Cursor 101 Demo Flow (with Screenshots)

An introductory walkthrough of Cursor's core features for new users.

Prerequisites

Before starting the demo:

1. **Start the server:** Run /start-demo to launch the development server and open the blog app in the browser
2. **Reset workspace:** Use /reset between sections to ensure a clean state for deterministic results

> 📝 **Deep Links:** Throughout this demo, clickable links like `[Example Link]` open a "Create chat with prompt" dialog. Click **"Create Chat"** to start the agent with the pre-filled prompt.

---

1. Layout Walkthrough

> Cursor is an IDE (Integrated Development Environment) forked from VS Code, now fully built out with AI-native features.
>
> Understanding the layout helps you navigate efficiently and customize your workspace for different tasks.

Overview

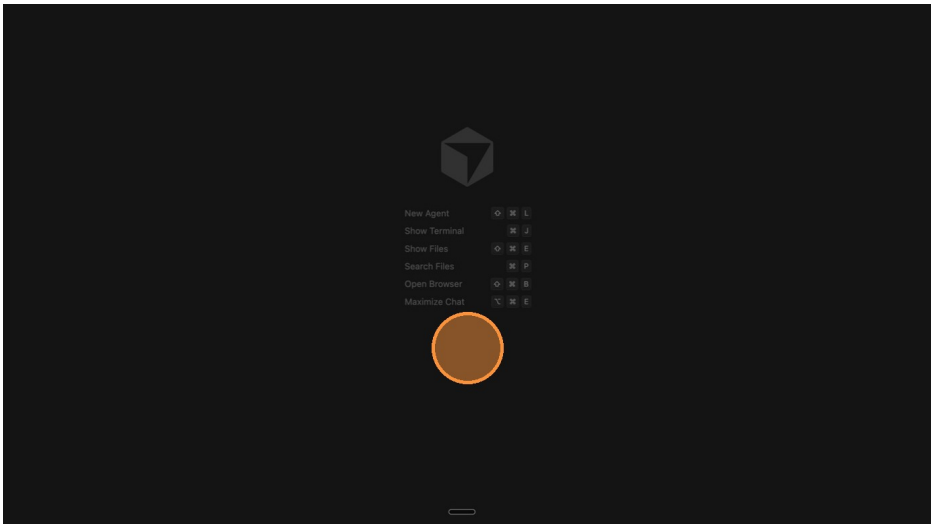
Cursor extends VS Code with AI-powered panels and features. The main areas include:

- **File Explorer** (left sidebar) - Navigate your project files
- **Editor** (center) - View and edit code
- **Terminal** (bottom panel) - Run commands
- **Agent Panel** (right sidebar) - AI chat and agent interactions

Demo

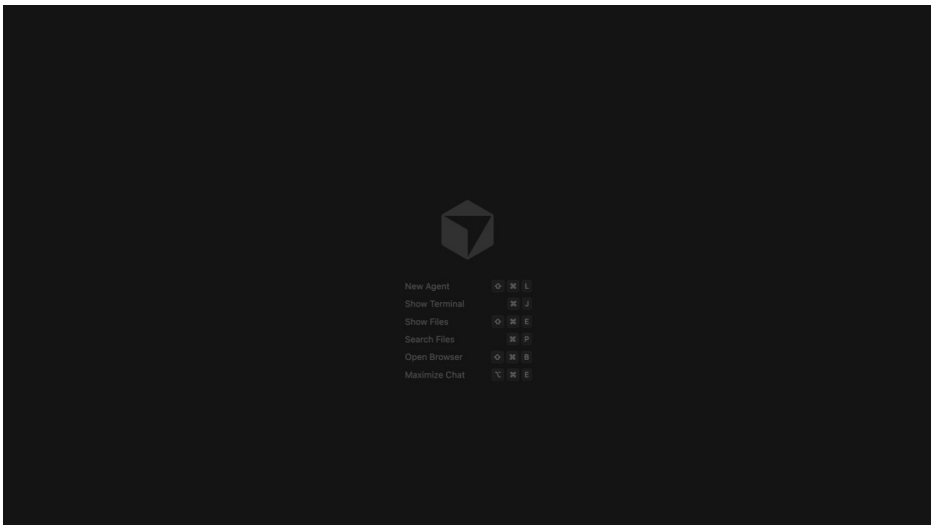
Opening the Layout

When you first open Cursor, you'll see an empty editor:



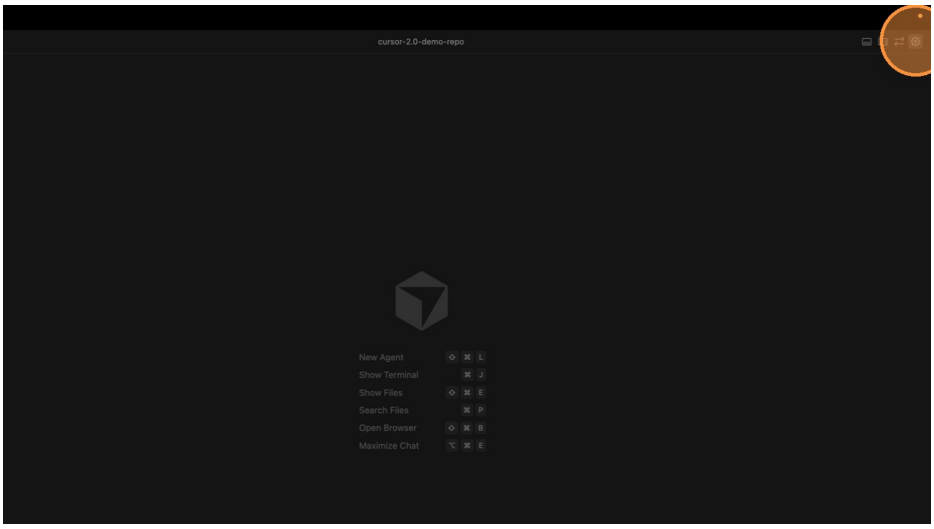
Empty editor view

Click "Change Layout" to customize your view:



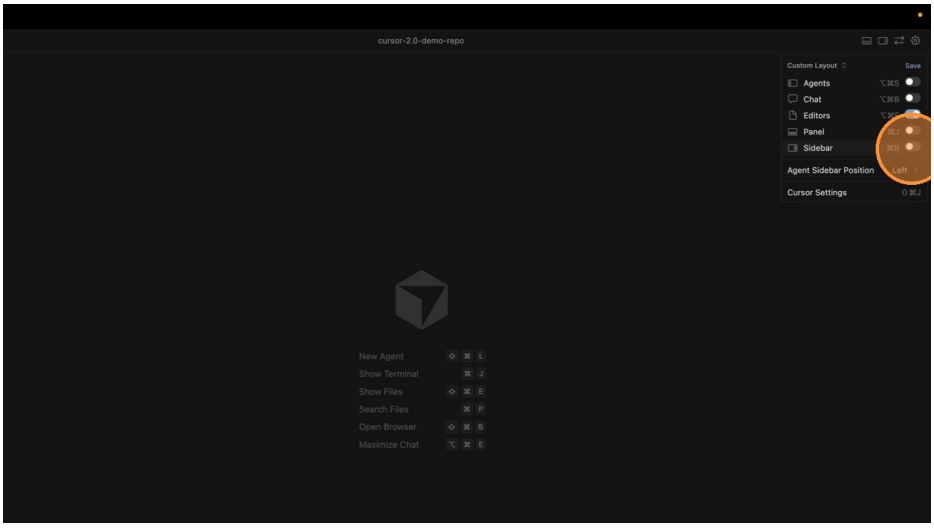
Change layout button

The layout options appear—Cursor is a fork of VS Code, now fully AI-native:



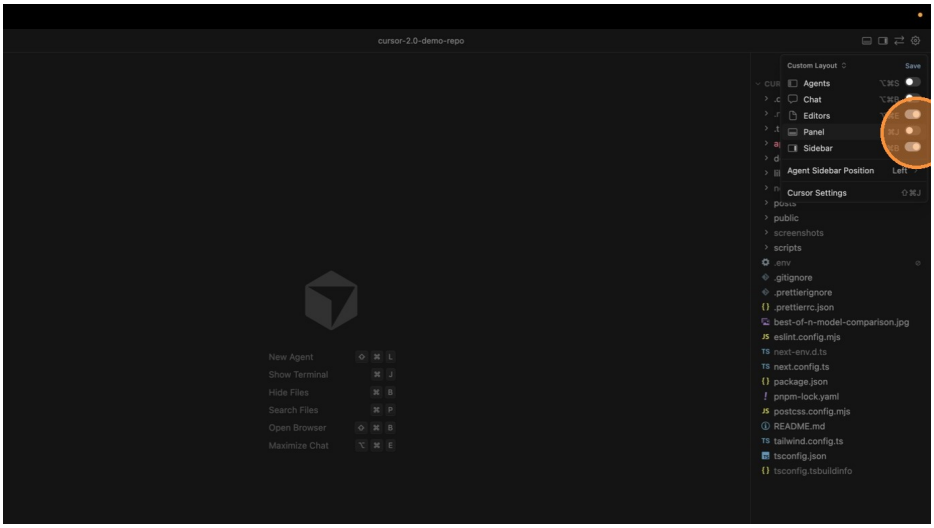
Layout options

Click the layout menu to see available options:



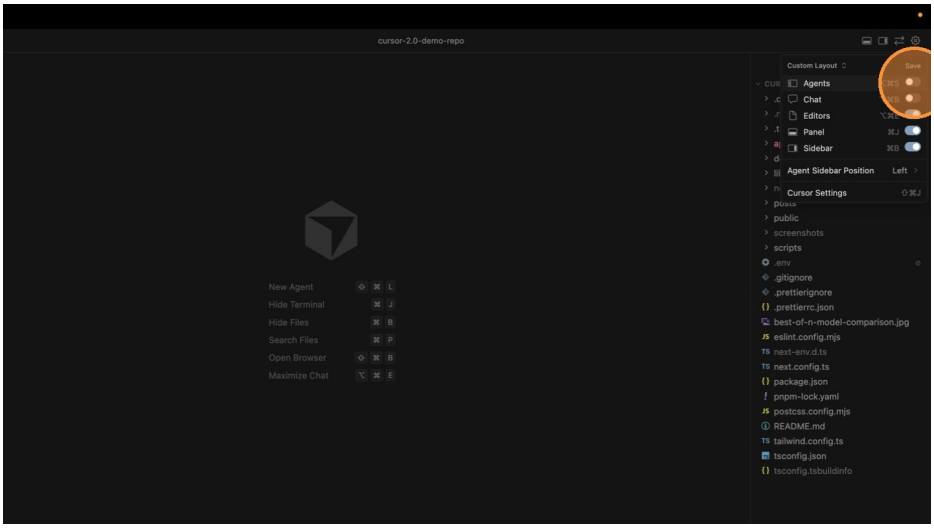
Layout menu

Click the File Explorer icon to see your project's file system:



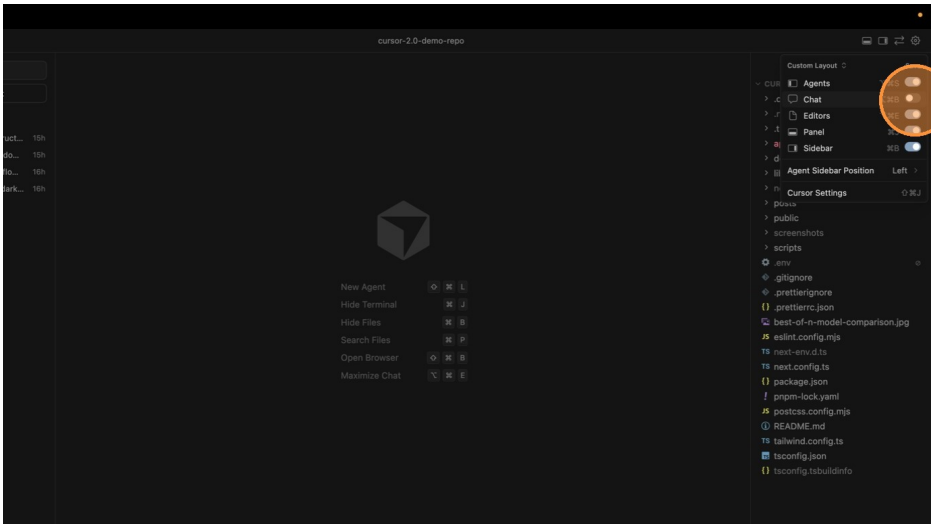
File Explorer panel

Click the Terminal area to open the integrated terminal:



Terminal panel

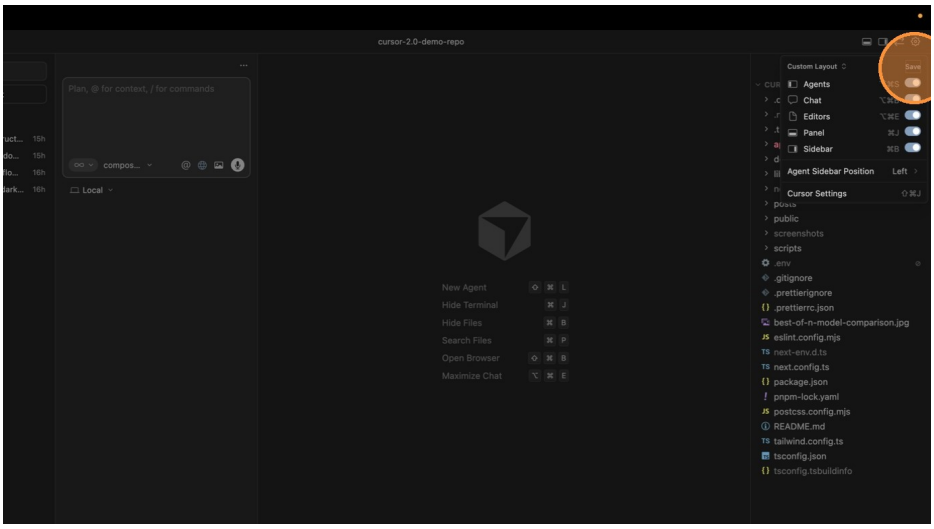
Click to open the Agent panel for AI chat and interactions:



Agent panel

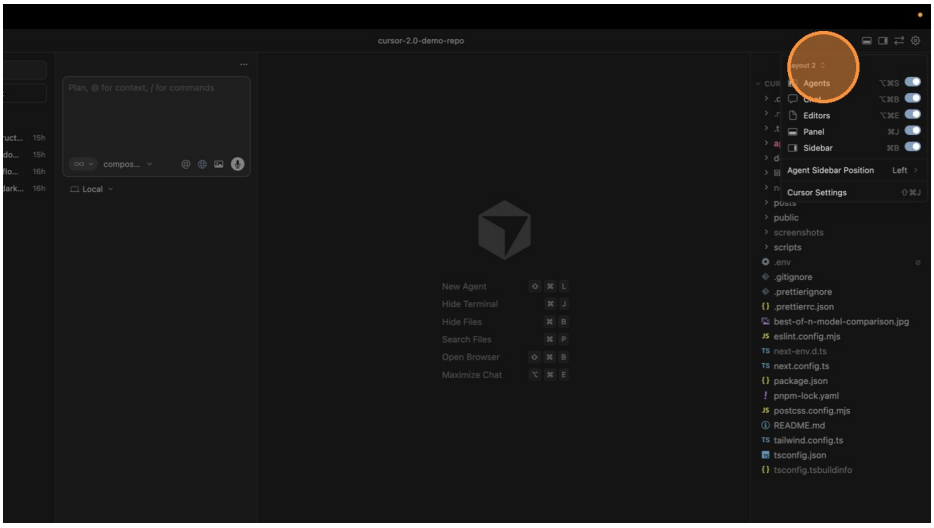
Saving Custom Layouts

Save your preferred layout for quick switching:



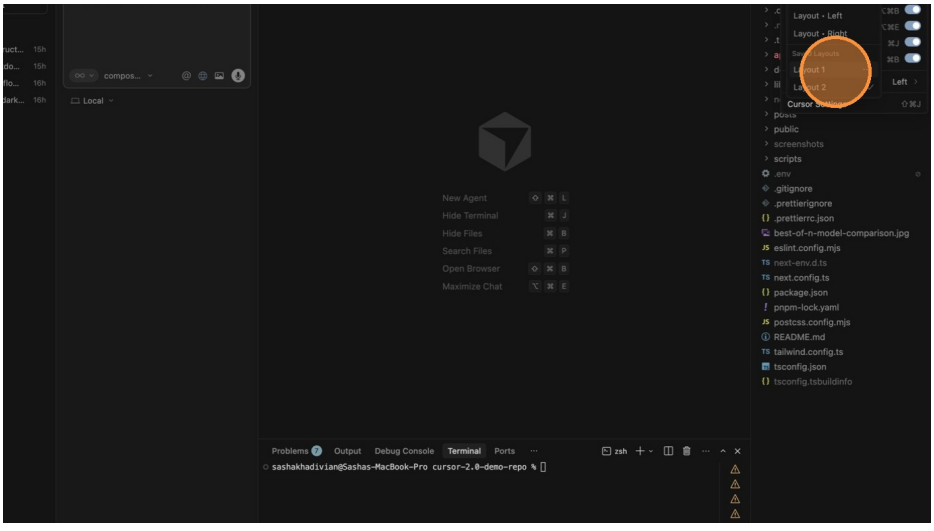
Save custom layout

Select a saved layout from the menu:



Select saved layout

Switch between different layouts as needed:



Switch layouts

What to Highlight

- Familiar VS Code experience with AI enhancements
- Customizable panel arrangement
- Quick layout switching for different workflows
- Integrated terminal for running commands alongside AI

Tips

- Use keyboard shortcuts to toggle panels quickly
- Save different layouts for coding vs. AI-assisted tasks
- The agent panel can be resized or detached

---

2. Tab + Quick Edit (Cmd+K)

> Cursor Tab provides context-aware, multi-line code suggestions as you type. It can modify multiple lines at once, add import statements when missing, and predict your next editing location within or across files.
>
> Inline Edit (Cmd+K) lets you edit code or generate new code directly within your editor using natural language. With code selected, it edits that specific code; without a selection, it generates new code at your cursor position.

Docs: Tab Overview | Inline Edit

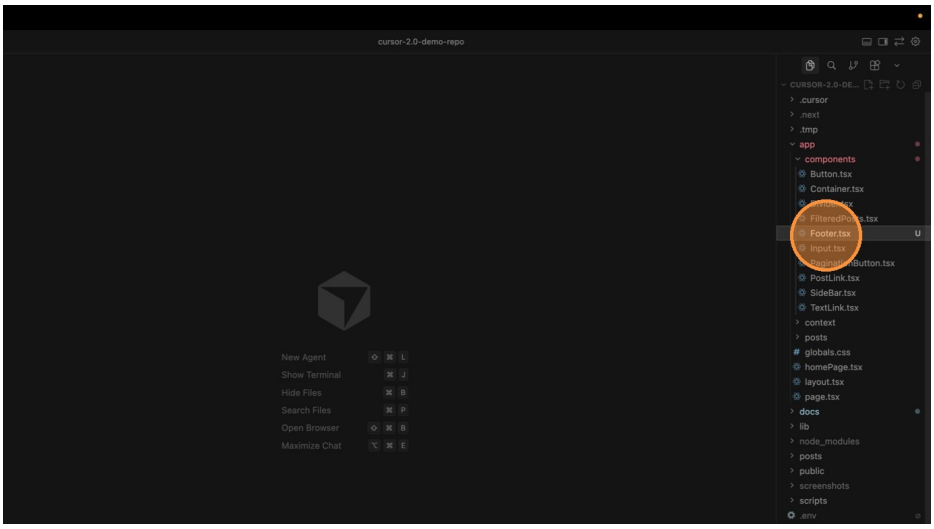
Tab Autocomplete

Overview

Tab predicts what you want to write next and suggests complete code blocks. The more you use it, the more it learns your style.

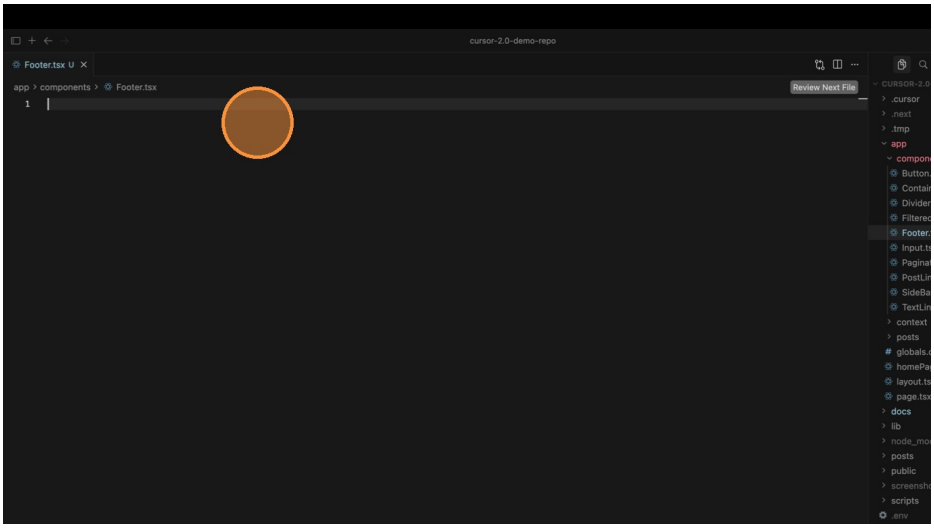
Demo

Open an empty component file like `Footer.tsx`:



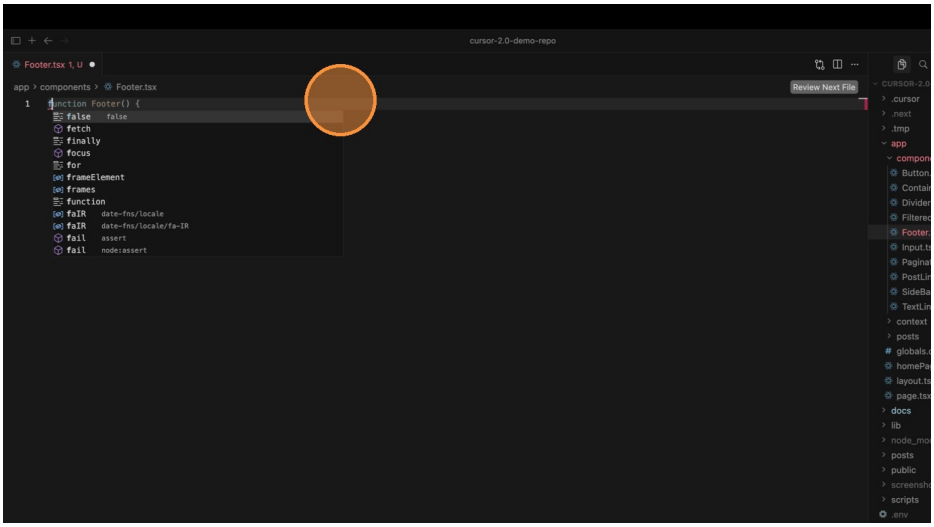
Open Footer.tsx

Type the letter “F”—Tab detects you’re in `Footer.tsx` and suggests the component:



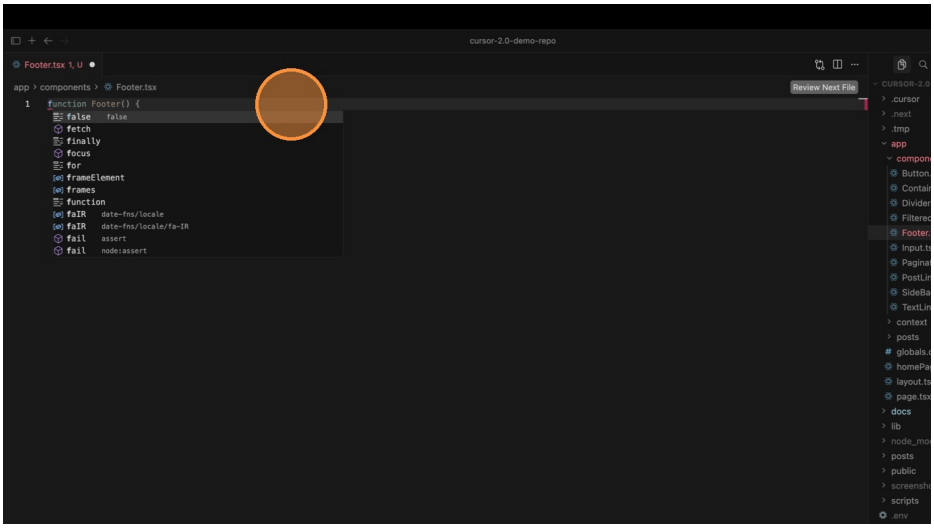
Start typing F

Tab suggests a complete Footer component structure:



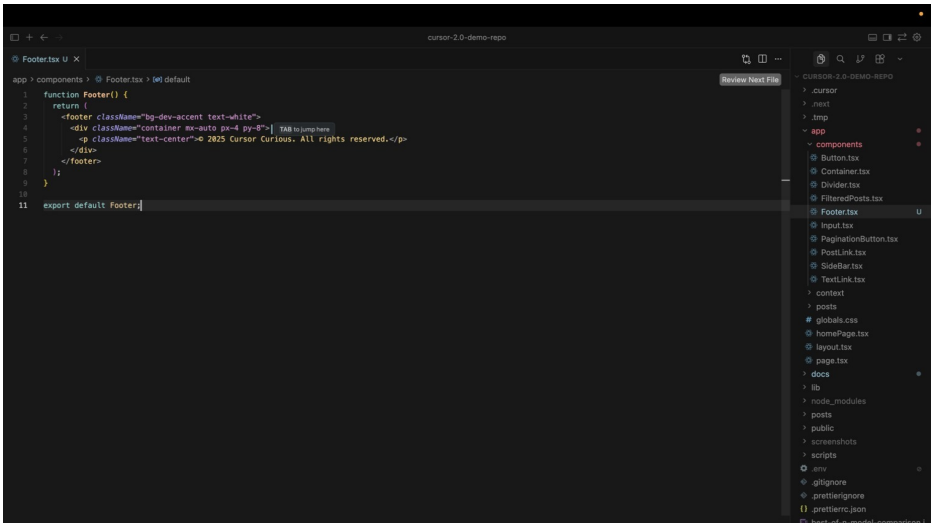
Tab suggestion appears

Press Tab to accept and continue building:



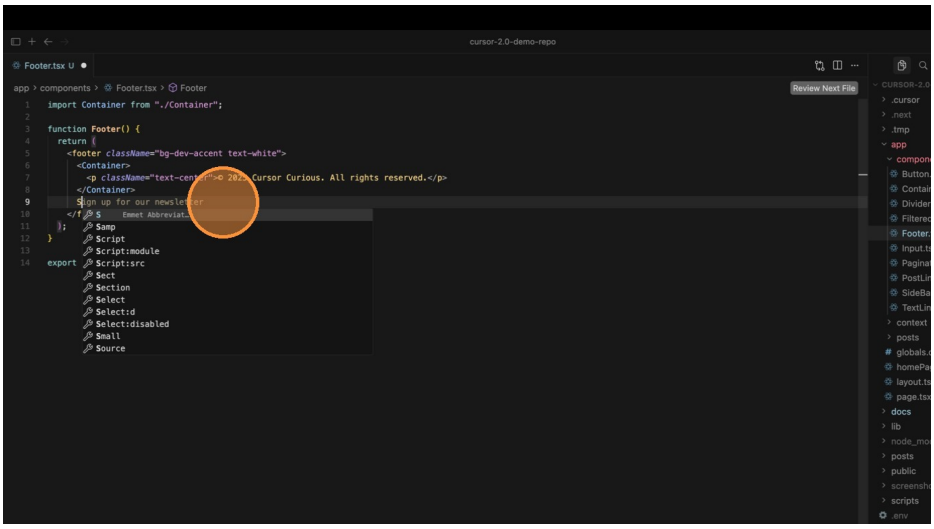
Accept Tab suggestion

Keep pressing Tab—it builds out the component iteratively:



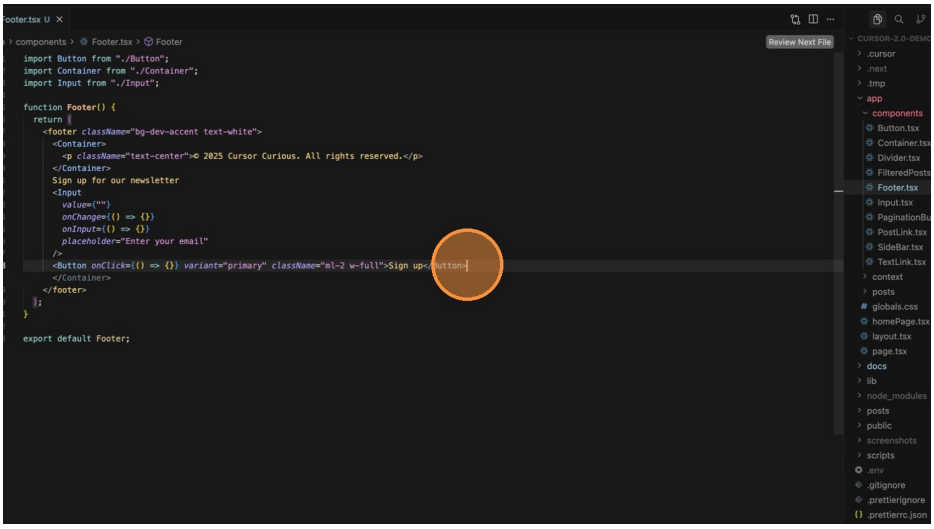
Continue with Tab

Click on the line above the closing footer tag and type “sign up”—Tab starts to auto-complete:



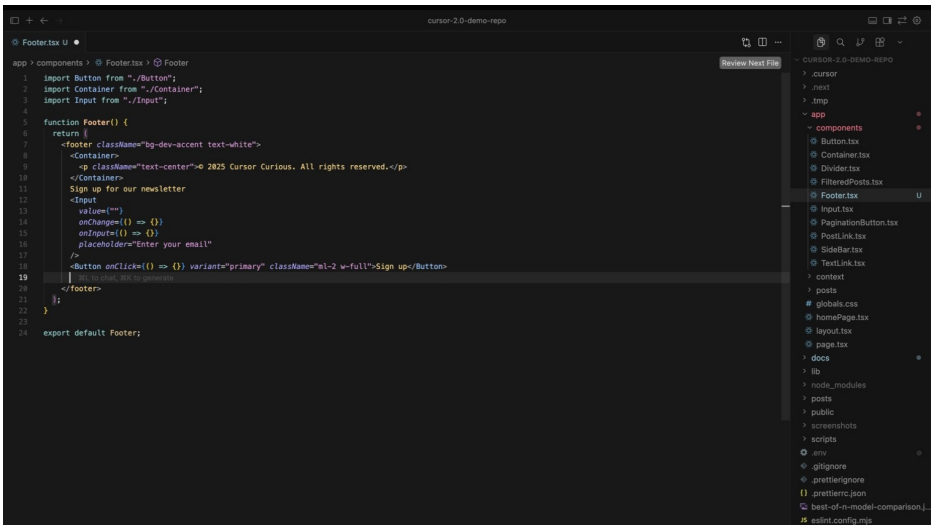
Type sign up hint

Press Tab to accept—it uses components from your codebase to build the feature:



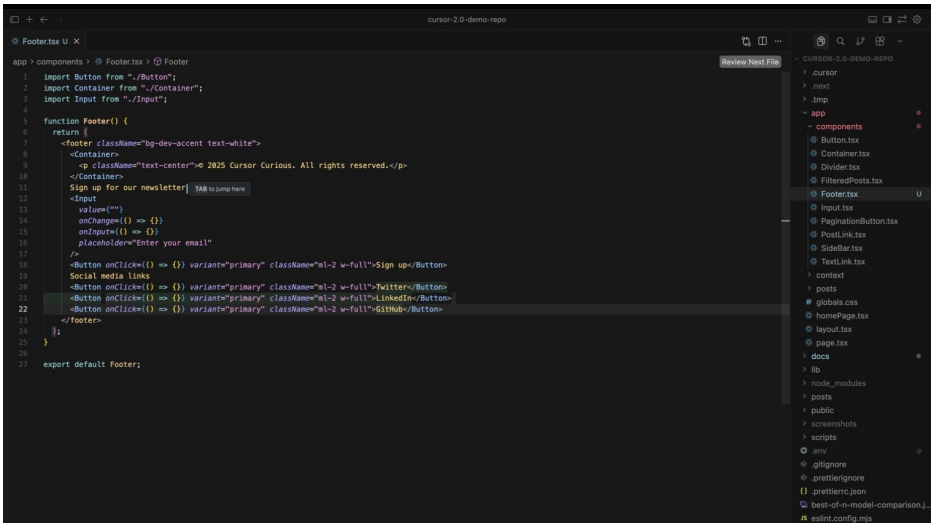
Tab uses existing components

Continue pressing Tab to iterate quickly—express ideas and Tab builds them out:



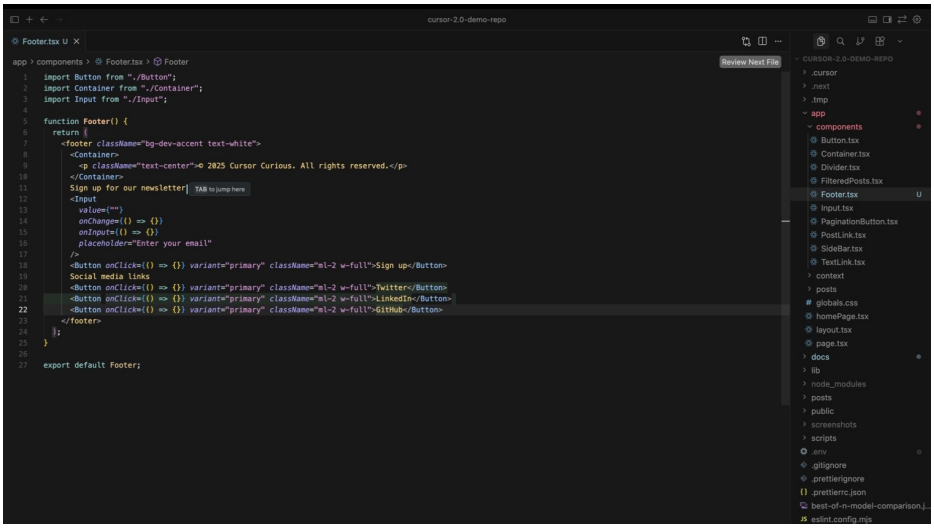
Rapid iteration with Tab

Type “social media links” and Tab adapts to your intent—press Tab as it auto-completes:



Type social media links

Tab now uses social links instead of the signup form:



Tab adapts to changes

What to Highlight

- Contextual awareness—knows what file you’re in
- Uses your existing components and patterns
- Gets smarter the more you use it
- Feels like “reading your mind” for power users

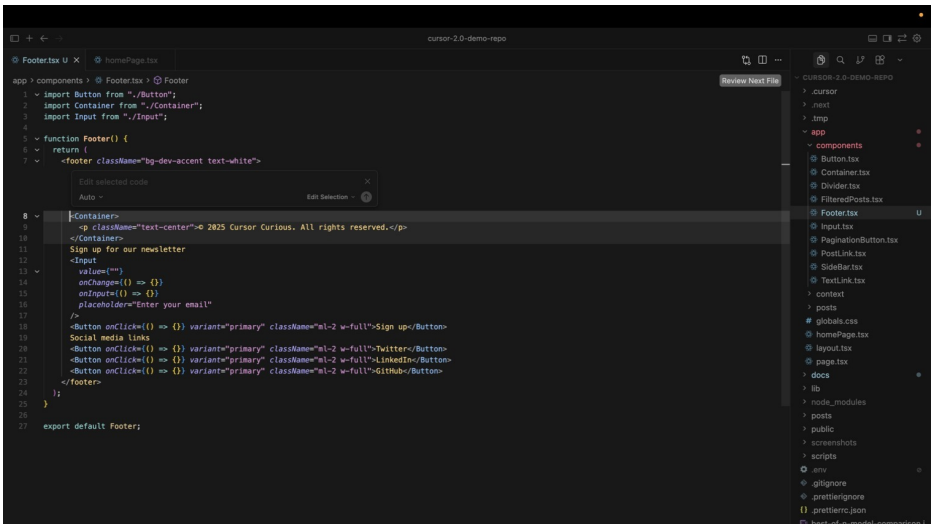
Quick Edit (Cmd+K)

Overview

Select code and press `Cmd+K` to make targeted edits using natural language. Only the selected section is modified.

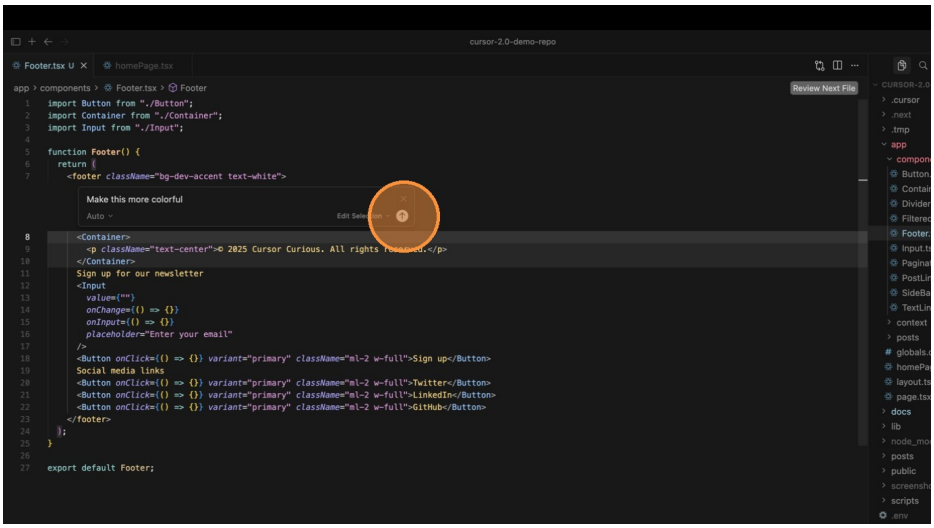
Demo

Select a section of code you want to edit and press Cmd+K to open the quick edit panel:



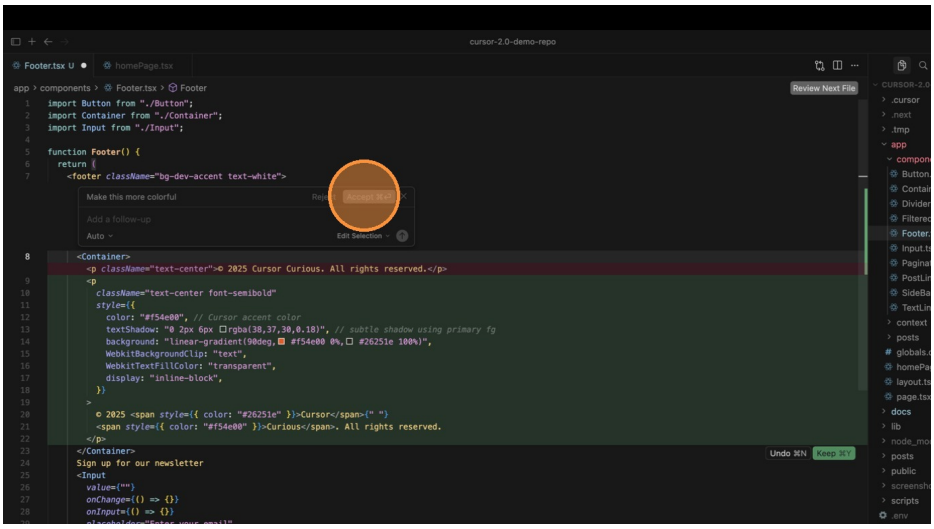
Quick edit prompt

Type your instruction—“make this more colorful”:



Type instruction

Cursor edits only the selected section and shows the diff:



View the diff

What to Highlight

- Surgical edits—only changes what you select
- Natural language instructions
- Preview changes before accepting
- Great for refactoring specific sections

Best Practices

- **Tab:** Start with hints in comments or partial code to guide suggestions
- **Quick Edit:** Be specific about what you want changed
- **Combine them:** Use Tab for building, Quick Edit for refining

---

3. Agent Chat

> Agent Chat is the command center for agentic workflows within Cursor. An agent is a large language model (LLM) with access to tool calls—in this case, the ability to make changes directly to your codebase.

Docs: Agent Chat

Overview

This demo covers all the key configurations and features of Agent Chat.

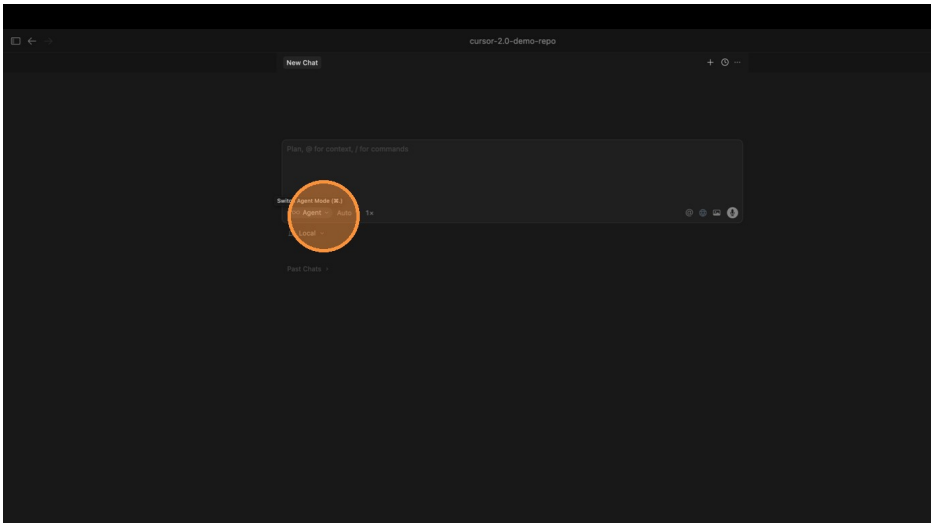
3.1 Agent Modes

> Cursor provides different modes optimized for different tasks. Choose the right mode for your workflow.

Mode	Use Case
Agent	

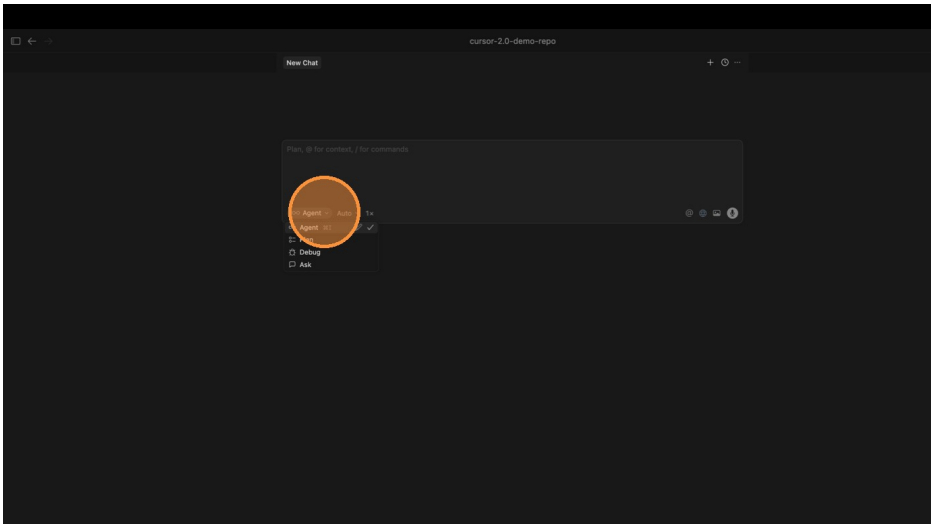
Mode	Use Case
	Execute code changes across your codebase (default)
Plan	Create editable implementation plans for complex features
Debug	Find, test, and resolve difficult bugs
Ask	Learn about the codebase without making changes (read-only)

Click the mode dropdown to see available options:



Agent mode selector

Select from Agent, Plan, Debug, or Ask mode:



Mode options

3.2 Model Selection

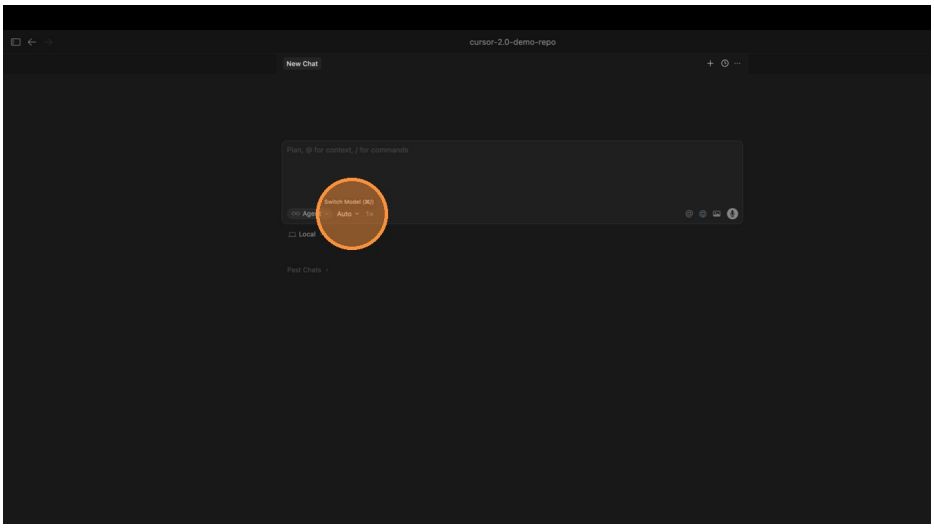
> You have full control over which model to use. Choose based on your task complexity and speed requirements.

- **Composer 1**: Cursor's own coding model, optimized for speed

- **Auto**: Let Cursor smartly select the best model for each task
- **Frontier models**: Claude, GPT-4, Gemini, and more

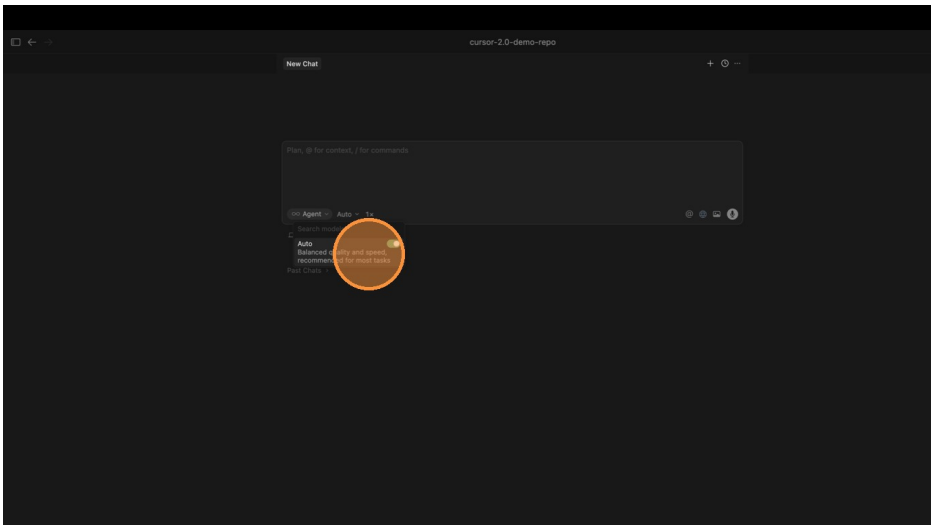
Quick Selection

Click the model selector:



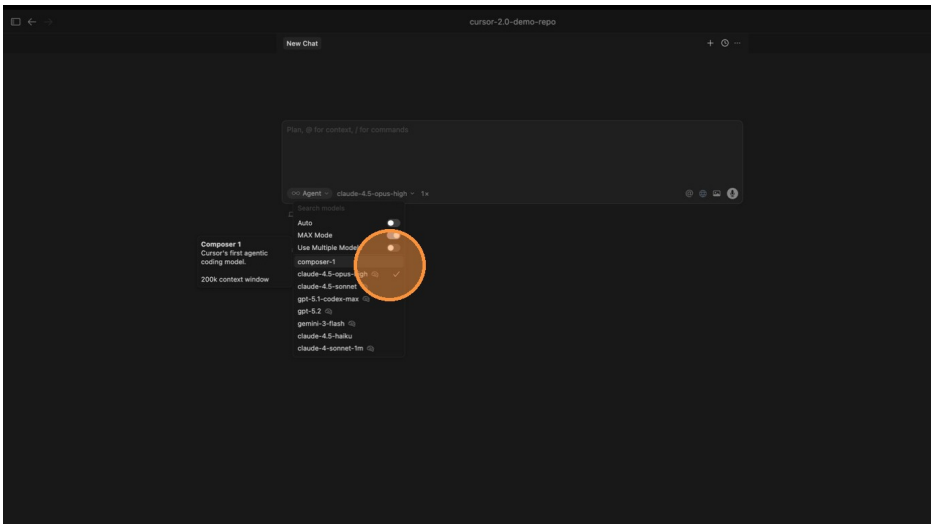
Model selector

Choose from available models:



Model options with descriptions

Model categories and Auto selection:



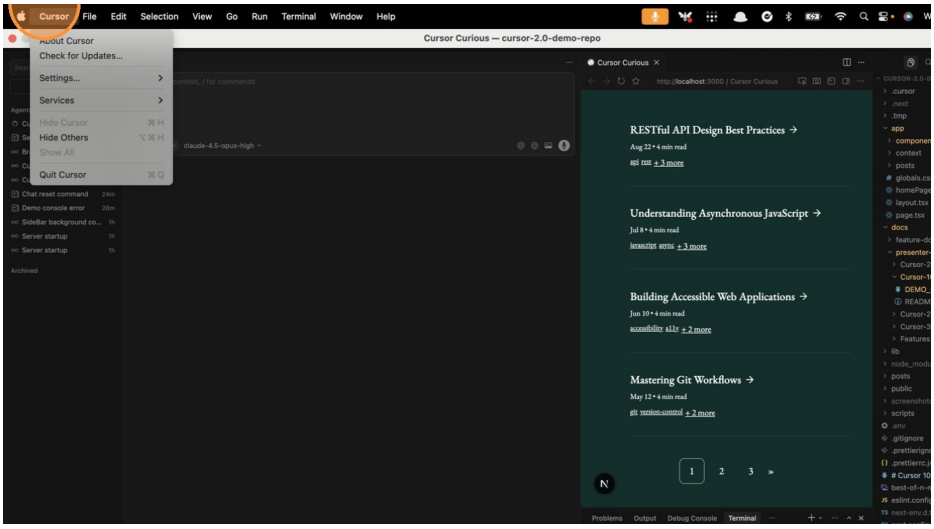
Model categories

Model Settings

For more control, configure models in **Cursor Settings > Models**:

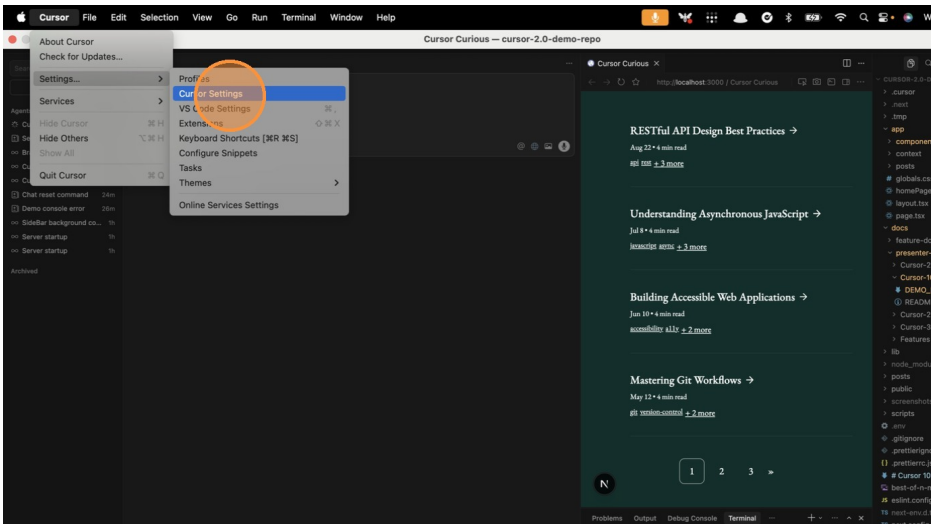
1. Open **Cursor** menu > **Cursor Settings**
2. Navigate to **Models**
3. Click **View All Models** to see all available options
4. Enable or disable models based on your needs

Open Cursor Settings:



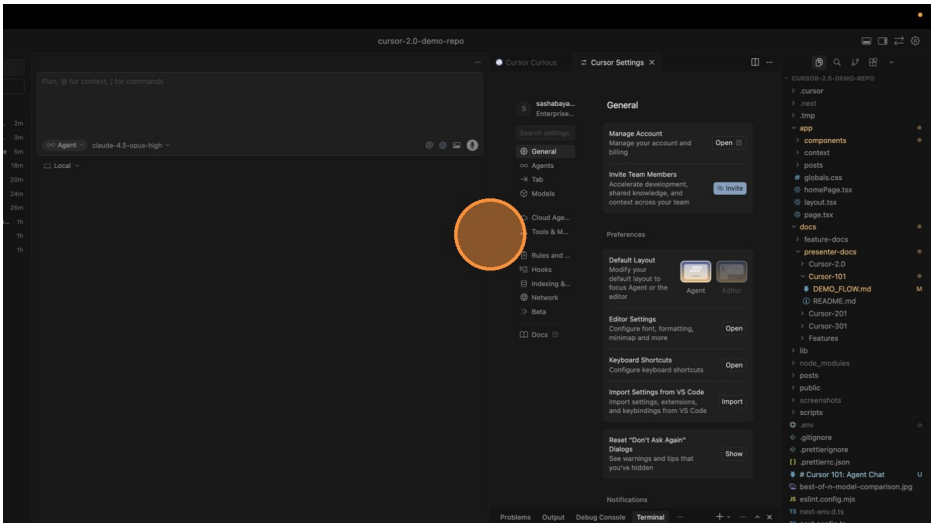
Cursor menu

Click Cursor Settings:



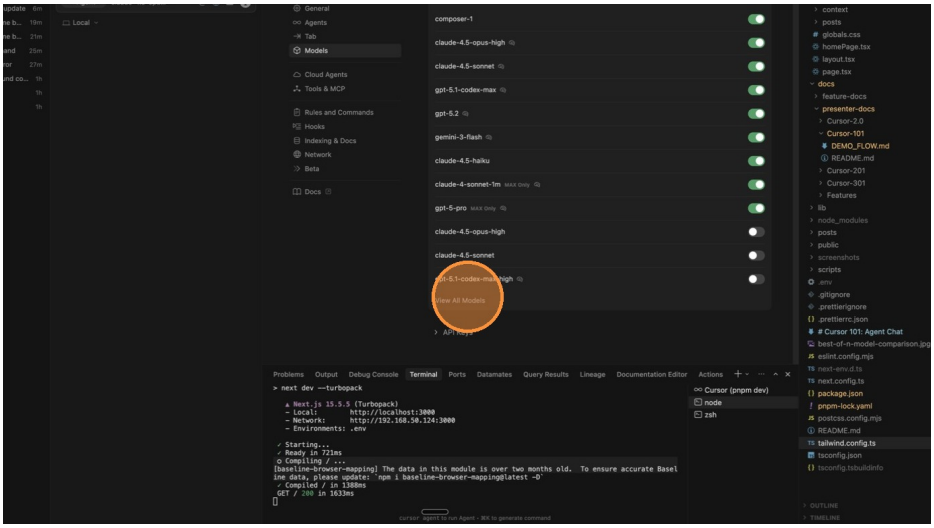
Cursor Settings option

Navigate to Models section:



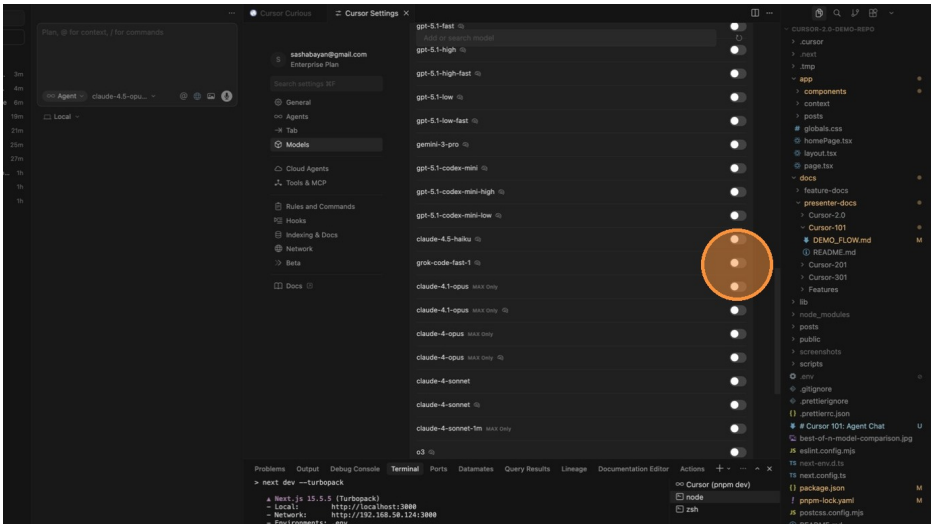
Models section

Click View All Models:



View All Models button

Enable or disable models:



Model list with toggles

3.3 Adding Context

> Provide the agent with relevant context so it can make informed decisions.

Why This Matters

Context is everything when working with AI. Without the right context, even the best model will give generic or incorrect answers. The friction of adding context—switching between apps, copying documentation from ChatGPT or Claude's web interface, pasting error messages—creates constant context switching that slows you down more than you realize.

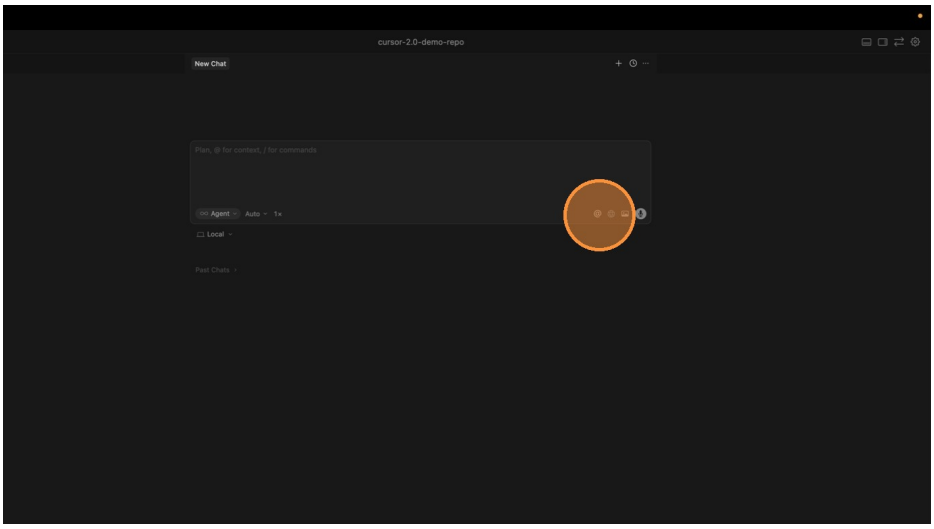
Cursor makes adding context fast and seamless. Instead of juggling browser tabs and copy-pasting between applications, you simply type `@` and reference what you need. This eliminates the hidden productivity tax of context switching, letting you stay in flow while giving the agent exactly what it needs to help you.

How to Add Context

Use the **Add Context** button (or type `@`) to include:

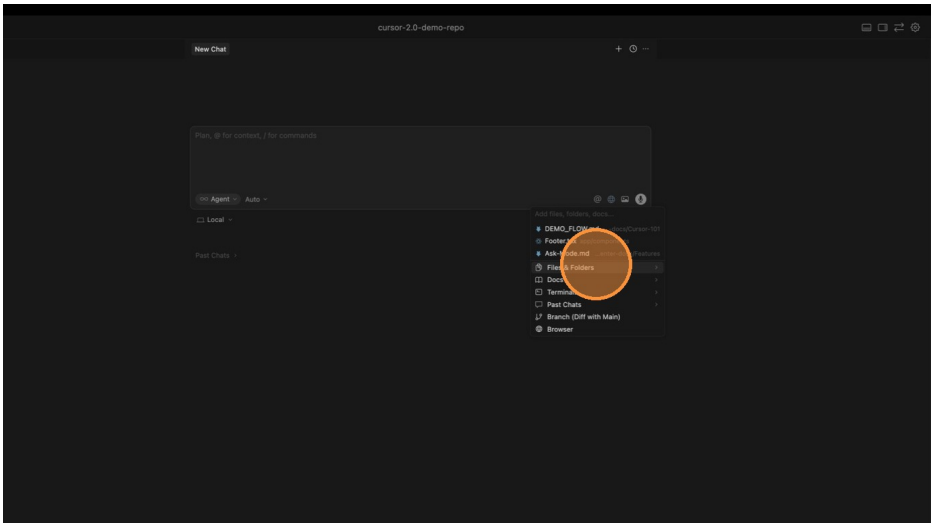
- **Files** – Specific files to reference or modify
- **Folders** – Entire directories for broader context
- **Docs** – Up-to-date documentation from popular libraries
- **Terminal** – Terminal output for debugging errors
- **Past Chats** – Reference previous conversations

Click the Add Context button:



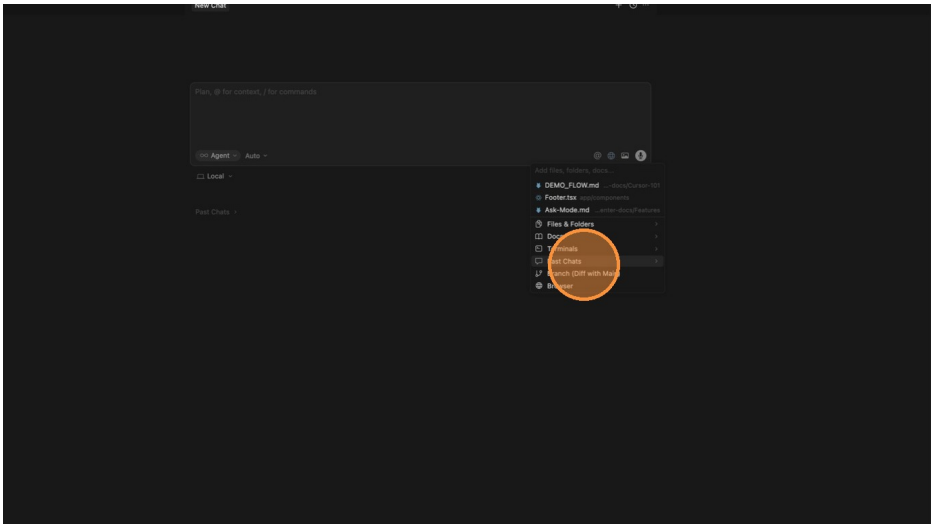
Add Context button

Context options menu:



Context options

Available context types:



Context types expanded

3.4 Image Upload

> Upload images as reference for the agent—mockups, screenshots, or error messages.

Click the image upload button:

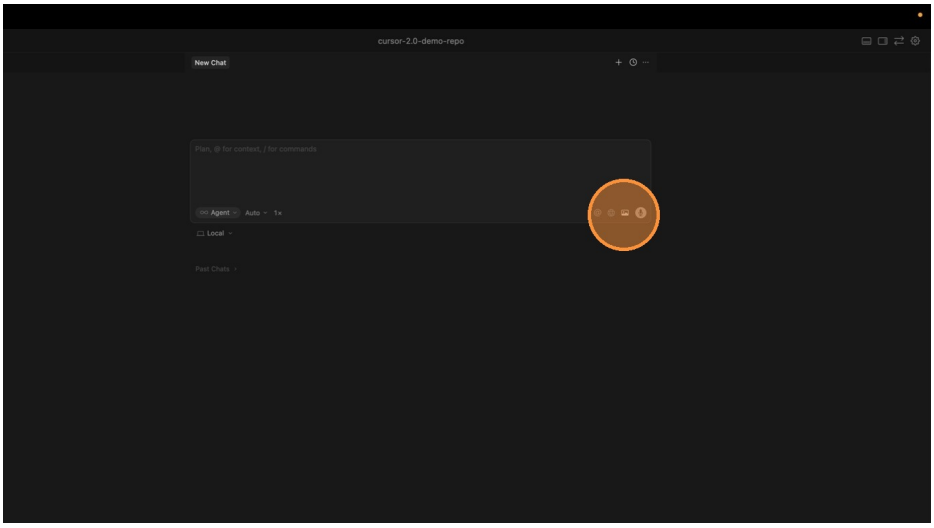


Image upload button

Image attached to chat:

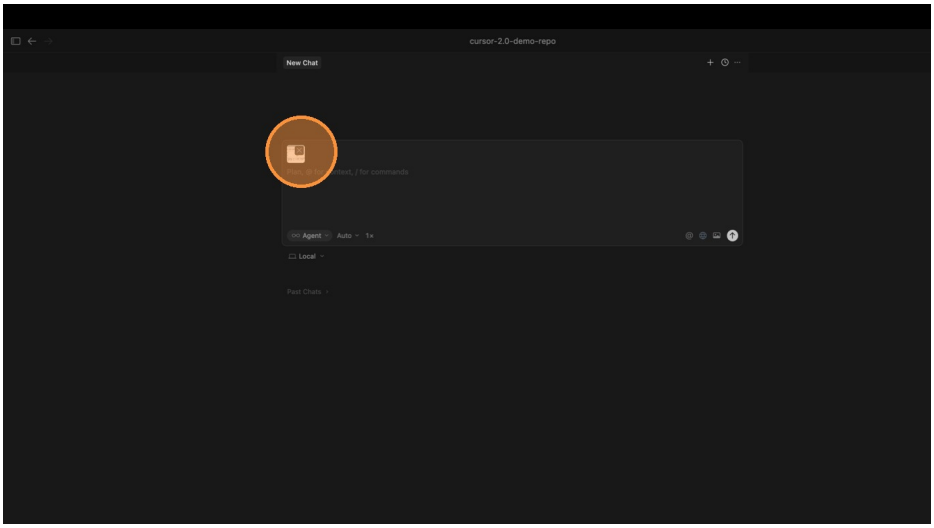
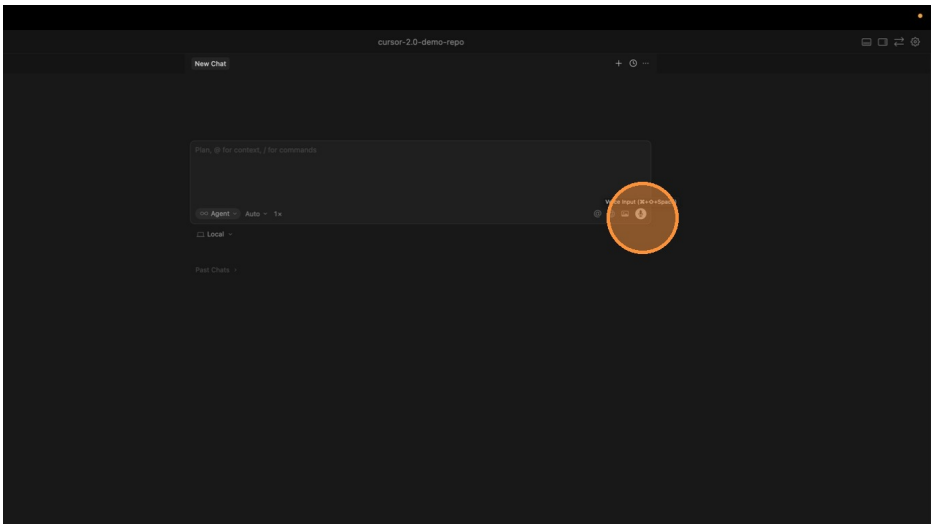


Image in chat

3.5 Voice Input

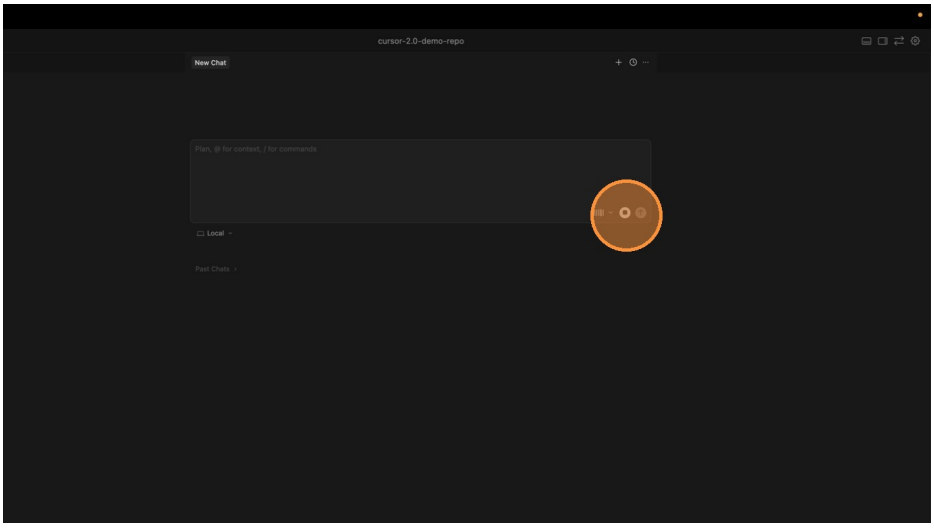
> Speak your prompts instead of typing for faster iteration.

Click the microphone button to record:



Voice input button

Recording in progress:



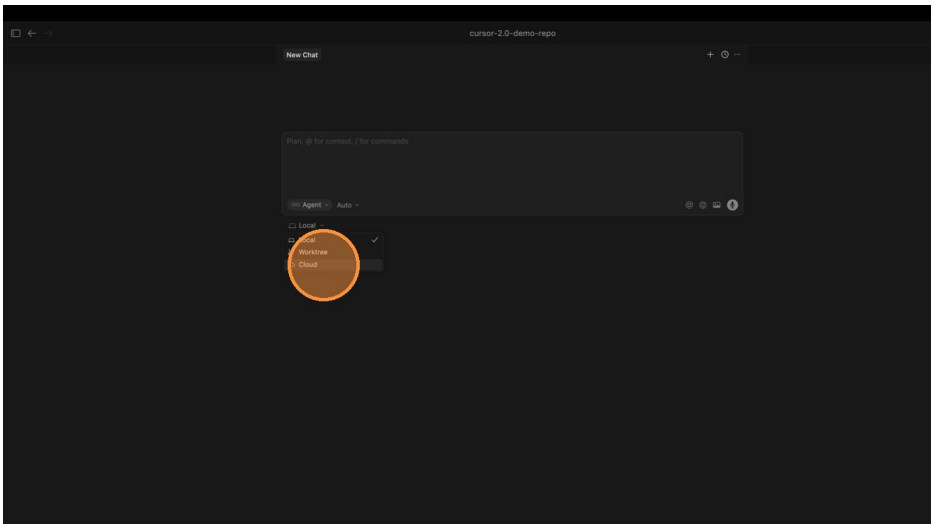
Voice recording active

3.6 Execution Mode

> Choose where your agent runs based on your needs.

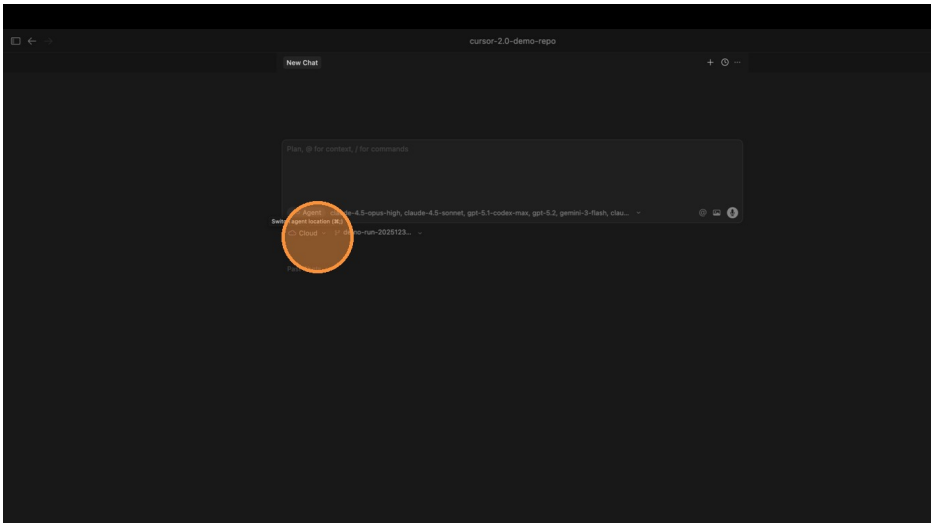
Mode	Description
Local	Runs on your machine with full access to local files
Cloud	Runs on Cursor’s cloud infrastructure
Worktree	Run parallel agents in isolated worktrees (advanced)

Local execution mode:



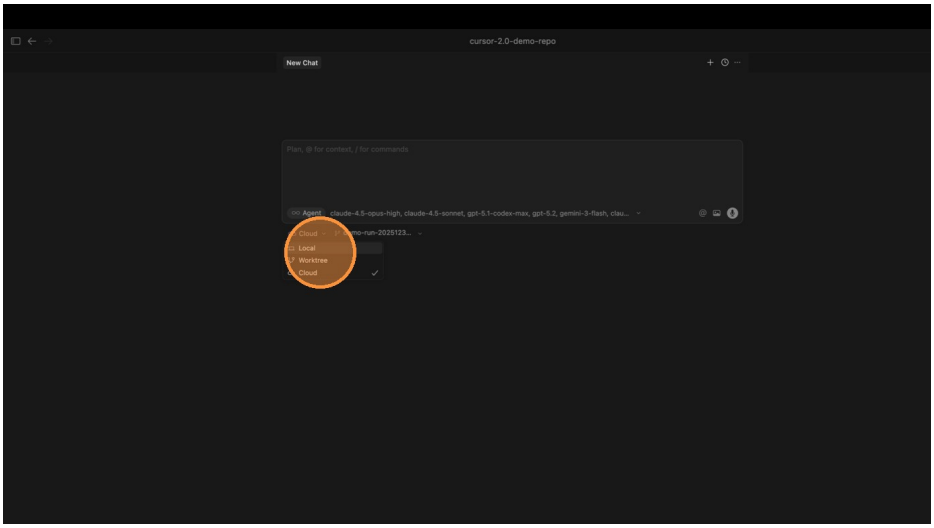
Local mode

Cloud execution mode:



Cloud mode

Worktree mode (advanced):



Worktree mode

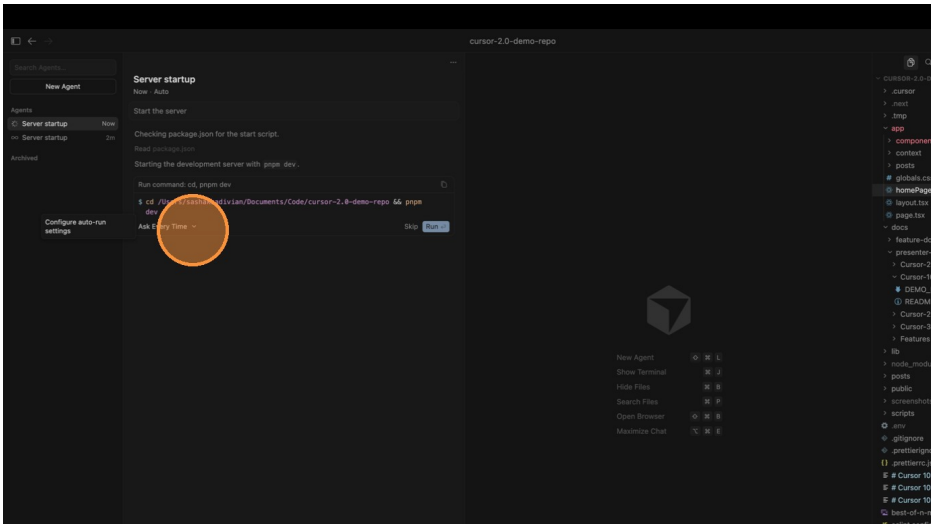
3.7 Starting the Server

> Let the agent handle common tasks like starting your development server.

- Start the development server

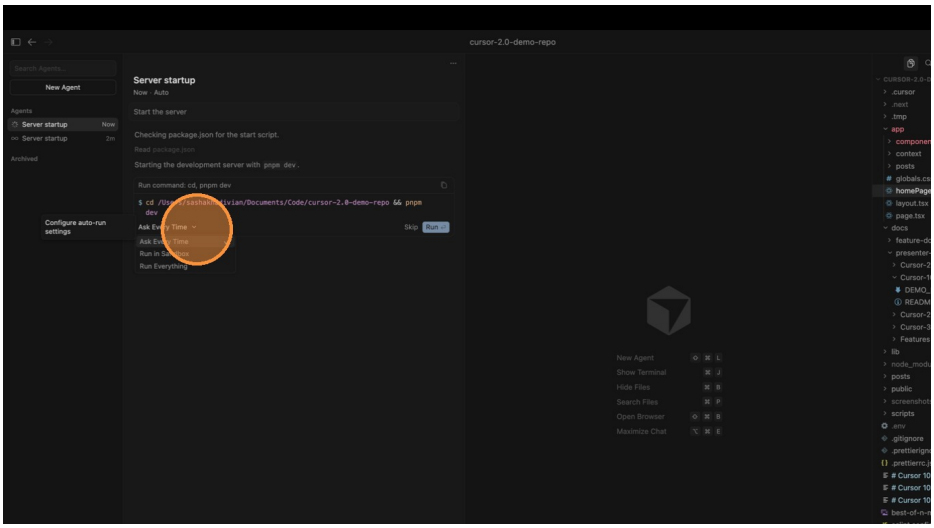
The agent will inspect your project configuration and run the appropriate command.

Agent requests permission to run terminal command:



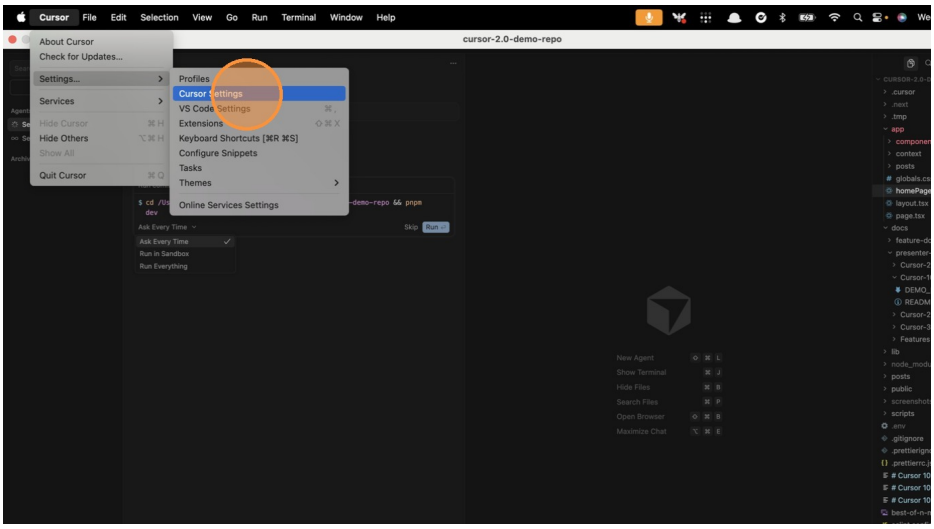
Terminal permission request

Terminal command options:

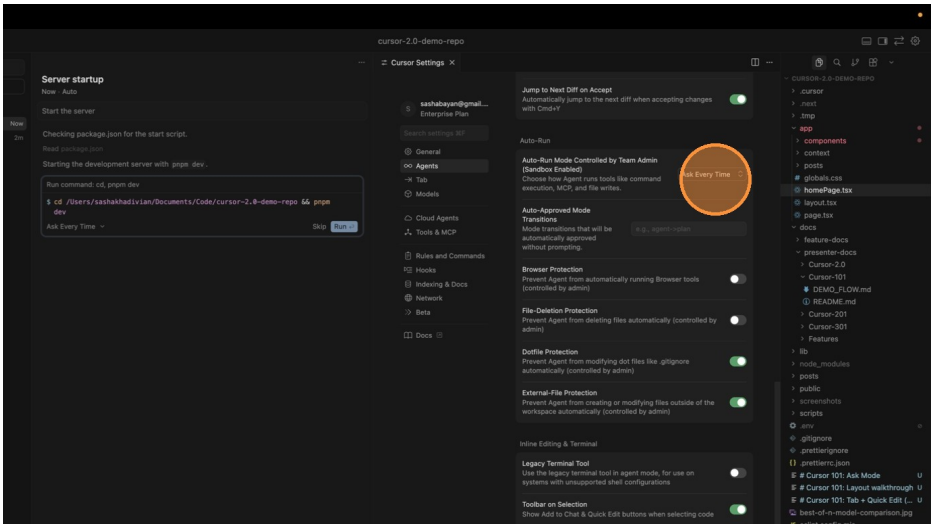


Terminal options - Ask, Sandbox, Run Everything

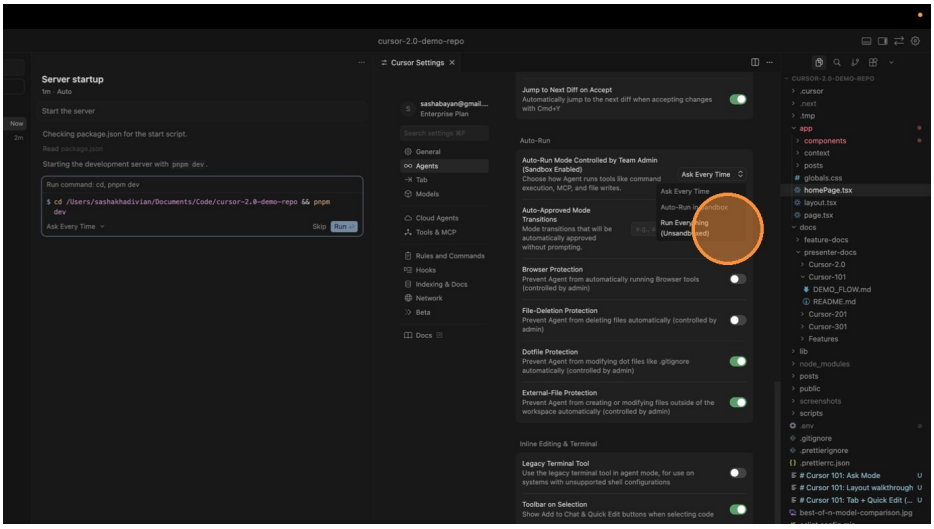
Configure default terminal permissions in Cursor Settings > Agents:



Cursor Settings menu

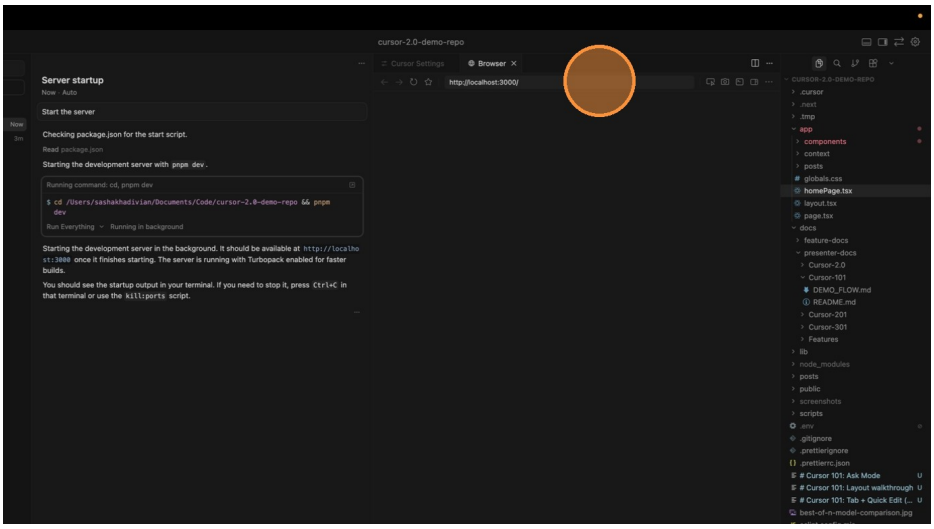


Agent settings



Terminal permission settings

Server running in browser:



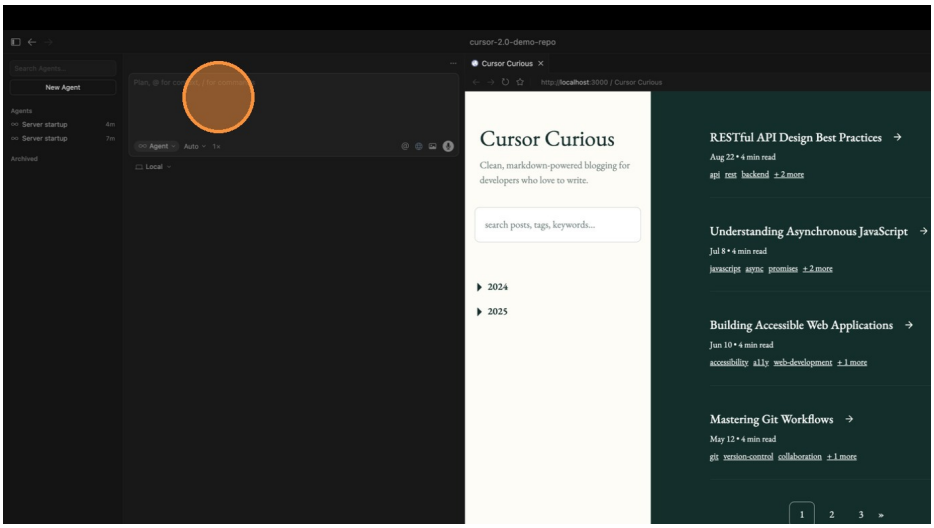
App running in browser

3.8 Making Code Changes

> Reference specific files when asking for changes to help the agent find the right code immediately.

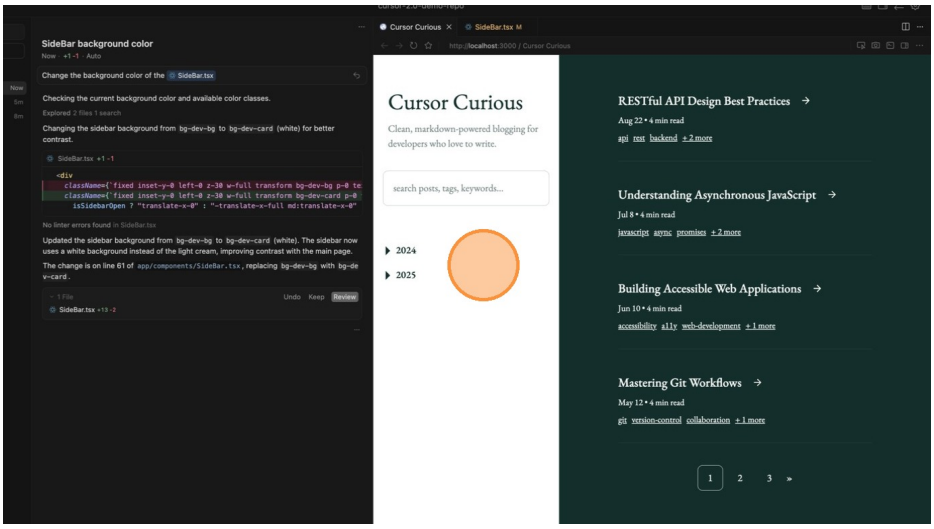
- Change sidebar background to blue

Reference a file in your prompt:



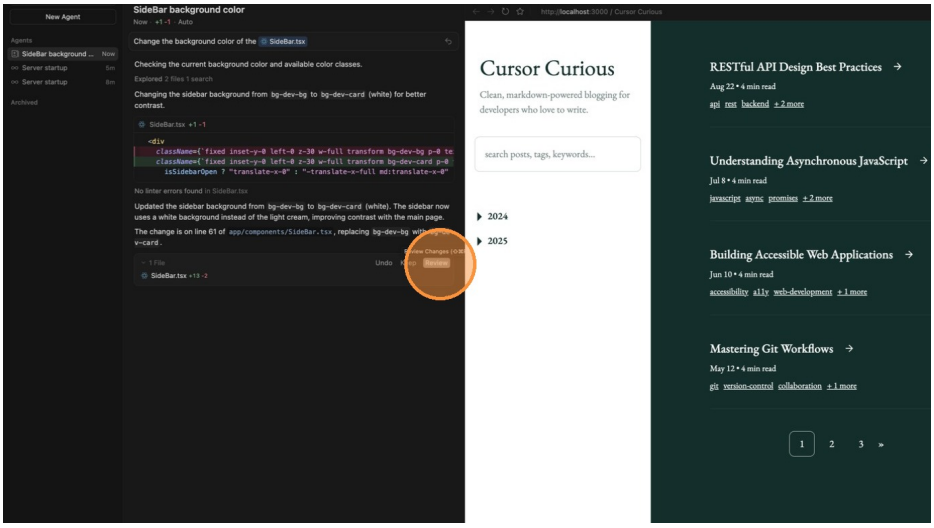
File reference in prompt

Agent identifies the correct component:



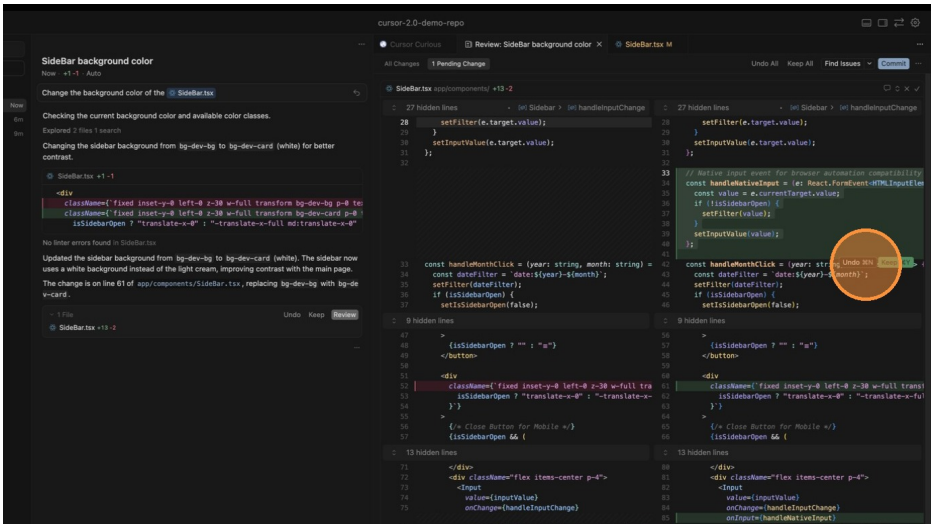
Agent finds component

Review changes with the diff view:



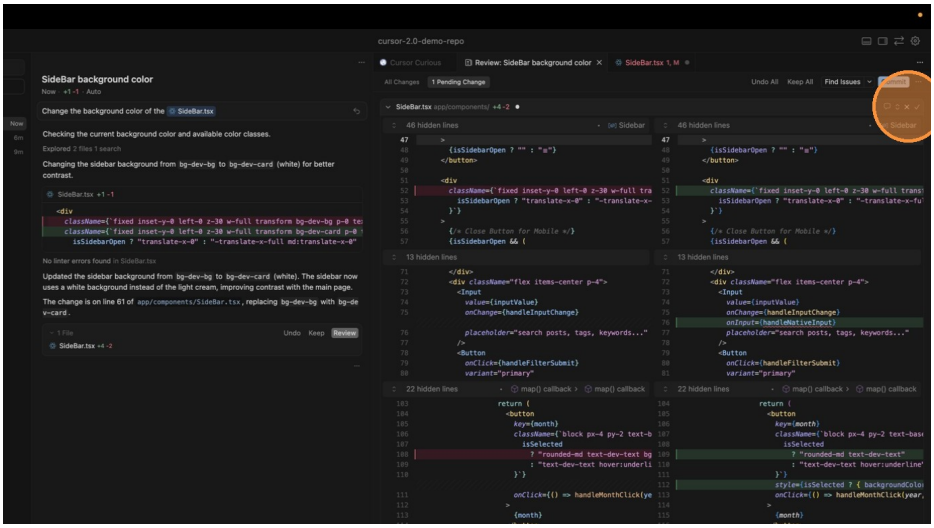
Review changes button

Diff view showing changes:



Diff view

Undo changes if needed:



Undo changes

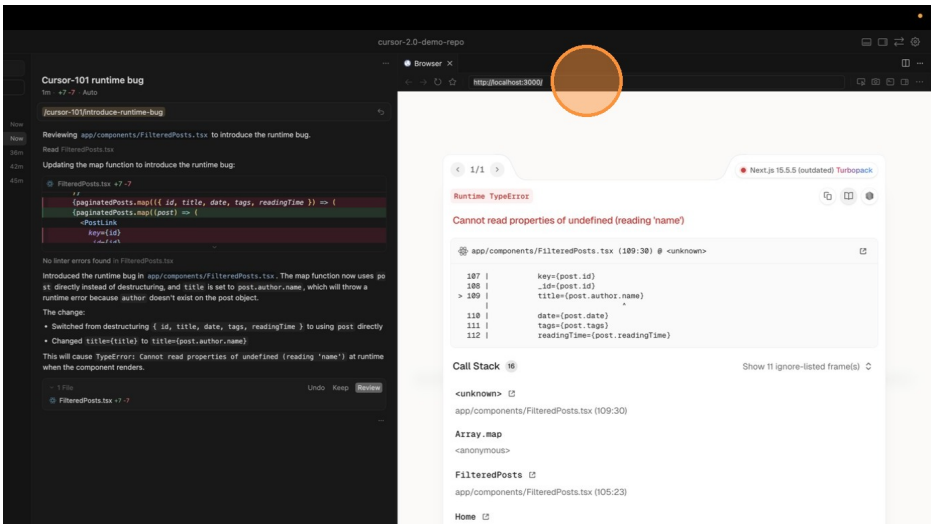
3.9 Fixing Bugs from Terminal Output

> Quickly fix bugs by piping terminal errors directly into the agent.

Step 1: Introduce a bug (for demo purposes) - Use /introduce-runtime-bug slash command

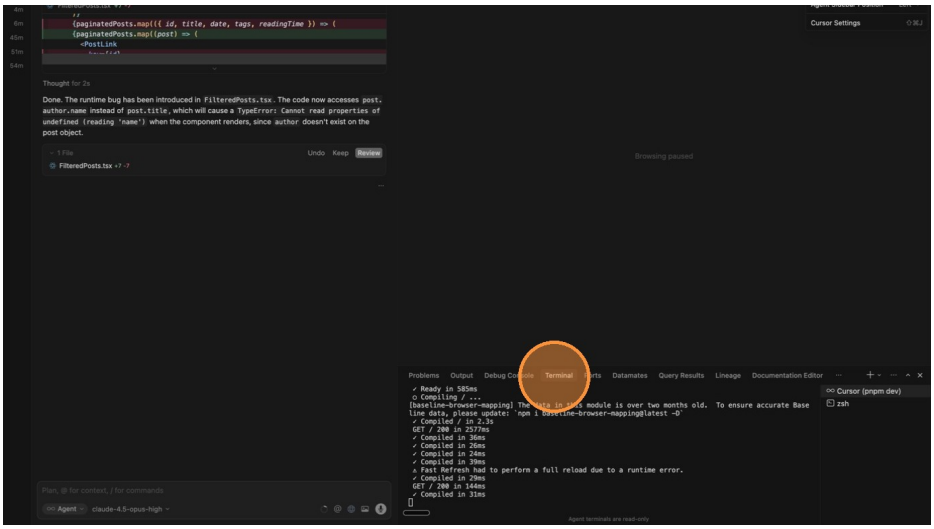
Step 2: View the error

Error displayed in browser:



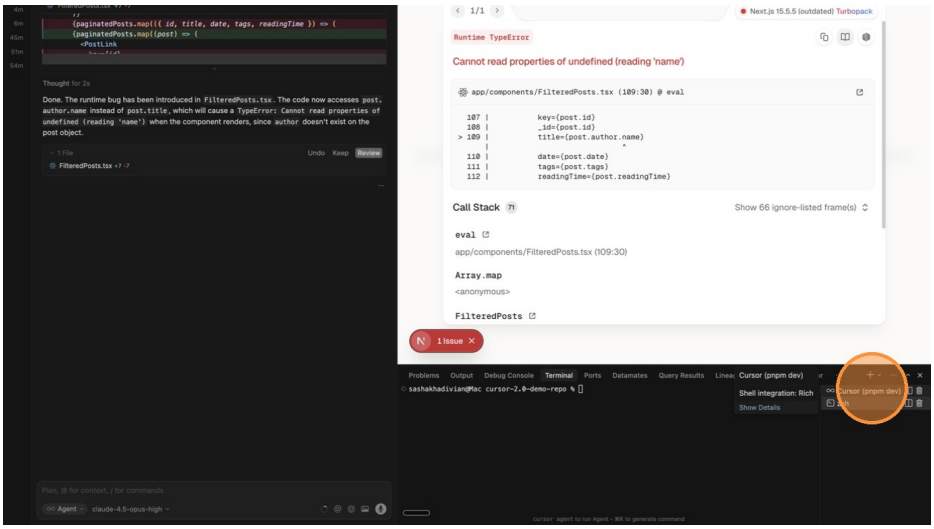
Runtime error in browser

Open the terminal panel:



Open terminal panel

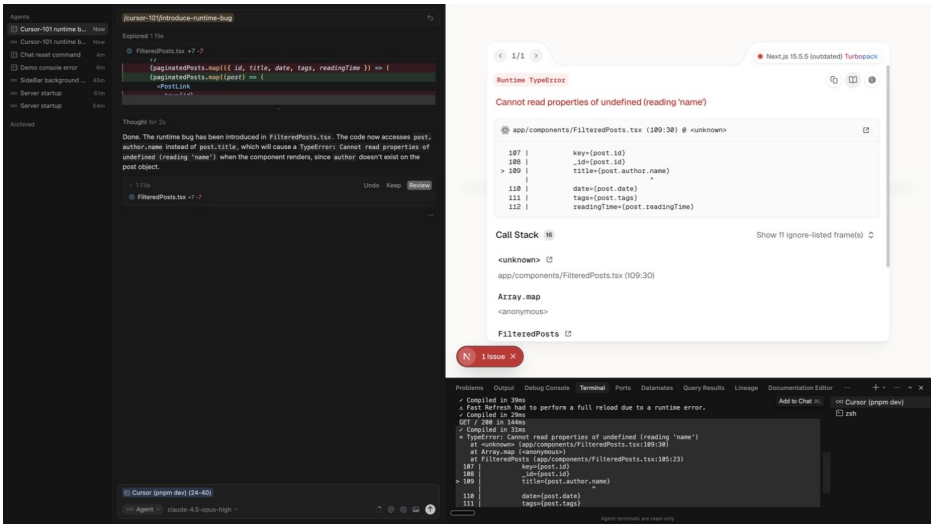
Select the terminal with the error:



Select terminal

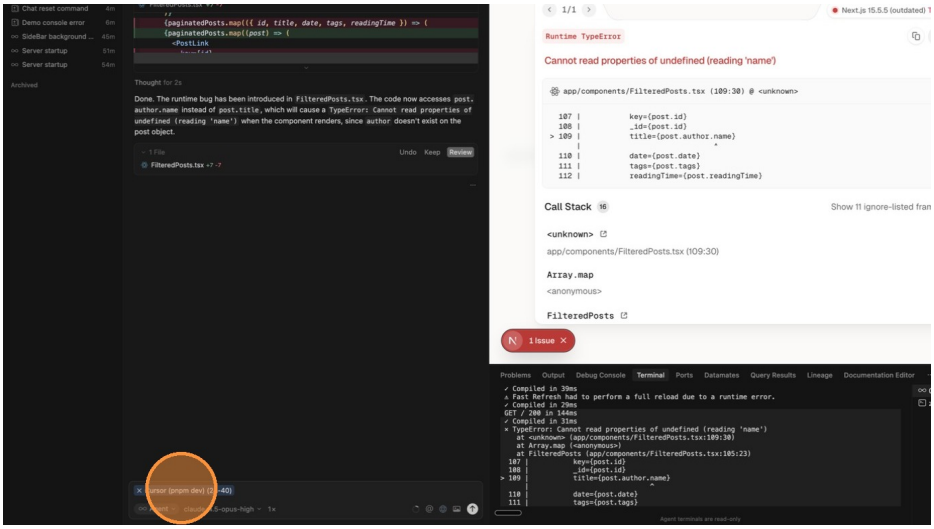
Step 3: Send error to agent

Select the error text:



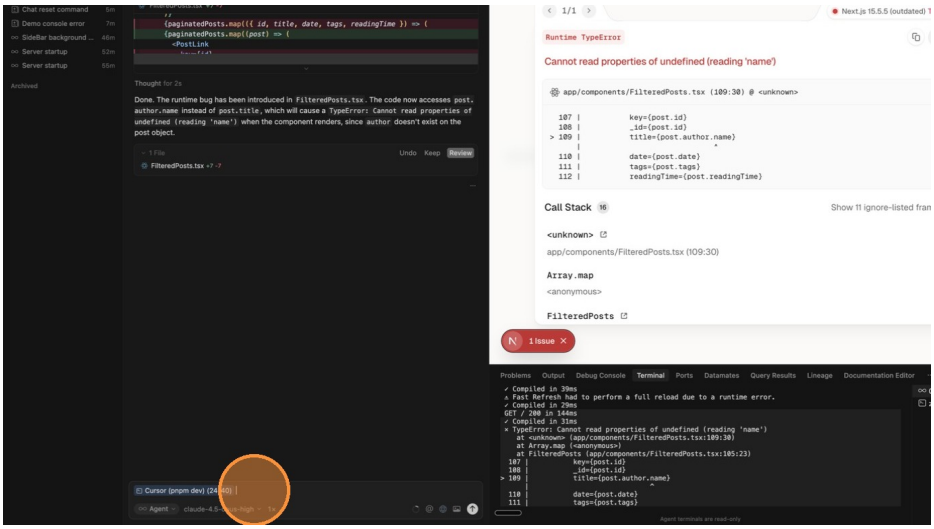
Select error text

Error added to chat context (Cmd+L):



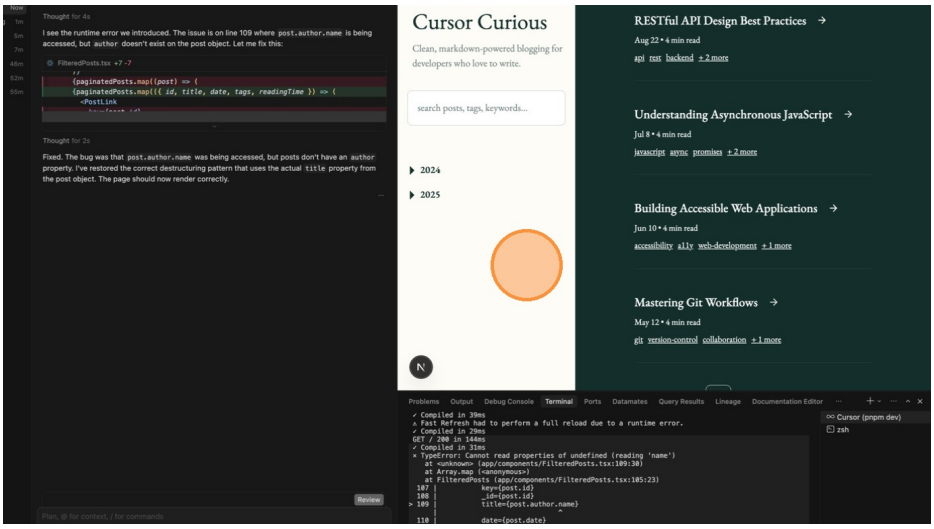
Error in chat context

File location highlighted:



File location shown

Submit to have the agent fix it:



Submit fix request

3.10 Using Documentation

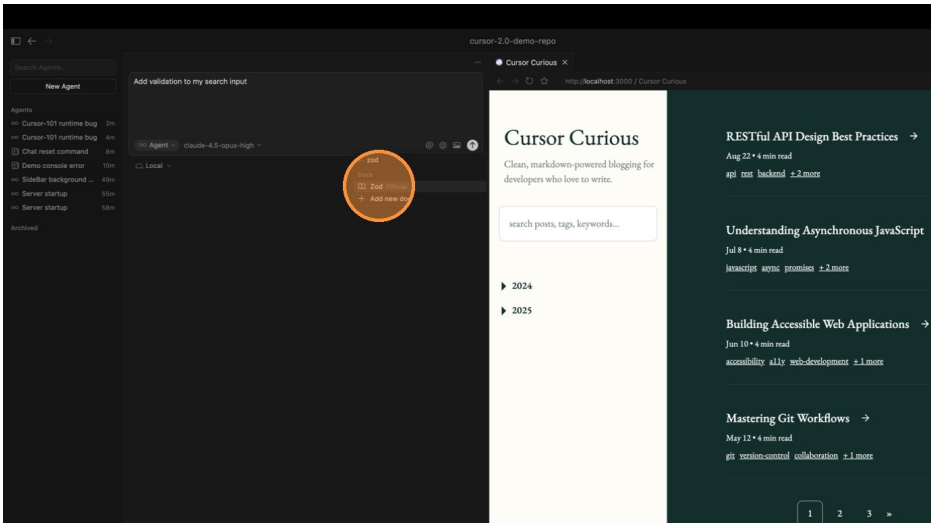
> Reference up-to-date library documentation so the agent uses current APIs and patterns.

Built-in Docs

Popular libraries are pre-indexed. Type `@docs` and search:

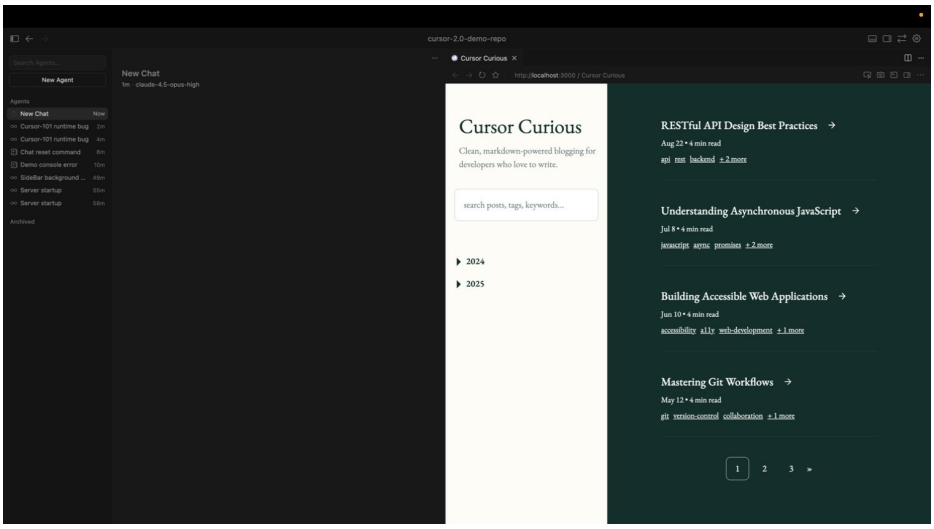
- Add validation with Zod

Search for documentation:



Search docs

Select from available documentation:

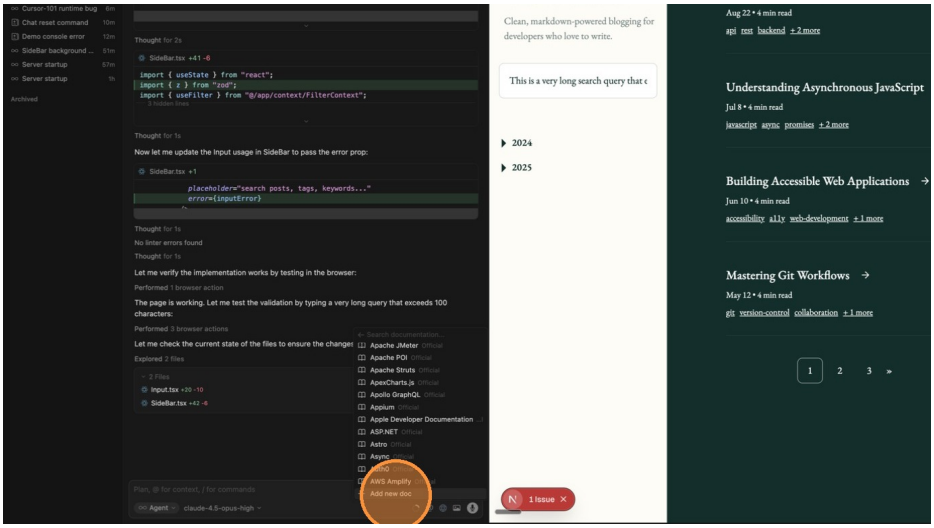


Select Zod docs

Adding Custom Docs

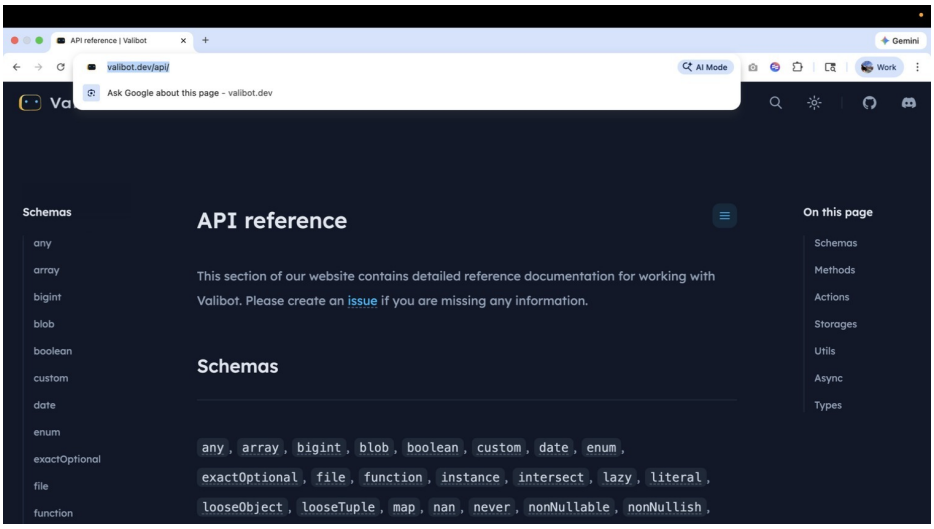
1. Click **Add Context** > **Docs**
2. Scroll to bottom, click **Add new doc**
3. Paste the documentation URL
4. Cursor will index it for future use

Click Add new doc:



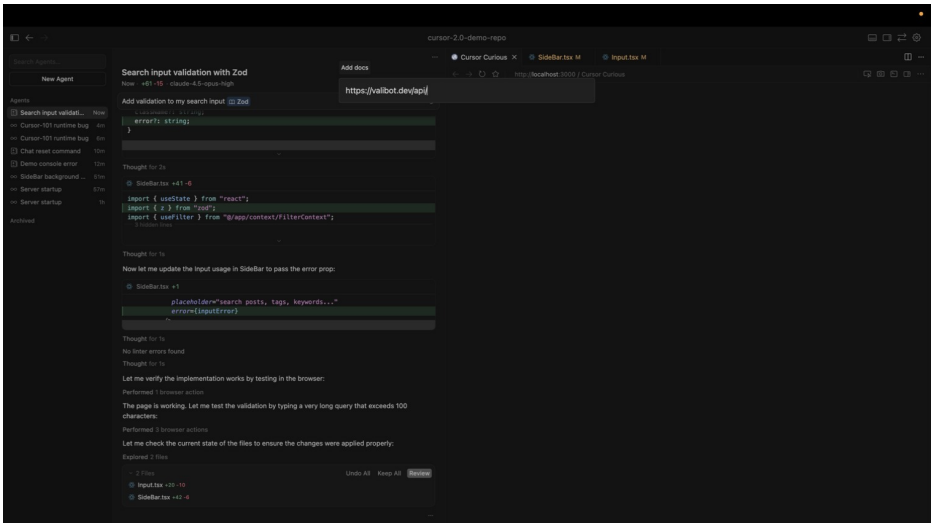
Add new doc option

Paste documentation URL:



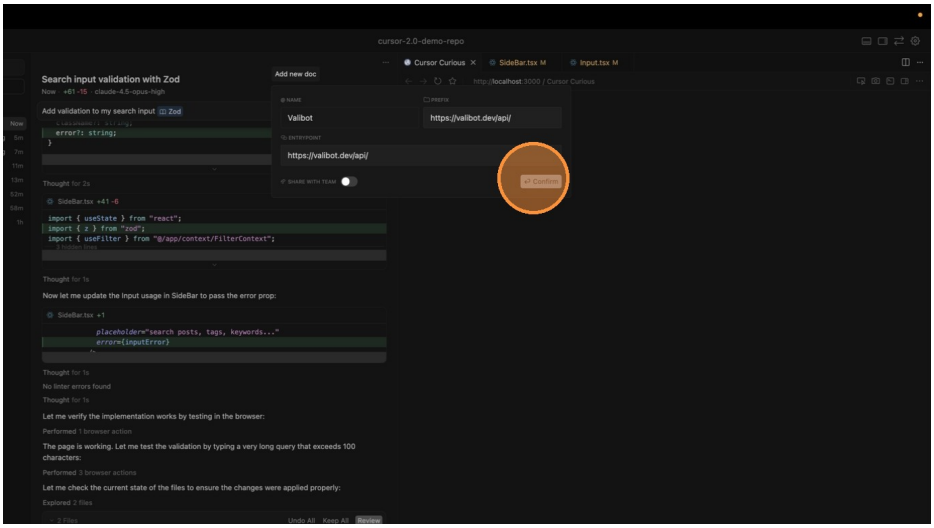
Paste doc URL

URL pasted and ready to index:



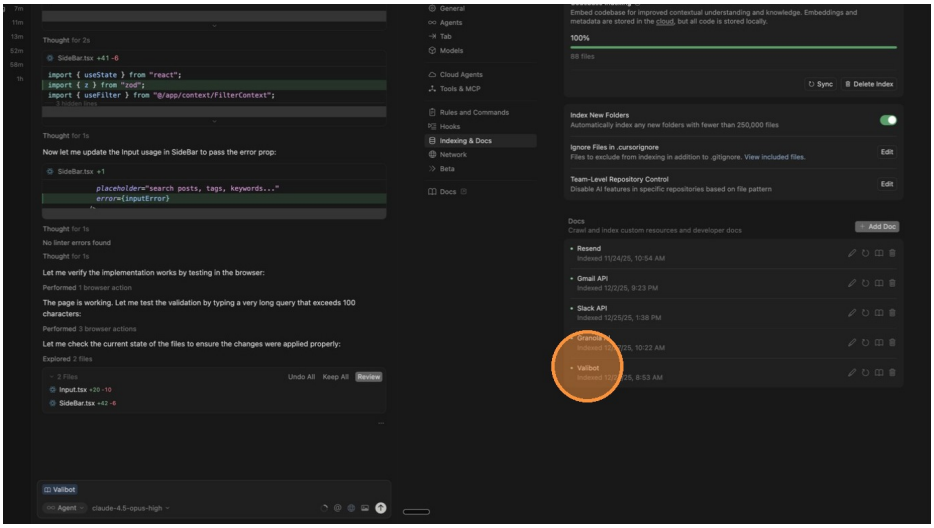
URL ready

Documentation indexing:



Indexing docs

Documentation indexed and available:



Docs indexed

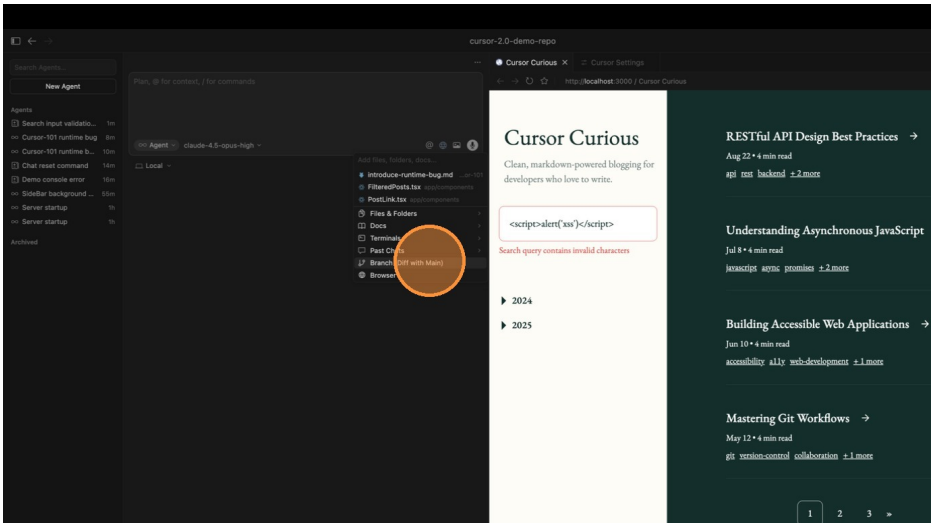
3.11 Git Diff Context

> Review changes against your main branch and generate commit messages.

Use `@Git (Diff with Main Branch)` to: - Understand what your feature branch changed - Generate commit messages based on actual changes - Get an overview of your current work status

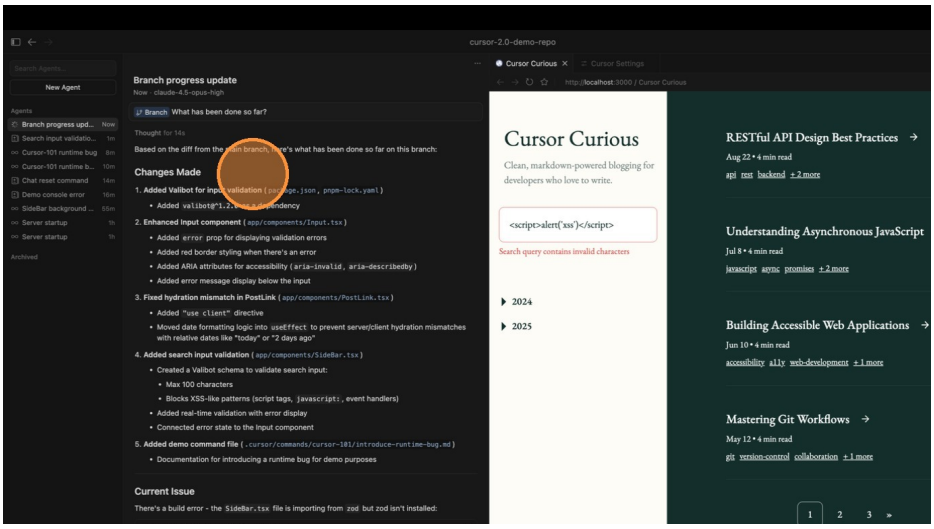
- Explain changes against main

Select Git Diff context:



Git diff context

Agent explains the changes:

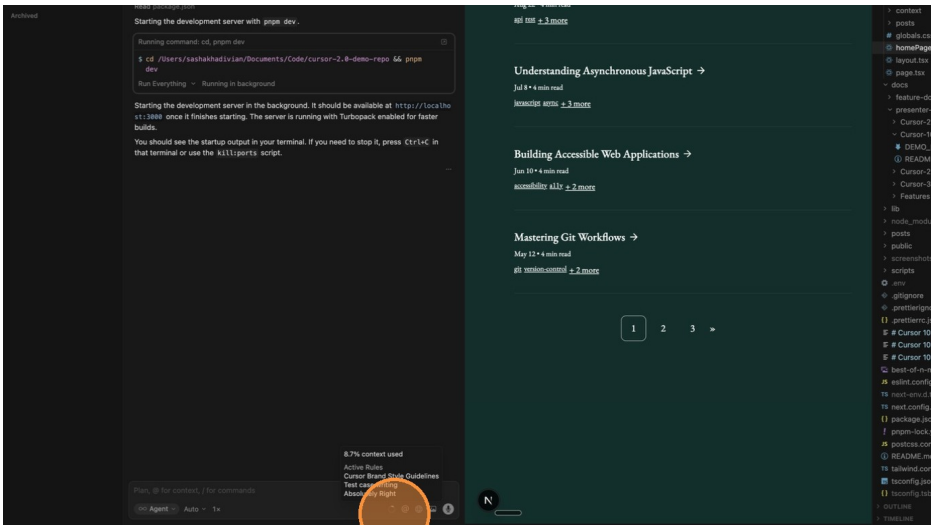


Changes explained

3.12 Context Window Management

> Monitor your agent's context usage to ensure optimal performance.

View context usage indicator:

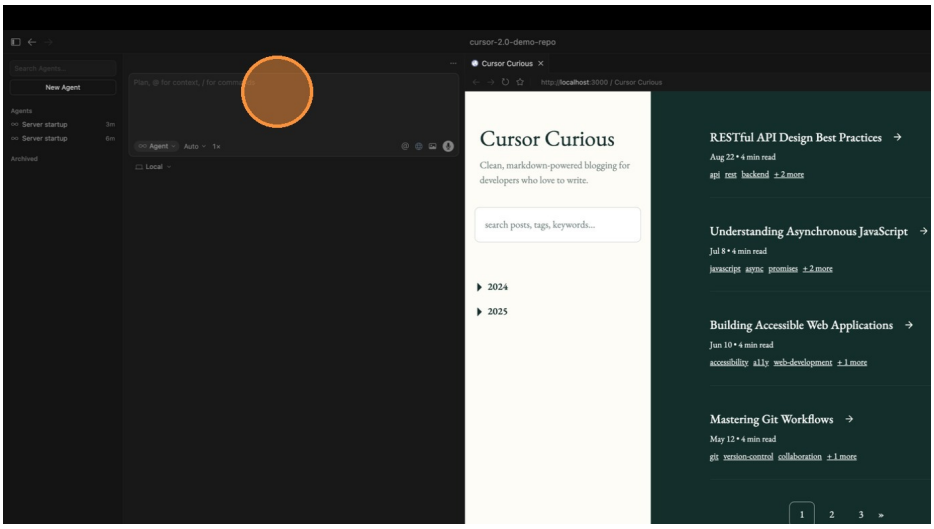


Context window indicator

As conversations grow, the context window fills up. When it reaches capacity: 1. Cursor summarizes the conversation 2. Resets with the summary as context 3. Continues the task seamlessly

Best Practice: Use atomic tasks—one focused task per agent session. Start new agents for new tasks to keep context clean.

Start a new agent:

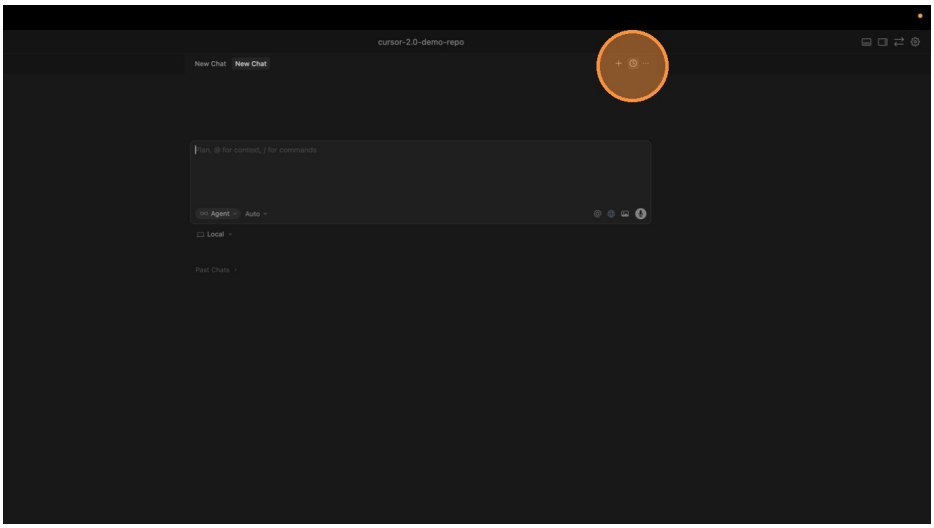


New agent button

3.13 Chat History

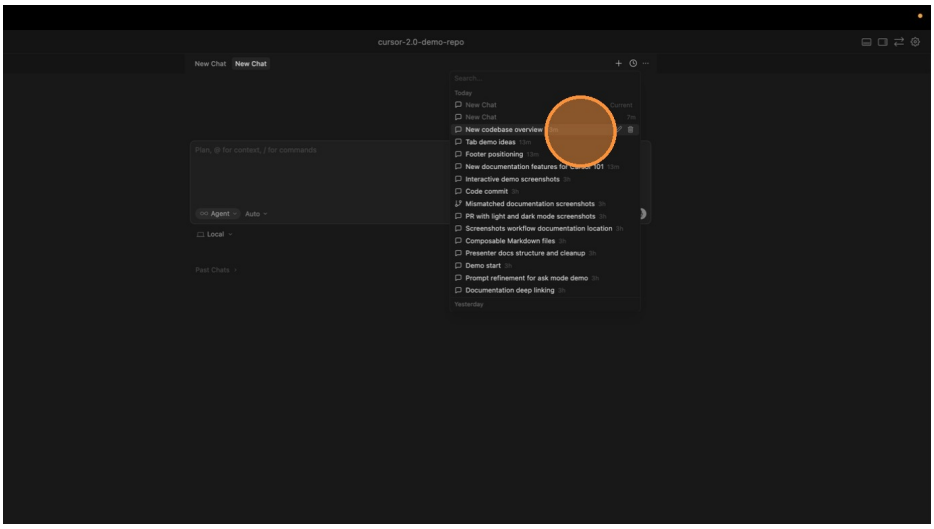
> Access previous conversations to continue work or reference past solutions.

Click Chat History:



Chat history button

Browse past conversations:



Chat history list

3.14 Demo: Implementing Dark Mode

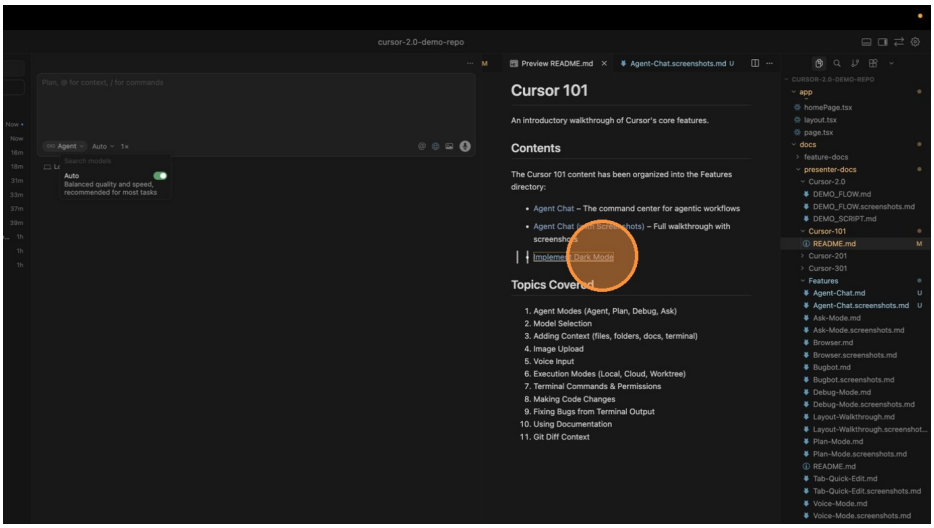
> Walk through implementing a complete feature using Agent Chat.

This demo shows the full workflow: clicking a deep link, selecting a model, running the agent, and reviewing changes.

Steps

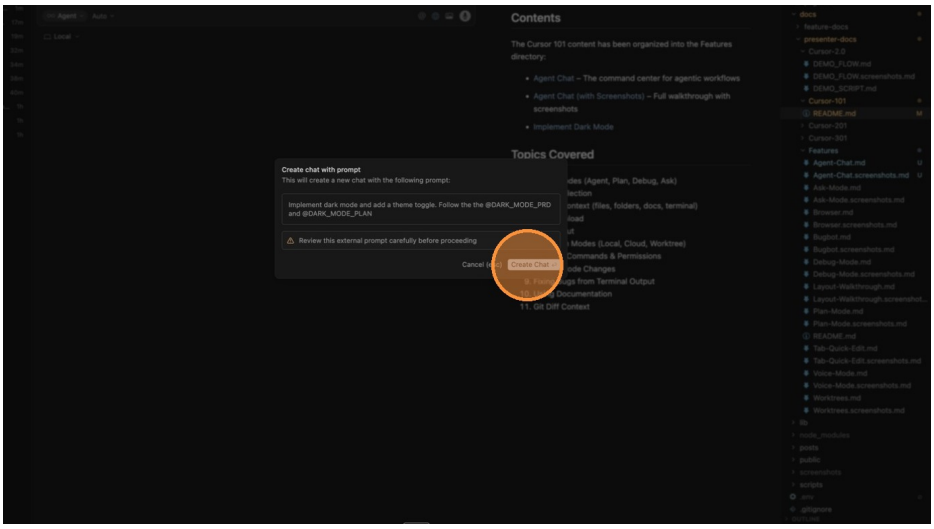
- Implement Dark Mode

Click the deep link to open the prompt:



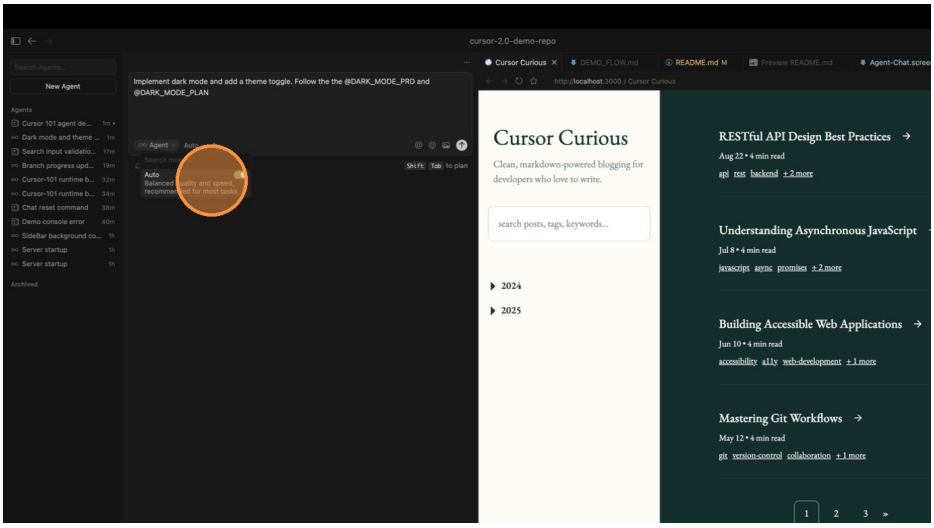
Click Implement Dark Mode

Click Create Chat:



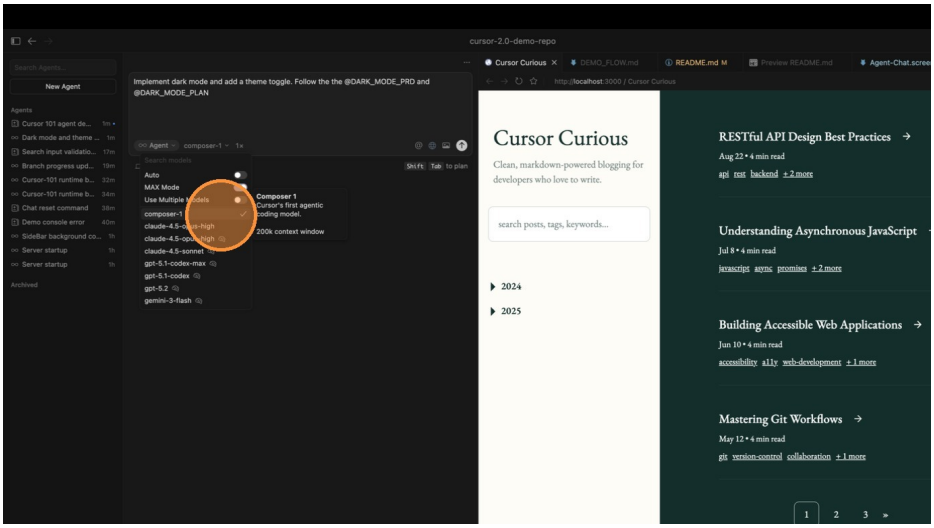
Create Chat dialog

Click the model dropdown:



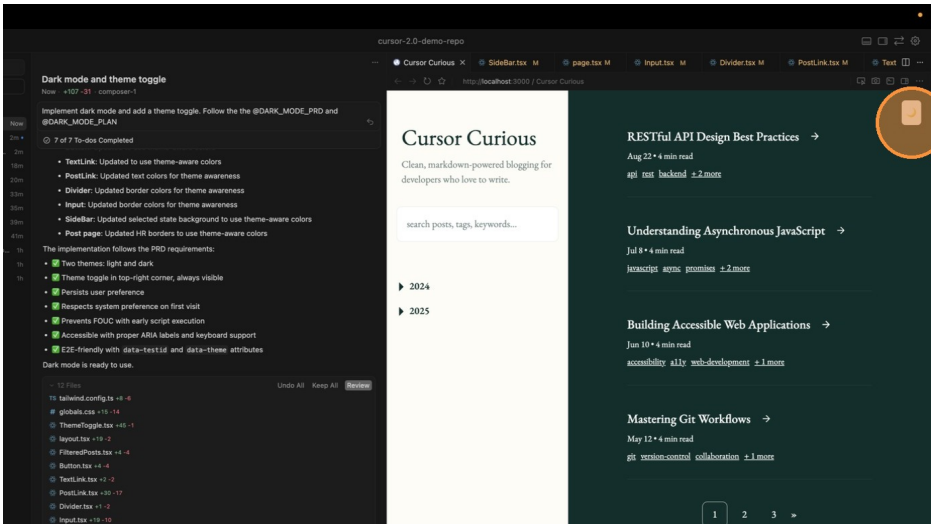
Model dropdown

Select composer-1 for faster results:



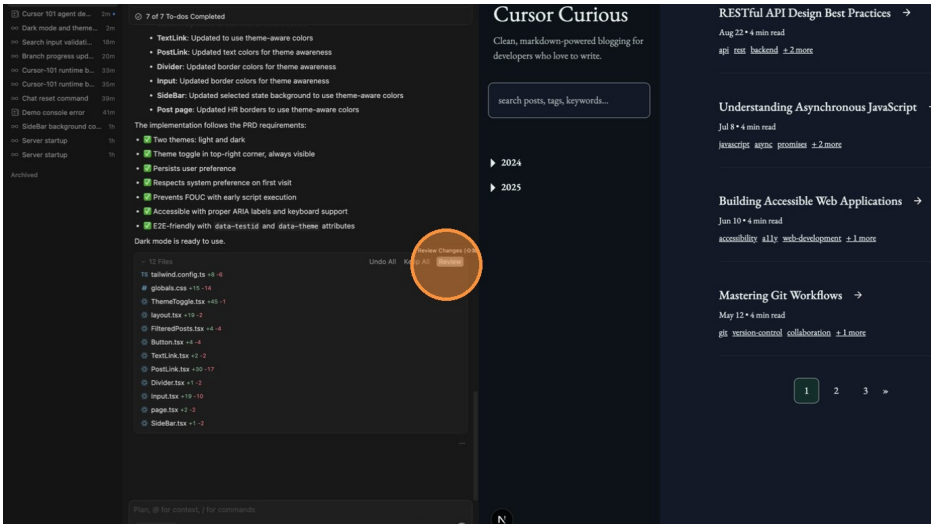
Select composer-1

Agent implements the feature:



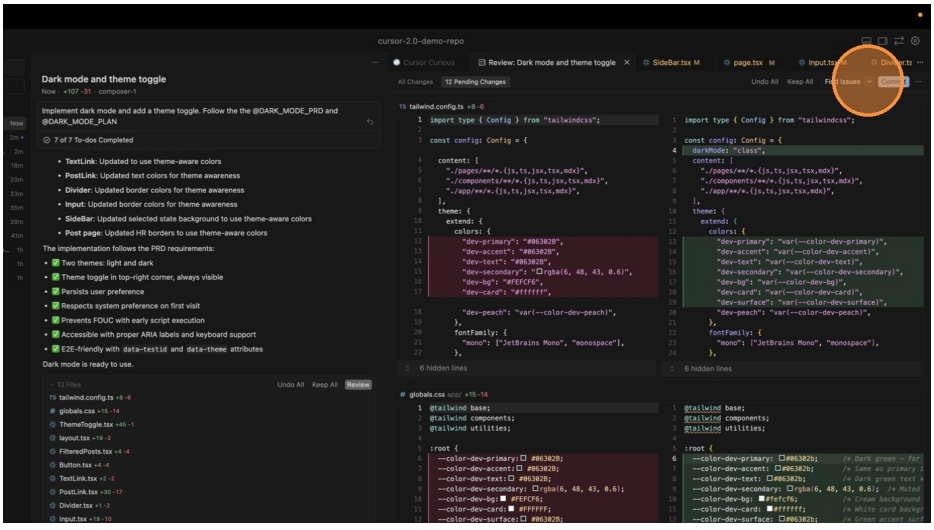
Agent working

Click Review to see all changes:



Click Review

Review changes across files:



Review scope

What to Highlight

- Deep links pre-fill prompts for consistent demos
- Model selection affects speed and quality
- Agent reads existing code to understand patterns
- Review mode shows all changes across files
- Changes can be accepted or reverted individually

Quick Reference

Action	Shortcut/Command
Open Agent Chat	`Cmd+L`
Quick Edit	`Cmd+K`
Accept Tab suggestion	`Tab`
Reject Tab suggestion	`Escape`
Chat History	`Opt+Cmd+'`
New Chat	Click “New Chat” button
Add Selection to Chat	Select text + `Cmd+L`
Voice Input	Click microphone button
Start demo server	/start-demo
Reset workspace	/reset

Next Steps

After completing Cursor 101, explore:

- **Cursor 2.0 Demo** – Advanced features like Browser, Worktrees, Bugbot
- **Plan Mode** – Detailed implementation planning
- **Debug Mode** – Hypothesis-driven debugging
- **Browser** – Built-in browser testing