

Building with an AI coding agent

1 What these tools are

You have probably used Claude or ChatGPT in a chat window: you ask, it answers. **Claude Code** and **Codex** are different. They work *inside a folder on your computer*. They can create files, edit them, run the program, see the error, and fix it, all on their own. You describe what you want; the agent does the typing.

Your job (the team)

- Decide what to build and for whom
- Approve the plan before it builds
- Try every change yourself
- Check that everything is right
- Save your work and keep the session log

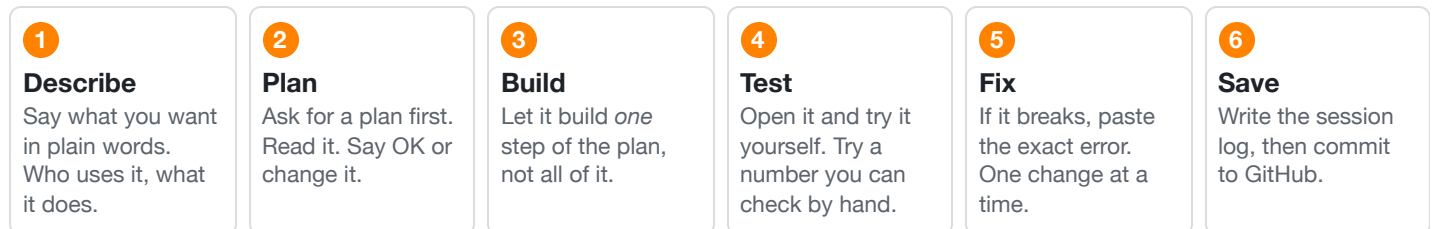
The agent's job

- Propose a plan when you ask
- Write and edit the code files
- Run the code and read the errors
- Explain what it did in plain English
- Write the save message for GitHub

You do not need to read code to use these tools. You do need to **test what they build** and **be able to explain it**; judges will ask.

2 The build loop

Every working session follows the same six steps. Small steps are the secret: one feature at a time, tested before the next.



Then go back to step 1 for the next feature. Think and plan in chat; build in the agent.

3 Defining Often Used Terms

Folder / project	The one folder on your computer that holds your app. The agent works only in it.
File	One piece of the app, such as <code>index.html</code> (the page) or <code>app.js</code> (the logic).
GitHub	A website that stores your team's code online, keeps every past version, and lets teammates share it.
Repository (repo)	Your project's home on GitHub, one per team. Each teammate keeps a copy (a "clone") on their own computer, pushes their changes up to the repo, and pulls teammates' changes down.
Commit	A saved snapshot of your work, with a short note on what changed. You can always go back to one.
Pull / push	Pull = get your teammates' latest changes. Push = send your commits up to GitHub.
Deploy	Put the app on the internet with a link anyone can open (Vercel does this for free).
API key	A password that lets your app use a data service (FRED, CoinGecko). Treat it like a password.
Terminal	A text window for typing commands. Optional: the desktop Claude/ChatGPT apps let you skip it.

More prompts: fintech.ekkfinance.com/resources.html. Start with "How the workflow works," "Put your work on GitHub," "Ship it with Vercel," and the "Make your test data" track.

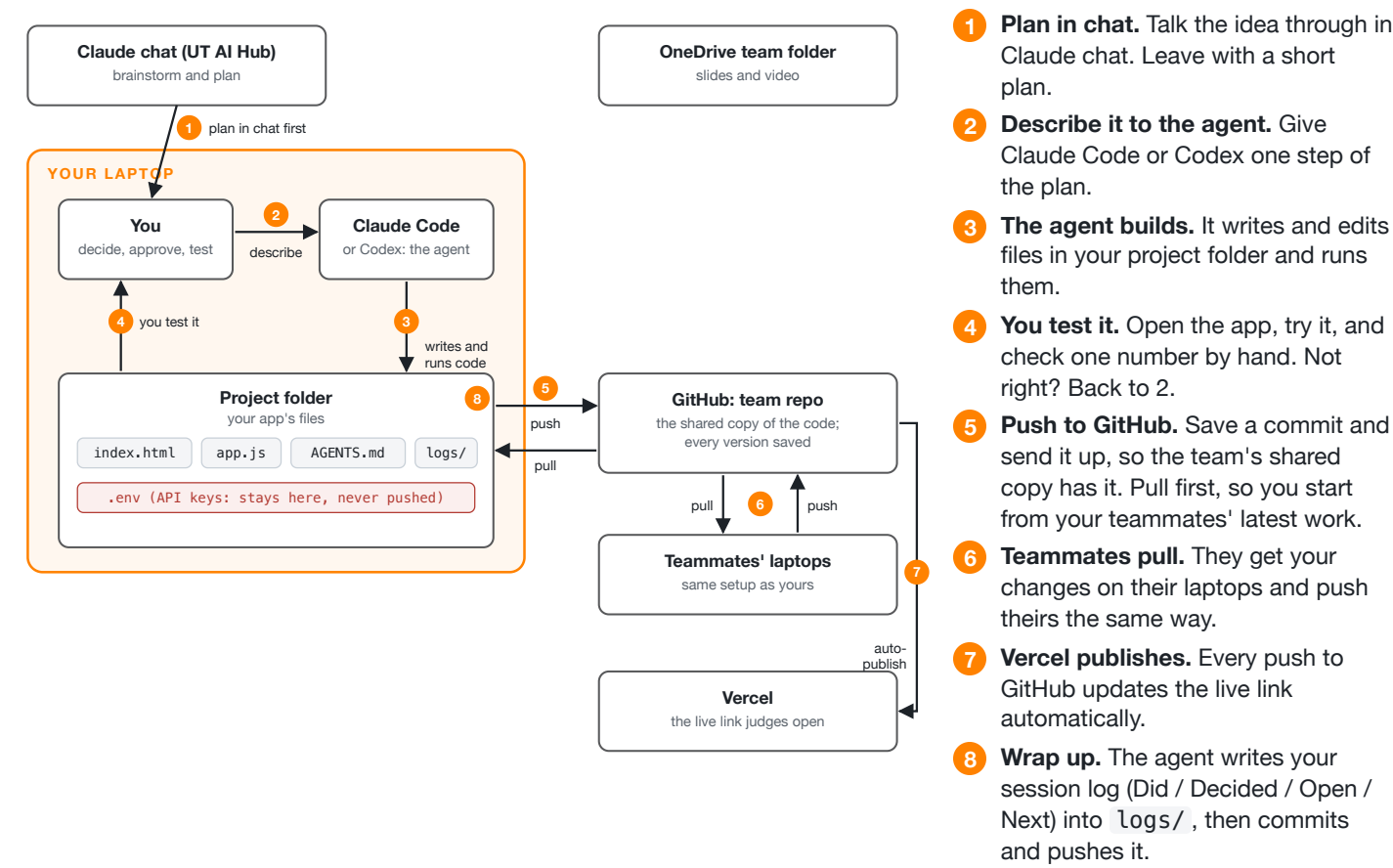
4 Where your team's things live

WHAT	WHERE	RULE
Code (the app itself)	A GitHub repo, one per team	Pull before you start; commit and push when you stop. Never keep the code folder inside OneDrive: syncing breaks it.
Session logs	A <code>logs/</code> folder in the repo	One short file per person per session: Did / Decided / Open / Next. The agent reads the latest ones when you start.
Slides and video	Your team's OneDrive or Google Drive folder	For the files that are not code.
Thinking and planning	Claude chat (UT AI Hub) or ChatGPT	Brainstorm, compare ideas, draft the plan.
Building	Claude Code or Codex	Pick one tool as a team so your setup matches.
Team instructions for the agent	A file called <code>AGENTS.md</code> in the repo	What the app is, how to run it, your rules. Every teammate's agent reads it first.
Live app	Vercel (free)	Connect it to GitHub once; every push goes live.

Two people, one file, same time = trouble. Split the work by feature, and say in your group chat who is working on what.

5 How the pieces connect

Follow the numbers through one work session. The orange box is your own computer; everything outside it is shared with your team.



Pull before you start, push when you stop. Then nobody overwrites anybody.

Code and logs in GitHub; slides and video in OneDrive or Drive. Never put the code folder in OneDrive.

Keys stay on your laptop. The `.env` file never goes to GitHub or into a chat.

6 Set up step by step

Do this once, in the first build session. The team lead does the steps marked **Lead**; everyone does the steps marked **Everyone**. Plan on 45 minutes. Do the parts in order: A, B, C, D.

A Your AI tool: account, plan, app

Pick **one** tool as a team: **Claude Code** (from Anthropic, maker of Claude) or **Codex** (from OpenAI, maker of ChatGPT). The steps below are for Claude; Codex works the same way.

EVERYONE

1. **Make an account.** Go to **claude.ai** → Sign up, using your UT email. (Codex: **chatgpt.com**.) The free UT AI Hub is for chat only; the build tool needs a paid personal plan.
2. **Pick a plan.** In **claude.ai**: your name (bottom left) → **Settings** → **Billing**, or the **Upgrade** button. Organizers suggest **Max at \$100 a month, for two months**, paid from your \$200 AI scholarship. Organizers will explain how to claim it.
3. **Protect your card.** Plans renew every month until you cancel. Put a reminder in your calendar to **cancel during month two** (you keep access to the end of the month). Leave any **extra usage** option off, so you can never be charged past \$200.
4. **Download the app.** Go to **claude.ai/download**, pick Mac or Windows, install it, and sign in. (Codex: install the Codex app from OpenAI and sign in with your ChatGPT account.)
5. **Find the Code tab.** Open the Claude app and switch from **Chat** to **Code**. Chat is for talking; Code is the agent that works in a folder. It will ask you to choose a project folder: you will make that folder in part B, step 7.
6. **Allow it to work.** The first time, the app asks permission to read and edit files in that folder. Say yes for your project folder only.

Mentors at the session can help with any step. If a payment or sign-in step fails, skip ahead to B and come back.

B GitHub: your team's shared code

EVERYONE

1. Sign up at **github.com** with your UT email. Optional: apply for the free **GitHub Student Developer Pack** (education.github.com).
2. Install **GitHub Desktop** (desktop.github.com). It gives you buttons for pull and push, so you never need the terminal.
3. Make one folder for code that is **not** inside OneDrive: **C:\code** on Windows, or a **code** folder in your home folder on a Mac.

LEAD

4. On **github.com**: **+** → **New repository**. Name it after your project, choose **Private**, tick **Add a README**, and under ".gitignore" pick **Node** (it already blocks **.env**).
5. In the repo: **Settings** → **Collaborators** → **Add people**, and add each teammate's GitHub username.

EVERYONE

6. Accept the invitation (check your email).
7. In GitHub Desktop: **File** → **Clone repository**, pick your team's repo, and set the local path to the code folder from step 3.
8. In the Claude app's **Code** tab (or Codex), choose this cloned folder as your project folder. It is the "project folder" on page 2.

Windows check: GitHub Desktop suggests **Documents\GitHub**. On many Windows laptops, **Documents** is synced to OneDrive. Change the path to **C:\code**.

C OneDrive: slides and video

LEAD

1. Sign in at **onedrive.com** with your UT account. **New** → **Folder**, named after the team.
2. **Share** → enter each teammate's UT email → set to **Can edit** → Send.
3. Inside it, create folders for slides and video. (Session logs live in the repo, not here.)

EVERYONE

4. Open the link from the email. Click **Add shortcut to My files**, so the folder shows up in your own OneDrive.
5. Keep documents here. Keep code out of here.

Prefer Google Drive? A shared Google Drive folder works the same way: create it, share it with each teammate as an editor, and keep slides and video there. Pick one as a team, not both.

D AGENTS.md: instructions for the agent

Agents need context about your project: what it is, how to run it, and your rules. They get it by reading a **Markdown file**, a plain text file with simple formatting whose name ends in `.md`. Claude Code looks for `CLAUDE.md`; Codex (ChatGPT) looks for `AGENTS.md`.

Put it at the **top level of the repo**, next to the README, not in a subfolder. Every teammate gets it when they pull, so every agent starts with the same background.

your-project/	← the cloned repo
AGENTS.md	← instructions (Codex reads it)
CLAUDE.md	← one line: @AGENTS.md
logs/	← session logs, one file each
README.md	
.gitignore	← lists .env
.env	← keys: your laptop only
index.html ...	← the app itself

Claude Code reads CLAUDE.md; its one line pulls in AGENTS.md, so both tools follow the same instructions.

1. **Lead:** open the agent in the cloned folder and run prompt 1 from page 5. It writes AGENTS.md with your session routine and makes the `logs/` folder. Also ask: "Create CLAUDE.md containing only the line @AGENTS.md."
2. Read AGENTS.md. Fix anything wrong. Commit and push.
3. **Everyone:** pull. From now on, type **start** when you sit down and **wrap up** when you stop. Update AGENTS.md whenever the team decides something the agent should know.

✓ Before you leave the first session

Each person checks every box. If one fails, a mentor can help before you go.

- | | |
|---|--|
| ☐ I can open the Claude app's Code tab (or Codex) with our project folder selected. | ☐ I pulled and can see my teammates' names in README.md. |
| ☐ That folder is in <code>C:\code</code> or my Mac <code>code</code> folder, not in OneDrive. | ☐ The team folder (OneDrive or Google Drive) shows up in my own drive. |
| ☐ I added my name to README.md, committed, and pushed, and it shows on github.com. | ☐ I typed wrap up , and my log file shows in <code>logs/</code> on github.com. |
| | ☐ I put a calendar reminder to cancel my plan during month two. |

7 Seven prompts to copy

Prompt 1 is run once, by the team lead, during setup. Prompts 2 and 7 open and close every session; once your AGENTS.md has the routine, typing **start** or **wrap up** is enough.

1 · Start the project **LEAD ONLY · ONCE**

Create AGENTS.md for this project: [one paragraph about the app]. Include how to run it; our rules (fake data only, API keys only in a .env file that is never committed); and a session routine. On "start": pull, read AGENTS.md and the three newest files in logs/, then propose a plan. On "wrap up": write logs/[date]_[name].md with Did / Decided / Open / Next, update AGENTS.md if needed, then commit and push. Create logs/ with a short README.md explaining this.

2 · Start a session

Pull the latest changes. Read AGENTS.md and the three most recent files in logs/. Tell me where the project stands, what is open, and propose a plan for today.

3 · Plan first

Before writing any code, propose a plan. Start with the smallest version that works. Wait for my OK.

4 · Build one step

Do step 1 of the plan only. Then tell me exactly how to test it.

5 · Check it

Explain what you built in plain English and how we know the numbers are right. Work one example by hand.

6 · When it breaks

Here is the exact error: [paste it]. Fix one thing at a time and tell me what you changed.

7 · Wrap up

Write logs/[today's date]_[my name].md with Did / Decided / Open / Next. If we decided something the agent should always know, add it to AGENTS.md. Then commit and push.

8 Rules that protect you

- **No real personal data.** No real names, account numbers, or customer records in prompts or code. Use fake data; the website has prompts that make it for you.
- **No passwords or API keys in a chat or on GitHub.** Keys go in a .env file that is never committed.
- **Check before you trust.** The agent can be confidently wrong, especially about finance formulas.
- **Say how you used AI.** Judges score "AI implementation," and that includes being clear about it in your log and demo.
- **Stuck for 20 minutes?** Ask a mentor, or run "My session is going in circles" from the website.