# Claude Chat Export

Topic: Claude Projects, Claude Code, and Multi-Folder Development

---

**Jim**

How can I create a Claude project that includes several sibling folders from local file system

---

**Claude**

There's no built-in way to sync local folders directly into a Claude Project — Claude.ai's Projects don't have a file system integration that can watch or pull from local paths automatically.

Here's what you can do:

Manual upload — The simplest approach. In a Project, you can upload files directly via the Project's knowledge section. You'd manually upload files from each sibling folder. The limitation is you have to re-upload when files change.

Concatenate/bundle content programmatically — Write a script that walks your sibling folders, concatenates relevant files (with path headers), and produces a single text or markdown file you upload to the Project.

Use the API instead of claude.ai — If you're building a tool, you can pass file contents directly in the system prompt or user messages via the API. This gives you full programmatic control — your app reads the folders at runtime and injects the content. This is probably the best fit for your dev workflow given your automation background.

Claude Code — Since you've been exploring it, claude in the terminal can read your local files directly during a session using its file tools. It doesn't persist as a "Project" but it can naturally work across sibling folders in a single session without any manual bundling.

The API + bundling script approach is probably the most practical if you want persistent project context that reflects your local repo structure.

---

**Jim**

Can I create an agent that supports software development for all folders in a specified list

---

**Claude**

Yes, absolutely — and this fits well with what you've already been building (your Software Development Agent with the Anthropic API). Here's the conceptual breakdown:

Core Architecture

Your agent gets a list of folders at startup, then can read/write/analyze across all of them in a unified session. The key pieces:

1. Multi-repo context builder — Build a map of all relevant files across folders using rglob with filtered extensions (.py, .cpp, .rs, .cs, .h, .md, .toml, .json).

2. Tool definitions for the agent — Give Claude tools that operate across all registered folders: read_file(path), write_file(path, content), search_code(pattern, folder=None), list_files(folder=None), run_command(cmd, cwd).

3. Agent loop — Use the Anthropic client with a system prompt listing all folders, then loop on tool_use stop_reason, executing tools and appending results until end_turn.

Practical Enhancements

Selective context loading — Don't dump all files into every prompt. Instead, load files lazily via read_file tool calls so the agent only pulls what it needs.

Cross-repo awareness — In your system prompt, describe the relationship between folders so the agent understands the architecture.

Config file — Drive the folder list from a JSON/TOML config rather than hardcoding it, with path and description for each folder.

Claude Code Alternative — If you want this without building it yourself, claude in the terminal can already do this — just start a session from a parent directory that contains all your sibling folders.

---

**Jim**

How do I upload a file to a Claude project

---

**Claude**

In Claude.ai:

1. Open or create a Project 2. In the left sidebar under the project name, look for "Project knowledge" 3. Click "Add content" (or the + button) 4. Choose "Upload files" — supports text files, PDFs, code files, markdown, etc.

Files you upload to Project knowledge are available as persistent context in every conversation within that project, so you don't have to re-paste them each session.

Limits to be aware of: There's a per-project storage limit on the knowledge base. Very large files may get truncated. Binary files (images, executables) have limited support — stick to text-based formats.

Programmatically via API — there's currently no API endpoint to upload files directly into a Claude.ai Project's knowledge base. If you're working via the API, you inject file contents directly in the system prompt or user messages at request time instead.

---

**Jim**

Are Claude projects just for claude.ai or are they used in Claude Code as well

---

**Claude**

Claude Projects are a claude.ai feature only — they're the organizational containers in the web/app interface with persistent knowledge, custom instructions, and conversation history.

Claude Code has a different but related concept: it uses a CLAUDE.md file in your project root (or ~/.claude/CLAUDE.md for global settings) as its persistent context. This is the Claude Code equivalent of Project knowledge — you put repo structure, coding conventions, build commands, architecture notes, etc. in there and Claude Code reads it automatically on startup. You can also have CLAUDE.md files in subdirectories for folder-specific context.

The mental mapping: - Claude.ai Project knowledge → CLAUDE.md file(s) - Claude.ai Custom instructions → CLAUDE.md + /configure - Claude.ai Conversation history → Session-based (not persisted by default) - Claude.ai File uploads → Direct file system access

For your multi-folder agent use case, you could put a CLAUDE.md in each sibling folder describing that folder's purpose, and Claude Code will pick them up as it navigates between them.

---

**Jim**

I want to use Claude Code in VS Code for working with projects. Do I need to ask code to create a claude.md file or will that happen some other way.

---

**Claude**

Claude Code can create the CLAUDE.md file for you — you don't have to write it manually. A few ways it happens:

Ask Claude Code to generate it — Just tell it: "Create a CLAUDE.md file for this project" and it will analyze your codebase and generate an appropriate one with repo structure, build commands, conventions, etc.

The /init command — Run /init in the Claude Code chat and it will scan your project and create a CLAUDE.md automatically. This is the intended "getting started" workflow.

Write it yourself — You can also just create CLAUDE.md manually in your project root and add whatever context you want Claude Code to always know about.

For your VS Code + multi-folder setup, the typical workflow would be: 1. Open your workspace in VS Code with your sibling folders 2. Start Claude Code in the terminal 3. Run /init in each folder (or the workspace root) to generate CLAUDE.md files 4. Optionally create a root-level CLAUDE.md that describes how the folders relate to each other

Claude Code in VS Code is currently available as an extension in beta. Since the tooling has been evolving quickly, it's worth checking the current installation state for the latest docs on the VS Code integration.

---