My Journey with Claude Code: 9 Lessons from Building MVPs That Changed My Workflow

I've been coding for years, throwing together MVPs for startups, side hustles, and that one app I thought would make me rich (yeah, right).

Lately, I've been using Claude Code, a terminal-based tool from Anthropic.

After two MVPs and way too many energy drinks, I'm kinda into it. It's like a nerdy buddy who's smart but needs a nudge.

Here's what I figured out.

Photo by Jakub Zerdzicki on Unsplash

1. Claude Needs a Game Plan

First time I used Claude, I was like, "Build me a login thing!" Total mess. Code worked, but it looked like my cat wrote it. Then I found out Claude can plan stuff before coding. Changed everything.

Now, I tell it to sketch a plan first. Like this for a Node.js login:

```
"Plan a login for a Node.js app with JWT. Just the steps, no code."
```

Claude gave me:

```
- Make a POST /login route.
- Set up a user model.
- Add JWT for tokens.
- Write middleware to check auth.
```

I added, “Hey, hash passwords too,” and the code was solid. Takes longer, but it’s worth it.

2. Make a `claude.md` File or You’re Screwed

I skipped this at first. Big mistake. Claude started adding random crap to files I didn’t even know I had. Now, I start with a `claude.md` file. It’s like a cheat sheet for Claude.

I put:

- What the app’s for (like, “a to-do list so I don’t forget client deadlines”).
- Tech I’m using (React, Node, whatever).
- How my folders are set up.
- How the app works for users.

Here’s one I did:

```
# To-Do App for Freelancers
Goal: Track tasks and clients without losing my mind.
Tech: React, Tailwind, Node.js, SQLite.
Folders:
- /src/client: React stuff
- /src/server: API routes
```

```
  - /src/db: Database junk
Flow: Log in, see tasks, add or edit them.
```

Claude checks this and stays on track. No more chaos.

3. Treat Claude Like a Newbie Coder

Claude's clever, but it's not a mind-reader. I treat it like a newbie coder who's eager but needs clear orders. Tell it exactly what file to touch and what to do.

Like:

```
"Add a signup route in /server/routes/auth.js. Use the User model from /serve
```

After it's done, I say:

```
"Check auth.js for mistakes."
```

Claude's weirdly good at catching its own screw-ups, like forgetting an async/await. Saved me from a late-night meltdown once.

4. Git Is Your Best Friend

Claude doesn't ask, "You cool with this?" It just edits. One time, it trashed a file, and I was this close to chucking my laptop. Now, I use Git like it's my lifeline.

My flow:

- New branch: `git checkout -b task-stuff`.
- Commit all the time: `git commit -m "Added task endpoint"`.
- If Claude messes up, I'm out: `git checkout main`.

Like:

```
git checkout -b add-task-api
# Claude does stuff
git add .
git commit -m "Task endpoint done"
# Claude breaks it
git checkout main
```

Git’s why I still have hair.

5. Screenshots Save Lives

I had this button in my React app that looked like it was doing the moonwalk. I tried explaining it to Claude — total fail. Then I sent screenshots: one of the broken button, one of how I wanted it. Claude fixed it in one shot.

I said:

```
"Fix the button in /client/components/TaskButton.js. It’s all wonky [screensh
```

Claude tweaked it:

```
/* Before */
.task-button {
  margin: 10px;
}

/* After */
.task-button {
  display: flex;
  justify-content: center;
  margin: 10px auto;
}
```

Now I screenshot every UI glitch. It’s like Claude got superpowers.

6. Throw Old Code at Claude

I was building a new app that felt like an old MVP I did. So, I gave Claude the whole folder from that project. It was like, “Oh, I get your vibe now.” Copied my style and setup perfectly.

I told it:

```
"Make a TaskCard component in /client/components/TaskCard.js. Copy the style
```

Boom — component done, and it looked like *my* code. Huge time-saver.

7. Tell Claude to Check Its Work

This is my favorite trick. After Claude writes code, I say:

```
"Look at /server/routes/tasks.js. Any bugs?"
```

One time, it caught a dumb mistake:

```
// Original
async function deleteTask(req, res) {
  Task.findByIdAndDelete(req.params.id);
  res.json({ message: "Deleted" });
}

// Claude's Fix
async function deleteTask(req, res) {
  try {
    const task = await Task.findByIdAndDelete(req.params.id);
    if (!task) return res.status(404).json({ message: "No task found" });
    res.json({ message: "Deleted" });
  } catch (error) {
    res.status(500).json({ error: "Something broke" });
  }
}
```

It’s like having a coder and a QA guy in one.

8. Claude's Slow, but It's Got Your Back

Claude's not fast. I once waited forever for it to sort out a tricky API route. I was ready to yeet my computer. But the code? Clean. No bugs. If you're in a rush, Claude'll stress you out. If you want solid code, it's your guy.

Tip: Give it time to think. It's like waiting for your mom to make dinner — slow, but so good.

9. You Gotta Talk Like a Coder

Claude's only as good as your instructions. I used to say stuff like "build an app." Garbage. Now, I act like I'm explaining to a coworker.

Bad:

```
"Make a to-do app."
```

Good:

```
"Build a React component for tasks in /client/components/TaskList.js. Use Tai
```

Give Claude details, and it'll give you gold.

I've been using Cursor forever — it's fast, lives in my IDE, and feels like a speedrun. But Claude's like that friend who's slow but always has your back. It gets my codebase, fixes its own mistakes, and loves screenshots. I'm working on a big Claude vs. Cursor post, so stay tuned.

TL;DR: Claude Code's awesome for MVPs if you keep it focused. Make a `claude.md`, use Git like a maniac, send screenshots, and make it check its work. It's slow, but the code's clean.