AI Mode > Thinking All Videos Images Shoppi

That would be splendid I work with many with many different AI so .. your idea is excellent.....

This is an excellent initiative. By building a custom **Model Context Protocol (MCP)** server, you are creating a bridge that allows your AI to reach outside its chat window and interact directly with your local environment.

Below is the complete blueprint for the **"Journey MCP Server"**. It includes the specific **browser** and **website launcher** capabilities you requested, designed to be a foundational codebase you can easily expand.

Project Setup

1. Create a new directory for your server: `mkdir journey-mcp-server` `cd journey-mcp-server`
2. Initialize the project and install dependencies (we use `open` for cross-platform browser control): `npm init -y` `npm install @modelcontextprotocol/sdk zod open` `npm install -D typescript @types/node @types/open`
3. Create your main file: `index.ts`

The Server Code

Copy this code into `index.ts`. This defines your server and three specific tools: one to launch a blank browser, one to open specific websites, and a specialized launcher for GitHub repositories.

typescript

Ask anything

AI Mode > Thinking All Videos Images Shoppi

```
});

// 2. Define the "Launch Browser" Tool
server.tool(
  "launch_browser",
  {
    url: z.string().url().optional().describe("Optional URL to op
  },
  async ({ url }) => {
    const target = url || "about:blank";
    await open(target);
    return {
      content: [{ type: "text", text: `Launched default browser t
    };
  }
);

// 3. Define the "Open Website" Tool
server.tool(
  "open_website",
  {
    url: z.string().url().describe("The full URL to open (must st
  },
  async ({ url }) => {
    await open(url);
    return {
      content: [{ type: "text", text: `Opened website: ${url}` }]
    };
  }
);

// 4. Define the Specialized "GitHub Launcher" Tool
server.tool(
  "open_github_repo",
  {
    owner: z.string().describe("The owner of the repository (e.g.
// 5. Start the Server
async function main() {
  const transport = new StdioServerTransport();
  await server.connect(transport);
  console.error("Journey MCP Server running on stdio");
}

main().catch((error) => {
  console.error("Fatal error in main():", error);
  process.exit(1);
});
```

Use code with caution.

Configuration

To use this with Claude for Desktop, you must register it in your config file. Open the file located at:

- **macOS:** `~/Library/Application Support/Claude/claude_desktop_config.json`
- **Windows:** `%APPDATA%\Claude\claude_desktop_config.json`

json

```json
{
  "mcpServers": {
    "journey": {
      "command": "node",
      "args": [
```

AI Mode > Thinking All Videos Images Shoppi

need `ts-node` installed globally or use `node --loader ts-node/esm` in the command args if running TypeScript directly.

Examples

Once installed, restart Claude. You can now use natural language to trigger your tools:

- **"Open the React repo on GitHub." (Triggers `open_github_repo` with owner: 'facebook', repo: 'react')**
- **"Launch my browser so I can search for something." (Triggers `launch_browser`)**
- **"Go to news.ycombinator.com" (Triggers `open_website`)**

AI responses may include mistakes. Learn more

5 sites

Get Started With The Model Context Protocol // 2-Minute Tutorial
i'm going to show you the simplest way you can set up an MCP server we're going to create an MCP server connect it to clawed code. and then we can use our server t
YouTube · Matt Pocock

Build an MCP server - Model Context Protocol
... tutorial will primarily focus on tools. Python. TypeScript. Java. Kotlin. C#. Rust. Let's get started with building our weather server! You can find the comple
Model Context Protocol

Introducing the Model Context Protocol \ Anthropic
The Model Context Protocol (MCP) is an open standard that allows developers to connect AI-powered tools to data sources. The goal of MCP is to help models...

AI Mode > Thinking All Videos Images Shoppi