

OBUDA UNIVERSITY YEAR OF JUBILEES



TRADITION AND INNOVATION



# Obuda University's Generative Artificial Intelligence platform

User Guide

genAI platform developer team

Revision 1.0

November 19, 2025

**Table 1.** Document Revision History

<b>Revision</b>	<b>Date</b>	<b>Reason for Revision Change(s)</b>
Number	Date	Add explanation
1.0	October 15, 2025	Introduction of user guide

## Table of Contents

- 1 Login** **5**
  - 1.1 Microsoft Login . . . . . 5
  - 1.2 Enter University Email . . . . . 5
  - 1.3 Stay signed in . . . . . 6
  
- 2 Sidebar and History** **7**
  - 2.1 Sidebar . . . . . 7
  - 2.2 Chat History . . . . . 7
  
- 3 Model Features** **8**
  - 3.1 Switching Models on an Empty Chat . . . . . 8
  - 3.2 Model List Overview . . . . . 9
  - 3.3 Switching Models During a Chat . . . . . 10
  - 3.4 Using Multiple Models Simultaneously . . . . . 10
  - 3.5 Web Search Integration . . . . . 11
  - 3.6 Code Interpreter . . . . . 11
  - 3.7 File Upload . . . . . 12
  
- 4 Deepseek Warning** **12**
  
- 5 Settings** **13**
  - 5.1 Opening Settings . . . . . 13
  - 5.2 General Settings . . . . . 14
  - 5.3 Advanced Parameters . . . . . 14
  - 5.4 Interface Settings . . . . . 15
  - 5.5 Personalization / Memory . . . . . 15
  - 5.6 Audio Settings (STT/TTS) . . . . . 16

<b>6 Chat Management</b>	<b>16</b>
<b>7 Account / API Keys</b>	<b>17</b>
<b>8 Cline Bot</b>	<b>18</b>
8.1 Step 1: Install the Cline Extension . . . . .	18
8.2 Step 2: Initial Setup and Configuration Choice . . . . .	19
8.2.1 Option A: Use Default Free Service . . . . .	19
8.2.2 Option B: Configure Custom University API (Recommended) . . . . .	20
8.3 Step 3: Configure the Open AI Compatible Endpoint . . . . .	21
<b>9 Example Workflow</b>	<b>23</b>
9.1 Step 4: Initiating the Task and Planning . . . . .	23
9.2 Step 5: Backend Setup and Command Execution . . . . .	25
9.3 Step 6: Iterative Development and Multimodal Refinement . . . . .	27
9.4 Step 7: Testing and Validation . . . . .	29
<b>10 Continue Extension</b>	<b>30</b>
10.1 Step 1: Installation and Configuration . . . . .	30
10.1.1 Recommended Models for Continue . . . . .	31
10.1.2 Example Configuration . . . . .	31
10.2 Step 2: Model Selection and Usage . . . . .	32
<b>11 Tool Comparison</b>	<b>34</b>
11.1 Cline: The Project-Level Agent . . . . .	34
11.1.1 Recommended Models for Cline . . . . .	35
11.2 Continue: The In-Code Assistant . . . . .	35
11.3 Summary: Which Tool to Use? . . . . .	35

## List of Tables

1 Document Revision History . . . . .	2
---------------------------------------	---

## List of Figures

1 Microsoft Login . . . . .	5
-----------------------------	---

2	Entering University Email Address . . . . .	6
3	Keep Signed In . . . . .	6
4	Sidebar . . . . .	7
5	Chat History . . . . .	8
6	Switching Models on an Empty Chat . . . . .	8
7	Model List and Details . . . . .	9
8	Switching Models in an Ongoing Chat . . . . .	10
9	Using Multiple Models in a Single Chat . . . . .	10
10	Web Search Functionality . . . . .	11
11	Running Code with the Code Interpreter . . . . .	11
12	Uploading Files for Model Analysis . . . . .	12
13	Deepseek-R1 Warning . . . . .	12
14	Opening Settings . . . . .	13
15	Settings Interface . . . . .	13
16	General Settings . . . . .	14
17	Advanced Parameters . . . . .	14
18	Interface Settings . . . . .	15
19	Memory and Personalization . . . . .	15
20	Audio Settings (STT and TTS) . . . . .	16
21	Chat Import/Export, Archiving, and Deletion . . . . .	17
22	Account Management, Profile Information, and API Keys . . . . .	17
23	Viewing (Blue), Copying (Green), and Regenerating (Red) API Keys . . . . .	18
24	Searching for the Cline Extension . . . . .	18
25	Installing the Cline Extension . . . . .	18
26	Choosing the Configuration Method . . . . .	19
27	Cline Browser Sign-up/Login . . . . .	20
28	Accessing Cline Settings via the Gear Icon . . . . .	21
29	Completing the API Configuration . . . . .	22
30	The initial request to Cline. . . . .	24
31	Cline's generated development plan. . . . .	25
32	Cline proposes a command for execution. . . . .	25
33	An example of a command execution error. . . . .	26

34	Cline's self-correction and successful re-execution. . . . .	26
35	Configuring Cline's permissions. . . . .	27
36	Cline proposing code for a new file. . . . .	28
37	The initial functional, but unstyled, UI. . . . .	28
38	The visually refined To-Do List application. . . . .	29
39	Cline testing the application via the browser. . . . .	29
40	Flexibility to switch between available models. . . . .	30
41	Accessing the Continue settings panel. . . . .	31
42	The config.yaml file (right) and the graphical model setup (left). . . . .	31
43	Switching between configured models in the chat interface. . . . .	33
44	Continue answering a code-based question by searching the repository. . . . .	34

## 1 Login

### 1.1 Microsoft Login

To log in to OpenWebUI, you need a university account.

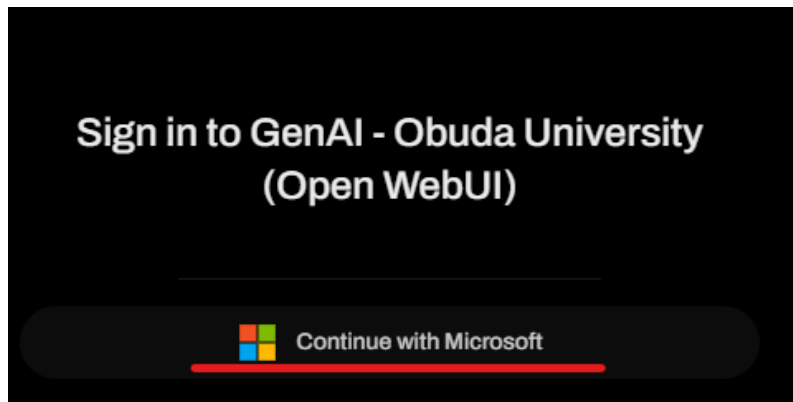
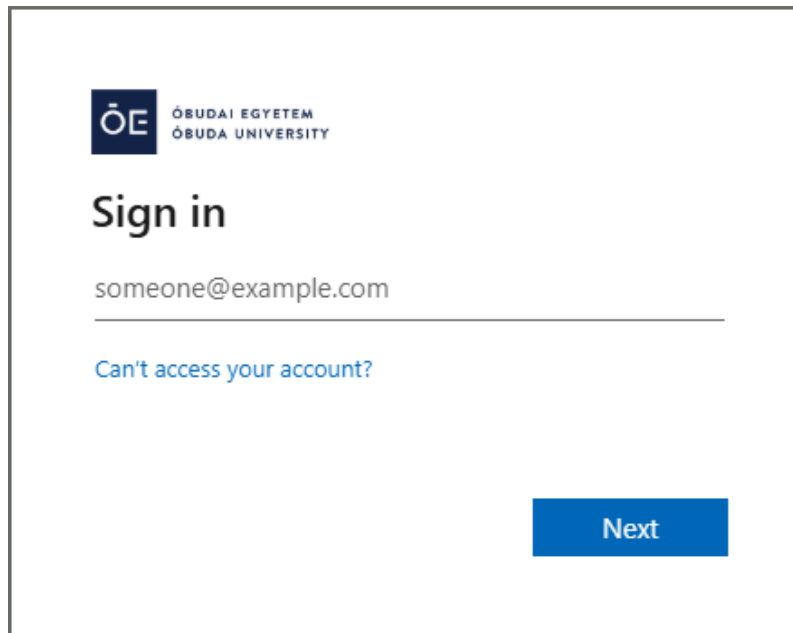


Figure 1. Microsoft Login

### 1.2 Enter University Email

Enter your university email address to proceed with login. This ensures your university permissions are applied correctly.



The screenshot shows the login interface for Obuda University. At the top left is the university logo, consisting of a blue square with the white letters 'ÖE' and the text 'ÖBUDAI EGYETEM ÖBUDA UNIVERSITY' to its right. Below the logo, the heading 'Sign in' is displayed in a large, bold, black font. Underneath the heading, the email address 'someone@example.com' is entered into a text input field. A horizontal line is positioned below the input field. Below the line, the text 'Can't access your account?' is written in a smaller, blue font. At the bottom right of the form, there is a blue rectangular button with the white text 'Next'.

**Figure 2.** Entering University Email Address

### 1.3 Stay signed in

If you don't want to sign in every time, choose the "Stay signed in" option.



The screenshot shows a dialog box for the 'Stay signed in?' option. At the top left is the university logo, consisting of a blue square with the white letters 'ÖE' and the text 'ÖBUDAI EGYETEM ÖBUDA UNIVERSITY' to its right. Below the logo, the email address '@stud.uni-obuda.hu' is displayed. The heading 'Stay signed in?' is shown in a large, bold, black font. Below the heading, the text 'Do this to reduce the number of times you are asked to sign in.' is written in a smaller, black font. Below this text, there is a checkbox followed by the text 'Don't show this again'. At the bottom of the dialog, there are two buttons: a grey button with the text 'No' and a blue button with the text 'Yes'.

**Figure 3.** Keep Signed In

## 2 Sidebar and History

### 2.1 Sidebar

The left sidebar gives you access to history, search, and notes. It provides easy navigation between different functions.

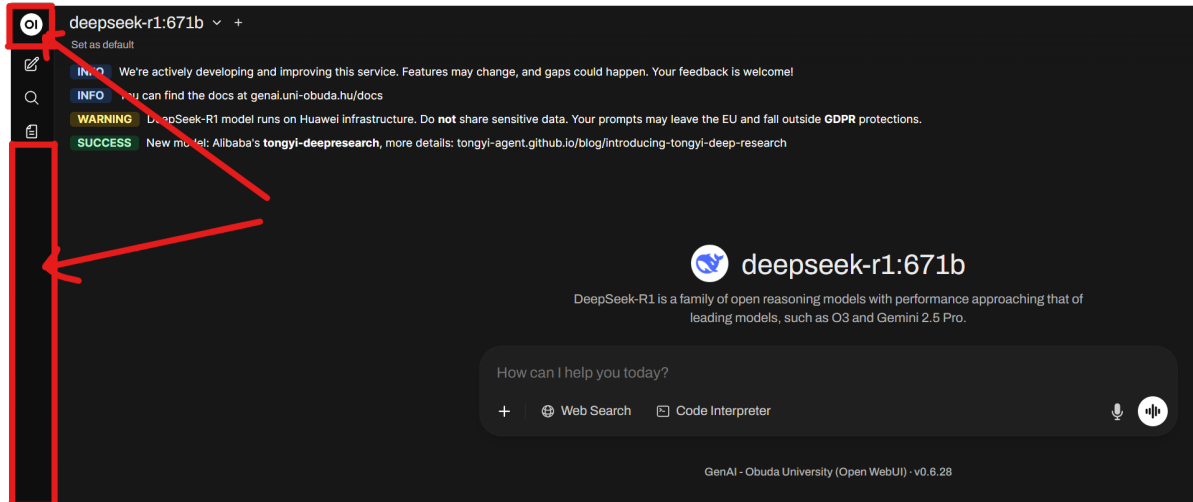


Figure 4. Sidebar

### 2.2 Chat History

By navigating to the chat history, you can easily review previous conversations. This helps maintain continuity in your workflow.

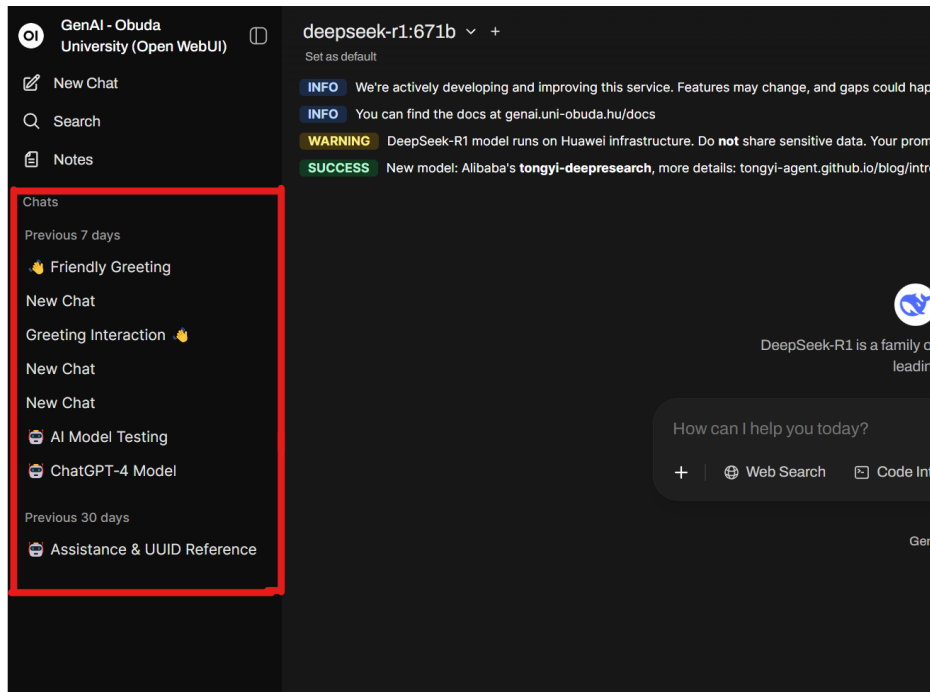


Figure 5. Chat History

### 3 Model Features

#### 3.1 Switching Models on an Empty Chat

You can switch models directly from an empty chat. The highlighted button allows you to select a model before starting a new conversation.

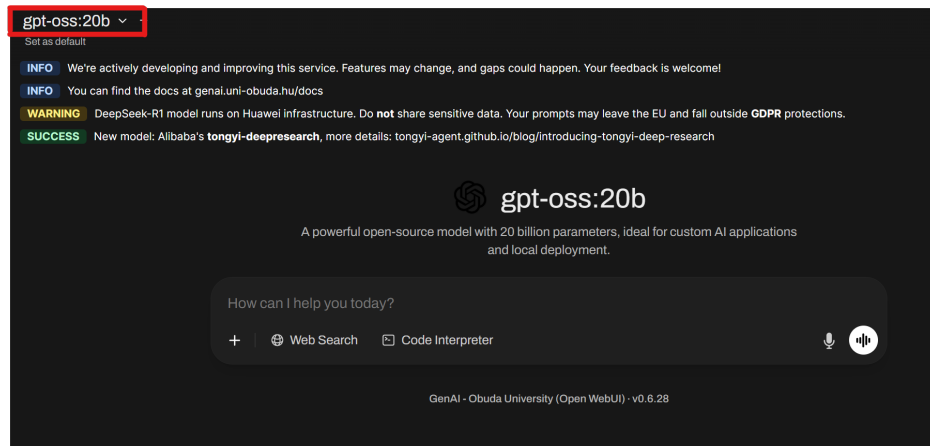


Figure 6. Switching Models on an Empty Chat

## 3.2 Model List Overview

The model list displays all available language models. Each model can have different capabilities, such as general reasoning, programming, or research. The small *i* icon displays a detailed description of the model, while the tag icon shows categories (tags) such as:

- **code** – optimized for programming, debugging, and explaining code.
- **huawei** – runs on Huawei infrastructure (outside the EU; GDPR may not apply).
- **OE** – university-hosted models, processed within the university environment.
- **reasoning** – models designed for complex logical or multi-step reasoning tasks.

Available models include:

- **gpt-oss:20b** – a balanced open-source model for general-purpose reasoning, explanations, and creative tasks.
- **deepseek-r1:671b** – a large reasoning-oriented model for deep analysis, multi-step logic, and advanced problem-solving.
- **tongyi-deepresearch** – specialized in information gathering and deep research tasks; can summarize and cross-reference external data.
- **llama3.1:8b** – a lightweight, efficient model for everyday chat, summaries, and light reasoning tasks.
- **qwen2.5-coder:7b** – a model focused on programming and code assistance, ideal for code generation, explanation, and debugging.

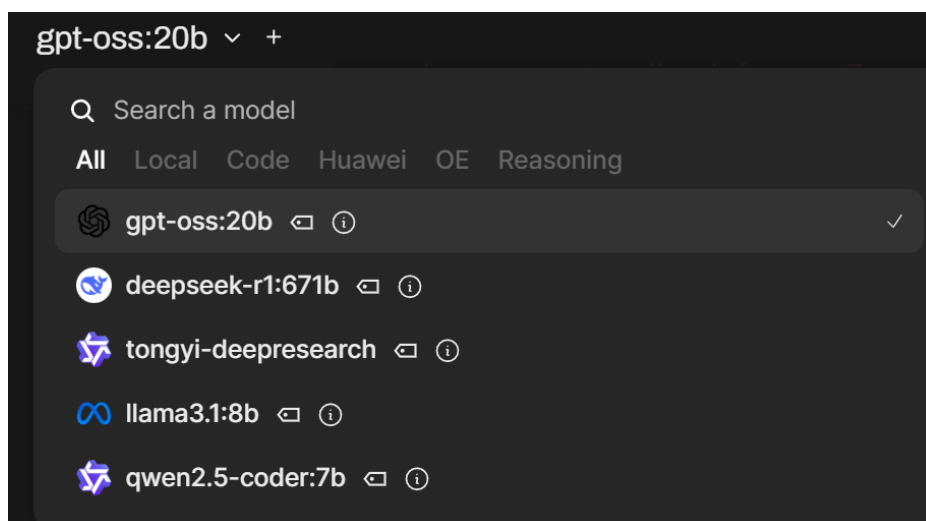


Figure 7. Model List and Details

### 3.3 Switching Models During a Chat

You can also switch models while chatting. When you select a new model during an active conversation, the new model will receive the context of the previous chat so the discussion can continue seamlessly.

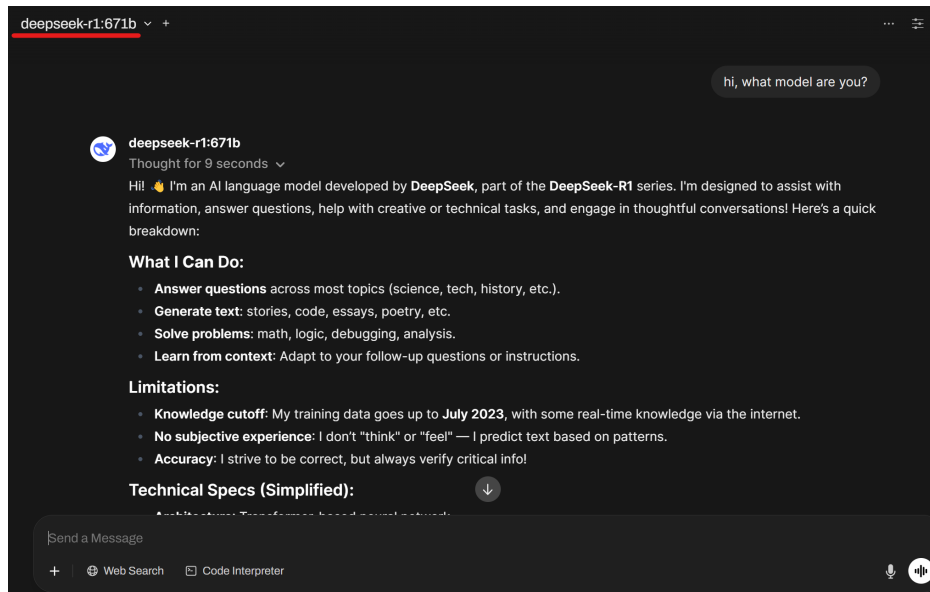


Figure 8. Switching Models in an Ongoing Chat

### 3.4 Using Multiple Models Simultaneously

It is possible to use multiple models at the same time. Each model will generate its own response, allowing you to compare answers or perspectives side by side.

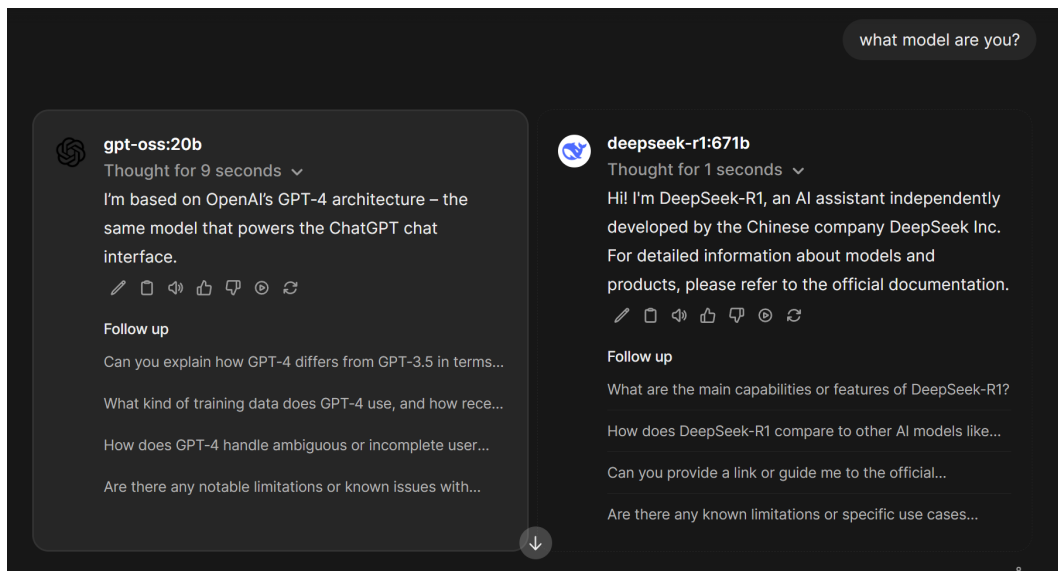


Figure 9. Using Multiple Models in a Single Chat

### 3.5 Web Search Integration

You can perform web searches directly from the chat interface. The example in the screenshot shows how search results are integrated into the conversation for more up-to-date information.

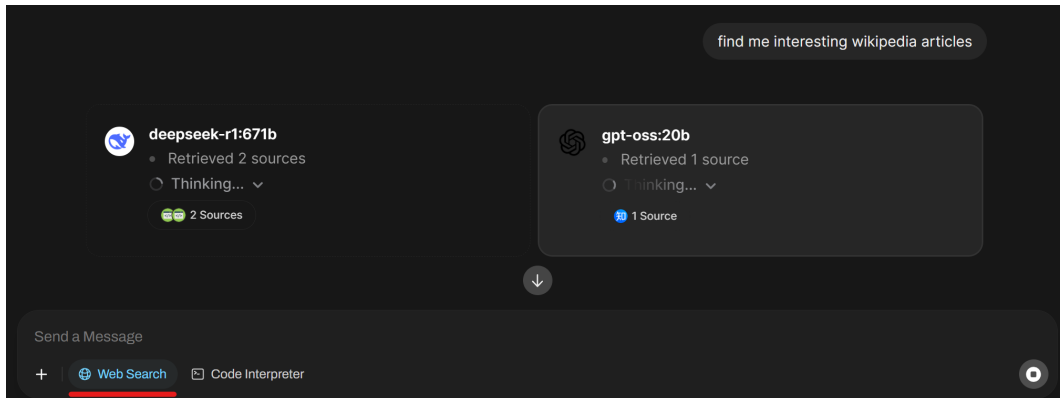


Figure 10. Web Search Functionality

### 3.6 Code Interpreter

You can run code directly from within the chat using the built-in code interpreter. This allows you to execute, test, and visualize code outputs as part of your interaction with the model.

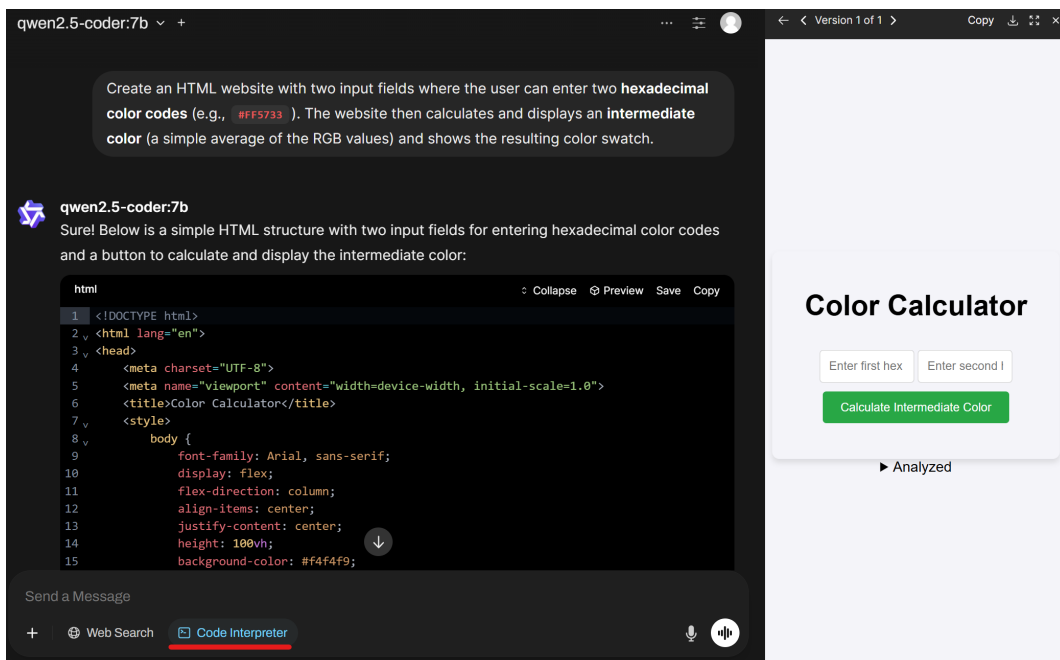


Figure 11. Running Code with the Code Interpreter

### 3.7 File Upload

You can upload files for the model to analyze or reference using the upload button. This feature allows you to:

- Upload documents (PDF, TXT, DOCX) for summarization or question answering.
- Provide datasets or source code for code analysis.
- Attach notes, essays, or research materials for context-aware discussions.

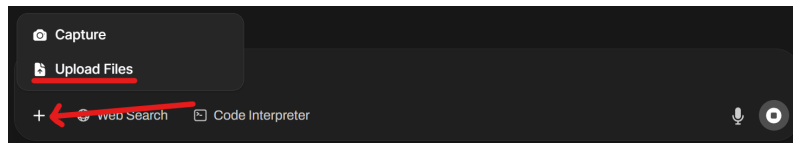


Figure 12. Uploading Files for Model Analysis

## 4 Deepseek Warning

The DeepSeek-R1 model runs on Huawei's infrastructure, so it's important **not to share sensitive data** on the platform. The prompts and data you provide may leave the EU region, and thus may not be protected under GDPR.

This warning is visible to all users on the OpenWebUI interface and is prominently displayed to highlight data protection risks. Users are advised not to share personal or sensitive information, and to use only data that can be safely made public.

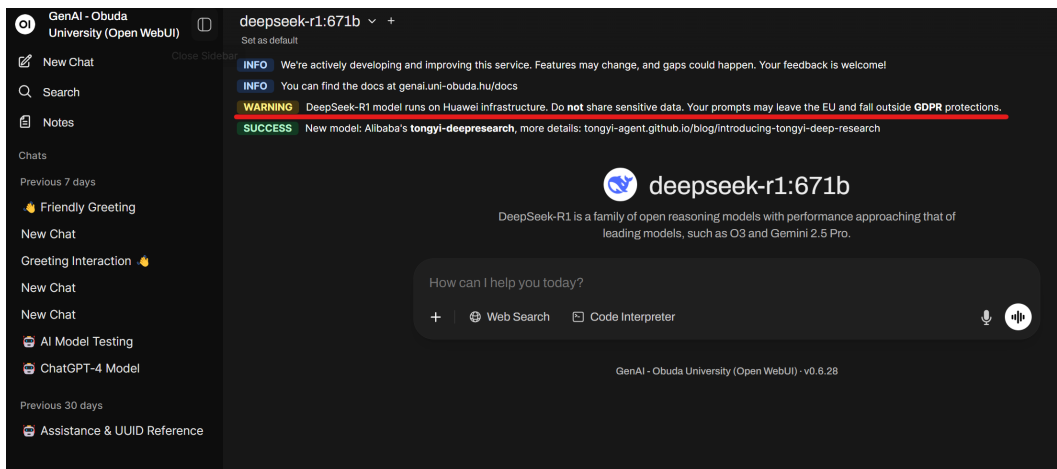


Figure 13. Deepseek-R1 Warning

# 5 Settings

## 5.1 Opening Settings

Click the profile icon to access settings, where you can modify user account options and system parameters.

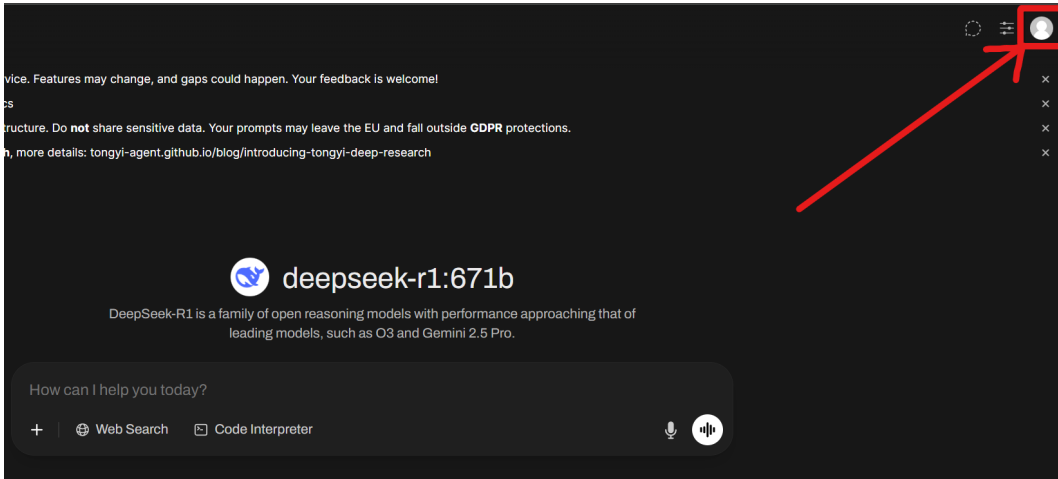


Figure 14. Opening Settings

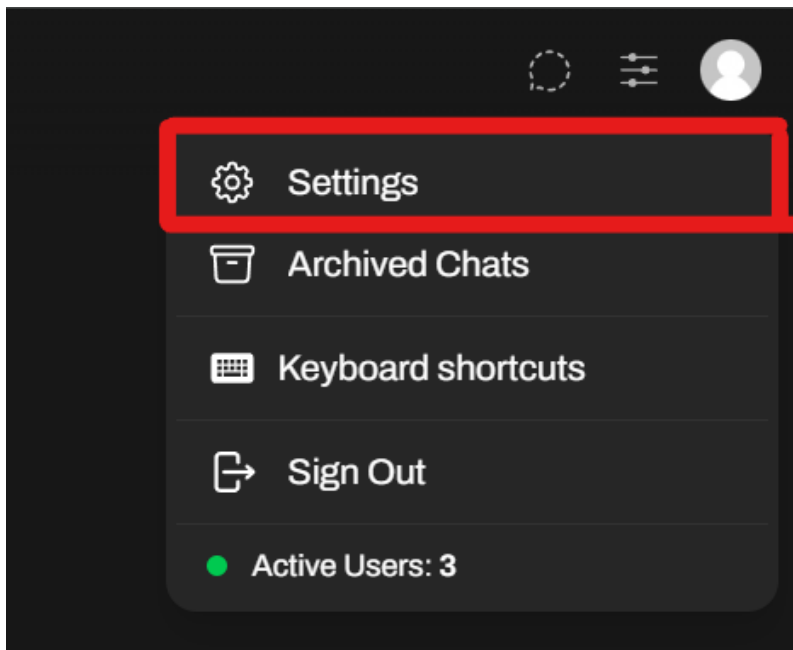
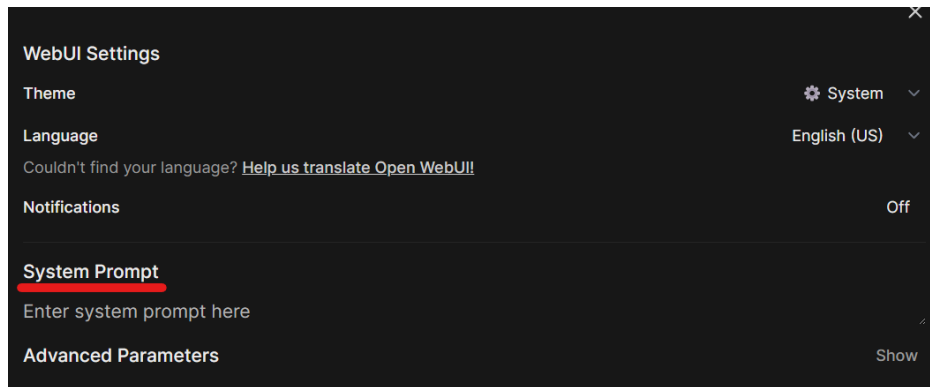


Figure 15. Settings Interface

## 5.2 General Settings



**Figure 16.** General Settings

Here you can modify options such as:

- Theme, language, notifications
- System prompt configuration

## 5.3 Advanced Parameters



**Figure 17.** Advanced Parameters

These parameters allow you to fine-tune chat behavior, such as response generation style, temperature, and stop sequences.

## 5.4 Interface Settings

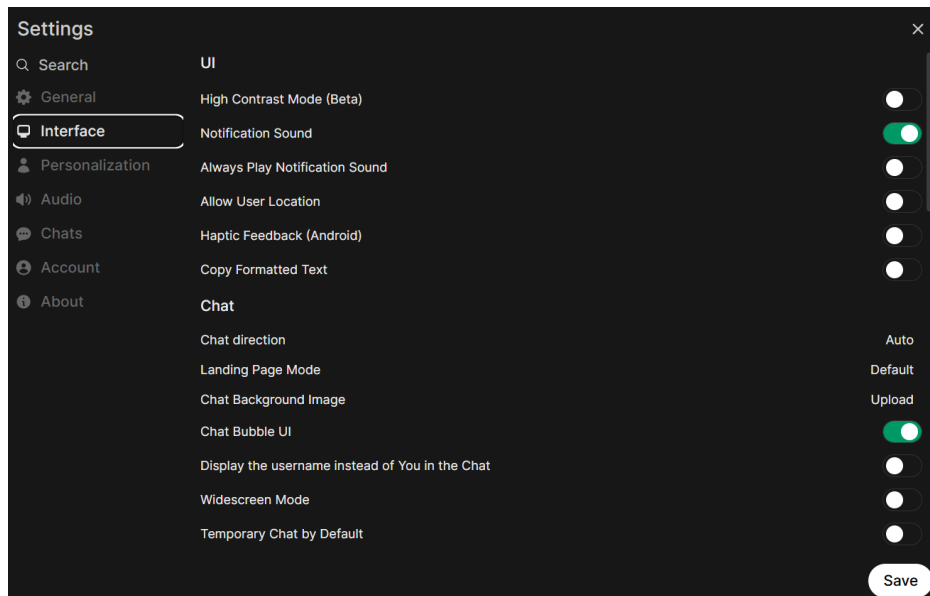


Figure 18. Interface Settings

Here you can configure:

- UI modifications (High Contrast, Chat Bubble UI)
- Chat options (direction, landing page, auto-copy)
- Voice, File, and Artifact handling

## 5.5 Personalization / Memory

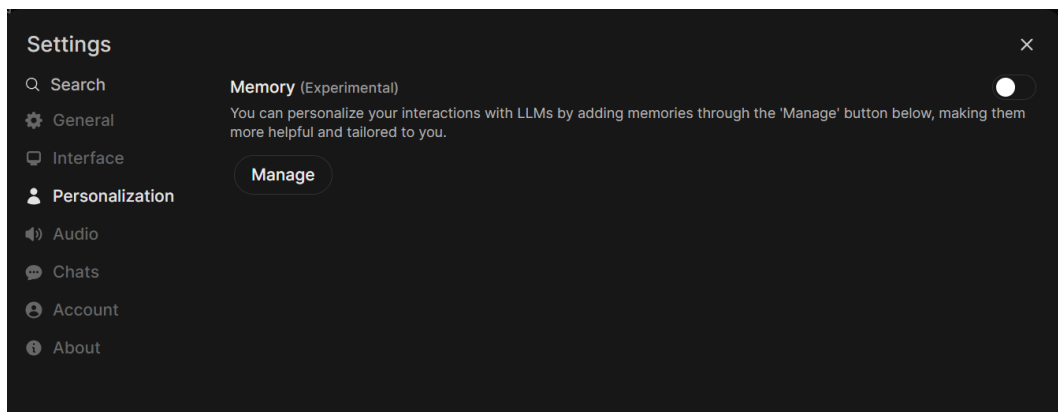
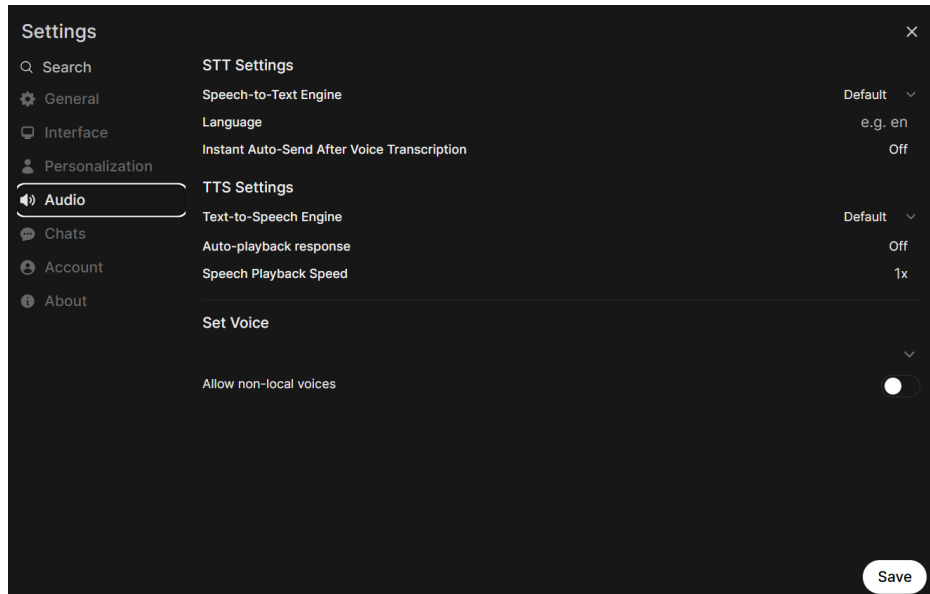


Figure 19. Memory and Personalization

This allows the model to remember previous interactions, making responses more personalized and contextually relevant.

## 5.6 Audio Settings (STT/TTS)



**Figure 20.** Audio Settings (STT and TTS)

Here you can configure speech recognition, voice generation, speed, and preferred voice options to enhance audio interactions.

## 6 Chat Management

You can manage your chats in OpenWebUI here. You can import previous conversations, export current chats, archive, or delete them.

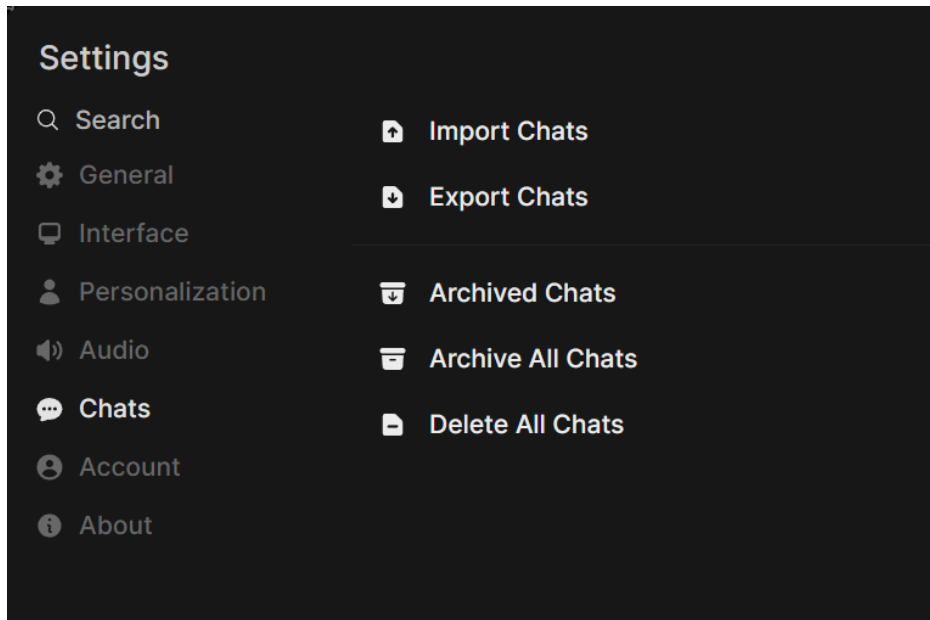


Figure 21. Chat Import/Export, Archiving, and Deletion

## 7 Account / API Keys

Users can manage their accounts and API keys linked to OpenWebUI. It's important to keep API keys secure, as they grant access to the system.

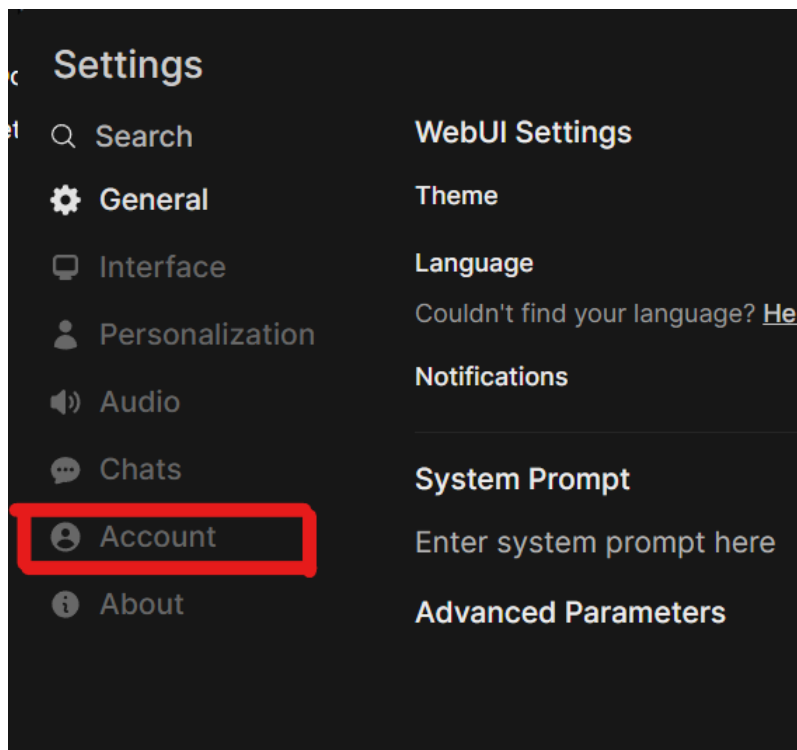


Figure 22. Account Management, Profile Information, and API Keys

Users can view, copy, or regenerate their API keys, which can also be used by external applications to connect to OpenWebUI.

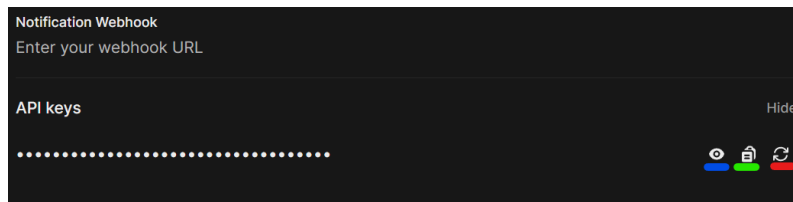


Figure 23. Viewing (Blue), Copying (Green), and Regenerating (Red) API Keys

## 8 Cline Bot

This section details how to configure the Cline VS Code extension to connect to our university's local OpenWebUI server, allowing you to use DeepSeek, Qwen, and other powerful models for your coding tasks.

### 8.1 Step 1: Install the Cline Extension

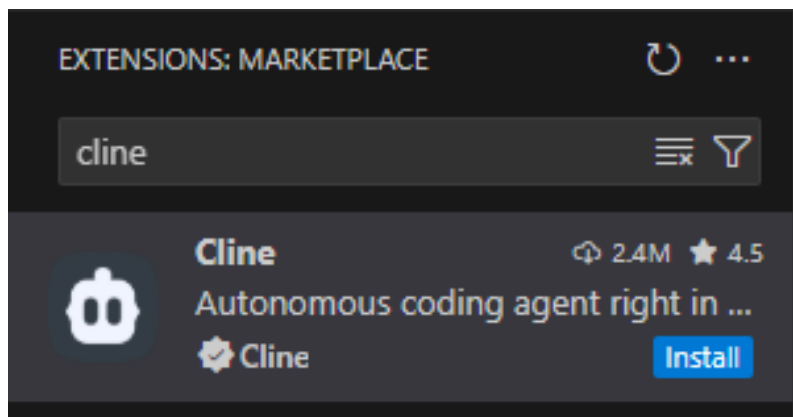


Figure 24. Searching for the Cline Extension

- Open the **Extensions** view in VS Code (Ctrl+Shift+X or Cmd+Shift+X).
- Search for **cline** in the marketplace.

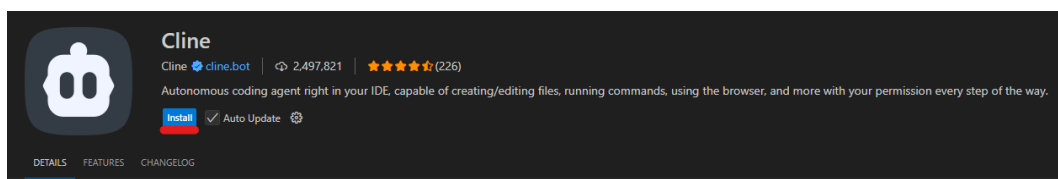
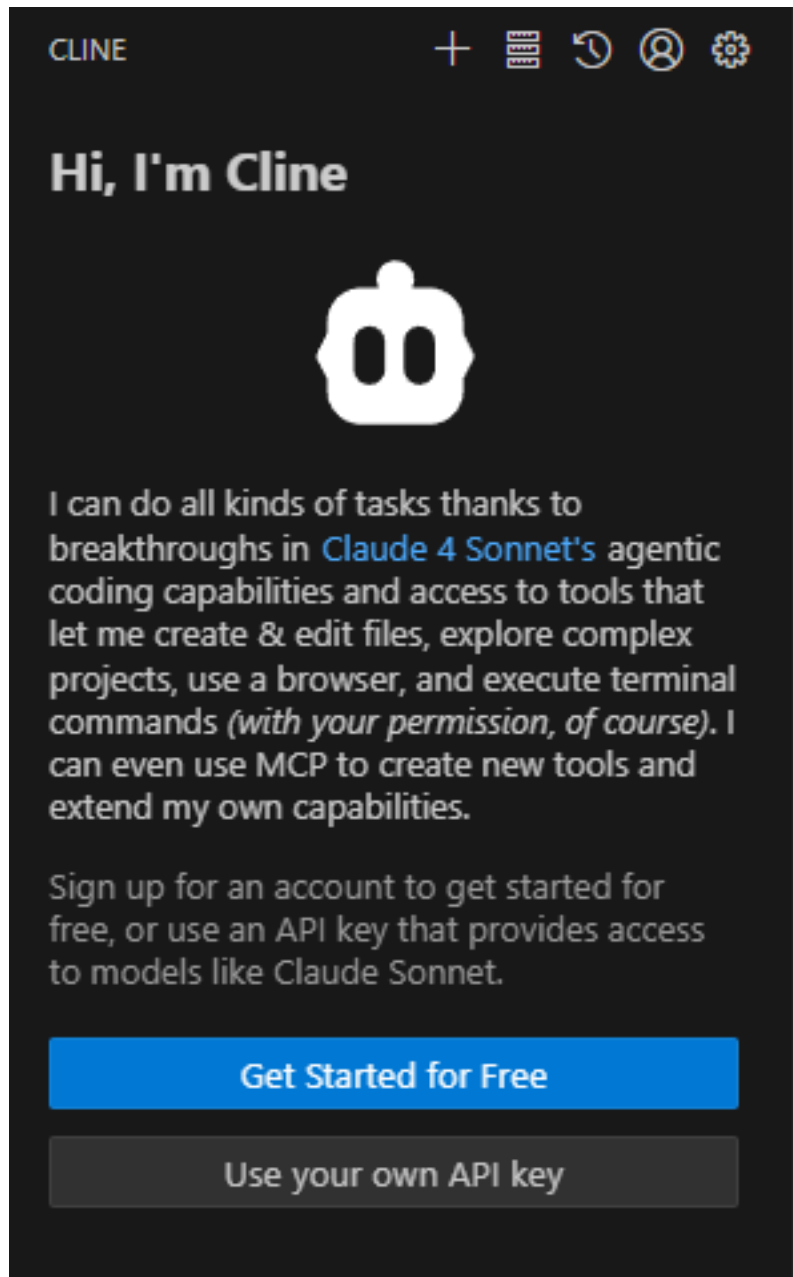


Figure 25. Installing the Cline Extension

- Click the **Install** button for the **Cline** extension by `cline.bot`.

## 8.2 Step 2: Initial Setup and Configuration Choice

After installing the Cline extension, you have two options for initial setup: using the default, free service for quick testing, or configuring our university's custom API endpoint.



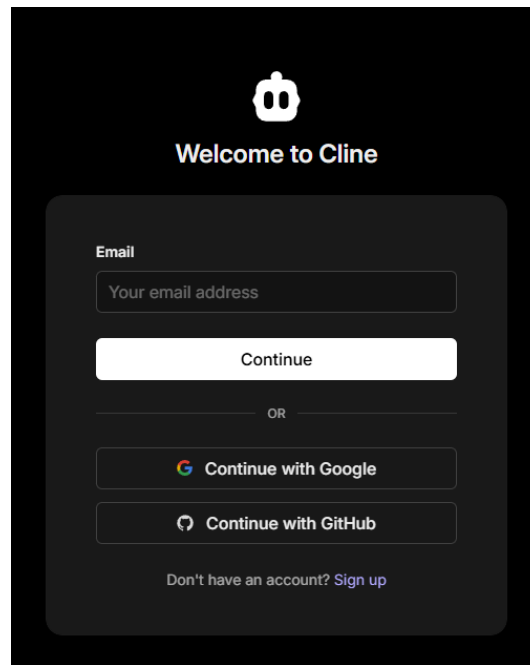
**Figure 26.** Choosing the Configuration Method

- Click the **Cline** icon on the left Activity Bar to open the extension pane.

### 8.2.1 Option A: Use Default Free Service

If you wish to use Cline's default model (e.g., Claude 4 Sonnet) for free, follow these steps:

- On the welcome screen, click the **Get Started for Free** button.
- This will redirect your browser to the Cline sign-up page (Figure 4).



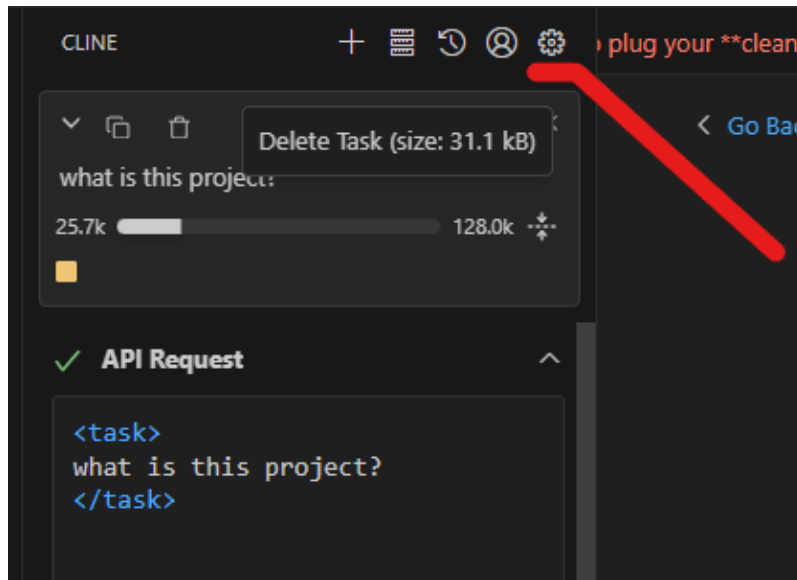
**Figure 27.** Cline Browser Sign-up/Login

- Complete the sign-up process in your browser using an email, Google, or GitHub account.
- Once authenticated, the extension should automatically connect, allowing you to use the default model.

### 8.2.2 Option B: Configure Custom University API (Recommended)

To use the powerful models on our university's server (DeepSeek, Qwen, etc.), you must manually enter the API details.

- On the welcome screen, click the **Use your own API key** button.



**Figure 28.** Accessing Cline Settings via the Gear Icon

- Once the main Cline panel is open, click the **gear icon** (⚙️) in the top-right corner of the Cline panel to open the **API Configuration** settings.

### 8.3 Step 3: Configure the Open AI Compatible Endpoint

In the **Settings** view, navigate to **API Configuration** and enter the details for our university's server.

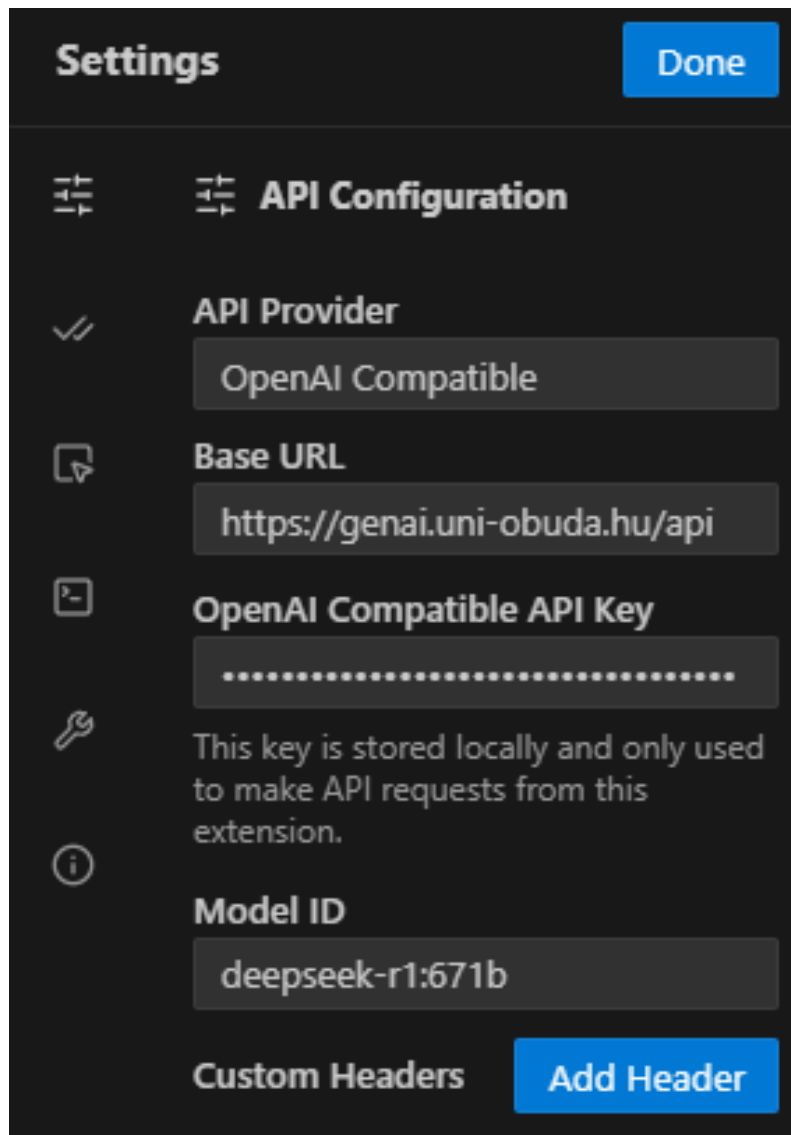


Figure 29. Completing the API Configuration

- **API Provider:** Select **OpenAI Compatible**. This is crucial for connecting to our Open-WebUI server.
- **Base URL:** Enter the full URL for our server's API endpoint:

`https://genai.uni-obuda.hu/api`

- **OpenAI Compatible API Key:** Paste the API key you generated from the OpenWebUI platform here. (It will appear as masked/hidden text).
- **Model ID:** Enter the exact identifier for the model you wish to use. The example shows the DeepSeek model:

`deepseek-r1:671b`

- Click the blue **Done** button in the top-right corner to save your configuration.

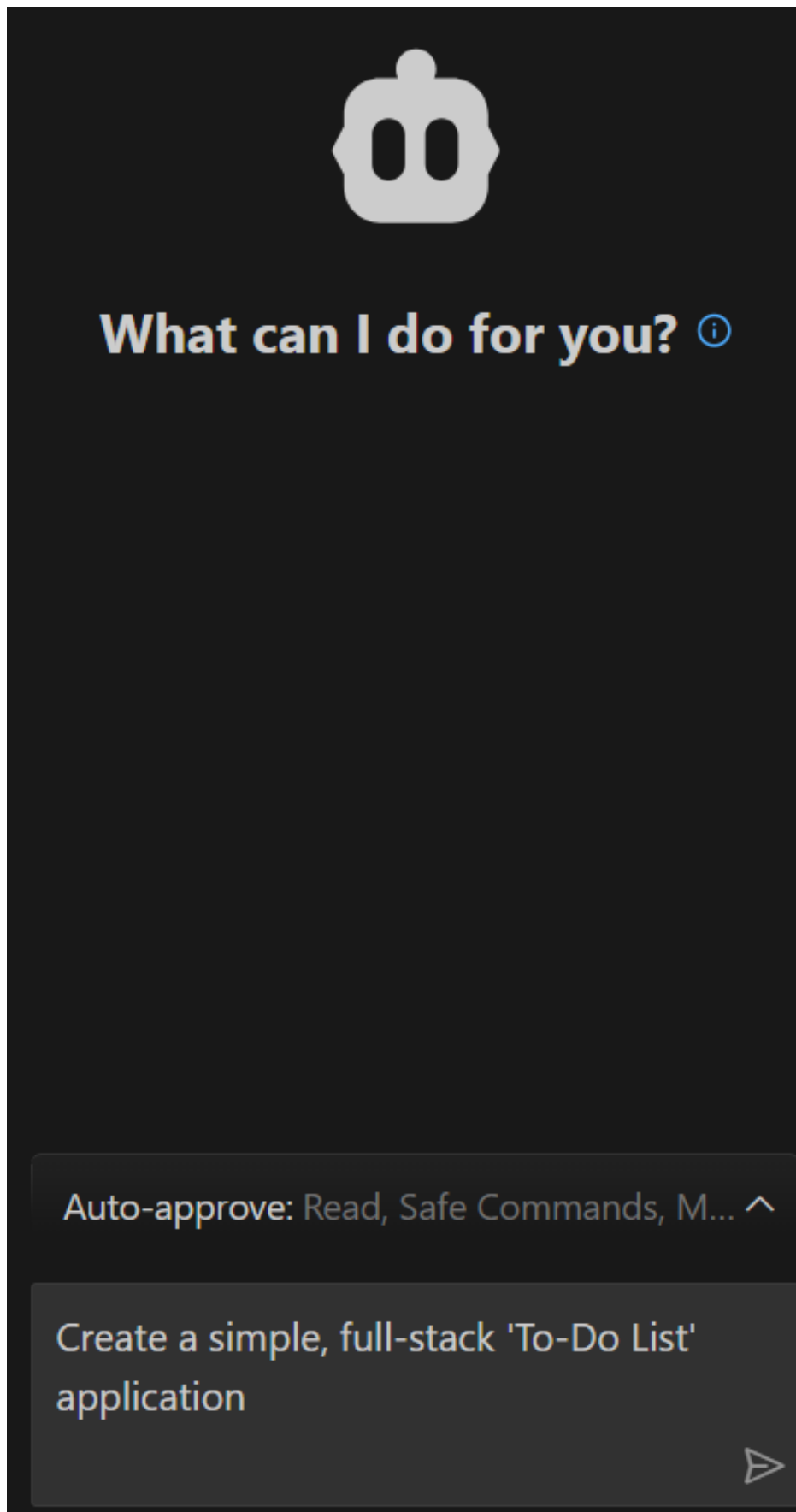
**Cline is now configured and ready to use our custom AI models for advanced coding tasks.**

## 9 Example Workflow

Once Cline is configured with the university's custom API, it becomes a powerful partner for end-to-end development. This section demonstrates the process of creating a full-stack "To-Do List" application, leveraging the connected high-performance models.

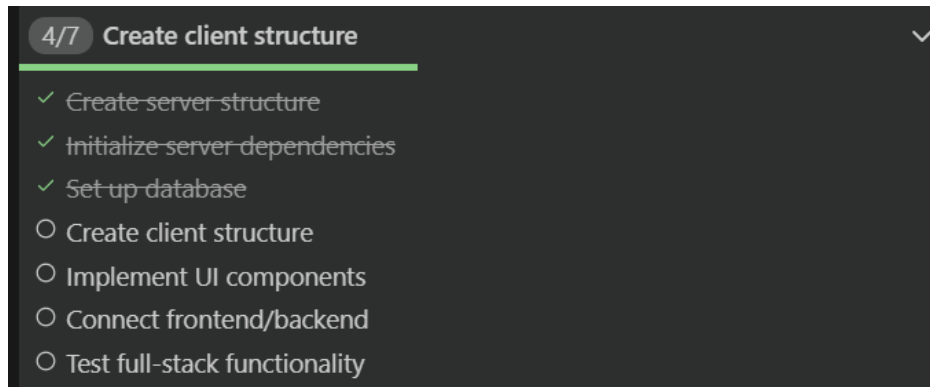
### 9.1 Step 4: Initiating the Task and Planning

The development process starts with a simple, high-level prompt to Cline. This prompt defines the overall goal: building a working application.



**Figure 30.** The initial request to Cline.

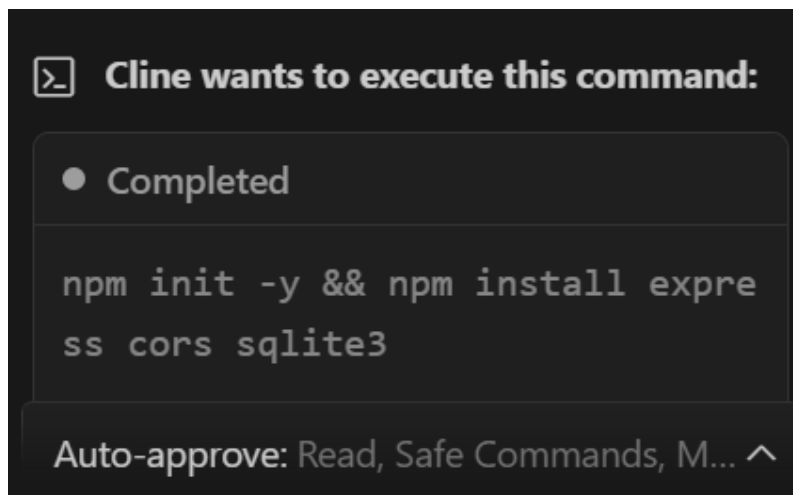
The AI immediately processes this request and generates a **structured, step-by-step plan**. This plan breaks the complexity into manageable, trackable phases, moving logically from the server setup to client testing.



**Figure 31.** Cline's generated development plan.

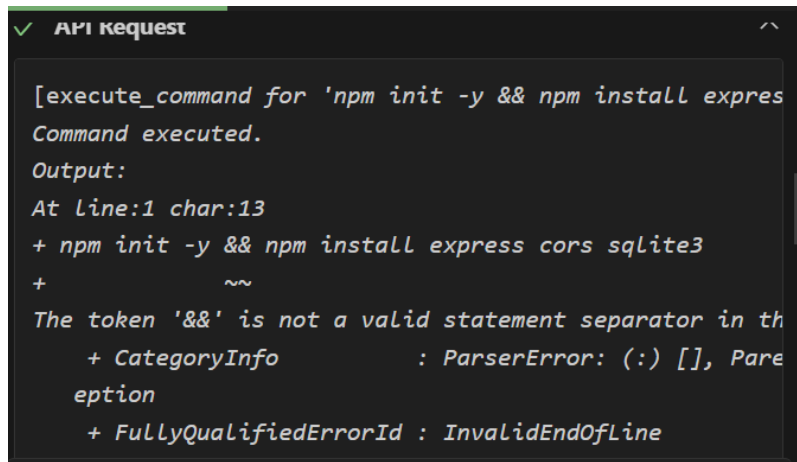
## 9.2 Step 5: Backend Setup and Command Execution

For system initialization, Cline proposes necessary terminal commands to install dependencies. Before any action, Cline presents the command for user review and eventual approval.



**Figure 32.** Cline proposes a command for execution.

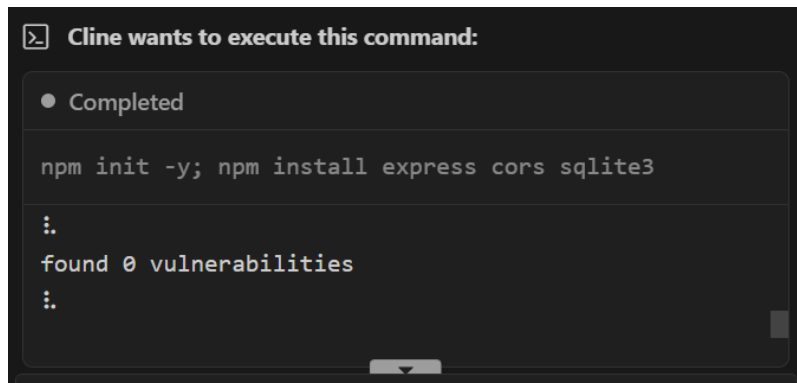
The AI's advanced capability includes **self-correction**. When an environment or syntax issue causes a command failure (like the misplaced '&&' token shown here), the model reads the error output.



```
✓ API Request  
[execute_command for 'npm init -y && npm install expres  
Command executed.  
Output:  
At line:1 char:13  
+ npm init -y && npm install express cors sqlite3  
+ ~~~~~  
The token '&&' is not a valid statement separator in th  
+ CategoryInfo          : ParserError: (:) [], Pare  
ption  
+ FullyQualifiedErrorId : InvalidEndOfLine
```

**Figure 33.** An example of a command execution error.

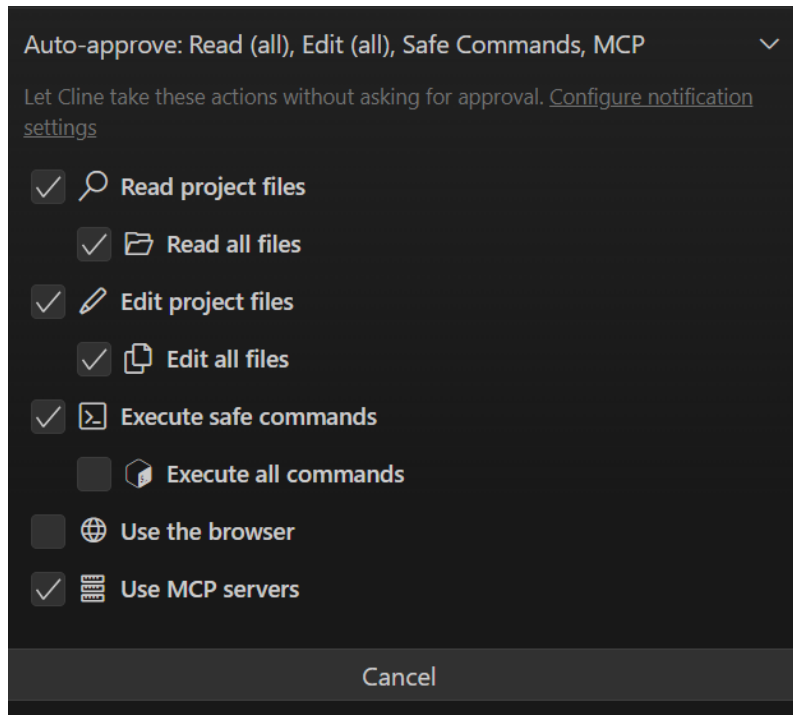
Cline then re-issues a modified, correct command that resolves the initial terminal incompatibility. This significantly streamlines the setup phase by handling common platform-specific errors.



```
☒ Cline wants to execute this command:  
● Completed  
npm init -y; npm install express cors sqlite3  
:  
found 0 vulnerabilities  
:  
⏏
```

**Figure 34.** Cline's self-correction and successful re-execution.

You retain complete control over the AI's actions via configurable permissions. This includes granting or revoking permission to execute commands and access files.



**Figure 35.** Configuring Cline's permissions.

### 9.3 Step 6: Iterative Development and Multimodal Refinement

As per the plan, Cline generates new code and proposes creating or modifying files within the project structure. The user is prompted to review the generated code before confirming the changes with the **Save** button.

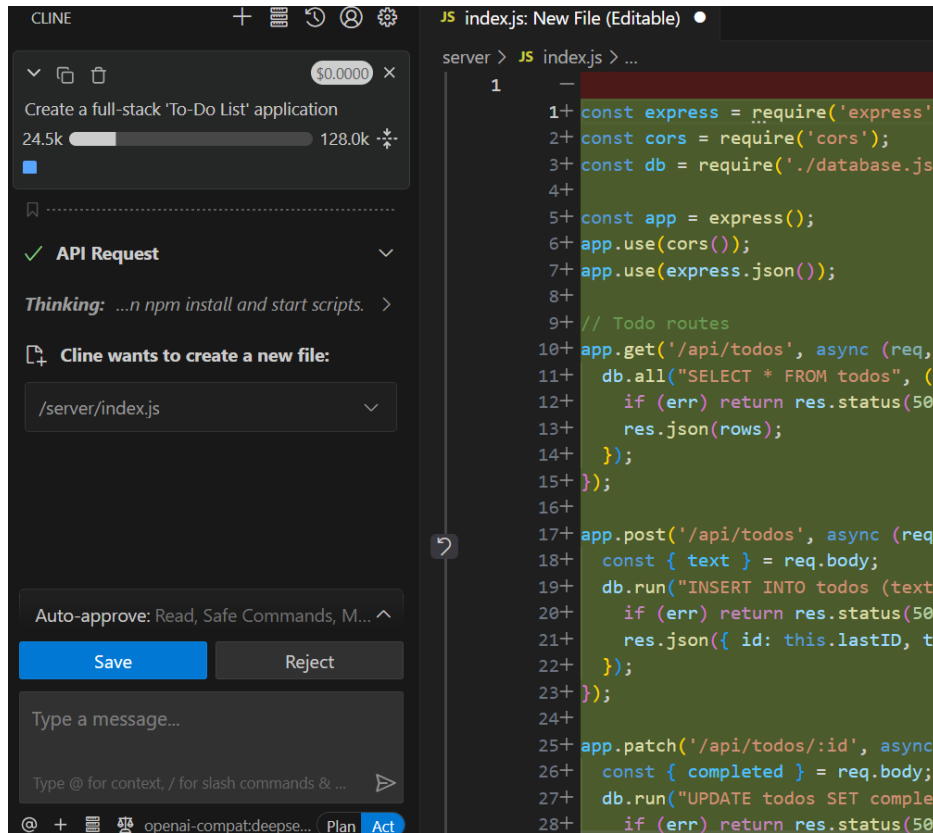


Figure 36. Cline proposing code for a new file.

Using our multimodal API, Cline can understand and act on visual input. The initial functional UI, while complete, often lacks aesthetic polish.

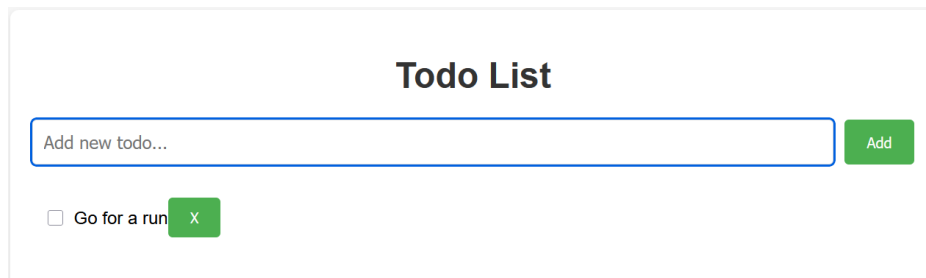
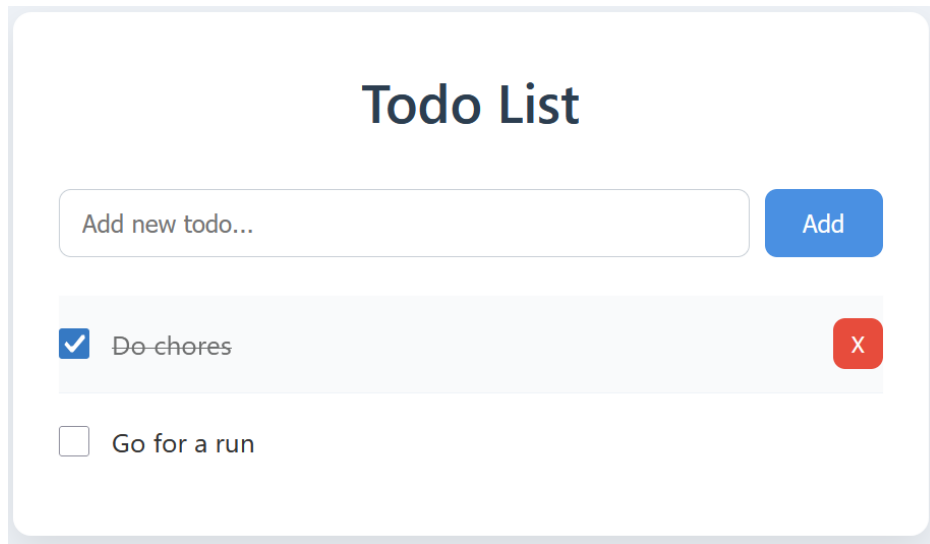


Figure 37. The initial functional, but unstyled, UI.

By submitting the screenshot with a prompt for improvement ("make it more modern"), Cline generates refined HTML and CSS. The result is a polished, professional-looking application.

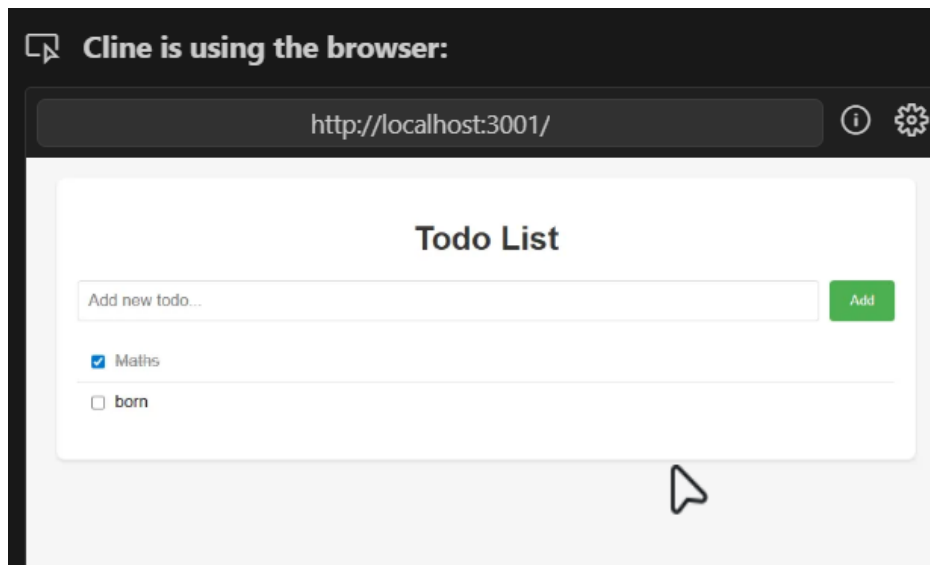


**Figure 38.** The visually refined To-Do List application.

## 9.4 Step 7: Testing and Validation

Finally, Cline validates the application's functionality by using an integrated browser to interact with the running server. This confirms that the front-end, back-end, and database connections are all working together correctly.

Note that the browser testing can give false negatives, meaning it detects errors even though the website is running correctly.



**Figure 39.** Cline testing the application via the browser.

The university's configuration allows you to swap between powerful models, such as Qwen or DeepSeek, throughout the development cycle. This ensures you always use the optimal tool

for the current task.

The screenshot displays the configuration panel for the Continue extension. It includes the following fields and options:

- API Provider:** OpenAI Compatible
- Base URL:** https://genai.uni-obuda.h
- OpenAI Compatible API Key:** A masked field with a note: "This key is stored locally and only used to make API requests from this extension."
- Model ID:** qwen3-coder:30b
- Custom Headers:** A section with an "Add Header" button and a checkbox for "Set Azure API version".
- MODEL CONFIGURATION:** A section with a checked checkbox for "Supports Images" and an unchecked checkbox for "Enable R1 messages format".
- Context Window Size:** 128000
- Max Output Tokens:** -1
- Input Price / 1M tokens:** openai-compat
- Output Price / 1M tokens:** qwen3-coder:30b

At the bottom, there are icons for adding, deleting, and refreshing configurations, and a "Plan" button.

**Figure 40.** Flexibility to switch between available models.

## 10 Continue Extension

This section details the setup and a common use case for the **Continue** extension, which serves as an excellent in-code assistant, especially for refactoring, explanation, and iterative changes.

### 10.1 Step 1: Installation and Configuration

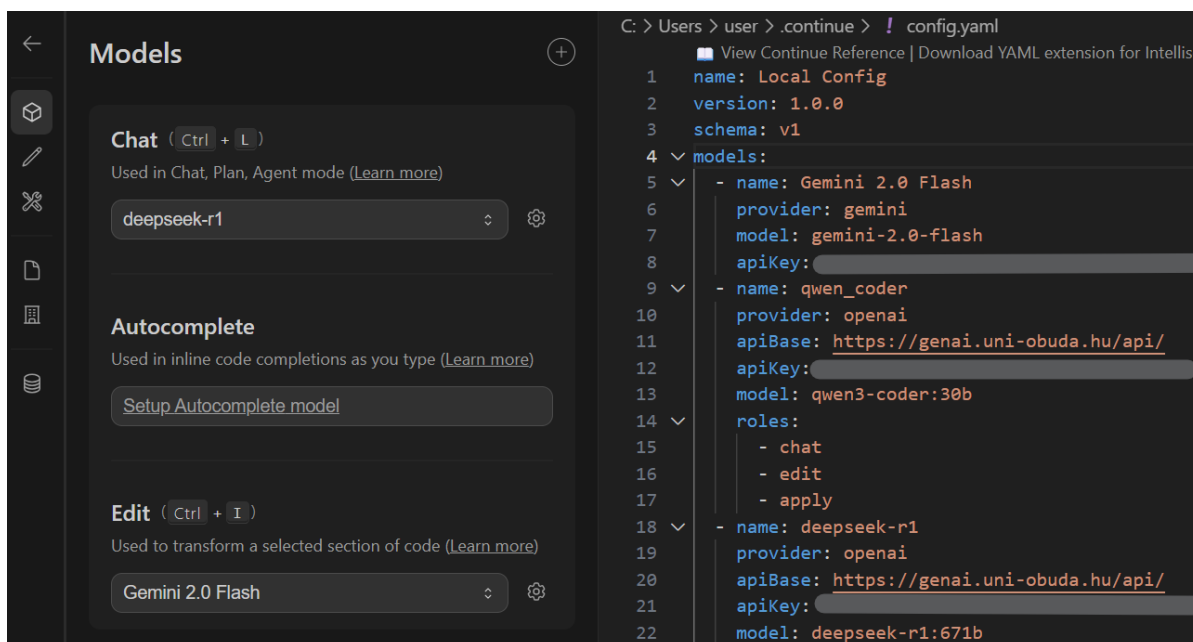
First, install the **Continue** extension from the VS Code Marketplace. Once installed, open the Continue panel from the Activity Bar.

To configure the extension, click the **gear icon** in the top-right corner.



**Figure 41.** Accessing the Continue settings panel.

This action opens a settings tab and, most importantly, the `config.yaml` file in your editor. You must manually edit this file to add the university's API endpoint (`apiBase`) and models. The settings tab provides a convenient graphical overview of the models defined in your YAML file.



**Figure 42.** The `config.yaml` file (right) and the graphical model setup (left).

### 10.1.1 Recommended Models for Continue

Continue is less sensitive to model formatting, allowing for broader compatibility.

- **Works Well:** `qwen-coder` models
  - **Result:** This combination is effective for direct code generation and modification tasks. We also plan to test `gpt-oss` models, which are expected to perform similarly well for these less-structured requests.

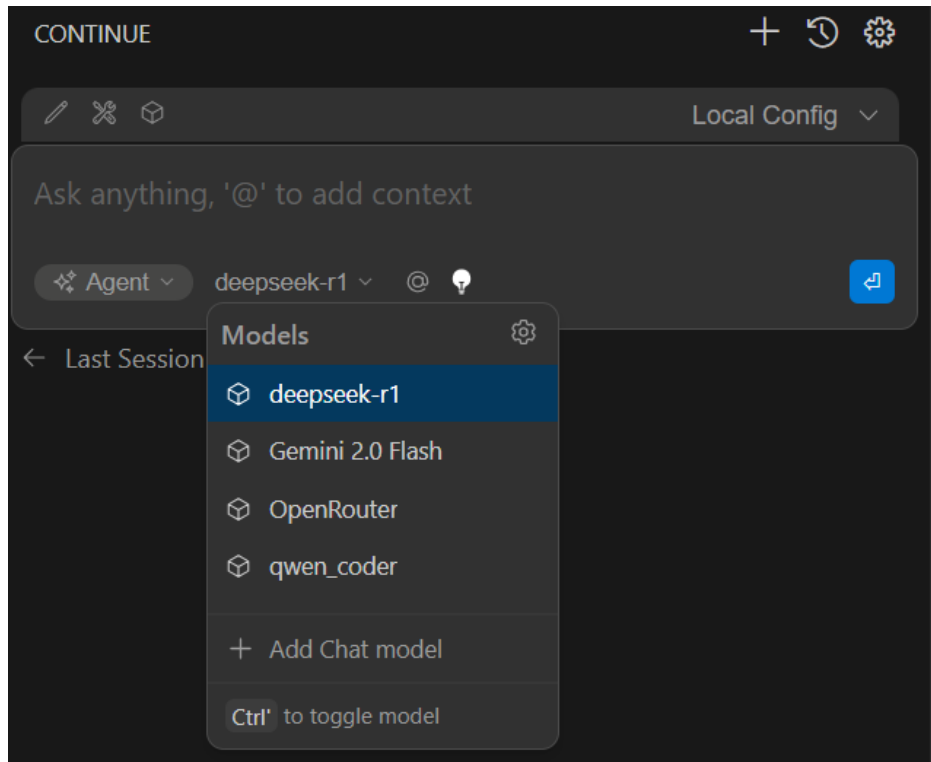
### 10.1.2 Example Configuration

Unlike Cline, Continue is configured using a `config.yaml` file. You can set up multiple models from our university's API and assign them different roles.

```
name: Local Config
version: 1.0.0
schema: v1
models:
  - name: qwen_coder
    provider: openai
    apiBase: https://genai.uni-obuda.hu/api/
    apiKey: sk-...
    model: qwen3-coder:30b
    roles:
      - chat
      - edit
      - apply
  - name: deepseek-r1
    provider: openai
    apiBase: https://genai.uni-obuda.hu/api/
    apiKey: sk-...
    model: deepseek-r1:671b
    roles:
      - chat
      - edit
      - apply
```

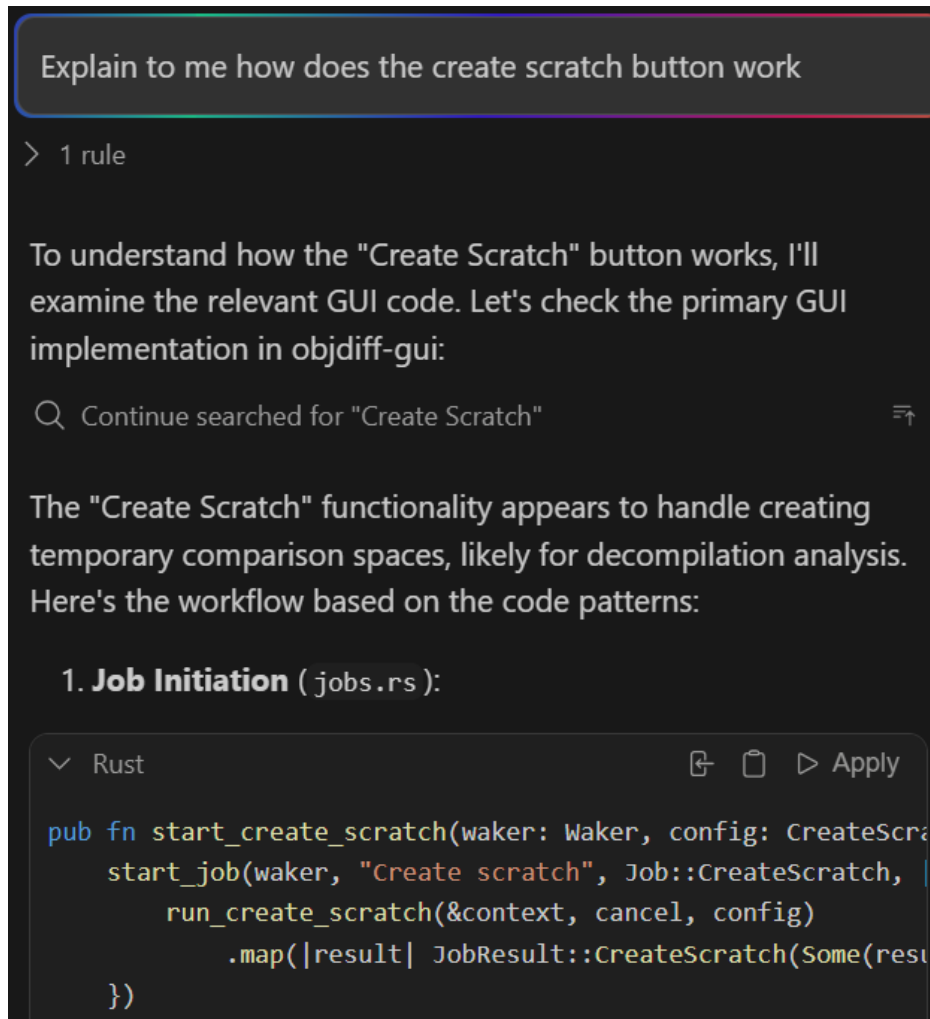
## 10.2 Step 2: Model Selection and Usage

Once your `config.yaml` is saved, the models you defined (e.g., `deepseek-r1`, `qwen_coder`) will be available for use. You can easily switch between your active models directly from the chat panel's dropdown menu.



**Figure 43.** Switching between configured models in the chat interface.

Continue excels at code-aware tasks. You can ask questions about your repository (e.g., "Explain how this button works"), and it will automatically search your local codebase. It then provides a detailed, code-based explanation by referencing the exact functions and files it found.



**Figure 44.** Continue answering a code-based question by searching the repository.

## 11 Tool Comparison

While the previous sections focused on configuring and using **Cline**, it is not the only AI coding assistant available. We have also tested the **Continue** VS Code extension with our university's API. This section provides a comparison based on our experiences to help users choose the right tool for their needs.

### 11.1 Cline: The Project-Level Agent

Cline operates as a "strong-thinking" agent. It excels by taking a high-level goal, breaking it down into a formal, step-by-step plan (as shown in the workflow example), and executing that plan sequentially.

- **Methodology:** Best for large, complex tasks that require significant planning, multi-file operations, and command execution (e.g., "create a full-stack application").

- **Token Usage:** This agent-based approach is powerful but consumes a high number of tokens, as it involves continuous, detailed prompting between Cline and the backing LLM.
- **LLM Sensitivity:** Cline is very sensitive to the output format of the language model it uses. It requires the LLM to follow a strict format for tasks like tool usage and plan generation (e.g., TODO list formatting). If the model's response deviates even slightly, Cline's process may fail.

### 11.1.1 Recommended Models for Cline

Our testing found that model compatibility is the most critical factor for success with Cline.

- **Highly Recommended:** `deepseek-r1:671b` (and similar DeepSeek variants)
  - **Result:** Works quite well. This model appears "smart" enough to consistently follow Cline's precise formatting and instructions, leading to a high success rate for complex tasks.
- **Not Recommended:** `qwen` and `gpt-oss` models
  - **Result:** These models struggled to follow Cline's strict instructions 100% of the time. This formatting incompatibility caused Cline's agentic workflow to fail frequently.

## 11.2 Continue: The In-Code Assistant

Continue operates more as a direct assistant for smaller, in-context tasks. It does not perform the same level of high-level, multi-step planning as Cline.

- **Methodology:** Best for modifications to existing code. It excels at tasks like "refactor this selected function," "add a new API endpoint to this file," or "explain this block of code."
- **Planning:** It involves less planning and is not recommended for complex, project-wide tasks.

## 11.3 Summary: Which Tool to Use?

- **Use Cline** when you need to perform **large-scale, planned tasks**, such as scaffolding a new project, implementing a complex multi-part feature, or executing commands.
  - **Recommended Model:** `DeepSeek`
- **Use Continue** for **smaller, iterative changes** within your existing codebase, such as refactoring, adding individual functions, or generating documentation.
  - **Recommended Model:** `Qwen Coder`