Open in app ↗

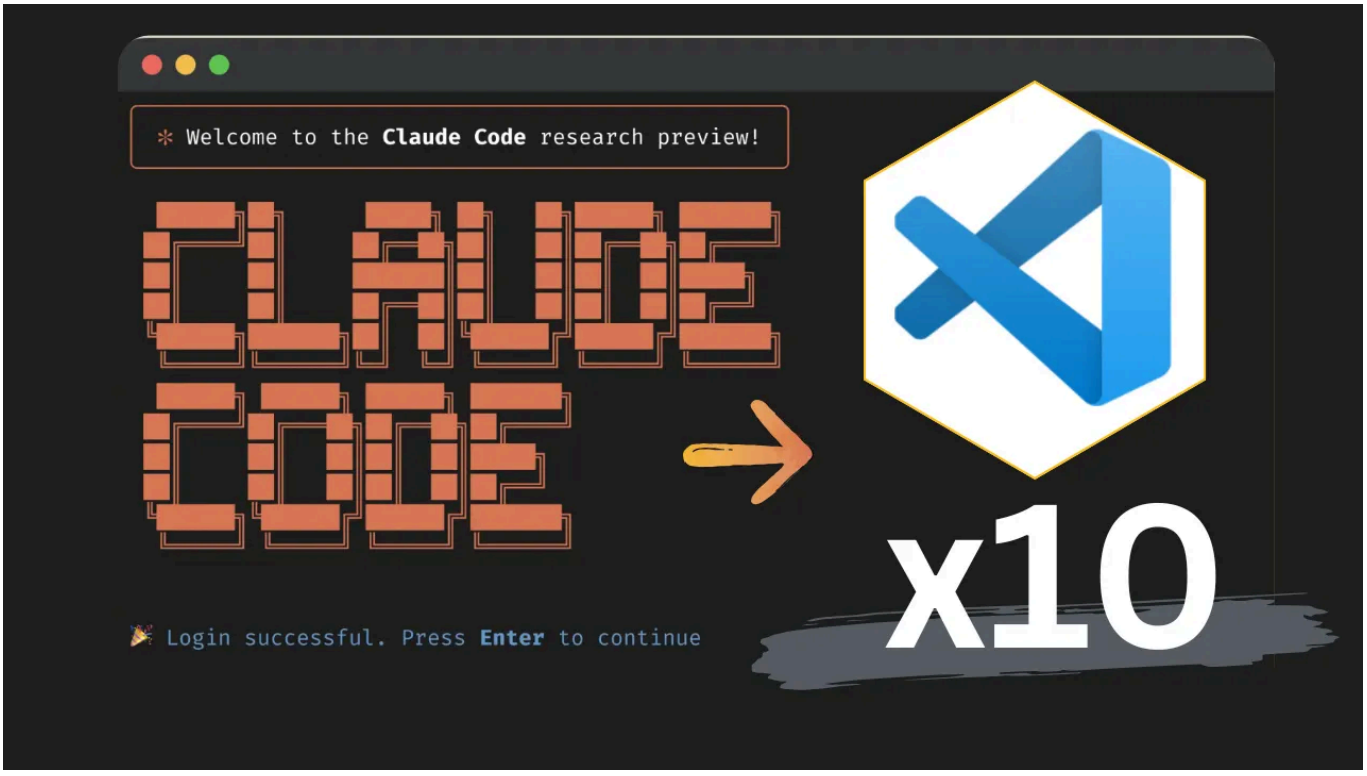
Medium Search Write

Member-only story

How I'm Using (New) VS Code Claude Code 2.0 Extension to Code 10x Faster

Joe Njenga Follow 10 min read · 1 day ago

35



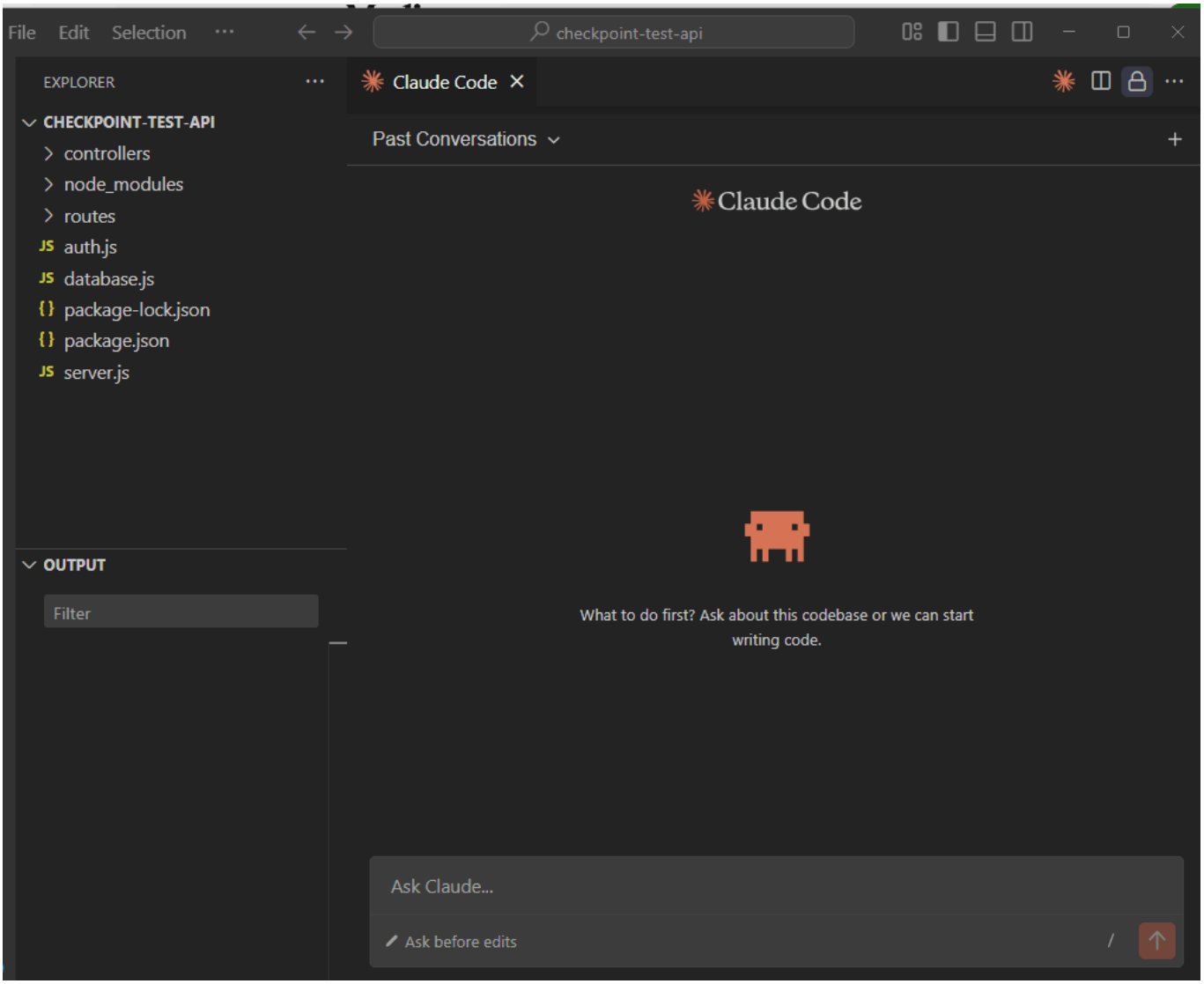
When I first reviewed the Claude Code VS Code extension, I was very disappointed.

> *But the new version is something different and will impress you!*

If you've been following my tutorials, you know I've been exploring Claude Code extensively.

We've covered all the new features as they are released, but if you missed out, here is the list of all the Claude Code tutorials.

The new Claude Code VS Code extension, launched with Sonnet 4.5, completely changes the development experience.



Claude Code started as a command-line tool. If you're comfortable with terminals, great.

But some developers avoid the terminal.

Anthropic released a native VS Code extension with the same power and visual interface.

Let's start with a quick comparison of the two terminals and the VS Code extension.

Extension vs Terminal: What's the Difference?

Quick breakdown since I get asked this often:

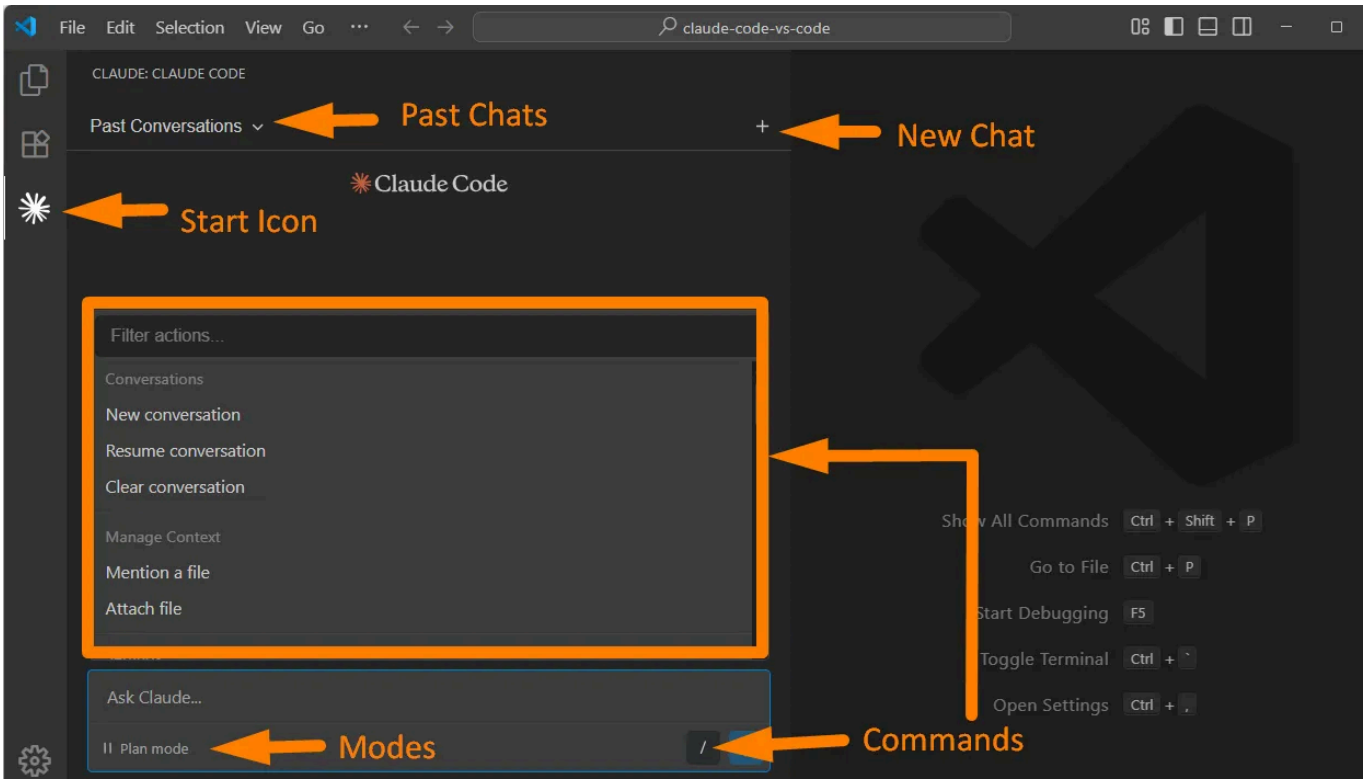
The VS Code Extension gives you:

- *Sidebar chat panel (no terminal needed)*
- *Inline diffs showing changes in real-time*
- *Plan Mode with visual reviews*
- *Drag-and-drop files directly into chat*
- *One-click accept/reject changes*

Still CLI-only (for now: Oct 2025):

- *Checkpoint system (save states and rewind)*
- *Some advanced shortcuts*
- *Direct MCP server configuration*

Now the new extension looks and feels better than what we had previously, and here is a quick map of what the UI and the features look like :



Testing Claude Code VS Code Extension

We will be testing to see how it performs and whether it matches our current Claude Code terminal experience.

Before we start, here's what you need:

Software:

- VS Code 1.98.0 or newer
- Node.js 18+ (we'll need this for some features later)

Subscription:

- Claude Max ($100/month) OR Anthropic API credits

Project Overview

We're building a Task Manager application from scratch to test and see how to use each of these UI features.

Tech stack:

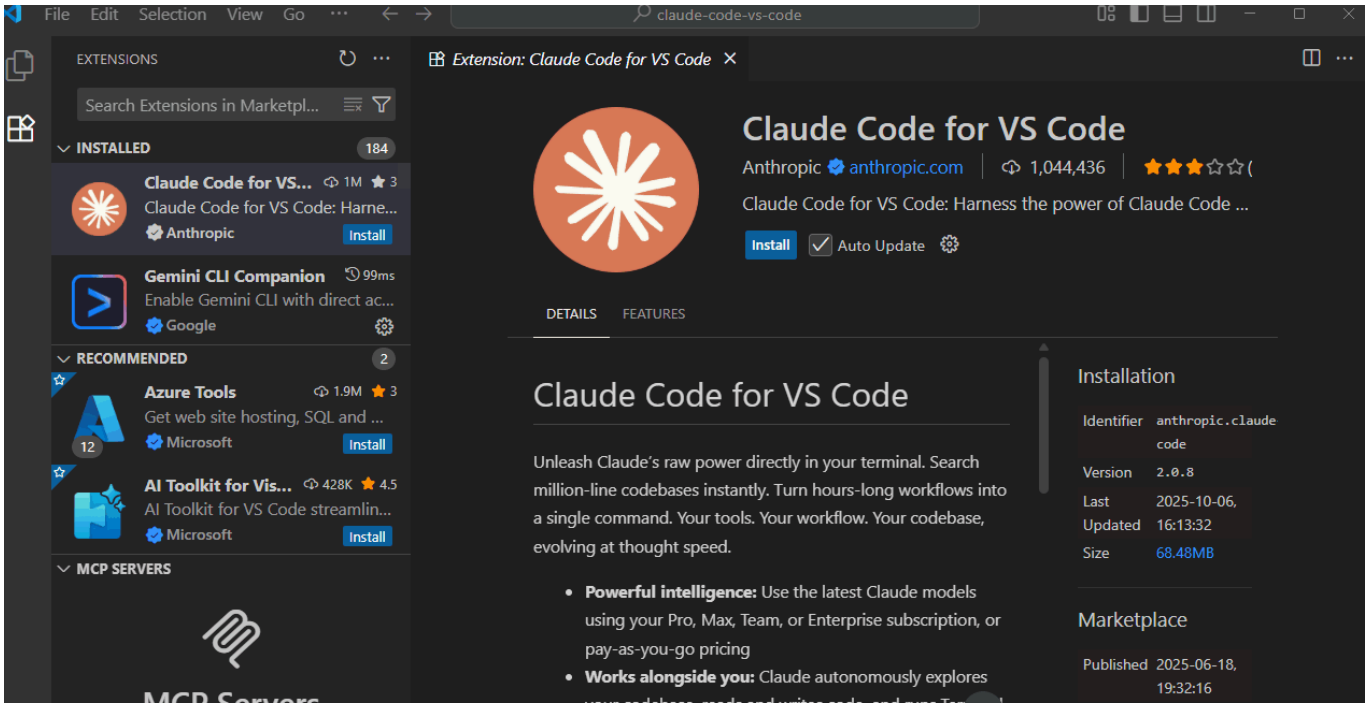
- **Frontend:** Vanilla JavaScript (no framework overhead)
- **Backend:** FastAPI (Python, super fast to build with)
- **Database:** SQLite (simple, no setup required)

I want you to see how Claude Code handles everything without getting lost in framework complexity.

We'll go from an empty folder to a working application, and I'll show you exactly how the extension makes each step faster.

Alright, let's get into it. First up: installation.

Installation & Setup



Let's get Claude Code running in your VS Code. Takes about 5 minutes.

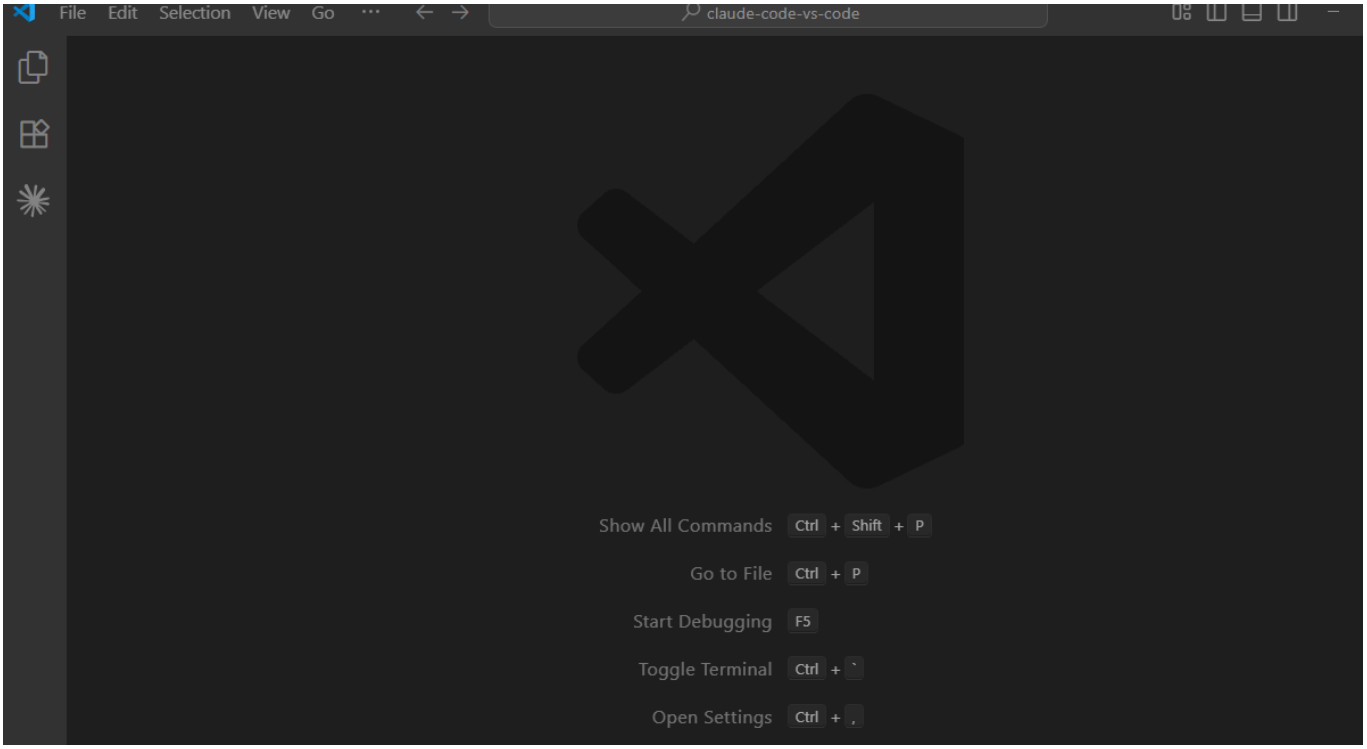
Installing the Extension

VS Code Marketplace:

1. *Open VS Code*
2. *Extensions icon (* `Ctrl+Shift+X` / `Cmd+Shift+X` *)*
3. *Search "Claude Code" by Anthropic*
4. *Click Install*

Or visit: VS Code Extension Marketplace

Authentication



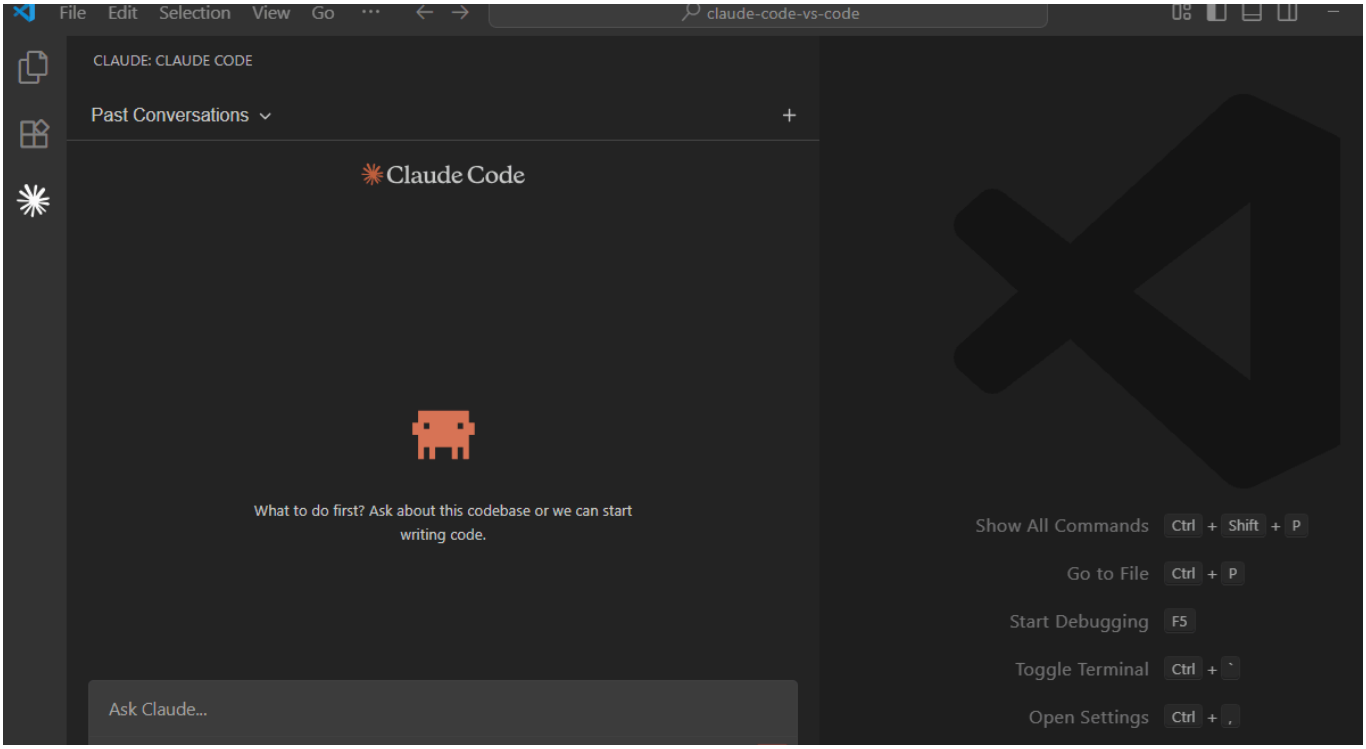
Click the Claude Code icon in the sidebar.

Set up flow:

1. *Click "Sign In" → Browser opens*
2. *Choose: Claude Max account OR API credits*
3. *Authorize VS Code access*
4. *Return to VS Code*

Important: API credits need active billing at console.anthropic.com.

Test it works: Type "*What version of Claude Code am I using?*" and press Enter.



Project Setup

Create project structure:

```
mkdir -p backend/routers frontend/js frontend/css
```

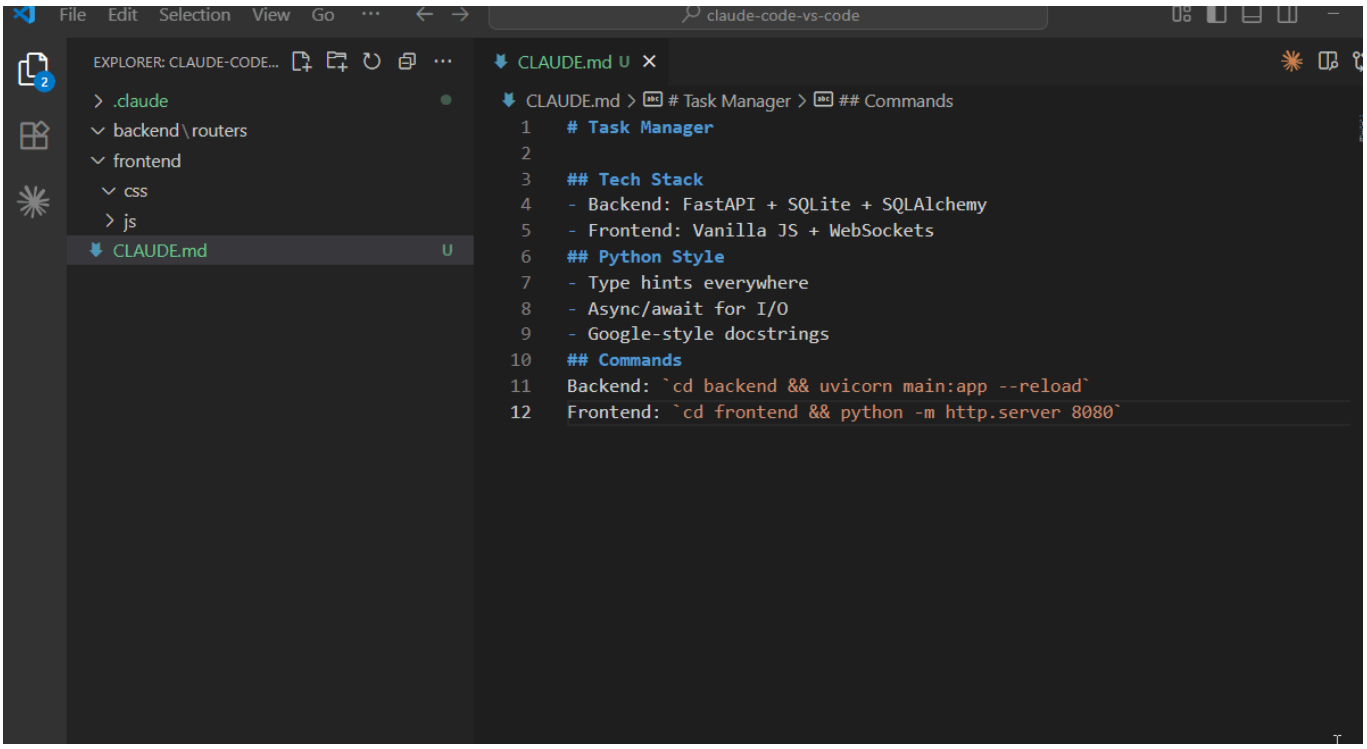
Initialize Git:

```
git init
```

Create CLAUDE.md in project root:

```
# Task Manager

## Tech Stack
- Backend: FastAPI + SQLite + SQLAlchemy
- Frontend: Vanilla JS + WebSockets
## Python Style
- Type hints everywhere
- Async/await for I/O
- Google-style docstrings
## Commands
Backend: `cd backend && uvicorn main:app --reload`
Frontend: `cd frontend && python -m http.server 8080`
```

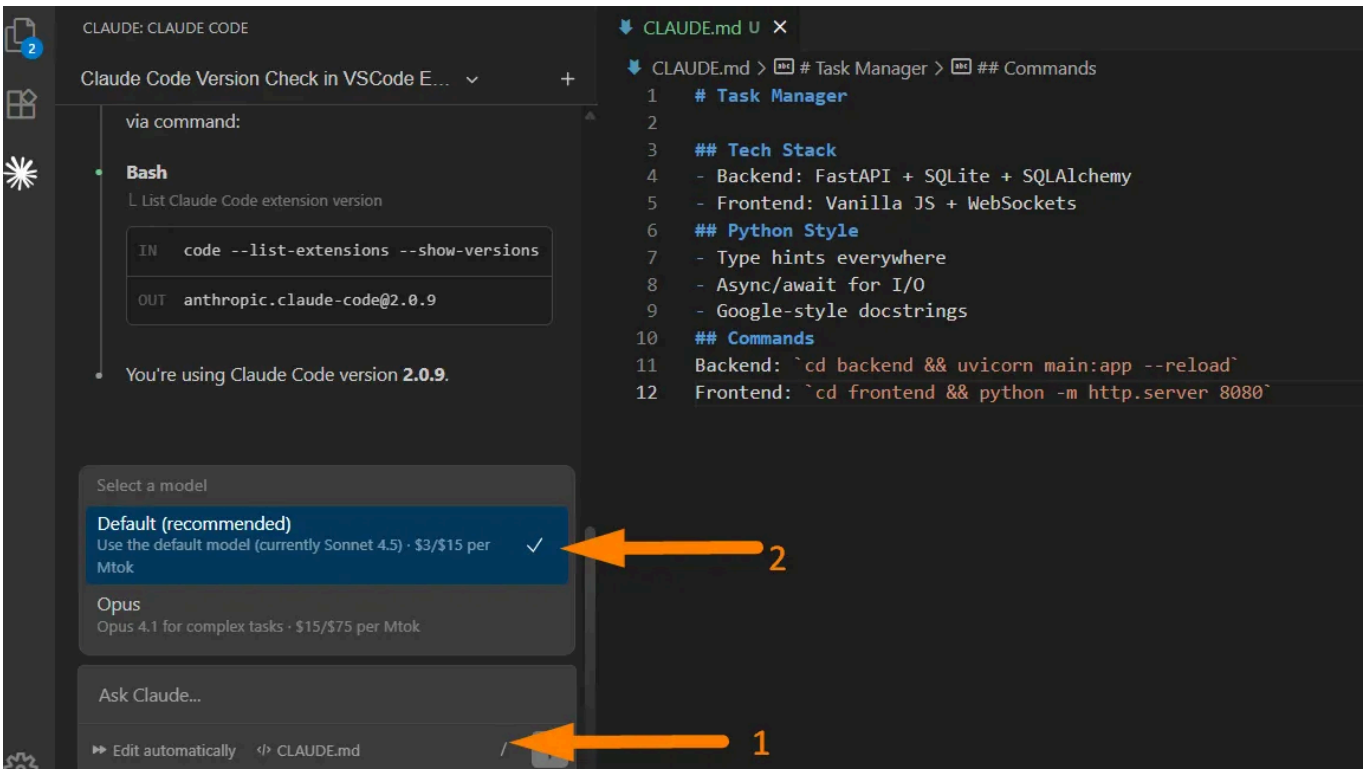


Claude reads this automatically.

Set Model

Click / in the Claude Code chat to set the Model. The model by default is set to Claude 4.5

- **Model:** claude-sonnet-4.5 (default, best for coding)

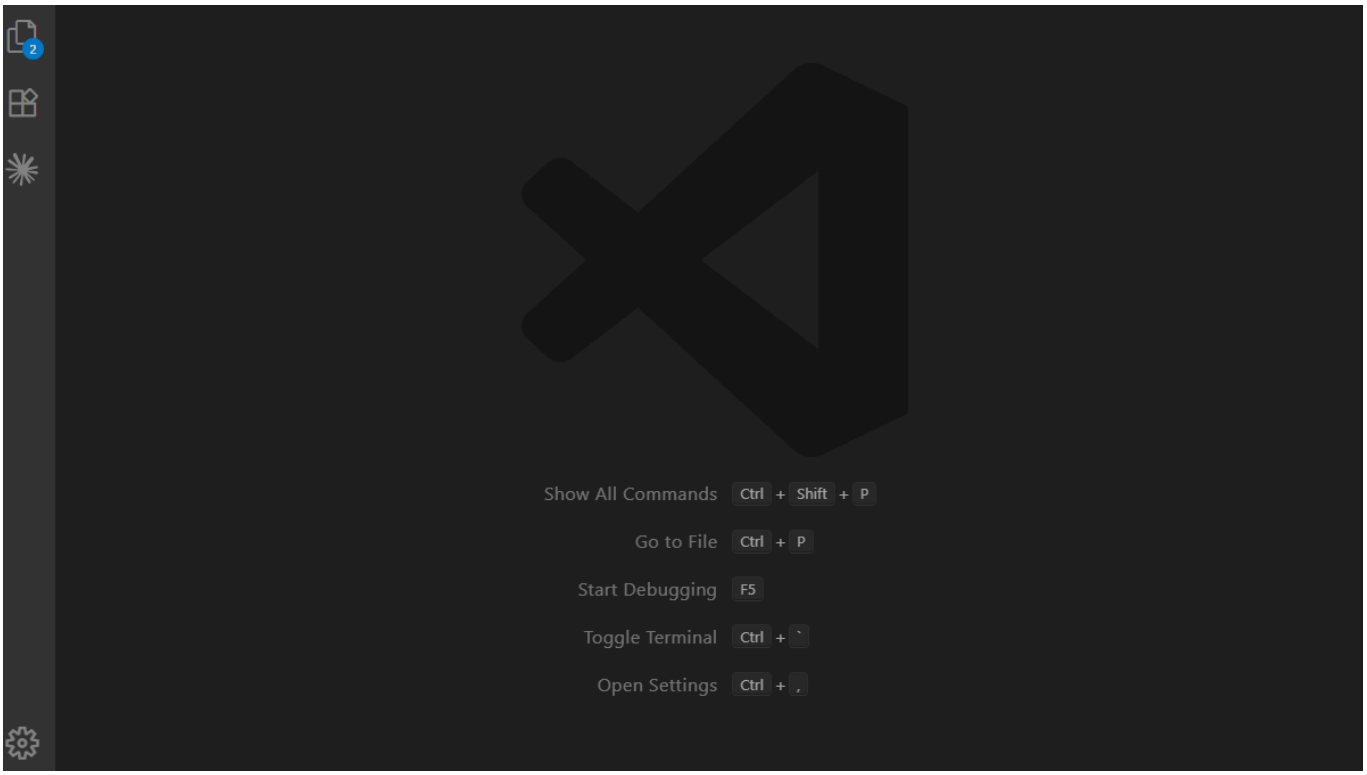


Understanding the Interface

Let me show you how the interface works

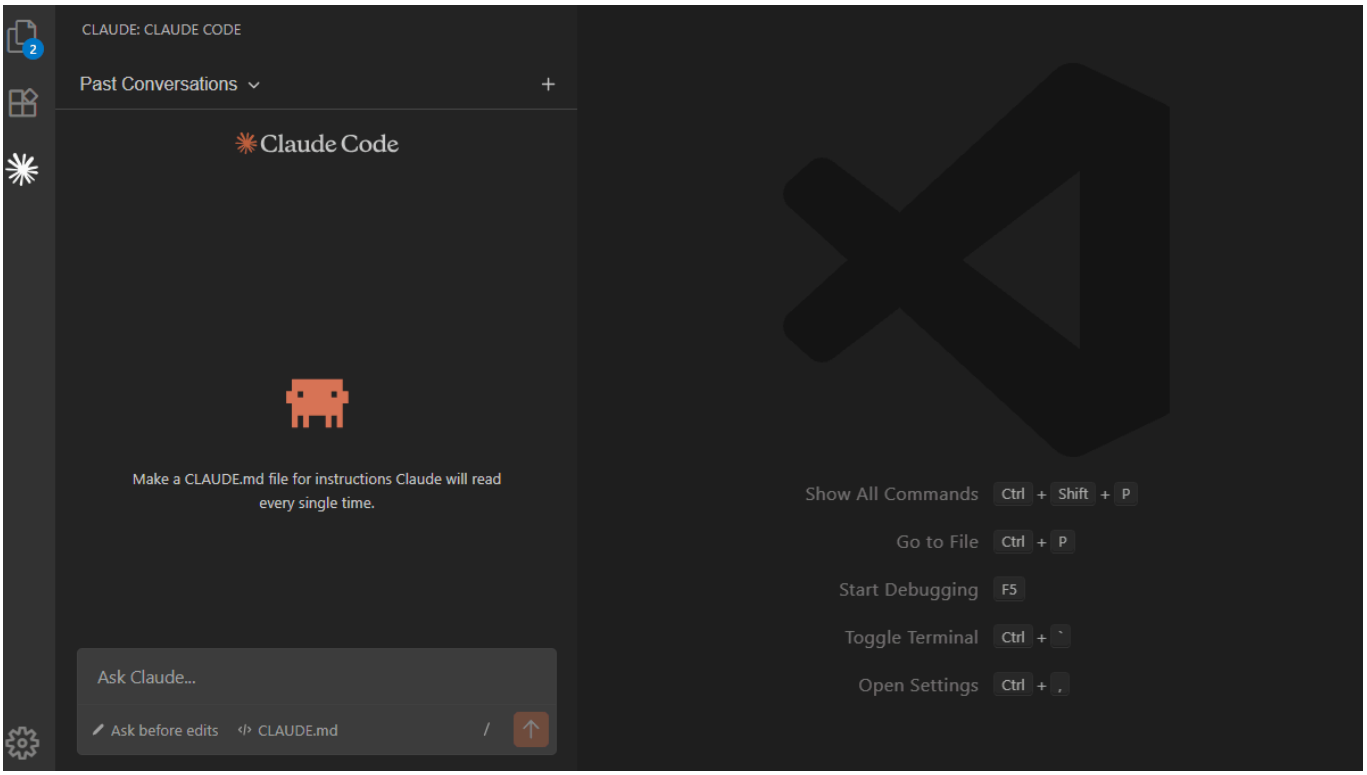
Command Menu

Click the Claude Code icon in the sidebar or press `Ctrl+Shift+P` and type `/` to see the command menu.



Conversations:

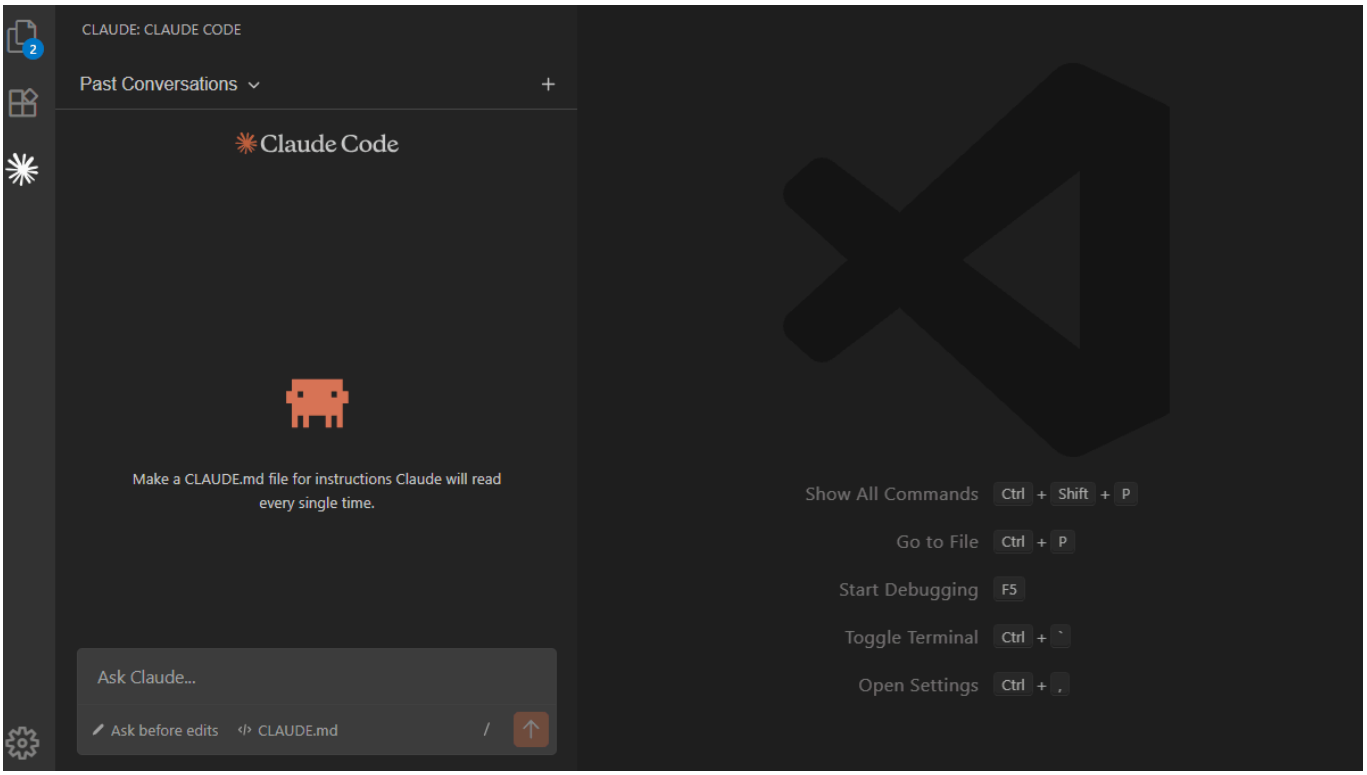
- New conversation
- Resume conversation
- Clear conversation



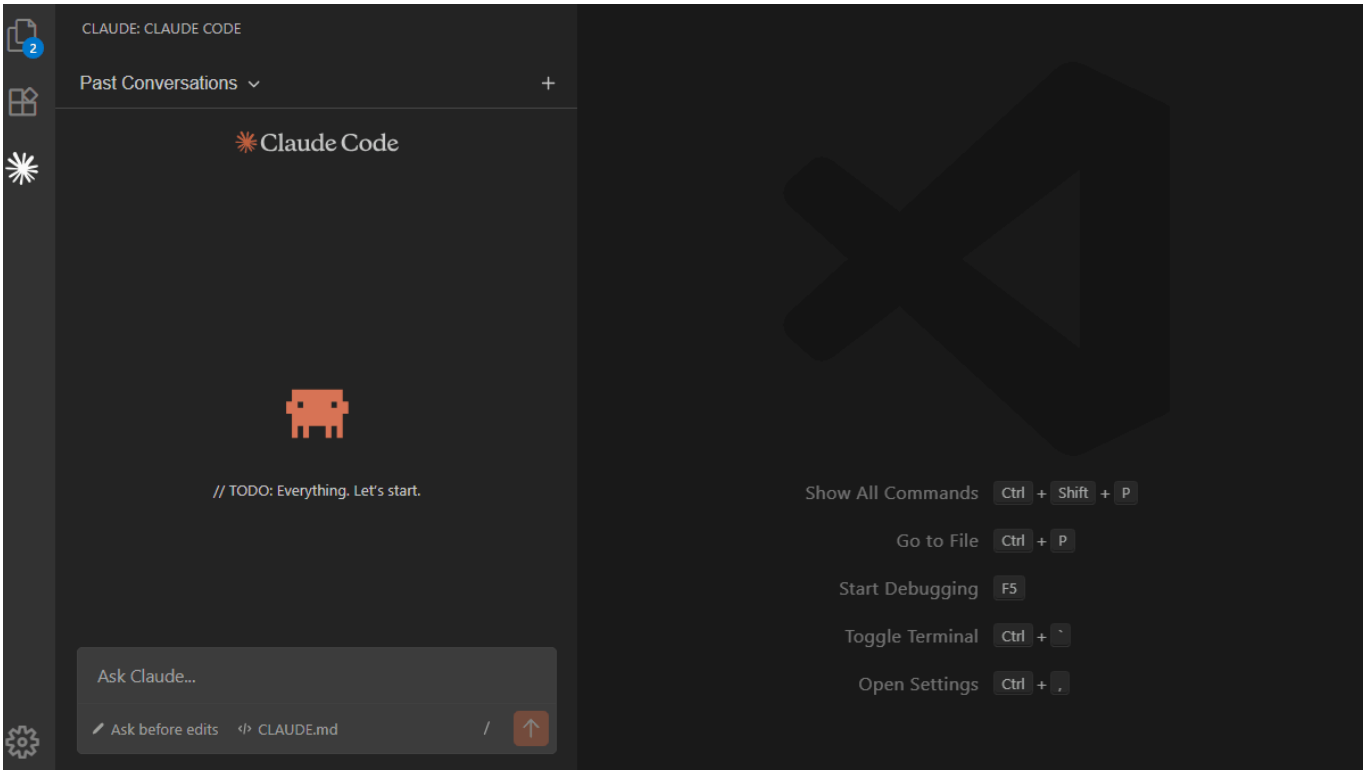
Manage Context:

- Mention a file (@ symbol in chat)
- Attach file (upload from anywhere)

Settings:



- Select model
- MCP (Model Context Protocol servers)
- Login



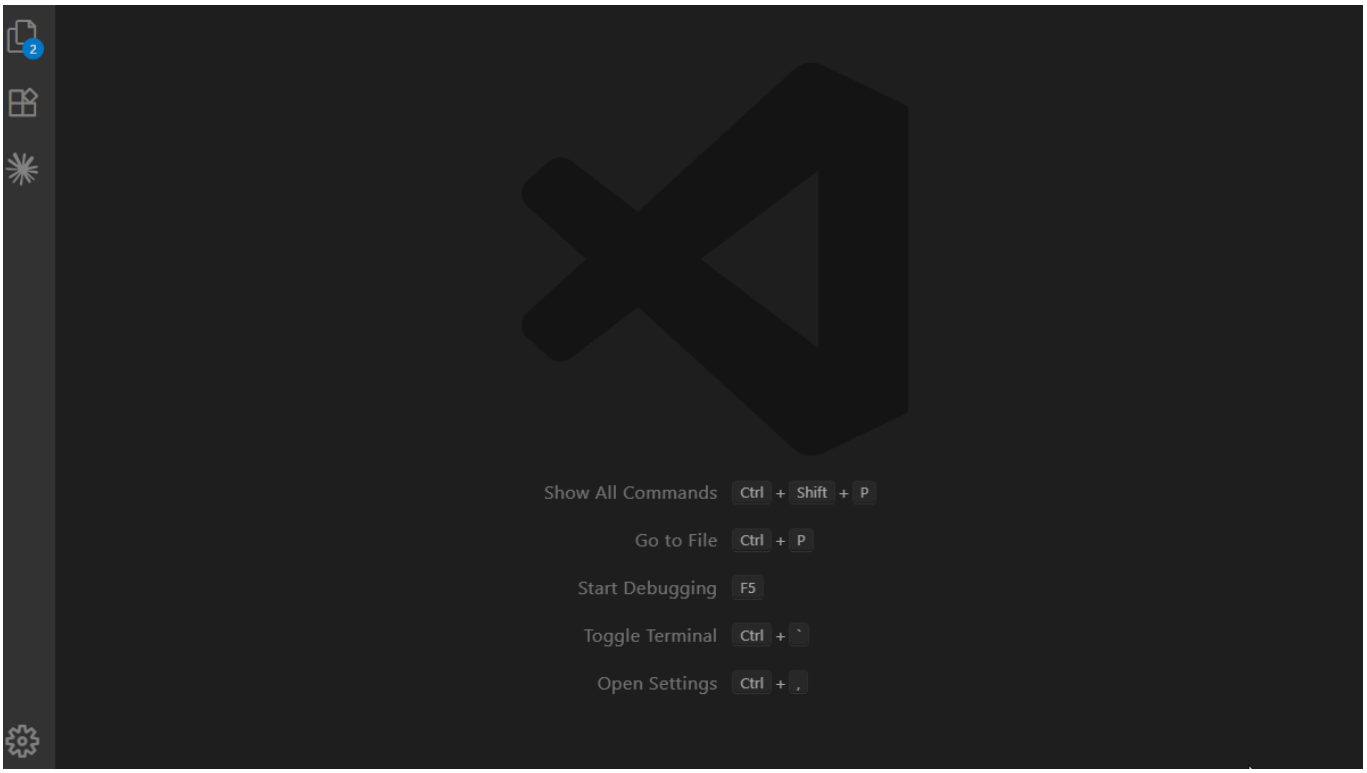
Slash Commands: You'll see a long list: `/cleanproject`, `/commit`, `/compact`, `/context`, `/create-todos`, `/docs`, `/format`, `/review`, `/test`, `/undo`, and many more.

These are pre-built commands for common tasks.

Type `/` to browse them all.

Three Prompting Modes

Bottom left of the input box shows your current mode:



"Ask before edits" (Default)

- Claude asks permission for each change

- You approve or reject
- Safest mode

“Edit automatically”

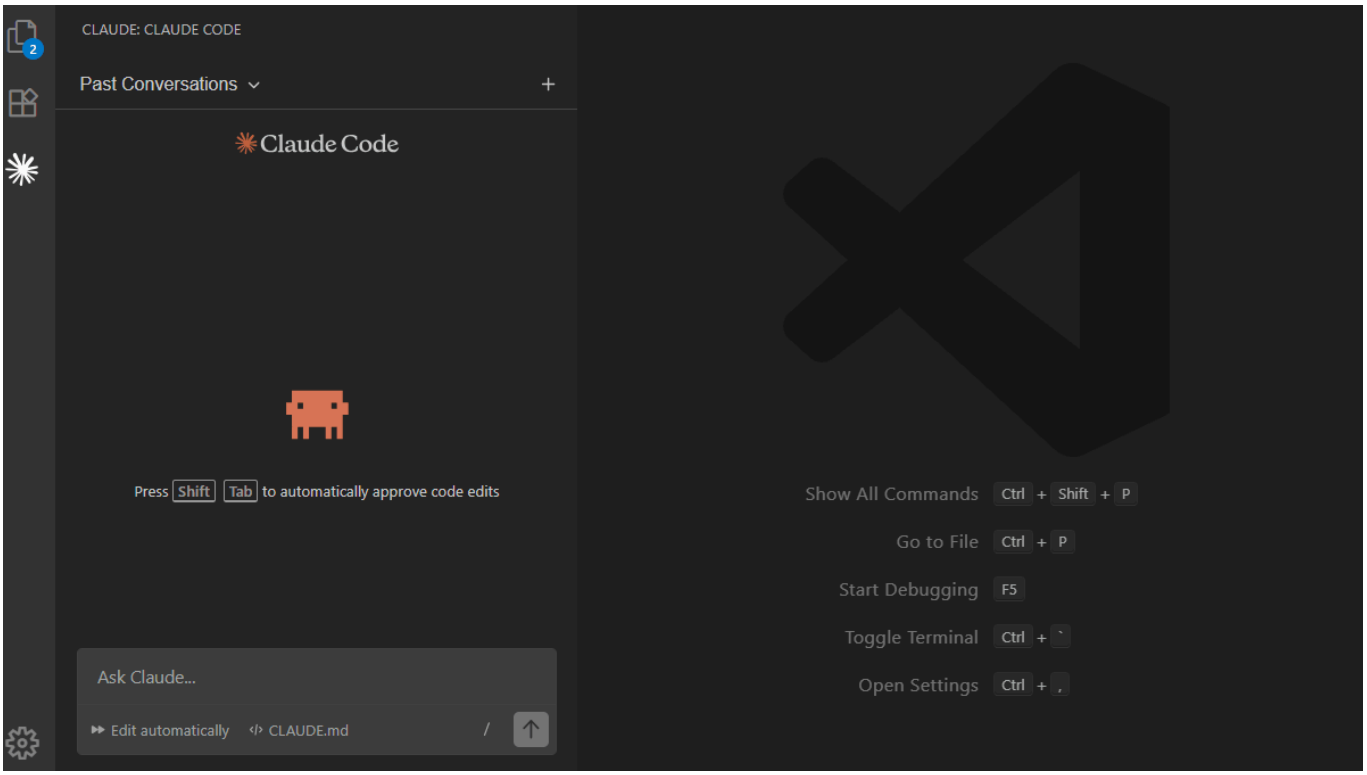
- Claude makes changes without asking
- Fastest mode
- Use with caution

“Plan mode”

- Claude creates a plan first
- You review the plan
- Then approve execution

Switch modes by clicking the mode indicator.

Prompt Input Box



Bottom of the panel:

Left side shows:

- Current mode (pencil icon)
- “Ask before edits” or “Edit automatically”
- CLAUDE.md indicator (shows it’s reading your project file)

Right side:

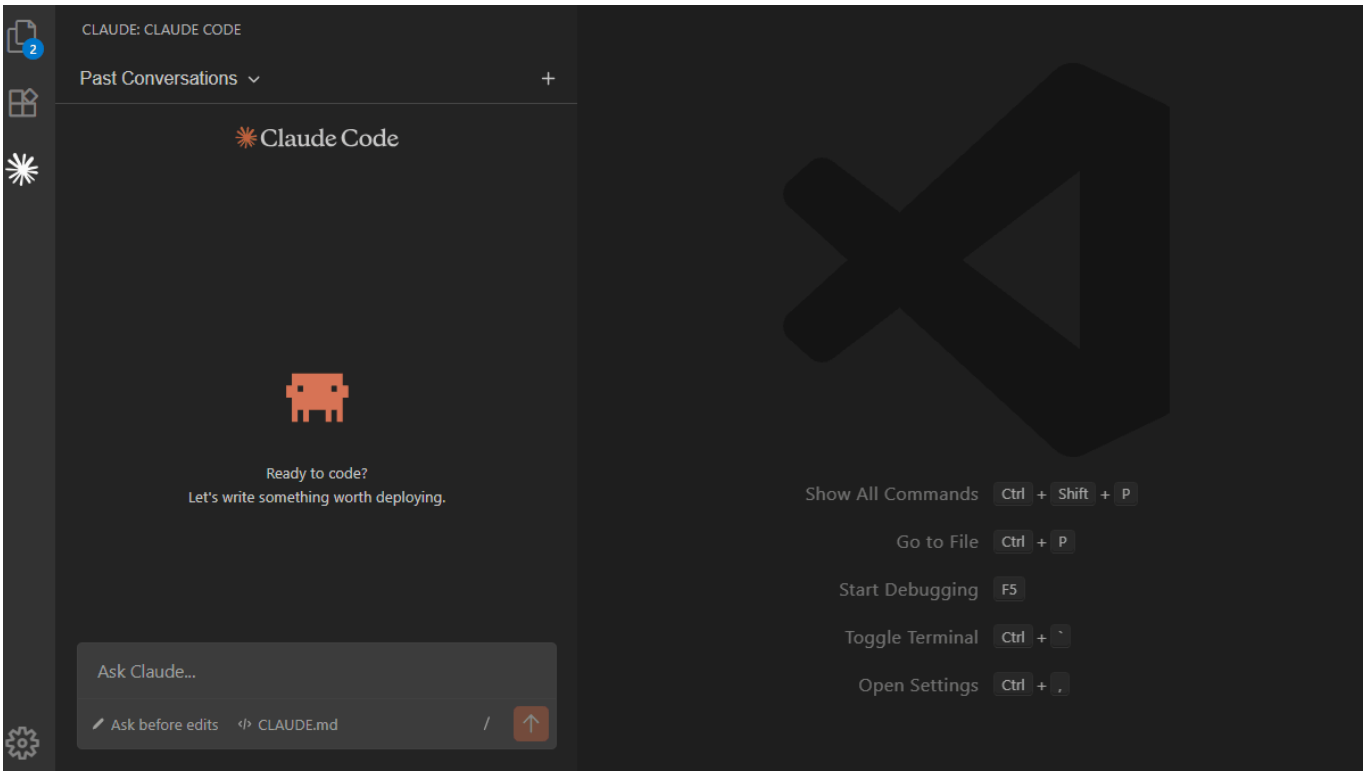
- Up arrow to send a message
- Type `/` for commands
- Type `@` to mention files

@-Mentions for Files

Type `@` in the input box:

- Start typing filename
- Autocomplete appears
- Select a file to add context

Example: `@CLAUDE.md what's in this file?`



How It Works

1. *Open Claude Code sidebar*
2. *Type your request in the "Ask Claude..." box*
3. *Claude reads CLAUDE.md automatically*
4. *Claude responds with a plan or makes changes*

5. *You see diffs in your files*
6. *Accept or reject changes*

What We'll Use Most

For our project build:

- **Ask before edits mode** — See every change
- **@-mentions** — Give Claude file context
- **Slash commands** — Quick actions like `/test` or `/review`
- **CLAUDE.md** — Project memory (Claude reads this automatically)

Building the Task Manager Project

Let's build.

I'll show you my exact prompts and what Claude does.

Build Strategy

For a large project, you can use a multiple-tab approach: open VS Code in tabs side by side. Run separate Claude Code conversations:

- *Window 1: Backend development*

- *Window 2: Frontend development*

Work on both simultaneously. Claude maintains a separate context in each conversation.

Save tokens:

- *Use @-mentions instead of pasting code*
- *Keep conversations focused (backend separate from frontend)*
- *Start a new conversation per central feature*
- *Let Claude read CLAUDE.md instead of repeating project details*

Modes we'll use:

- ***Ask before edits: New features (see every change)***
- ***Plan mode: Multi-file changes (review architecture first)***
- ***Edit automatically: Repetitive tasks (format, style fixes)***

Setup

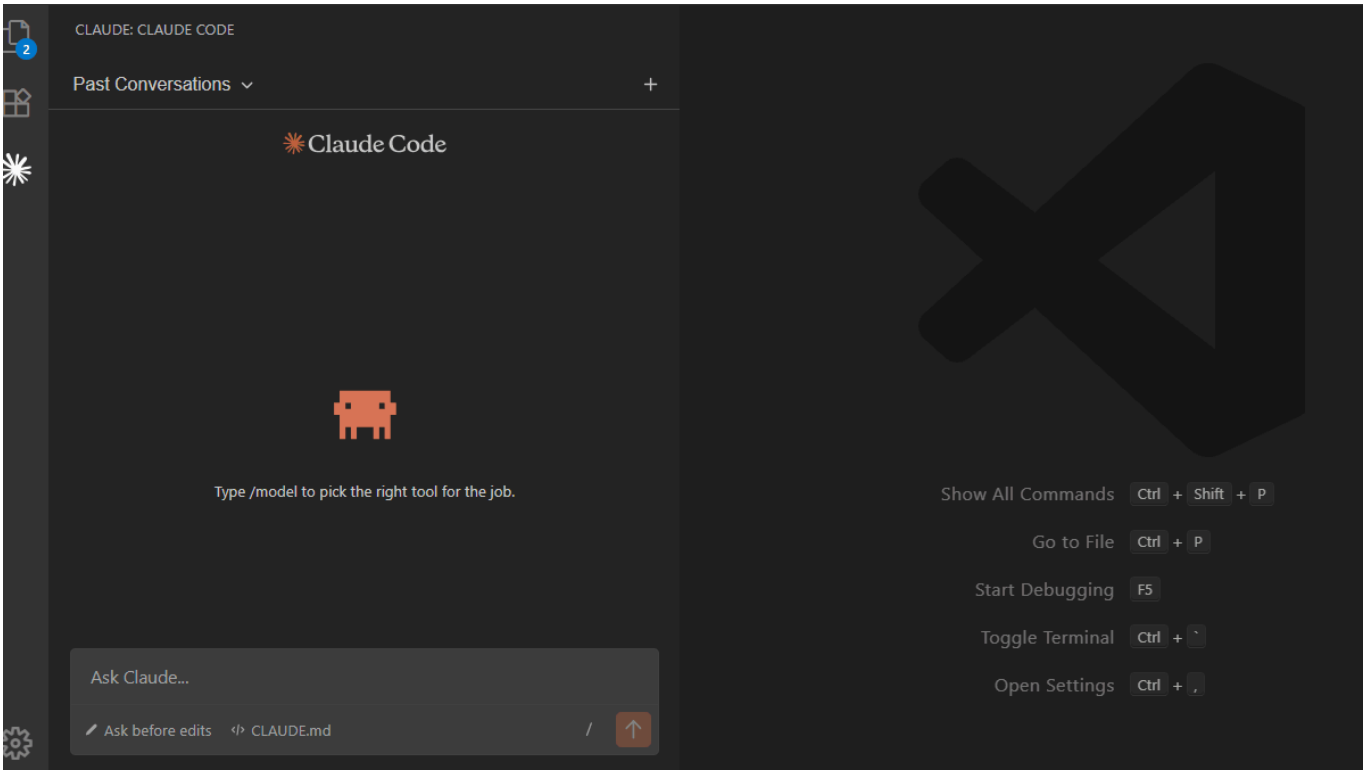
Project structure created. CLAUDE.md in place. Claude Code opens in two windows.

Backend — Database Layer

Prompt 1:

```
Create backend/database.py with SQLite + SQLAlchemy async setup.
Include AsyncSession factory, Base model, get_db dependency, and init_db functio
```

Mode: Ask before edits

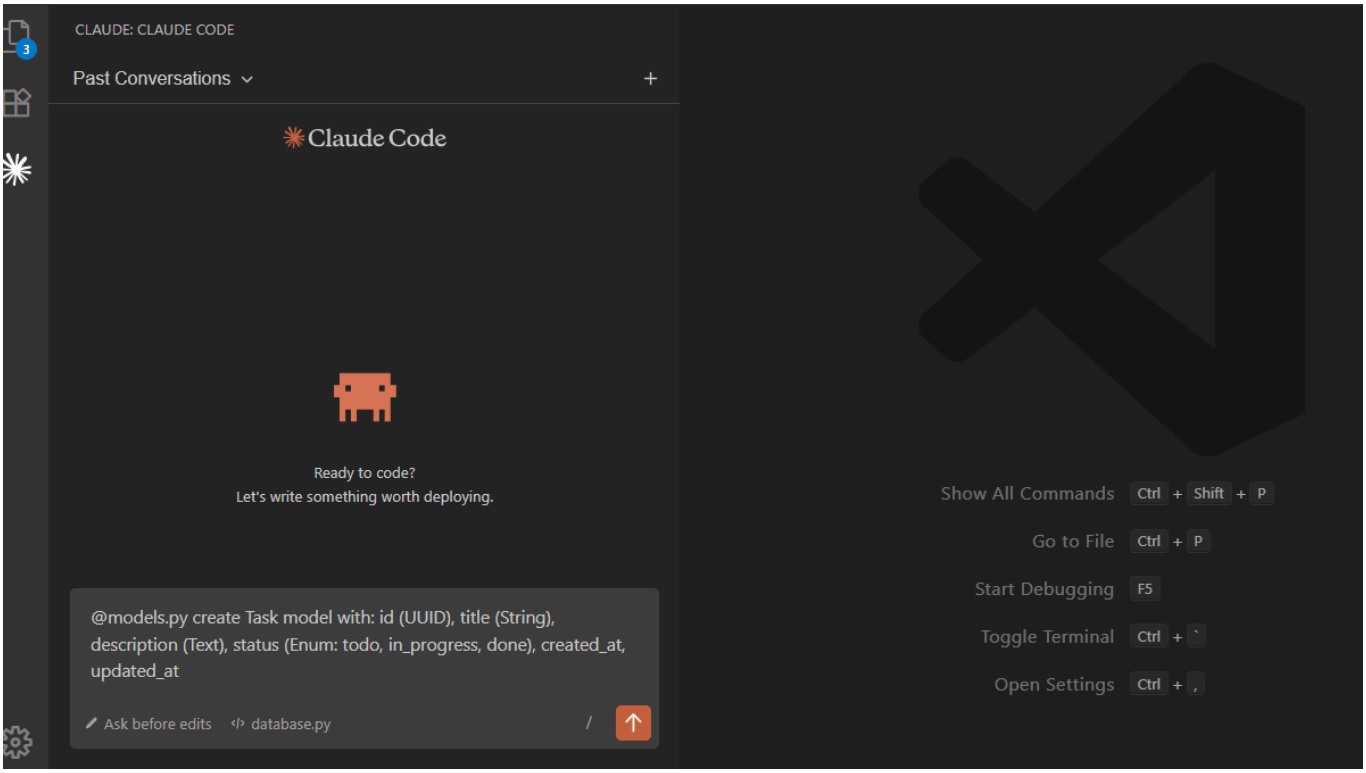


Claude creates the file. Shows diff. I accept.

Prompt 2:

```
@models.py create Task model with: id (UUID), title (String),
description (Text), status (Enum: todo, in_progress, done), created_at,
updated_at
```

Claude reads models.py (or creates it), adds the model. Accept.

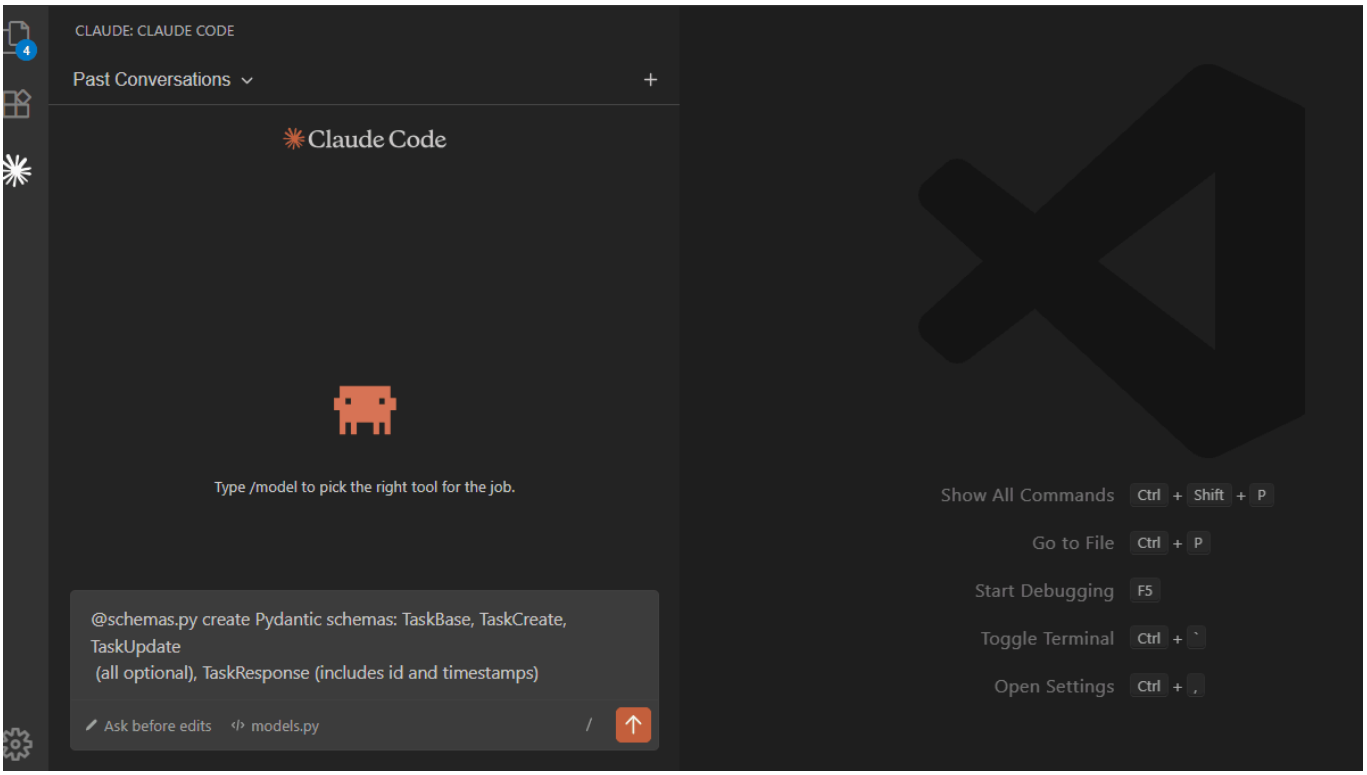


Prompt 3:

```
@schemas.py create Pydantic schemas: TaskBase, TaskCreate, TaskUpdate
 (all optional), TaskResponse (includes id and timestamps)
```

Accept.

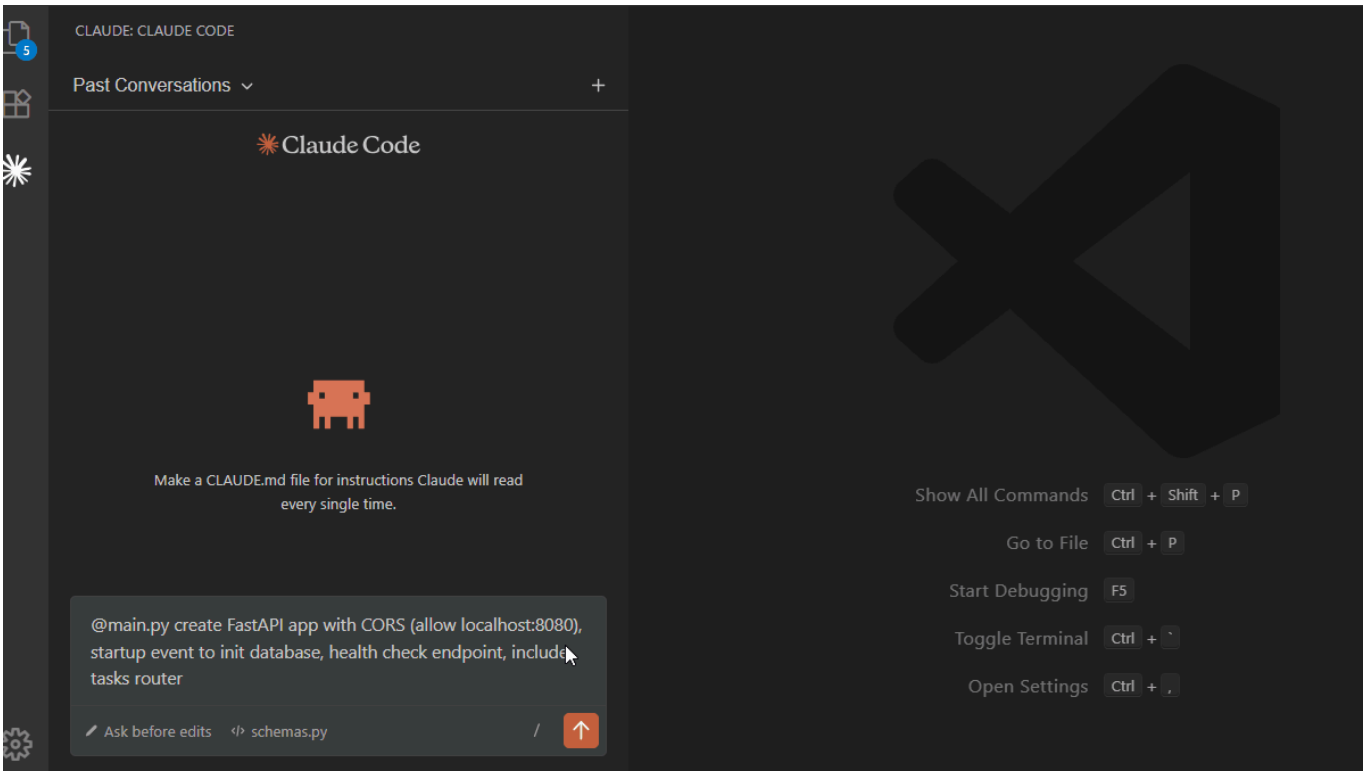
Time: 2 minutes. Database layer done.



Backend — API Endpoints

Prompt 4:

```
@main.py create FastAPI app with CORS (allow localhost:8080),
 startup event to init database, health check endpoint, include tasks router
```



Prompt 5:

```
Create backend/routers/tasks.py with:
- POST /api/tasks (create)
- GET /api/tasks (list, optional status filter)
- GET /api/tasks/{id} (get one)
- PATCH /api/tasks/{id} (update)
- DELETE /api/tasks/{id} (delete)
Handle 404 errors.
```

Mode: Plan mode (Shift+Tab twice)

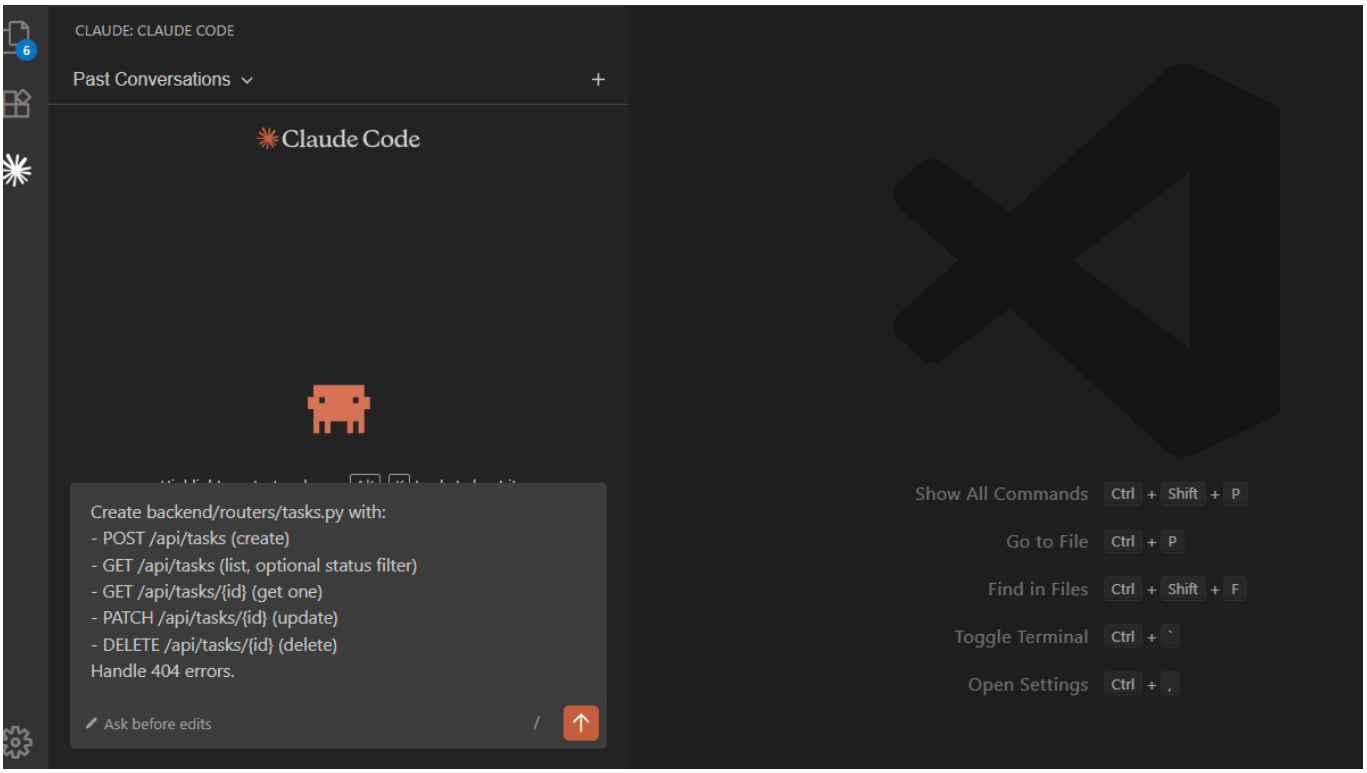
Claude shows the plan:

1. Create the routers directory
2. Create tasks.py

3. Add all 5 endpoints
4. Import in main.py

I review. Accept. Claude executes.

Time: 3 minutes. CRUD API is complete.



Backend — WebSocket

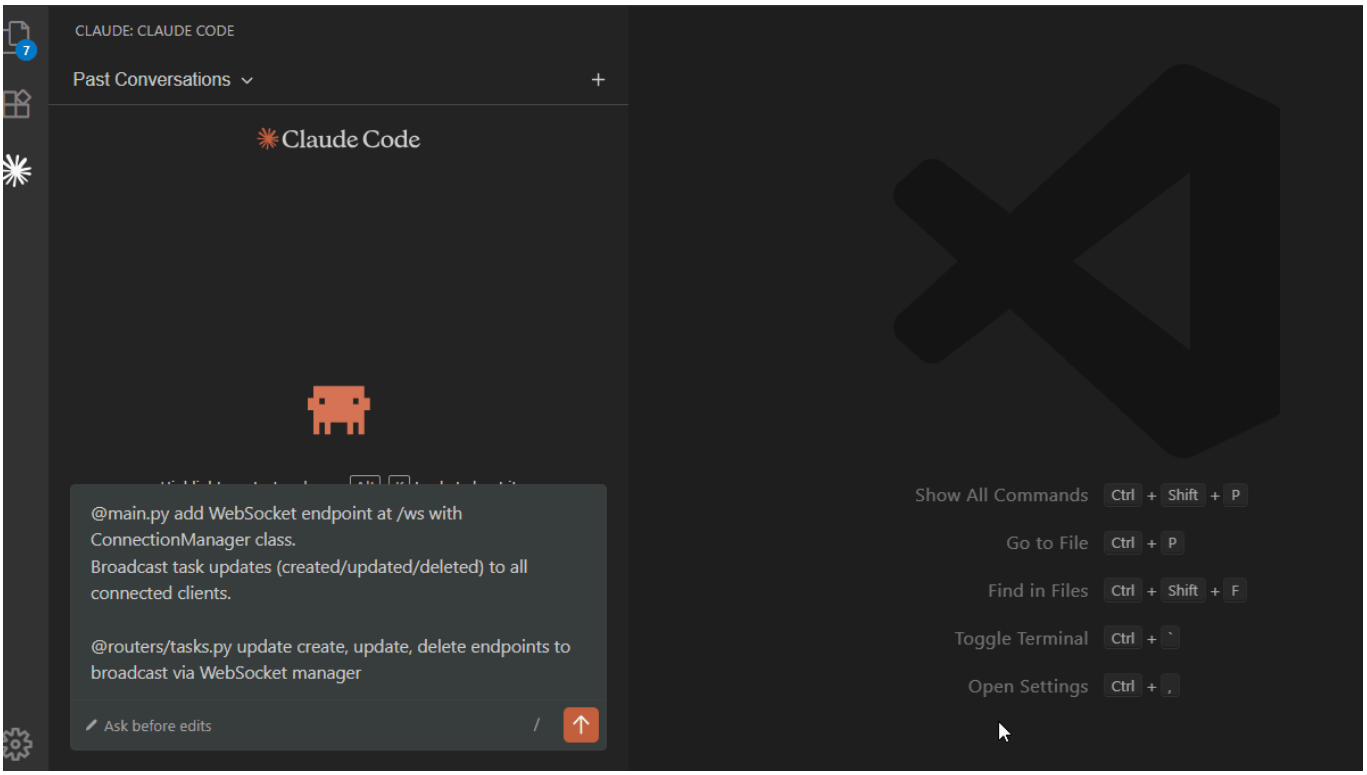
Prompt 6:

```
@main.py add WebSocket endpoint at /ws with ConnectionManager class.
Broadcast task updates (created/updated/deleted) to all connected clients.
```

Prompt 7:

```
@routers/tasks.py update create, update, delete endpoints to
broadcast via WebSocket manager
```

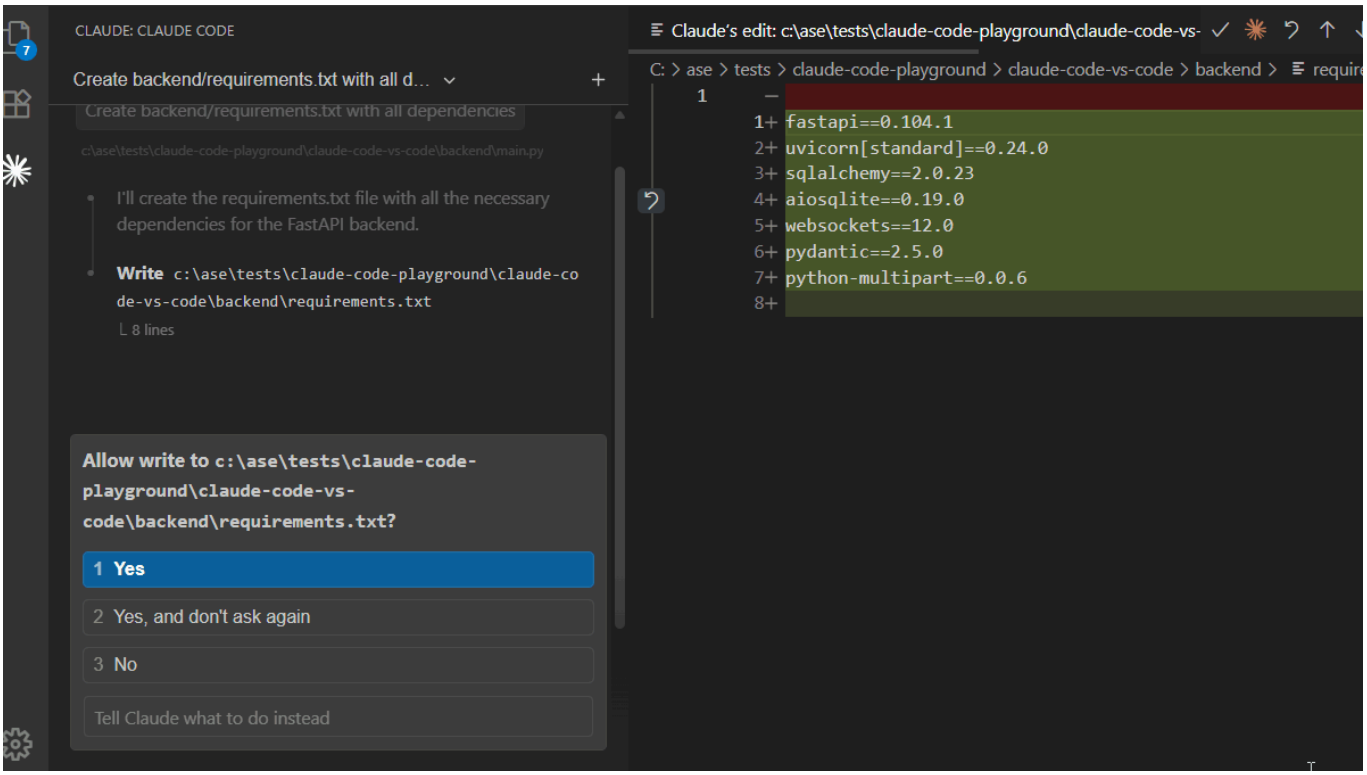
Accept both.



Prompt 8:

```
Create backend/requirements.txt with all dependencies
```

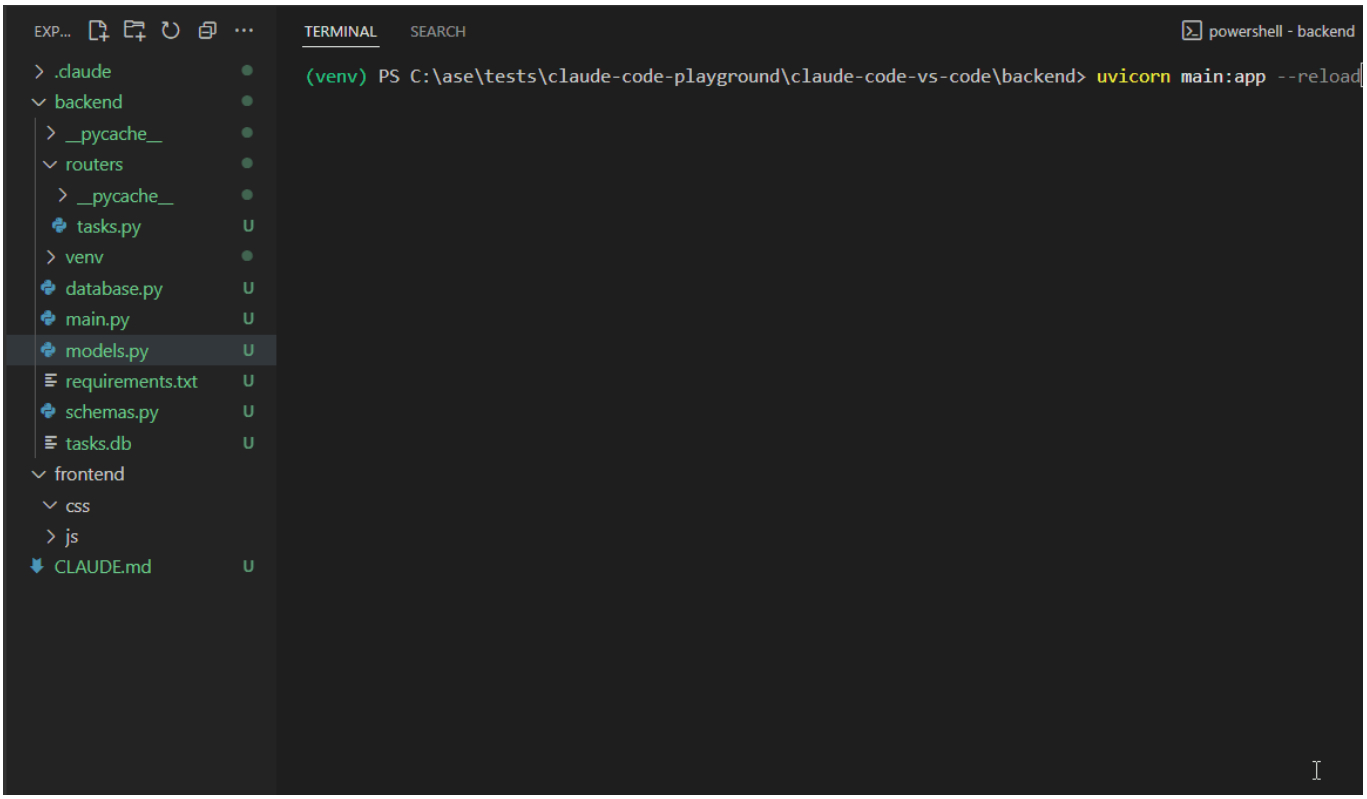
Done.



Test backend:

```
cd backend
python -m venv venv
source venv/bin/activate
pip install -r requirements.txt
uvicorn main:app --reload
```

Visit: http://localhost:8000/docs



API works.

Backend time: 8 minutes total.

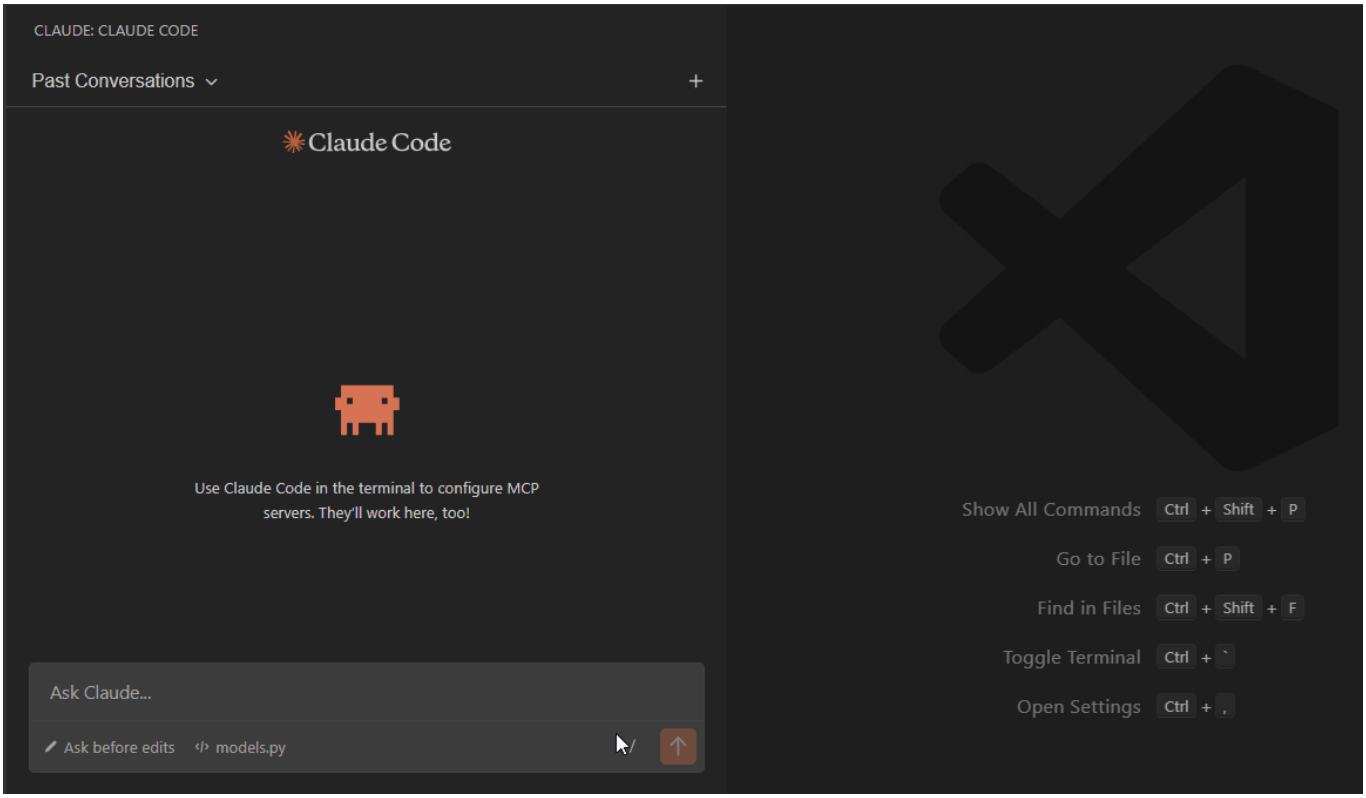
Frontend — HTML & CSS

Prompt 9:

```
@frontend/index.html create task manager UI with:
- Header
- Form (title input, description textarea, submit button)
- Filter buttons (All, Todo, In Progress, Done)
- Tasks container
- Link to style.css and app.js, api.js
```

Prompt 10:

```
@frontend/css/style.css create modern styles with:
- CSS variables for colors
- Responsive design
- Card-based task layout
- Status badges (todo: blue, in_progress: orange, done: green)
- Clean form styling
```



Both accepted.

Time: 3 minutes.

Frontend — JavaScript

Prompt 11:

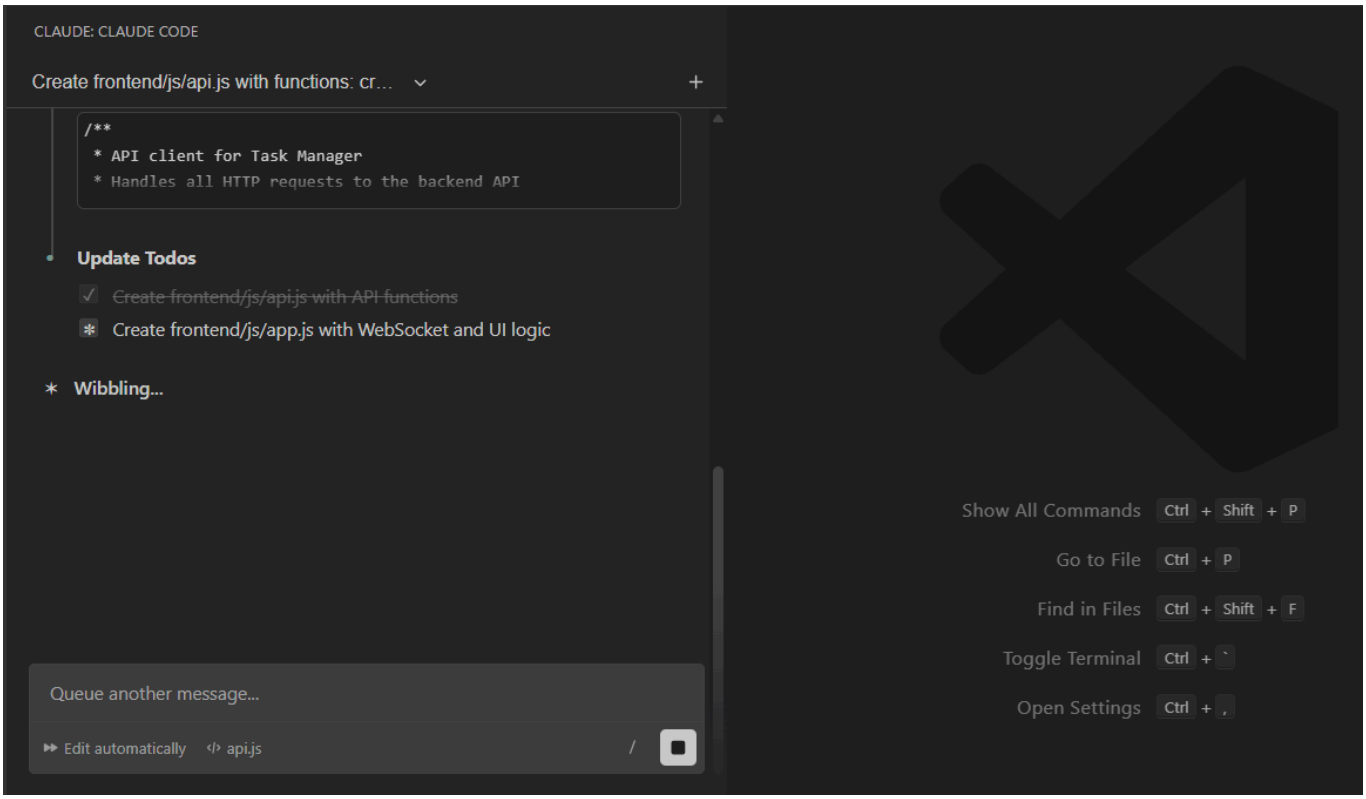
```
Create frontend/js/api.js with functions: createTask, getTasks,
updateTask, deleteTask. Use fetch, async/await, handle errors.
```

Prompt 12:

```
Create frontend/js/app.js with:
- WebSocket connection to ws://localhost:8000/ws
- Handle messages (task_created, task_updated, task_deleted)
- Reconnect logic
- renderTasks function with filtering
- Form submit handler
- Status change buttons
- Delete with confirmation
- Load tasks on page load
```

Mode: Ask before edits

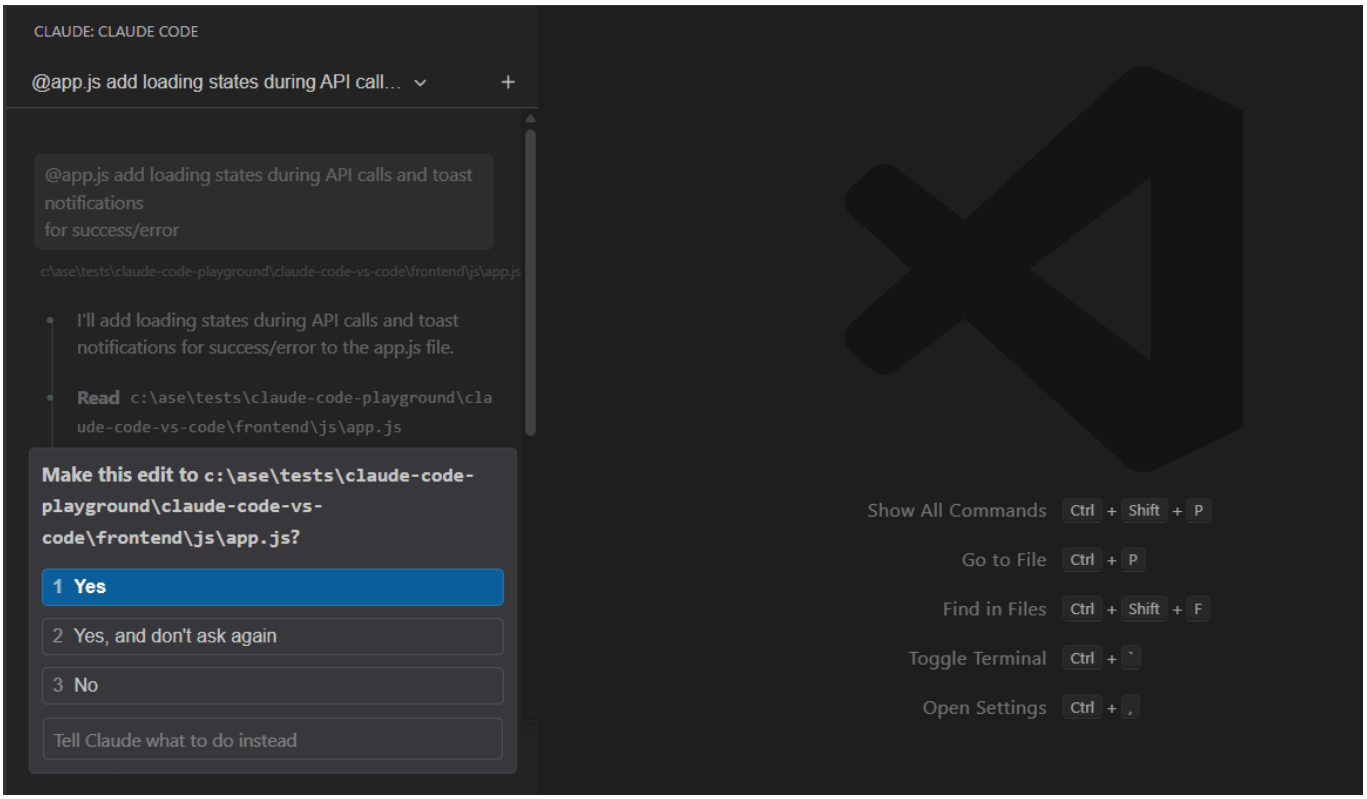
Claude creates both files. I review each. Accept.



Prompt 13:

```
@app.js add loading states during API calls and toast notifications
for success/error
```

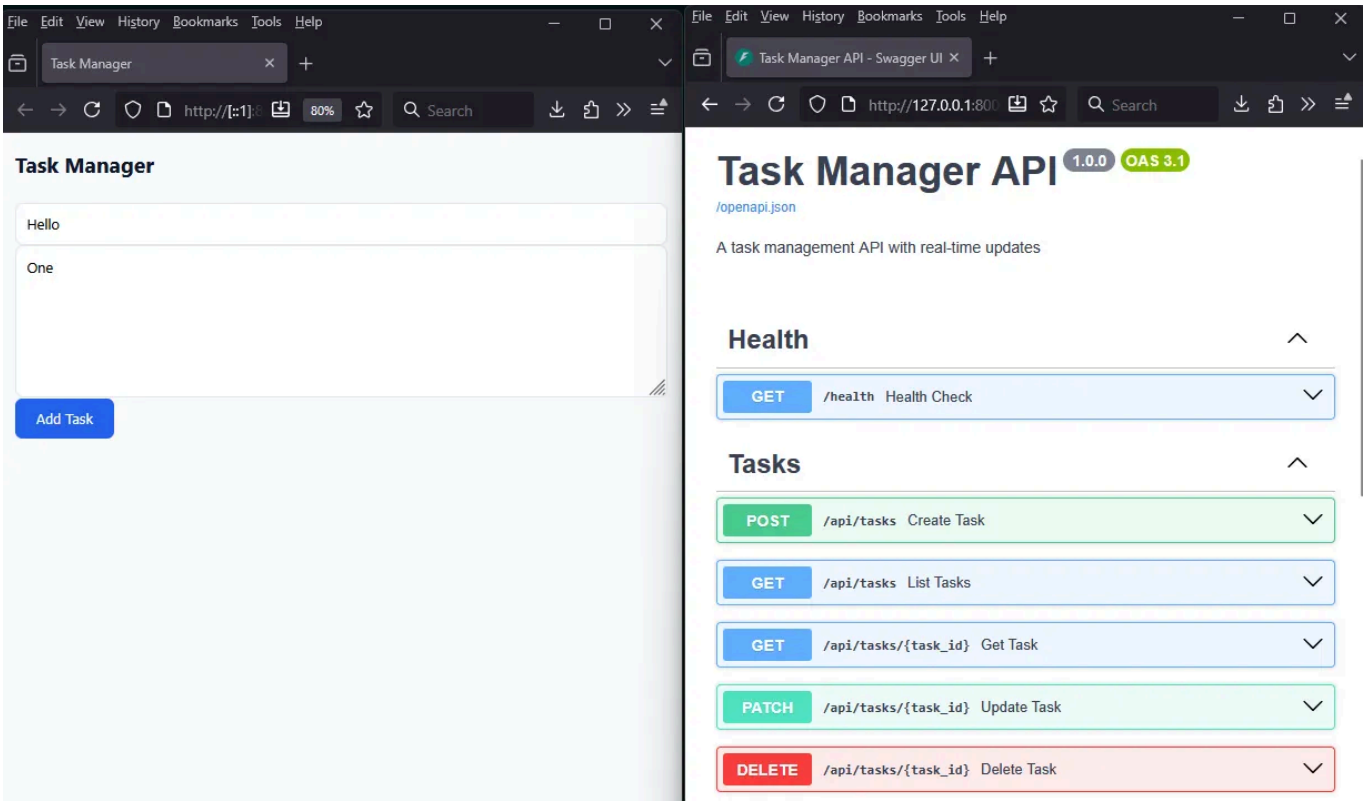
Accept.



Test frontend:

```
cd frontend
python -m http.server 8080
```

Visit: http://localhost:8080



Final Product Review

Total Build Time

- Backend: 8 minutes
- Frontend: 7 minutes
- **Total: 15 minutes**

What We Built

Backend:

- FastAPI with async SQLite
- Full CRUD API
- WebSocket real-time updates

- Pydantic validation
- Error handling
- CORS configured

Frontend:

- Vanilla JavaScript
- Real-time UI updates
- Status filtering
- Toast notifications
- Loading states
- Responsive design

Features:

- Create tasks
- Update status
- Delete tasks
- Filter by status
- Real-time collaboration
- Clean modern UI

Key Observations

Ask before edit mode:

- Best for new features
- See every change
- Catch issues early

Plan mode:

- Perfect for multi-file changes
- Review the architecture first
- One approval for the entire feature

@-mentions:

- Give Claude the file context
- More accurate changes
- Less back-and-forth

Slash commands:

- `/test` - Instant test generation
- `/review` - Code review
- `/docs` - Documentation

What I did:

- Wrote clear prompts
- Reviewed diffs
- Made decisions
- Tested features

Manual coding would take hours for this project, but with Claude Code vs Code extensions, it would take 20 minutes.

> *That's the difference.*

Final Thoughts

We built a full-stack task manager that fast.

In my final testing, it was good enough with all the functionality and with a few tests, and polishing up would make it a deployable project.

> *The key is using the right approach: specific prompts with @-mentions, choosing the correct mode for each task, and keeping conversations focused.*

Ask before edits for new features, Plan mode for multi-file changes, and save tokens by separating backend and frontend work into different conversations.

The extension isn't perfect; it does not have Checkpoints available yet.

However, it has genuinely improved, and the current state is good for improved productivity.

> ***You can install it and try it on a new project, and let us know about your experience.***

Let's Connect!

If you are new to my content, my name is Joe Njenga

> *Join thousands of other software engineers, AI engineers, and solopreneurs who read my content* ***daily on Medium*** *and on* ***YouTube where I review the latest AI engineering tools and trends.*** *If you are more curious about my projects and want to receive detailed guides and tutorials,* ***join thousands of other AI enthusiasts in my weekly AI Software engineer newsletter***

If you would like to connect directly, you can reach out here:

AI Integration Software Engineer (10+ Years Experience)

Software Engineer specializing in AI integration and automation. Expert in building AI agents, MCP servers, RAG...

njengah.com

Follow me on Medium | YouTube Channel | X | LinkedIn

Claude Code | Anthropic Claude Code | Vscode Extension | Vs Code Extensions

Vscode

Written by Joe Njenga

4.5K followers · 99 following

Follow

Software & AI Automation Engineer, Tech Writer & Educator. Vision: Enlighten, Educate, Entertain. One story at a time. Work with me: mail.njengah@gmail.com

No responses yet

Bgerby

What are your thoughts?

More from Joe Njenga

Joe Njenga

I Tested Upgraded Claude Code 2.0 (And Discovered 7 New Features...

Claude Code just dropped version 2.0, and they packed it with the most requested...

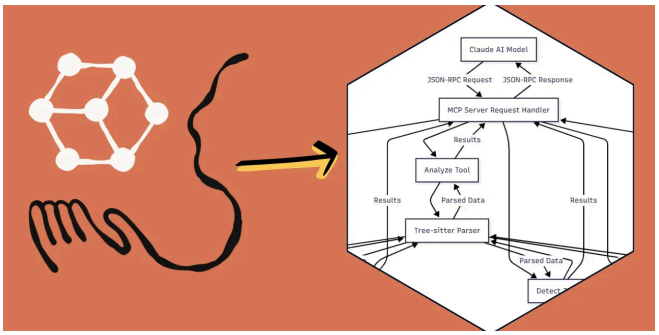
Sep 30 361 12

Joe Njenga

12 Little-Known Claude Code Commands That Make You a Whiz

What if I told you there are some Claude Code commands that, although not very popular,...

Sep 21 250 4

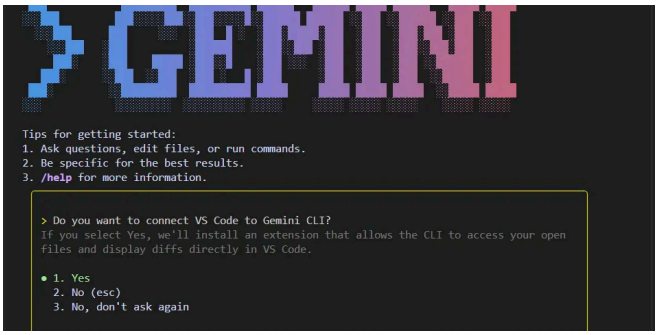


Joe Njenga

I Asked Claude 4.5 to Build Me a Complex MCP Server (It Was So...

If you want to build your custom MCP server, stop wasting time thinking, coding, or trying...

4d ago 361 7



Joe Njenga

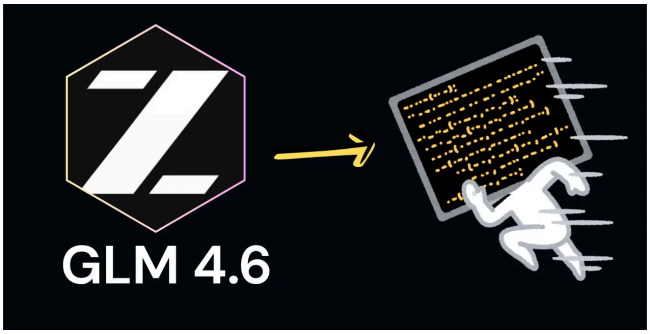
I Tested The Updated Gemini CLI (And Found These New Features...

Gemini CLI has these newly updated features that are making it edge closer to dethroning...

Sep 12 161 6

See all from Joe Njenga

Recommended from Medium

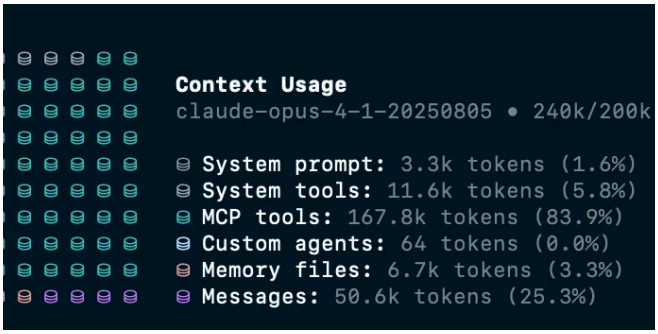


Joe Njenga

I Tested Newly Released GLM 4.6 (And Discovered a Cheaper Way t...

GLM 4.6 has been released just a few days after Claude 4.5, and it's another coding...

2d ago 39 1



Sevak Avakians

Claude Code Limit Hit on Max Plan?! What to do next:

$200/m plan blocks me for a week!

5d ago 63 15

Reza Rezvani

The ultimate Code Modernization & Refactoring prompt for your...

Transform your legacy codebase chaos into a strategic modernization roadmap with this...

Akshay khale

Setting up your MacBook for Development.

A refresher on my previous article posted 5 years ago.

4d ago 77

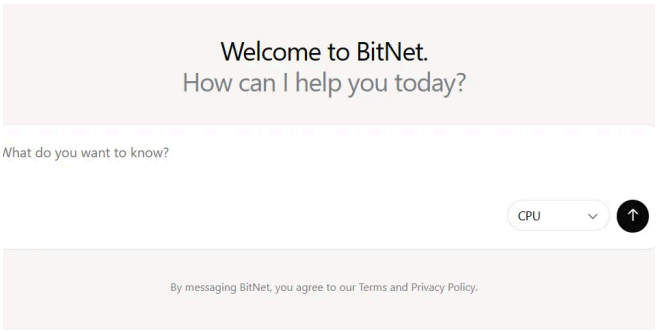
In Artificial Intelligence in Plain E... by Prithwish N...

5 Essential MCP Servers That Give Claude & Cursor Real Superpower...

Transform Claude & Cursor into web scraping, browser-controlling automation engines. 5...

Sep 27 182 1

Jul 28 105 5



In Coding Nexus by Algo Insights

Microsoft Just Declared War on the GPU Mafia: Meet bitnet.cpp

Bitnet.cpp will break the GPU Lock

4d ago 244 3

See more recommendations