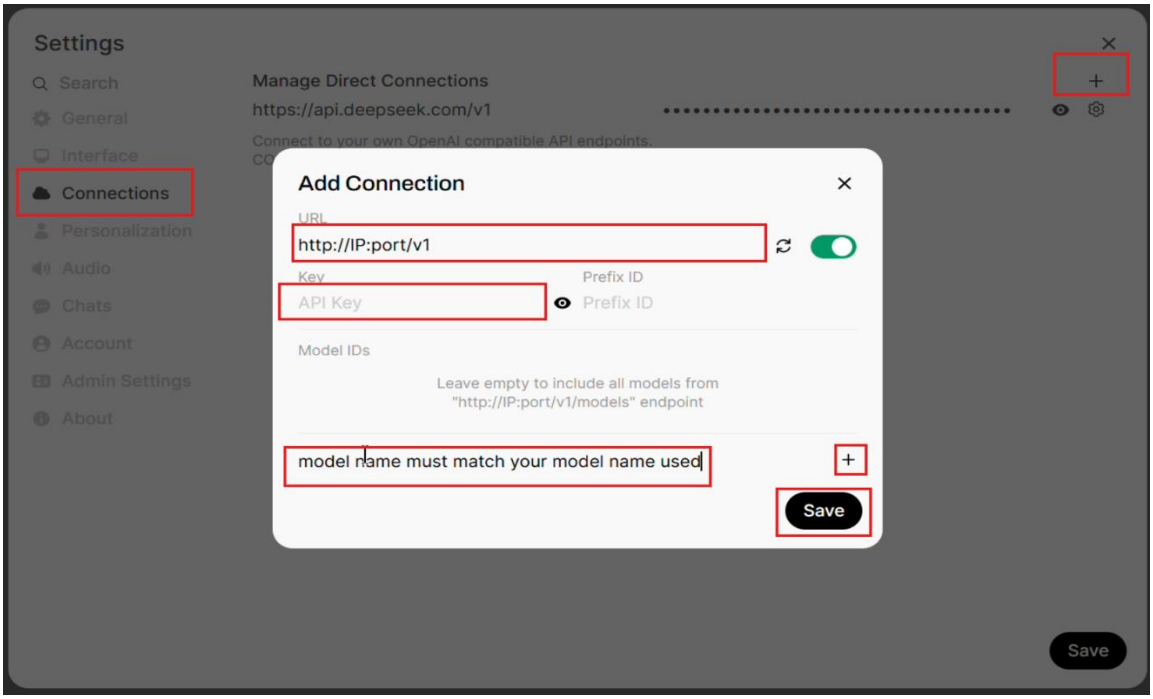
You can open Open-WebUI client by accessing Open-WebUI server IP addr + server port number from your laptop, like http://Server IP :Server Port/.
If you installed Open-WebUI server in your laptop, you can access it using local host directly, like http://localhost:5009/

when you first login Open-WebUI client, you may need to register your name, mailbox and password. Once login Open-WebUI successfully, you need to configure your endpoint URL in the Open WebUI client as follows:

- Navigate to Settings as shown in the image below:



- Select Connections form the left tab.
  - Enter URL that is matching this format: http://YOUR_SERVER_PUBLIC_IP:PORT_NUMBER/v1, like http://llmcluster.embeddedllm.com/serve/1/v1
  - Enter API Key matching the key you chose in step 2 and passed to vllm serve.
  - Enter model name that matches exactly the argument passed to vllm serve in step 2. For example deepseek-ai/DeepSeek-R1-Distill-Qwen-32B.
  - Click on + button.
  - Click on Save button.



In my demo, I configured it as below settings:

When the configuration is saved , you can find the Chatbot has been set up on your laptop successfully, and AMD MI300x of cloud instance machine is providing the strong DeepSeek Ri inference through vLLM.