

Grokking the Agents

How LLM Coding Assistants Actually Work Under the Hood — A Case Study with a Simple Example

Oliver Dürr, Sep 2025

Introduction

Currently (Sep 2025) there are many Agents for Coding

Agents (Selection)

Plugins inside IDEs

- GitHub Copilot Agent (VS Code)
- Cursor (VS Code fork “AI-First”)
- ➔ **Roo Code (VS Code extension, open-source)**

Command-line Agents

- Copilot CLI
- Windsurf CLI
- Claude CLI (Anthropic)

Hybrid (both plugin + CLI)

- Codex OpenAI

• ...

LLMs and Providers

Cloud / Commercial LLMs

- **GPT-5** (OpenAI) – flagship, multitask/responsive model
- **Claude 4** (Anthropic) – includes Code-oriented variants
- **Gemini 2.5 Pro/Flash** (Google DeepMind) – multimodal
- **Grok 4** (xAI) – recent fast-reasoning model

Open-Source LLMs (Local / Self-hosted)

- **Code LLaMA** (Meta) – code-specialized LLaMA-2 variants (7B–70B)
- **Llama 4** (Meta) – mixture-of-experts, large context, multimodal, open-licensed
- **DeepSeek Coder V2** – open-source MoE code models, handles 338 languages and 128K context



We will use [Roo-Code](#) since it's open source, so we can look into it.

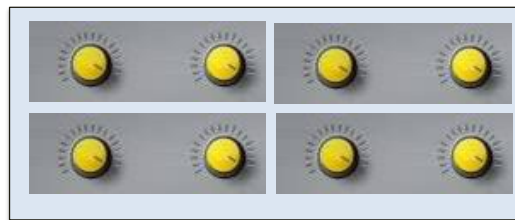


Recap (principle of LLM)

LLMs are probabilistic models

The weather is
really nice today.
I'm thinking
about going for a

$y_{t:0}$ (Context)

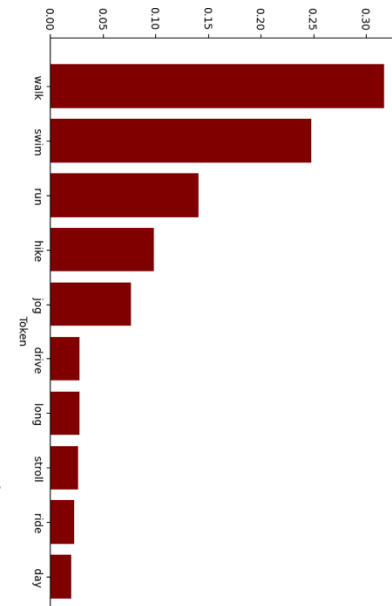


A model with
parameters θ



y_{t+1}

Probability for y



In math the output of a LLM is

$$p_{\theta}(y_{t+1}|y_{t:0})$$

GTP (Generative Pretrained Transformer) and Llama are causal/autoregressive

Creating the response sequence



Describe your technology in one sentence.



I am a Generative Language model based on the transformer architecture that uses self-attention mechanisms and is trained with maximum likelihood principle to predict the next token in a sequence of tokens and generate new text.



At each step, the model samples the next token from $p(y_{t+1}|y_{t:0})$
or takes most likely token (when $T = 0$)

Steps 1: Describe your technology in one sentence. → I

Steps 2: Describe your technology in one sentence. I → am

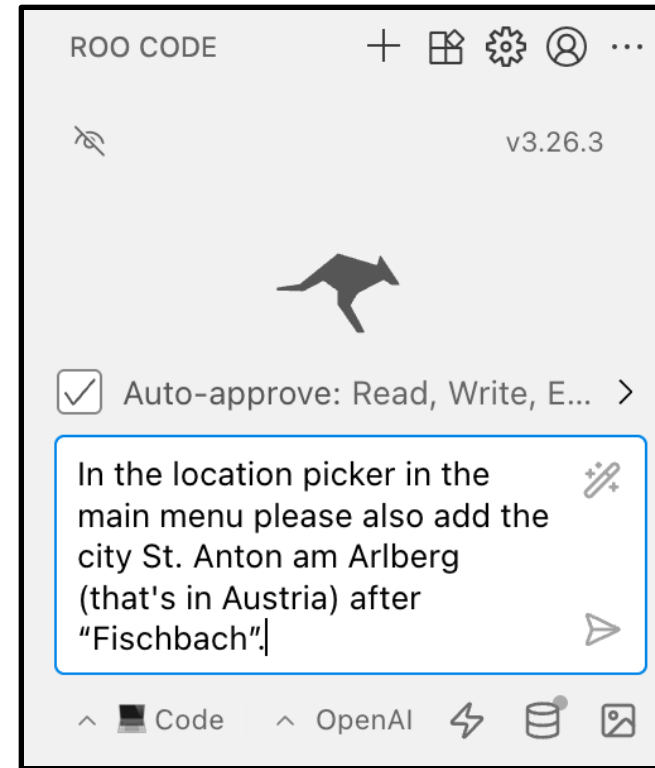
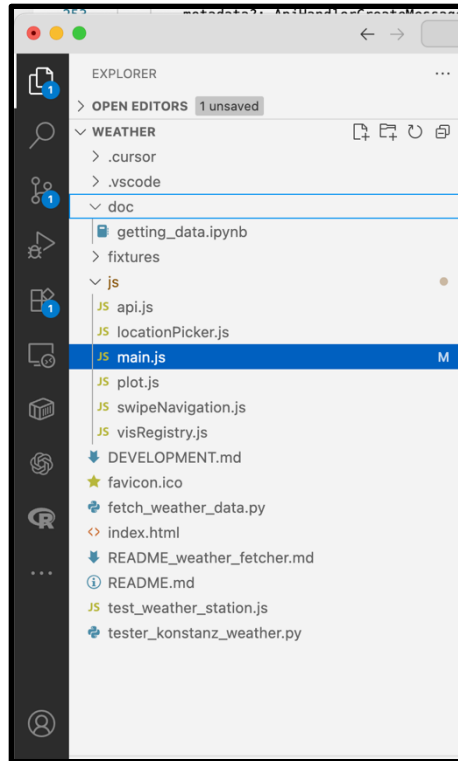
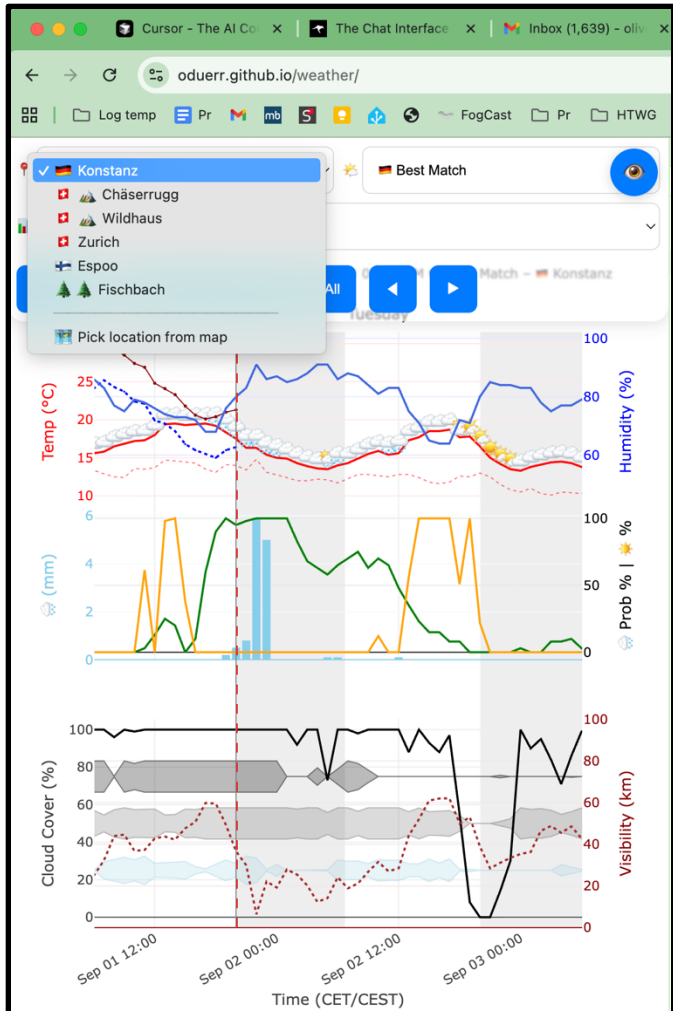
Steps 3: Describe your technology in one sentence. I am → a

Steps 4: Describe your technology in one sentence. I am a → generative

Step 36 Describe your technology in one sentence. I am a ... new text. → END

Case Study

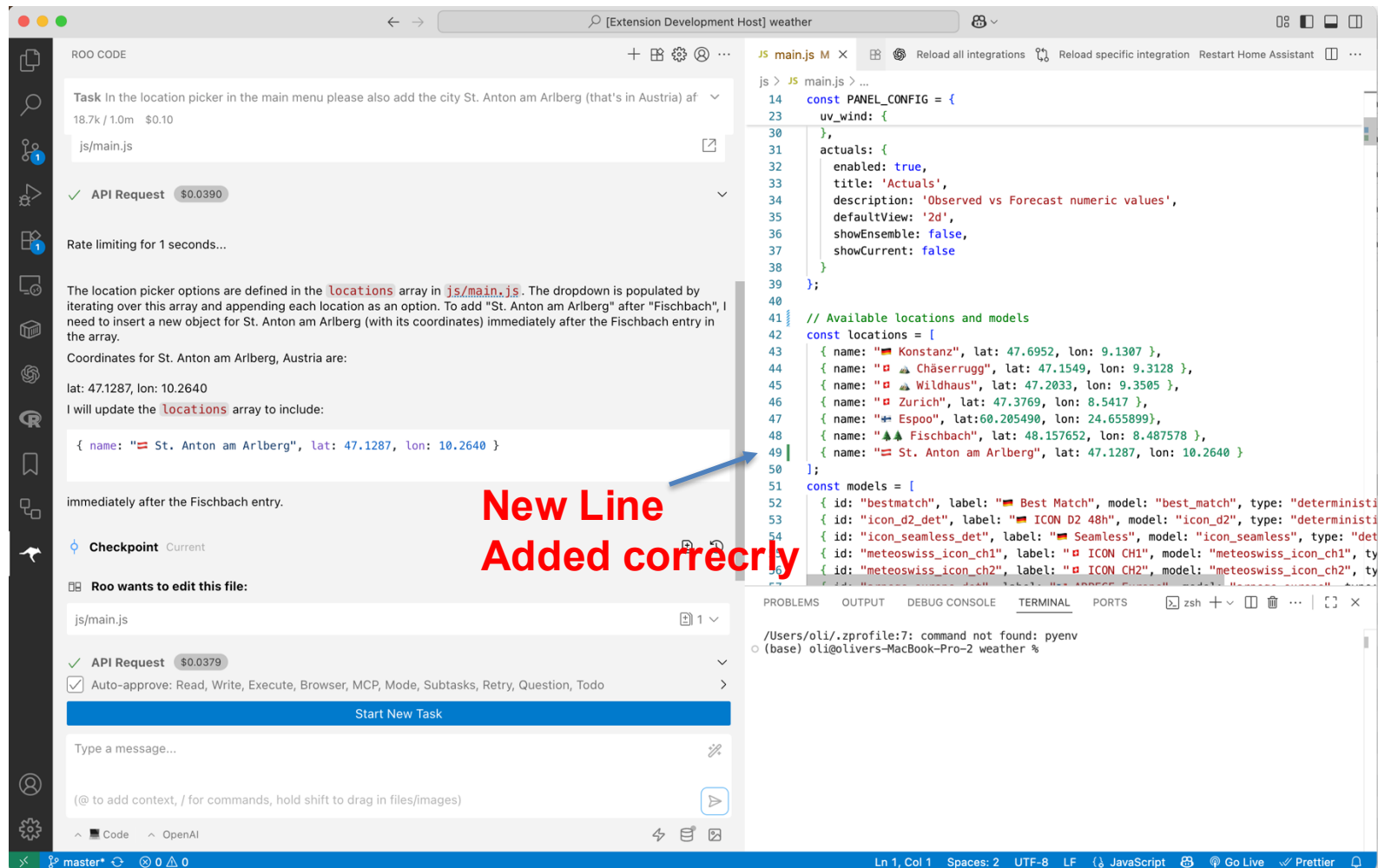
Very Simple Task in existing code base



Mini refactoring of a small, partly vibe-coded project. Showing results of weather models for certain locations.

<https://oduerr.github.io/weather/> (gh tag is demo-baseline)

Result, using Roo Code and gpt-4.1



Only one line of code has been changed, perfect!
Using mistral-instruct 8b and deepseek-r1:8b via Ollama did not work out!

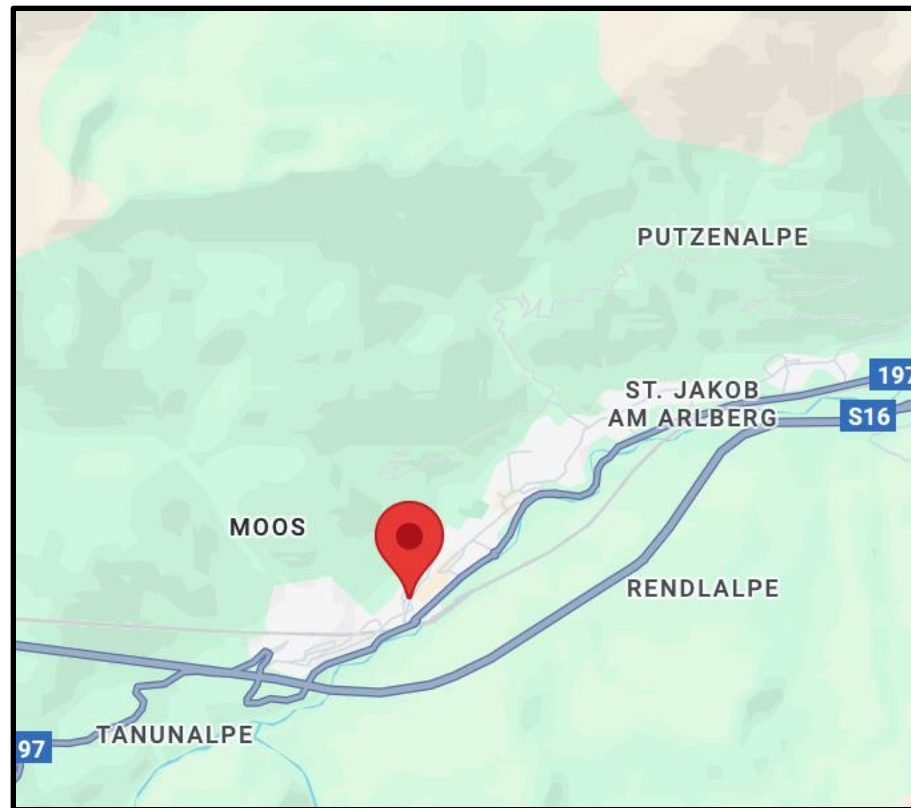
Be skeptical and don't trust the LLMs!

- The line added was
{ name: "AT St. Anton am Arlberg",
lat: 47.1287, lon: 10.2640 }

Generate google maps link for lat: 47.1287, lon: 10.2640

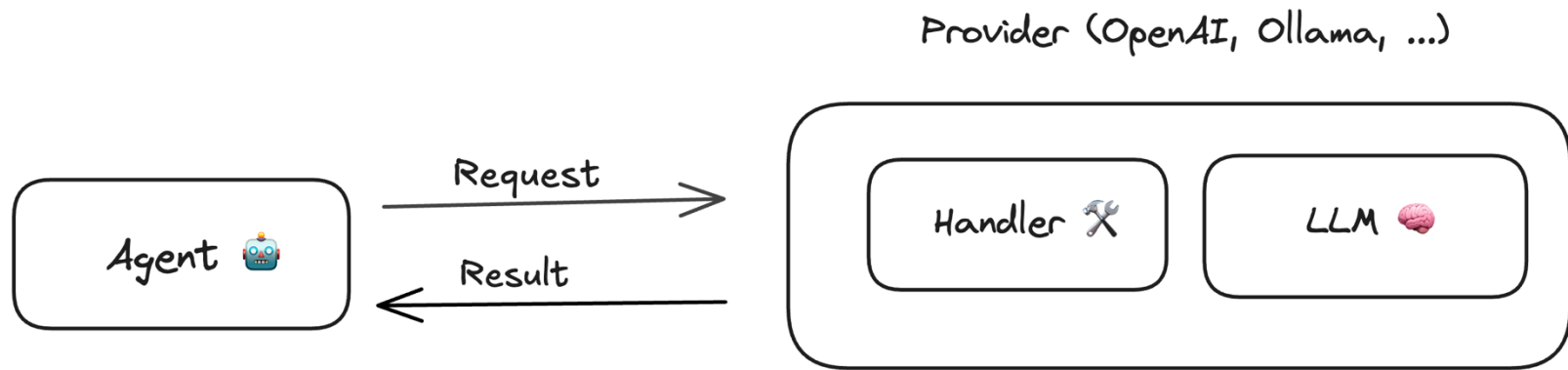
- This time it worked but I also had runs with hallucinations.

<https://www.google.com/maps/search/?api=1&query=47.1287%2C10.2640>



How does this work (highlevel)

How Agents Communicate with LLM Providers



- The Agent 🤖 (Roo/Cursor) sends a request to the Provider (e.g. OpenAI, Ollama, ...)
- The Handler 🛠️ makes a single string from the request (e.g. system prompt / user prompt*), tokenizes it to $y_{t:0}$ and sends it to the LLM.
- The LLM 🧠 calculates the probability distribution $p(y_{t+1}|y_{t:0})$ of next token y_{t+1}
- The Handler repeatably samples the tokens and sends them it to the agent

*Also, might add context from previous conversations.

Creating the prompt (Conceptually)

- The system prompt and the user prompt (optionally other prompts) are sent to the provider
- A **single prompt / string** is then built using a template like

```
<|system|>
You are Roo, a helpful...
<|user|>
Please add the location St. Anton...
```
- Important: after all a single prompt enters the LLM
- This prompt is **tokenized** to $y_{t:0}$ and feed through to the LLM.
- The handler then **repeatedly samples tokens** until the response is complete.

Under the Hood

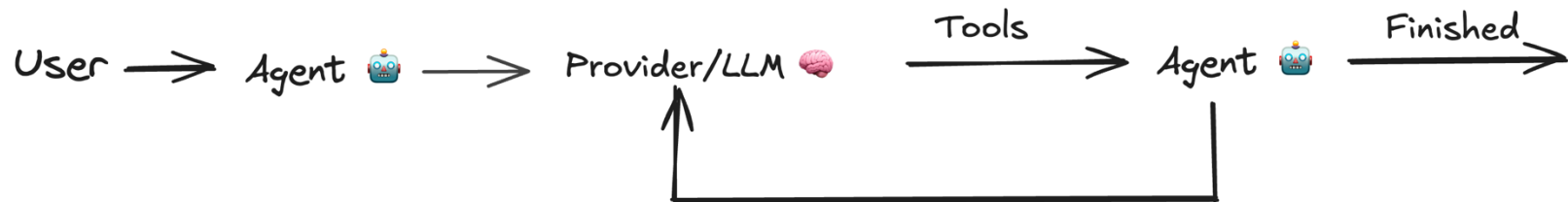
- Providers use **optimizations and tricks** beyond simple prompt assembly:
 - **Prompt caching** (avoid re-sending repeated context).
 - **KV-cache reuse** (store hidden states to accelerate)
 - Caching internals of the transformer
 - **Context management** (truncate or select relevant history)
 - Especially when we reach the limit of the context window
 - **Batching & parallelism** (serve many requests efficiently).
 - **Speculative decoding & other sampling tricks** (speed up generation)
 - E.g. a smaller model and if results are uncertain, fallback to the larger model.

Likely providers such as OpenAI also have a secret sauce.

The agentic loop

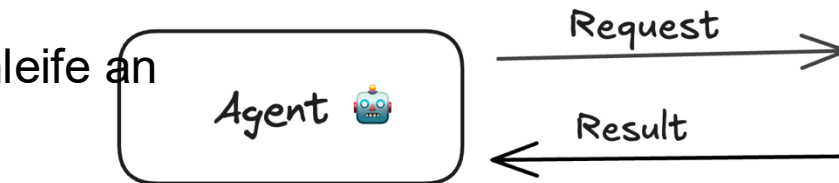
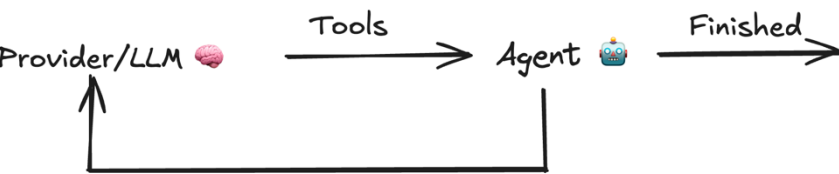
- This usually happens in a loop (aka Reason Act Loop)
 1. The Agent sends **the system prompt** (“You are Roo, a helpful...”) and the **user prompt** (“Please add the location St. Anton...”) to the Provider.
 2. The Provider returns the response of the LLM to the Agent (**usually request to use a tool**).
 3. The Agent fulfills the request (e.g. calls the tool) and sends the result back to the LLM.

Step 2 and 3 are iterated until success (kind of while loop)*

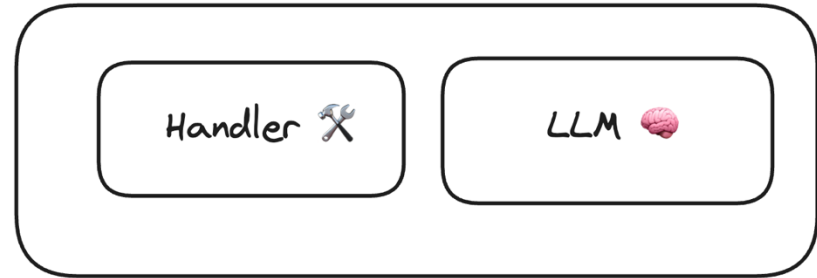


*In principle, also other non-loop-like calls are possible.

Eavesdropping the communication



Provider (OpenAI, Ollama, ...)



Here we log the json file for each communication

```

// native-ollama.js
let Ollama$1 = class Ollama {
  async processStreamableRequest(endpoint, request) {
    request.stream = request.stream ?? false;
    const host = `${this.config.host}/api/${endpoint}`;

    // ----- logging setup (single timestamp per request) -----
    const t0 = new Date();
    //const stamp = isoStamp(t0); // e.g. 2025-08-31T09-15-22.184Z
    //const reqDir = path.join(dayDir, stamp);
    // get both reqDir and stamp from the helper
    const { reqDir, stamp } = await makeRequestDirs();
    // if you need the dayDir as well:
    const dayDir = path.dirname(reqDir);

    // Write outgoing request once
    await writeJson(path.join(reqDir, "request.json"), {
      timestamp: t0.toISOString(),
      host,
      endpoint,
      request
    });

    if (request.stream) {
      const abortController = new AbortController();
      const response2 = await post(this.fetch, host, request, {
        signal: abortController.signal,
        headers: this.config.headers
      });
    }
  }
}










```

Writing out the request and result (shown is Ollama)

[Installing Roo-Code in development mode](#)

Three rounds happend

- Using openAI's GPT-4.1, three rounds in the loop happened (see json files)

 openai-log-viewer.html	Yesterday at 22:41	35 KB	HTML text
▼  llm_input	Yesterday at 21:50	--	Folder
 openai-native-input-2025-09-01T17-56-34-674Z.json	Yesterday at 19:56	44 KB	JSON File
 openai-native-input-2025-09-01T17-55-33-852Z.json	Yesterday at 19:55	76 KB	JSON File
 openai-native-input-2025-09-01T17-54-33-066Z.json	Yesterday at 19:54	44 KB	JSON File
▼  llm_output	Yesterday at 21:50	--	Folder
 openai-native-output-2025-09-01T17-56-43-623Z.json	Yesterday at 19:56	874 bytes	JSON File
 openai-native-output-2025-09-01T17-55-42-902Z.json	Yesterday at 19:55	2 KB	JSON File
 openai-native-output-2025-09-01T17-54-39-349Z.json	Yesterday at 19:54	1 KB	JSON File

- openai-log-viewer.html is a vibe-coded simple viewer for the json files
- Files are available at
- <https://www.dropbox.com/scl/fo/8gtzr2qi3boil0r8fhiqy/AK-dfu4WPI7M9PXlufd348M?rlkey=z46p92u1ps2tao7rtarg7wd0x&st=s3zjdnwd&dl=0>

Step 1 Agent to provider/LLM (Principle) →

Raw Request

44167 characters

```
{
  "requestBody": {
    "model": "gpt-4.1",
    "input": [
      {
        "role": "user",
        "content": [
          {
            "type": "input_text",
            "text": "<task>\nIn the location picker in the main menu please also add the city St. Anton am Arlberg (that's in Austria) after",
          },
          {
            "type": "input_text",
            "text": "<environment_details>\n# VSCode Visible Files\njs/main.js\n\n# VSCode Open Tabs\njs/main.js\n\n# Current Time\nCurrent",
          }
        ]
      }
    ],
    "stream": true,
    "store": true,
    "instructions": "You are Roo, a highly skilled software engineer with extensive knowledge in many programming languages, frameworks, des",
    "temperature": 0,
    "max_output_tokens": 32768
  },
  "model": {
    "id": "gpt-4.1",
    "info": {
      "maxTokens": 32768,
      "contextWindow": 1047576,
      "supportsImages": true,
      "supportsPromptCache": true,
      "inputPrice": 2,
      "outputPrice": 8,
      "cacheReadsPrice": 0.5,
      "supportsTemperature": true
    },
    "format": "openai",
    "maxTokens": 32768,
    "temperature": 0
  },
  "metadata": {
    "mode": "code",
    "taskId": "43f2c6e9-cfd9-4813-b90b-d9ef7329d341"
  }
}
```

User msg

System msg

Round 1 System msg 41415 chars, first entries

You are Roo, a highly skilled software engineer with extensive knowledge in many programming languages, frameworks, design patterns, and best practices.

==== MARKDOWN RULES

...

TOOL USE

You have access to a set of tools that are executed upon the user's approval. You can use one tool per message, and will receive the result of that tool use in the user's response. You use tools step-by-step to accomplish a given task, with each tool use informed by the result of the previous tool use.

Round 1 System Msg cont'd (Tools)

...

Tools

read_file

Description: Request to read the contents of one or more files. The tool outputs line-numbered content (e.g. "1 | const x = 1") for easy reference when creating diffs or discussing code. Supports text extraction from PDF and DOCX files, but may not handle other binary files properly.

****IMPORTANT:** You can read a maximum of 5 files in a single request. ****** If you need to read more files, use multiple sequential read_file requests.

Parameters:

- args: Contains one or more file elements, where each file contains:
 - path: (required) File path (relative to workspace directory /Users/oli/Documents/GitHub/weather)

Usage:

<read_file>

<args>

<file>

<path>path/to/file</path>

</file>

</args>

</read_file>

Round 1 System Msg cont'd

Addition Tools (in system prompt, GPT-summarized)

From the JSON you showed (the system prompt's **Tools** section), the LLM is allowed to call these tools:

read_file — read up to 5 files (with line numbers).

fetch_instructions — fetch “how-to” instructions for tasks (e.g., `create_mcp_server`, `create_mode`).

search_files — regex search across a directory, returns matches with context.

list_files — list directory contents (optionally recursive).

list_code_definition_names — extract function/class/method names from files/dirs.

apply_diff — make **surgical** search/replace edits in a file via precise diff blocks.

write_to_file — create/overwrite a file with full content (single-shot).

insert_content — insert new lines into a file at a given line (or append).

search_and_replace — find/replace (literal or regex) within a file, with optional line range.

execute_command — run a CLI command (with optional working directory).

ask_followup_question — ask the user for missing details (with suggested answers).

attempt_completion — finalize the task and present the result (only after confirming prior tool success).

switch_mode — switch to another mode (code / ask / architect / debug / orchestrator).

new_task — start a new task instance in a chosen mode.

update_todo_list — replace the whole TODO checklist with updated statuses.

The green one have been picked up by the agent.

Round 1 User Msg

<task>
In the location picker in the main menu please also add the city St. Anton am Arlberg (that's in Austria) after “Fischbach”.
</task>

<environment_details>
VSCode Visible Files
js/main.js

VSCode Open Tabs
js/main.js

Current Time
Current time in ISO 8601 UTC format: 2025-09-01T17:54:33.033Z
User time zone: Europe/Berlin, UTC+2:00

....
Current Workspace Directory (/Users/oli/Documents/GitHub/weather) Files
DEVELOPMENT.md
favicon.ico
fetch_weather_data.py
index.html
README_weather_fetcher.md
README.md
test_weather_station.js
tester_konstanz_weather.py
.cursor/
.vscode/
doc/
doc/getting_data.ipynb
</environment_details>

Round 1 Response from LLM (GPT-4.1)

<thinking>

The user wants to add "St. Anton am Arlberg" (Austria) to the location picker in the main menu, specifically after "Fischbach". Based on the file structure, the location picker logic is likely in [js/locationPicker.js](js/locationPicker.js:1), but it could also be referenced or rendered in [js/main.js](js/main.js:1). Since the user has both files available and the picker is likely defined or at least configured in `locationPicker.js`, I should read both files to find where the list of locations is defined and how to insert the new city after "Fischbach".

I will read both [js/locationPicker.js](js/locationPicker.js:1) and [js/main.js](js/main.js:1) to get the full context and ensure I update the correct place.

</thinking>

<read_file>

<args>

<file>

<path>js/locationPicker.js</path>

</file>

<file>

<path>js/main.js</path>

</file>

</args>

</read_file>

Tool call for the two files

Summary of round 1

Usage & Costs				-
INPUT TOKENS	OUTPUT TOKENS	TOTAL TOKENS	TOTAL COST	
8,950	230	9,180	\$0.0197	

The agent now uses the tool / executes the command.
Here it reads the 2 files (js/locationPicker.js) and (js/main.js)
and appends them to the user msg

Round 2 (raw request)

```
76308 characters

{
  "requestBody": {
    "model": "gpt-4.1",
    "input": [
      {
        "role": "user",
        "content": [
          {
            "type": "input_text",
            "text": "[read_file for 'js/locationPicker.js', 'js/main.js'] Result:"
          },
          {
            "type": "input_text",
            "text": "<files>\n<file><path>js/locationPicker.js</path>\n<content lines=\"1-323\">\n  1 | /**\n  2 | * Location Picker Module
          },
          {
            "type": "input_text",
            "text": "<environment_details>\n# VSCode Visible Files\njs/main.js\n\n# VSCode Open Tabs\njs/main.js\n\n# Current Time\nCurrent
        ]
      }
    ],
    "stream": true,
    "store": true,
    "instructions": "You are Roo, a highly skilled software engineer with extensive knowledge in many programming languages, frameworks, des
    "temperature": 0,
    "max_output_tokens": 32768,
    "previous_response_id": "resp_68b5ddd9c7d4819193fc38d2d3b94d1f027f2fdceca5f9b3"
  },
  "model": {
    "id": "gpt-4.1",
    "info": {
      "maxTokens": 32768,
      "contextWindow": 1047576,
      "supportsImages": true,
      "supportsPromptCache": true,
      "inputPrice": 2,
      "outputPrice": 8,
      "cacheReadsPrice": 0.5,
      "supportsTemperature": true
    },
    "format": "openai",
    "maxTokens": 32768,
    "temperature": 0
  },
  "metadata": {
    "mode": "code",
    "taskId": "43f2c6e9-cfd9-4813-b90b-d9ef7329d341"
  }
}
```

User msg (new)

System msg, stays the same

Round 2 (User prompt, with additional file information)

[read_file for 'js/locationPicker.js', 'js/main.js'] Result:

<files>

<file><path>js/locationPicker.js</path>

<content lines="1-323">

```
1 | /**
2 |  * Location Picker Module
3 |  * Provides map-based location selection functionality
4 |  */
```





...

<file><path>js/main.js</path>

<content lines="1-482">

```
1 | let measured_temp = null;
2 | let measured_water_temp = null;
```

...

```
41 | // Available locations and models
42 | const locations = [
43 |   { name: "DE Konstanz", lat: 47.6952, lon: 9.1307 },
44 |   { name: "CH  Chäserrugg", lat: 47.1549, lon: 9.3128 },
45 |   { name: "CH  Wildhaus", lat: 47.2033, lon: 9.3505 },
46 |   { name: "CH Zurich", lat: 47.3769, lon: 8.5417 },
47 |   { name: "FI Espoo", lat: 60.205490, lon: 24.655899},
48 |   { name: "  Fischbach", lat: 48.157652, lon: 8.487578 },
49 | ];
```

Round 2: Response

<thinking>

The location picker options are defined in the `locations` array in [`js/main.js`](js/main.js:42-49). The dropdown is populated by iterating over this array and appending each location as an option. To add "St. Anton am Arlberg" after "Fischbach", I need to insert a new object for St. Anton am Arlberg (with its coordinates) immediately after the Fischbach entry in the array.

Coordinates for St. Anton am Arlberg, Austria are:

lat: 47.1287, lon: 10.2640

I will update the `locations` array to include:

```
```js
{ name: "AT St. Anton am Arlberg", lat: 47.1287, lon: 10.2640 }
```
```

immediately after the Fischbach entry.

</thinking>

<apply_diff>

<path>js/main.js</path>

<diff>

<<<<<< SEARCH

:start_line:47

{ name: "FI Espoo", lat:60.205490, lon: 24.655899},

{ name: "🌲🌲 Fischbach", lat: 48.157652, lon: 8.487578 }

];

=====

{ name: "FI Espoo", lat:60.205490, lon: 24.655899},

{ name: "🌲🌲 Fischbach", lat: 48.157652, lon: 8.487578 },


{ name: "AT St. Anton am Arlberg", lat: 47.1287, lon: 10.2640 }

];

>>>>>> REPLACE

</diff>

</apply_diff>



It knows the location of
Arlberg, out of the box.
This needs checking!



That's a tool call

Round 2: Summary

Usage & Costs

—

INPUT TOKENS

18,009

OUTPUT TOKENS

367

TOTAL TOKENS

18,376

TOTAL COST

\$0.0390

The diff is executed and send to the provider/LLM

Round 3 (Finishing off)

User (System stays like it is)

1113 characters

[apply_diff for 'js/main.js'] Result:

```
<file_write_result>
<path>js/main.js</path>
<operation>modified</operation>
<notice>
<i>You do not need to re-read the file, as you have seen all changes</i>
<i>Proceed with the task using these changes as the new baseline.</i>
</notice>
</file_write_result>
```

274 characters

```
<attempt_completion>
<result>
"St. Anton am Arlberg" (Austria) has been added to the location picker immediately after "Fischbach" in the main menu. The new entry uses the
correct coordinates and will now appear in the dropdown for selection.
</result>
</attempt_completion>
```

Costs

- Costs For the single line change

ROO CODE

Task

In the location picker in the main menu please also add the city St. Anton am Arlberg (that's in Austria) after "Fischbach".

Context Length

18.7k

1

✖

Tokens

↑ 45.6k ↓ 661

API Cost

\$0.10

Size

192 kB

$\text{Costs} = 2 * 0.0456 + 8 * 0.000661 = 0.096488 \text{ Dollar}$

Model

gpt-4.1

Context Window: 1,047,576 tokens

Max output: 32,768 tokens

✓ Supports images

✗ Does not support computer use

✓ Supports prompt caching

Input price: \$2.00 / 1M tokens

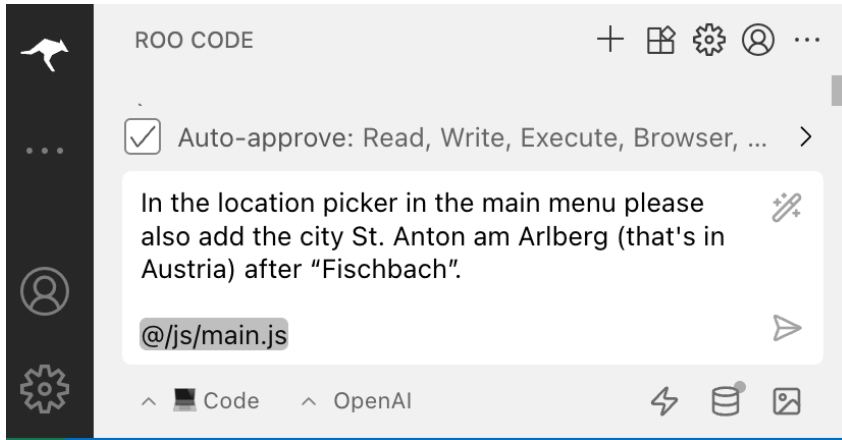
Output price: \$8.00 / 1M tokens

Cache reads price: \$0.50 / 1M tokens

GPT5 is (currently)
cheaper! Also
codex-mini-latest \$0.03

Practical takeaways

Adding Content



Adds the content of
js/main.js to the user prompt

USER

21553 chars

21553 characters

```
<task>
In the location picker in the main menu please also add the city St. Anton am Arlberg (that's in Austria) after "Fischbach".

'js/main.js' (see below for file content)
</task>

<file_content path="js/main.js">
1 | let measured_temp = null;
2 | let measured_water_temp = null;
3 |
```

GPT4.1 then does the job in a single round (Ollma models still fail).

Adding Content (My Workflow)

- Start with a requirement.md file describing your project
 - When starting from scratch, I talk to ChatGPT, develop ideas, and come up with the file.
 - Having a project might help when an agent looks into it can describe it (untested)
- Use this requirement document to also describe changes/issues.
From my project

Round 3.8

- In the discreet version indicator at the bottom add The Chinese label “莉娜老师的版本”
- Past Sessions / Replay UI changes
 - When listing sessions, show for each
YYYY-MM-DD-hh:mm • <finished>/<total> • <mistakes> mistakes • status: complete|incomplete
 - * Allow inline renaming of session titles, persist changes in session summaries.
 - * Move Export all sessions and Import sessions buttons into the Past Sessions / Replay dialog.
 - * For each session in the list, provide buttons:
 - Replay mistakes
 - Resume (if status = incomplete)
 - Rename
 - Delete

Round 3.9

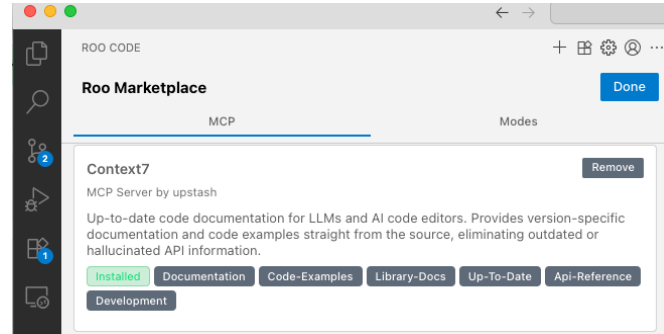
- Include a Back button to navigate to the previous card.
- Support keyboard arrow keys: ← (previous), → (next).
- When navigating back to a card previously marked as a mistake, clearly flag it as such (e.g., red border, icon).
- Allow changing the status of a previously marked mistake to correct; update the session log, mistake count, and replay list accordingly.

Adding / Handling content (Cursor)

- Use Cursor Rules
 - Define project-wide behaviors like “always use TypeScript import”
 - These are always included in the context
- Add Context
 - You can webpage with documentation and best practices (use @)
- Context Management by Cursor
 - Automatic context building: Cursor includes files you open or edit into the LLM context.
 - Smart truncation: If files are too large, only the most relevant parts are sent.
 - Session memory: Keeps track of recent edits & conversations, discards irrelevant history uses RAG (cursor’s secret sauce)

Adding Tools via MCP Protocol (Roo-Code)

- You can add your own tools using the MCP protocol (here context7)



- The added MCP Server now appears in the System Instruction.

SYSTEM INSTRUCTIONS

47238 chars

```
## context7 (`npx -y @upstash/context7-mcp`)  
  
### Instructions  
Use this server to retrieve up-to-date documentation and code examples for any library.  
  
### Available Tools  
- resolve-library-id: Resolves a package/product name to a Context7-compatible library ID and returns a list of matching libraries.  
  
You MUST call this function before 'get-library-docs' to obtain a valid Context7-compatible library ID UNLESS the user explicitly provides a library ID in the format '/org/project' or '/org/project/version' in their query.  
  
Selection Process:  
1. Analyze the query to understand what library/package the user is looking for  
2. Return the most relevant match based on:  
- Name similarity to the query (exact matches prioritized)  
- Description relevance to the query's intent  
- Documentation coverage (prioritize libraries with higher Code Snippet counts)  
- Trust score (consider libraries with scores of 7-10 more authoritative)  
  
Response Format:  
- Return the selected library ID in a clearly marked section  
- Provide a brief explanation for why this library was chosen
```

Special MCP Servers

- Context7 has documentation of 15k libraries (including R's ggplot :-)
 - Make sense to install
 - For Cursor install via <https://docs.cursor.com/en/tools/mcp>
 - Use something like “use context7” in prompt

Costs

I did several agentic / vibe coding projects to get a feeling. About 100\$, but sponsored.

Included Usage Summary

Current billing period: Jul 27, 2025 - Aug 27, 2025

Pro comes with at least \$20 of included usage per month. We work closely with the model providers to make this monthly allotment as high as possible. You'll be notified in-app when you're nearing your monthly limit.

During the promotional release period, initial GPT-5 usage does not count towards usage limits.

| MODEL | INPUT (W/ CACHE WRITE) | INPUT (W/O CACHE WRITE) | CACHE READ | OUTPUT | TOTAL TOKENS | API COST | COST TO YOU |
|--------------------------|------------------------|-------------------------|-------------|-----------|--------------|----------|-------------|
| gpt-5-high | 0 | 12,456,568 | 71,989,868 | 786,358 | 85,232,794 | \$30.22 | \$0 |
| gpt-5 | 0 | 12,522,734 | 30,413,821 | 324,782 | 43,261,337 | \$22.20 | \$0 |
| auto | 2,916,535 | 961,669 | 37,643,677 | 258,957 | 41,780,838 | \$15.81 | \$0 |
| gpt-5-fast | 0 | 524,755 | 4,945,088 | 26,819 | 5,496,662 | \$3.08 | \$0 |
| gpt-5-high-fast | 0 | 80,868 | 2,725,376 | 13,922 | 2,820,166 | \$1.16 | \$0 |
| claude-4-sonnet-thinking | 21,999 | 184 | 240,249 | 3,322 | 265,754 | \$0.20 | \$0 |
| gpt-5-2025-08-07 | 0 | 77,540 | 133,593 | 16,503 | 227,636 | \$0.07 | \$0 |
| Total | 2,938,534 | 26,624,318 | 148,091,672 | 1,430,663 | 179,085,187 | \$72.75 | \$0 |

Commercial vs. Locally Hosted LLM

- For the adding Arlberg task grok-code-fast-1 needed approx 12 seconds (using cursor), also running linter test

| | |
|--------------|------------------|
| Cache Read | 1,448,448 |
| Cache Write | 0 |
| Input | 3,477 |
| Output | 2,043 |
| <hr/> | |
| Total | 1,453,968 |

| | | | | |
|-----------------|------------------|----------|---------------|----------------------------|
| Sep 1, 10:40 PM | grok-code-fast-1 | Included | 1.5M tokens ⓘ | \$0.03 Included |
|-----------------|------------------|----------|---------------|----------------------------|

- Local LLM could not do the task
 - only tried 8B models
 - and did not try hard
- My belief, just the matter of time when small specialized local LLMs can better use the tools the agents provide them.

Outlook Other Agentic Workflows

- So far, coding agents are very successful for coding
 - Massive R&D efforts are currently focused on coding agents
 - A large motivation for GPT5 was to catch up with Anthropic Claude models
- These workflows will soon expand into other fields, such as scientific research, education, and medicine (IMO).
- We are exploring this future ourselves: together with colleagues, we submitted a grant proposal on agentic workflows that enable LLMs to use causal reasoning tools for the medical domain.