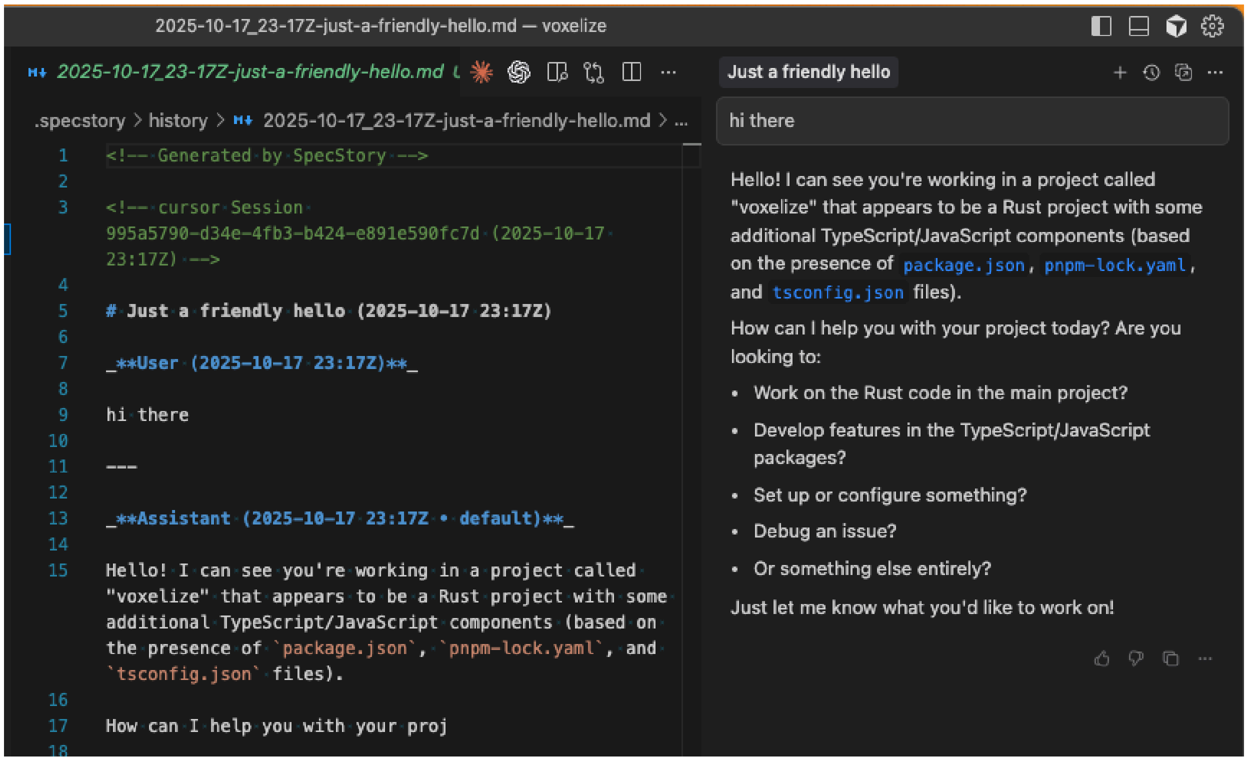
Instructions for Installing SpecStory

SpecStory is a Cursor extension and CLI that automatically captures your AI coding sessions locally as markdown files. It only records conversations and interactions that occur while a file is visible and open in your editor. All session data is stored locally in the .specstory/ directory within your project folder.

SpecStory Cursor Extension

Follow the following steps to install the **SpecStory Cursor extension** manually from a .vsix file.



2025-10-17_23-17Z-just-a-friendly-hello.md — voxelize

2025-10-17_23-17Z-just-a-friendly-hello.md

.specstory > history > 2025-10-17_23-17Z-just-a-friendly-hello.md > ...

```
<!-- Generated by SpecStory -->

<!-- cursor Session
995a5790-d34e-4fb3-b424-e891e590fc7d (2025-10-17
23:17Z) -->

# Just a friendly hello (2025-10-17 23:17Z)

_**User (2025-10-17 23:17Z)**_

hi there

---

_**Assistant (2025-10-17 23:17Z • default)**_

Hello! I can see you're working in a project called
"voxelize" that appears to be a Rust project with some
additional TypeScript/JavaScript components (based on
the presence of `package.json`, `pnpm-lock.yaml`, and
`tsconfig.json` files).

How can I help you with your proj
```

Just a friendly hello

hi there

Hello! I can see you're working in a project called "voxelize" that appears to be a Rust project with some additional TypeScript/JavaScript components (based on the presence of `package.json`, `pnpm-lock.yaml`, and `tsconfig.json` files).

How can I help you with your project today? Are you looking to:

- Work on the Rust code in the main project?
- Develop features in the TypeScript/JavaScript packages?
- Set up or configure something?
- Debug an issue?
- Or something else entirely?

Just let me know what you'd like to work on!

1. Download the Extension File
Get the installation file: specstory-vscode-0.23.0.vsix

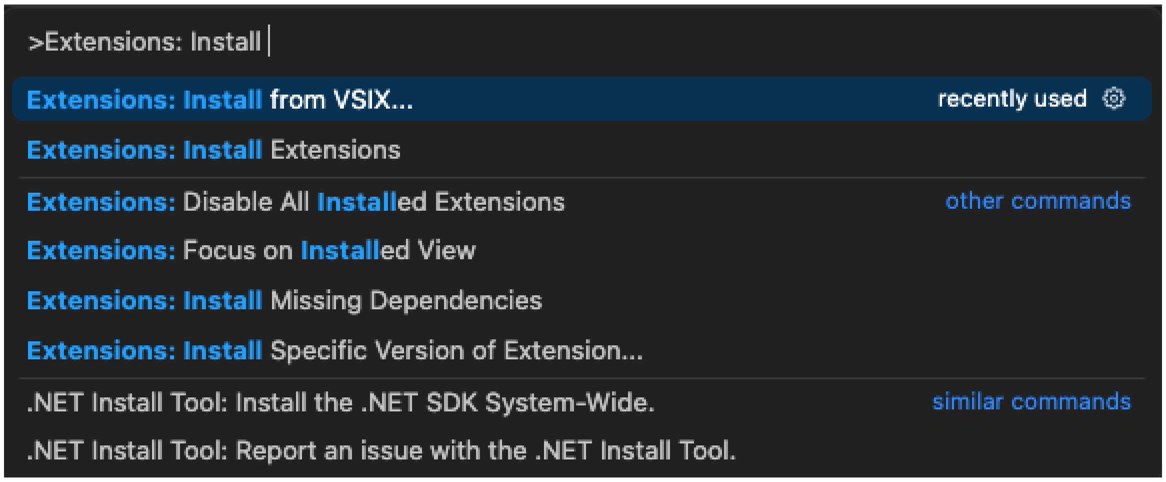
2. Open Your IDE (Cursor)
Launch your development environment. Both editors use the same underlying extension system, so the installation steps are identical.

3. Open Command Palette

- Windows/Linux: Ctrl + Shift + P
- macOS: Cmd + Shift + P

4. Search for "Extensions: Install from VSIX"

In the Command Palette, start typing: Extensions: Install from VSIX



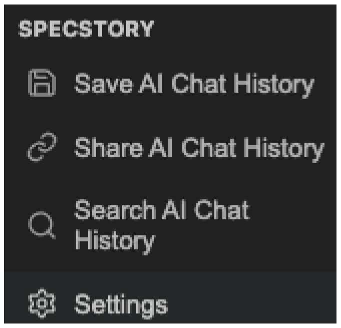
```
>Extensions: Install
Extensions: Install from VSIX...                                  recently used
Extensions: Install Extensions
Extensions: Disable All Installed Extensions                      other commands
Extensions: Focus on Installed View
Extensions: Install Missing Dependencies
Extensions: Install Specific Version of Extension...
.NET Install Tool: Install the .NET SDK System-Wide.              similar commands
.NET Install Tool: Report an issue with the .NET Install Tool.
```

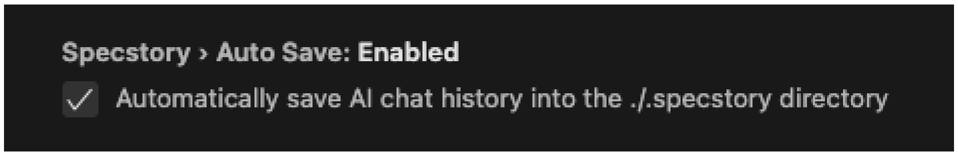
5. Select the Downloaded File

Choose specstory-vscode-0.23.0.vsix from your file browser and click Open. The installation will begin automatically. Wait until VS Code confirms that the extension was installed successfully.

6. Enable Auto Save

Open Command Palette again and search for Show SpecStory to bring up the left panel. From the side bar, select Settings and scroll to the Auto Save option. Click to enable it.





Specstory › Auto Save: Enabled

✓ Automatically save AI chat history into the ./.specstory directory

7. Verify That Conversations Are Being Saved

Open Cursor and start a quick conversation with an AI assistant.
Navigate to the .specstory/ directory within the forked repository assigned to you and locate the history folder. You should see a new `.md` file for your session, with timestamps.This confirms that SpecStory is active and capturing your coding sessions.

8. Submit your SpecStory data with .specstory/history
At the end of the study, zip your .specstory/history folder and share it with us through the Google form. Do NOT push it to the updated branch.

Note: If you want to restrict SpecStory to a specific project, you can create a separate Cursor profile and enable the SpecStory extension only in that profile. This keeps SpecStory active for that project while leaving your other workspaces unaffected.

SpecStory CLI

To capture interactions with coding agents used via your terminal (e.g., Claude Code), we also need to install specstory CLI. Follow the following steps to install the

1. Install SpecStory

- [MacOS] Install through Homebrew below
- [Linux/WSL] Download the binary here

```
None
brew install specstoryai/tap/specstory
```

2. Install the SpecStory Wrapper
Under tools/specstory/specstory_cli, You will find two files:
`install_specstory_wrapper.sh` and `specstory_wrapper.py`

Run the installer:

```
None
bash install_specstory_wrapper.sh
```

This script will:

- install the wrapper at `~/bin/specstory`

- install the Python logger at `~/.specstory_wrapper/specstory_wrapper.py`
- ensure `~/bin` is added to your PATH
- install Conda activation hooks so what when you run claude, it will be routed through specstory.

After installation, run `source ~/.zshrc` or `source ~/.bashrc`
You will be prompted to **choose the Conda environment** where SpecStory should run.

3. Use SpecStory

You can now run SpecStory through:

```None
specstory run claude
```

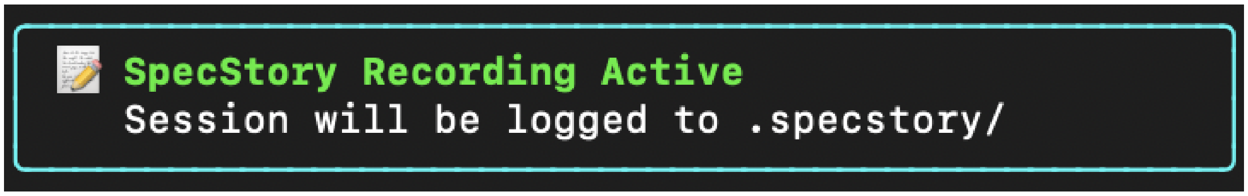
All interactions during these sessions will be captured and saved in:

```None
.yourproject/.specstory/history/
```

You should be able to use and prompt AI assistants just like you normally do during development. Just make sure you're prompting inside the assigned repository, so SpecStory saves the conversations in the right place.

4. Verify That Conversations Are Being Saved
Navigate to the .specstory/ directory within the forked repository assigned to you and locate the history folder. You should see a new `.md` file for your session, with timestamps. Check that your conversations and AI interactions are being saved there — this confirms that SpecStory is active and capturing your coding sessions.

You should see a banner like the one below when you run `specstory run claude`



```
📝 SpecStory Recording Active
   Session will be logged to .specstory/
```

You should also use the provided verifier script to automatically check that:

1. Both Claude and Cursor sessions are present
2. All User/Agent headers have timestamps

Run the verifier from the repository root:

```
None
python tools/specstory/verify_specstory.py
```

Or verify a specific project directory:

```
None
python tools/specstory/verify_specstory.py /path/to/project
```

The verifier will report which files (if any) are missing timestamps and need attention.

5. Submit your SpecStory data with .specstory/history
At the end of the study, zip your .specstory/history folder and share it with us through the Google form. Do NOT push it to the updated branch.