

PROMPT PACK · VOL. 1

Vibe Code Starter Pack

33 copy-and-paste prompts for building real apps & games with AI coding tools

For Cursor · Claude · ChatGPT · Windsurf · Bolt · Lovable · Replit

9 sections · Beginner to Intermediate

Welcome! How to Use This Pack

You've got an idea. You've got an AI that writes code. What's usually missing is **knowing what to ask for**, and asking in a way the AI can't misread. That's what this pack is for.

Each prompt here is written to get **clear, complete, beginner-friendly answers** from your AI tool. They're grouped by the problem you're dealing with right now, so you don't have to read the whole thing front to back. Just jump to the section you need.

How each prompt is laid out

Every prompt comes with:

- **Title:** what the prompt is.
- **Use this when...:** one or two sentences on what the prompt is for and when to reach for it.
- **Fill in:** the blanks to replace before you paste.
- **The prompt:** the box you copy.
- **Tips to customize:** optional ways to change it (included where it helps).

Filling in the placeholders

Anything in **[BRACKETS IN CAPS]** is a blank for you to fill in. Delete the brackets and type your own details. For example:

[APP IDEA] → a habit tracker for students that sends a daily reminder

Being specific makes a huge difference. "A to-do app" gets you a generic to-do app. "A to-do app for freelance designers that groups tasks by client and shows what's due this week" gets you something useful.

If you don't know what to put in a blank (for example, you have no idea which tech stack to pick), write **"you choose, and explain why"**. The AI will make a sensible choice and tell you its reasoning.

Common placeholders

Placeholder	What to put there	Example
[APP IDEA]	One or two sentences about what you're building	"A recipe app that suggests meals from what's in my fridge"

Placeholder	What to put there	Example
[TECH STACK]	The languages and frameworks you're using	"Next.js, Tailwind, Supabase"
[TARGET USER]	Who the app is for	"Busy parents who cook on weeknights"
[ERROR MESSAGE]	The full error text, copied exactly	"TypeError: Cannot read properties of undefined (reading 'map')"
[FILE NAME]	The file you're working in	"app/page.tsx"
[FEATURE]	The thing you want to add	"a dark mode toggle"
[SKILL LEVEL]	How much coding you know	"total beginner" / "I know some HTML and JS"

Five habits that make every prompt work better

1. **Give context.** In Cursor or Claude Code, tag the relevant files (with `@filename` or by opening them). In ChatGPT, paste the code in.
2. **Do one thing at a time.** Build one feature, test it, then go to the next one. Huge "build everything" requests are where AI tools go off track.
3. **Ask for a plan first.** Many prompts here ask the AI to plan before it writes code. Read the plan. You can fix a bad plan in seconds, but bad code takes much longer to untangle.
4. **Save your progress.** Commit to Git (or at least copy your project folder) every time something works. Then you can always roll back.
5. **Paste the real error.** Copy the full error text. Don't summarize it as "it's broken".

Quick Start: Your First 30 Minutes

New to vibe coding? Follow these five steps in order. You'll go from idea to a working app on your screen.

1. **Shape your idea.** Use **Prompt 1.1 (Idea to Build Plan)**. You'll get a short, clear plan and a recommended tech stack.
2. **Set up the project.** Use **Prompt 1.2 (Project Setup, Step by Step)**. Follow the steps until you see the app running in your browser.
3. **Build the first screen.** Use **Prompt 2.1 (Build a Screen from a Description)** for your main page.
4. **Hit an error? (You will, and that's normal.)** Use **Prompt 5.1 (Fix This Error)**. Paste the error and keep going.
5. **Add one real feature.** Use **Prompt 1.4 (Add One Feature Safely)**. Repeat this step for each feature until your app does what you want.

When you're ready to share it, go to **Section 6: Deploying & Going Live**.

Golden rule: If the AI gets confused after a long chat, start a new chat. Paste a short summary of your project (Prompt 1.3 makes one for you) and carry on.

What's Inside

1. Starting a New Project (4 prompts)
2. Building UI & Landing Pages (4 prompts)
3. Adding Login & Auth (3 prompts)
4. Databases & Saving Data (4 prompts)
5. Fixing Bugs & Errors (4 prompts)
6. Deploying & Going Live (3 prompts)
7. Making It Look Polished (3 prompts)
8. Adding AI Features & Agents (4 prompts)
9. Simple Browser Games (4 prompts)

SECTION 1

Starting a New Project

Turn a fuzzy idea into a project you can actually build.

1.1 Idea to Build Plan

Use this when you have an idea but don't know where to start or which tools to use. It's for turning a rough idea into a clear, step-by-step build plan with a recommended tech stack.

Fill in: [APP IDEA], [TARGET USER], [PROBLEM IT SOLVES], [SKILL LEVEL], [AI TOOL] (e.g. Cursor / Claude / ChatGPT), [WEB APP / MOBILE / DESKTOP / "NOT SURE"], [BUDGET FOR HOSTING AND SERVICES] (e.g. "free only" / "up to \$10/month"), [TECH STACK].

I want to build an app. Act as a senior developer and a patient mentor helping me plan it.

My app idea: [APP IDEA]

Who it's for: [TARGET USER]

The main problem it solves: [PROBLEM IT SOLVES]

My coding experience: [SKILL LEVEL]

The AI tool I'm building with: [AI TOOL]

Where I want it to run: [WEB APP / MOBILE / DESKTOP / "NOT SURE"]

Budget for hosting and services: [BUDGET FOR HOSTING AND SERVICES]

Please give me:

1. A one-paragraph summary of the app in plain English.
2. The MVP (minimum viable product): the smallest version worth building, with a list of 3-6 must-have features. Put "nice to have later" features in a separate list.
3. A recommended tech stack that's beginner-friendly and free or cheap to start. Explain each choice in one sentence. If [TECH STACK] is filled in, use it unless there's a strong reason not to: [TECH STACK]
4. The main screens or pages, with one line describing each.
5. The data the app needs to store (e.g. users, posts, scores), described simply.
6. A numbered build order of small steps, where each step ends in something I can see working.
7. The 3 biggest risks or tricky parts for a beginner, and how to handle each one.

Keep the language simple and don't write any code yet. At the end, ask me up to 3 questions that would most improve the plan.



Tips to customize: Add "I want to launch in [TIMEFRAME]" to get a plan scoped to your deadline. If you're building a game, mention the genre and the feel you want.

1.2 Project Setup, Step by Step

Use this when you have a plan and stack, but an empty folder. It's for getting a new project created, installed, and running on your computer without getting stuck.

Fill in: [SKILL LEVEL], [APP IDEA], [TECH STACK], [WINDOWS / MAC / LINUX], [CODE EDITOR / AI TOOL] (e.g. Cursor, VS Code, Replit), [ALREADY INSTALLED] (e.g. "Node.js, Git" / "nothing" / "not sure").

Help me set up a brand new project from scratch. Assume I'm a [SKILL LEVEL] and explain each step briefly.

Project: [APP IDEA]
Tech stack: [TECH STACK]
My computer: [WINDOWS / MAC / LINUX]
Code editor / AI tool: [CODE EDITOR / AI TOOL]
Already installed: [ALREADY INSTALLED]

Please:

1. List anything I need to install first, with how to check whether it's already installed (the exact command to run and what the output should look like).
2. Give me the exact terminal commands to create the project, one per code block, in order. Say which folder I should be in before each command.
3. Show the folder structure I should end up with, and explain what the important files and folders do in one line each.
4. Tell me how to start the app locally and what I should see in the browser when it works.
5. Set up a .gitignore and a .env.example file if this stack needs them, and explain what environment variables are in one sentence.
6. Set up Git and make a first commit, with the exact commands.
7. List the 3 most common setup errors for this stack and how to fix each one.

Don't add any features yet. I just want a clean, running starting point.



Tips to customize: If you're using an AI app builder like Bolt, Lovable, or Replit, replace step 2 with "Write the first message I should send to [TOOL] to generate this starting point."

1.3 Project Brief and Rules File

Use this when you're starting a project, or the AI keeps forgetting your stack, style, or decisions. It's for creating a reusable "memory" file so your AI tool stays consistent and remembers your project across chats.

Fill in: [APP IDEA], [TARGET USER], [TECH STACK], [WHAT'S BUILT SO FAR], [NEXT 1-3 FEATURES], [STYLE AND DESIGN PREFERENCES] (e.g. "clean and minimal, dark mode, rounded corners"), [THINGS I DON'T

WANT] (e.g. "no extra libraries without asking", "no TypeScript"), [FILE NAME] (e.g. .cursorrules / CLAUDE.md / PROJECT_BRIEF.md).

Create a project brief file that I'll give to my AI coding assistant at the start of every session. It should be short enough to paste in, but complete enough that a new AI chat understands my project immediately.

App: [APP IDEA]
Target user: [TARGET USER]
Tech stack: [TECH STACK]
Current progress: [WHAT'S BUILT SO FAR]
Next goals: [NEXT 1-3 FEATURES]
Style and design preferences: [STYLE AND DESIGN PREFERENCES]
Things I DON'T want: [THINGS I DON'T WANT]

Format it as Markdown with these sections:

- Project overview (2-3 sentences)
- Tech stack and versions
- Folder structure (key folders only)
- Coding rules (naming, file organization, comment style, keep components small, etc.)
- Design rules (colors, fonts, spacing, tone)
- Rules for the AI: always explain changes briefly, ask before adding new dependencies, don't delete existing features, change only the files needed, and show which files were changed
- Current status and next steps

I'll save this as [FILE NAME].



Tips to customize: Update this file whenever you finish a feature. When a chat gets long or confused, start a new chat with this file and the AI will pick up where you left off.

1.4 Add One Feature Safely

Use this when your app runs and you want to add the next feature on your list. It's for adding a new feature without breaking what already works.

Fill in: [APP IDEA], [TECH STACK], [FEATURE], [DESCRIBE STEP BY STEP] (e.g. "user clicks X, sees Y, then Z is saved"), [FILE NAMES, OR "PLEASE FIND THEM"].

I want to add a new feature to my existing app. Please plan before writing code.

App: [APP IDEA]
Tech stack: [TECH STACK]
Feature to add: [FEATURE]
How it should work from the user's point of view: [DESCRIBE STEP BY STEP]
Relevant files: [FILE NAMES, OR "PLEASE FIND THEM"]

Step 1, plan only:

- Explain in plain English how you'll build this.
- List every file you'll create or change, and why.
- Mention any new packages needed (and ask me before adding them).
- Point out anything that might break existing features.

Wait for me to say "go" before writing code.

Step 2, after I say go:

- Write the code in small, clearly labeled chunks, one file at a time.
- Keep my existing code style and don't rewrite unrelated code.
- Add short comments on any non-obvious parts.
- Finish with a checklist of how I can test the feature by hand, including one edge case to try.

 **Tips to customize:** For bigger features, add "Break this into 3-5 mini-steps and let me test after each one." Small steps are the best way to avoid AI spaghetti code.

SECTION 2

Building UI & Landing Pages

Make screens and pages that look good and work on any device.

2.1 Build a Screen from a Description

Use this when you know what a page should contain, but not how to build it. It's for turning a written description of a screen into working, responsive UI code.

Fill in: [APP IDEA], [TECH STACK] (e.g. React + Tailwind / plain HTML, CSS, JS), [SCREEN NAME] (e.g. Dashboard, Profile, Settings), [WHAT THE USER DOES HERE], [LIST THE PARTS] (e.g. "header with logo and menu; a welcome message with the user's name; 3 stat cards; a list of recent items with edit/delete buttons; an 'Add new' button"), [STYLE] (e.g. "modern, minimal, lots of white space, blue accent color" or "match my existing design").

```
Build this screen for my app.
```

```
App: [APP IDEA]
```

```
Tech stack: [TECH STACK]
```

```
Screen name: [SCREEN NAME]
```

```
Purpose of the screen: [WHAT THE USER DOES HERE]
```

```
What's on the screen, from top to bottom:
```

```
[LIST THE PARTS]
```

```
Style: [STYLE]
```

```
Requirements:
```

- Fully responsive: looks good on phone, tablet, and desktop. Tell me how the layout changes on mobile.
- Use placeholder or fake data for now, kept in one clearly marked spot so I can swap in real data later.
- Split it into small reusable components if the stack supports that.
- Include hover and focus states on buttons and links.
- Use semantic HTML and basic accessibility (alt text, labels, good contrast).
- Tell me exactly which file(s) to put the code in.

```
After the code, give me 3 quick ideas to make this screen feel more polished.
```

 **Tips to customize:** Got a screenshot of a design you like? Attach it and add "Match the layout and feel of the attached image."

2.2 High-Converting Landing Page

Use this when you need a homepage, waitlist page, or sales page for your app or product. It's for creating a complete landing page, copy included, that explains your product and gets sign-ups.

Fill in: [PRODUCT NAME], [ONE-LINE DESCRIPTION], [TARGET USER], [PROBLEM], [BENEFIT 1], [BENEFIT 2], [BENEFIT 3], [MAIN CALL TO ACTION] (e.g. "Join the waitlist" / "Buy now - [YOUR PRICE]" / "Start free"), [BRAND VIBE] (e.g. "playful and colorful" / "sleek dark tech" / "warm and friendly"), [TECH STACK], [PRICING DETAILS], [TONE], [WHAT THE BUTTON DOES] (e.g. "link to /signup", "email capture form", or "Stripe checkout link" plus the URL).

Create a complete, high-converting landing page.

Product: [PRODUCT NAME]. [ONE-LINE DESCRIPTION]

Target customer: [TARGET USER]

Their main pain point: [PROBLEM]

Top 3 benefits: [BENEFIT 1], [BENEFIT 2], [BENEFIT 3]

Main call to action: [MAIN CALL TO ACTION]

Brand vibe: [BRAND VIBE]

Tech stack: [TECH STACK]

Sections, in order:

1. Hero: a punchy headline (give me 3 options), a subheadline, the CTA button, and a spot for a product image or demo.
2. Problem: show I understand the customer's frustration.
3. Solution / How it works: 3 simple steps with icons.
4. Features and benefits: 3-6 cards that focus on outcomes, not tech jargon.
5. Social proof: placeholder testimonials or stats (clearly marked as placeholders).
6. Pricing or offer (if relevant): [PRICING DETAILS]
7. FAQ: 5 questions that answer real objections.
8. Final CTA section and a simple footer.

Requirements:

- Write all the copy in a [TONE] voice. No lorem ipsum.
- Mobile-first and responsive, with fast-loading lightweight code.
- Make the CTA button work as: [WHAT THE BUTTON DOES].
- Include SEO basics: a page title, meta description, and Open Graph tags for social sharing.



Tips to customize: Building a waitlist? Ask the AI to connect the email form to a free service such as Formspree, Google Sheets, or Supabase and show you how to check the sign-ups.

2.3 Turn a Screenshot into Code

Use this when you've sketched a layout, designed one in Figma, or found a site whose layout you want to use as inspiration. It's for recreating a design or layout from an image.

Fill in: [TECH STACK], [PURPOSE], [CHANGES FROM THE IMAGE] (e.g. "use my brand color #6C5CE7 instead of red", "remove the sidebar", or "none").

```
I've attached an image of a UI design. Recreate it as working code.
```

```
Tech stack: [TECH STACK]
```

```
What this screen is for: [PURPOSE]
```

```
Changes from the image: [CHANGES FROM THE IMAGE]
```

```
Please:
```

1. First, describe what you see in the image: the layout, sections, colors, fonts, and spacing. That way I can confirm you understood it correctly.
2. Then build it as closely as you can, splitting it into sensible components.
3. Make it responsive. The image may only show desktop, so make sensible choices for mobile and explain them.
4. Use real-looking placeholder text and images (e.g. from placeholder.co).
5. List anything you couldn't tell from the image and the guesses you made.

```
Don't copy logos, trademarks, or copyrighted text from the image. Use placeholders instead.
```

 **Tips to customize:** For a closer match, add "Use pixel-perfect spacing and estimate the exact font sizes and colors as hex codes."

2.4 Make My UI Responsive (Fix Mobile)

Use this when your app looks fine on desktop but bad on mobile. It's for fixing layouts that break, overflow, or look cramped on phones.

Fill in: [TECH STACK], [FILE NAME], [WHAT GOES WRONG ON MOBILE] (e.g. "page scrolls sideways", "buttons overlap", "text is tiny", "menu doesn't fit"), [SCREEN SIZES I CARE ABOUT] (e.g. iPhone SE up to large desktop), [PASTE THE CODE OR TAG THE FILES].

```
My app's layout breaks on mobile. Help me make it fully responsive.
```

```
Tech stack: [TECH STACK]
```

```
The file(s) with the problem: [FILE NAME]
```

```
What goes wrong on mobile: [WHAT GOES WRONG ON MOBILE]
```

```
Screen sizes I care about: [SCREEN SIZES I CARE ABOUT]
```

```
[PASTE THE CODE OR TAG THE FILES]
```

Please:

1. Explain in plain English WHY it breaks (e.g. fixed widths, missing wrap, wrong units).
2. Fix it with a mobile-first approach, changing only what's needed.
3. Add a mobile navigation pattern if the menu doesn't fit (e.g. a hamburger menu that opens and closes).
4. Make sure tap targets are at least 44px and text is readable without zooming.
5. Check for the usual causes of sideways scrolling and fix any you find.
6. Tell me how to test it using my browser's device toolbar.

SECTION 3

Adding Login & Auth

Let people sign up and log in, and keep their data private.

3.1 Add Sign Up and Login

Use this when your app needs user accounts. It's for adding a full, secure sign-up, login, and logout system without building it all from scratch.

Fill in: [SKILL LEVEL], [APP IDEA], [TECH STACK], [AUTH PROVIDER] (e.g. Supabase Auth / Clerk / Firebase Auth / NextAuth / "recommend one that's free to start"), [LOGIN METHODS I WANT] (e.g. email + password, Google login, magic link), [PAGE] (e.g. /dashboard).

Add user authentication to my app. Prioritize security and simplicity. I'm a [SKILL LEVEL].

App: [APP IDEA]

Tech stack: [TECH STACK]

Auth provider: [AUTH PROVIDER]

Login methods I want: [LOGIN METHODS I WANT]

After login, users should go to: [PAGE]

Please:

1. If I haven't chosen a provider, recommend one for my stack in 2-3 sentences. Use an established auth service instead of a hand-made password system.
2. Walk me through setting up the provider's dashboard step by step (what to click, which keys to copy).
3. Show me how to store the keys safely in a .env file. Tell me which keys are safe in the browser and which must stay secret.
4. Build these pages and components: sign up, log in, log out button, forgot password, and a page that only shows when logged in.
5. Show friendly error messages (wrong password, email already used, etc.).
6. Show the logged-in user's name or email in the header.
7. Give me a test checklist: sign up, log out, log back in, try to visit a protected page while logged out.



Tips to customize: Building an AI app or a game? Add "Also create a user profile record with [FIELDS] when someone signs up." For example: username, avatar, credits, high score.

3.2 Protect Pages and User Data

Use this when login works, but you're not sure your app is actually secure. It's for making sure users can only see and edit their own data, and that logged-out visitors can't reach private pages.

Fill in: [TECH STACK], [AUTH PROVIDER], [DATABASE], [DATA THAT SHOULD BE PRIVATE] (e.g. "each user's notes, projects, and settings"), [DATA THAT SHOULD BE PUBLIC] (e.g. "public profiles, leaderboard" or "none"), [LIST PAGES], [PAGES], [PASTE OR TAG RELEVANT FILES].

My app has login working. Now help me make sure each user's data is protected.

Tech stack: [TECH STACK]

Auth provider: [AUTH PROVIDER]

Database: [DATABASE]

Data that should be private: [DATA THAT SHOULD BE PRIVATE]

Data that should be public: [DATA THAT SHOULD BE PUBLIC]

Pages that need login: [LIST PAGES]

Please:

1. Explain the difference between hiding something in the UI and actually protecting it on the server/database, in plain English.
2. Add route protection so logged-out users are redirected to login from: [PAGES].
3. Set up database security rules (e.g. Row Level Security in Supabase, or Firebase Security Rules) so users can only read and write their own rows. Give me the exact rules and explain each one.
4. Check my API routes / server functions and make sure each one checks who the user is before doing anything.
5. Give me a simple way to test it: log in as two different test users and confirm neither can see the other's data.
6. List any other security gaps you notice in the code I've shared.

[PASTE OR TAG RELEVANT FILES]

3.3 Fix My Broken Login

Use this when login, sign-up, or "stay logged in" isn't working and you're stuck. It's for figuring out why auth is failing: redirect loops, sessions that don't stick, OAuth errors.

Fill in: [TECH STACK], [AUTH PROVIDER], [EXPECTED BEHAVIOR], [ACTUAL BEHAVIOR] (e.g. "redirect loop", "logged out on refresh", "Google login error"), [ERROR MESSAGE], [YES / NO / HAVEN'T DEPLOYED], [PASTE OR TAG THE AUTH-RELATED FILES].

My login isn't working. Help me find and fix the problem step by step.

Tech stack: [TECH STACK]

Auth provider: [AUTH PROVIDER]

What I expected: [EXPECTED BEHAVIOR]

What actually happens: [ACTUAL BEHAVIOR]

Error message (browser console, terminal, or screen): [ERROR MESSAGE]
Works locally but not in production? [YES / NO / HAVEN'T DEPLOYED]

[PASTE OR TAG THE AUTH-RELATED FILES]

Please:

1. List the most likely causes for these symptoms with this provider, ranked by likelihood.
2. For each cause, tell me exactly what to check (which setting, file, or dashboard field) and what "correct" looks like.
3. Pay special attention to: redirect / callback URLs, environment variables, cookie/session settings, and the difference between localhost and the live URL.
4. Once we find the cause, give me the fix with a clear explanation.
5. Tell me how to stop this from happening again when I deploy.

SECTION 4

Databases & Saving Data

Store, load, and organize your app's data so nothing gets lost.

4.1 Design My Database

Use this when your app needs to store data and you're not sure how to structure it. It's for planning tables and relationships before you build, so you don't have to redo them later.

Fill in: [SKILL LEVEL], [APP IDEA], [DATABASE] (e.g. Supabase (Postgres) / Firebase Firestore / SQLite / "recommend one"), [WHAT USERS CAN DO] (e.g. "create projects, add tasks to projects, comment on tasks, like posts"), [THINGS THE APP NEEDS TO REMEMBER] (e.g. users, projects, tasks, scores, messages).

Help me design the database for my app. Explain it simply, because I'm a [SKILL LEVEL].

App: [APP IDEA]

Database: [DATABASE]

What users can do: [WHAT USERS CAN DO]

Things the app needs to remember: [THINGS THE APP NEEDS TO REMEMBER]

Please:

1. If I haven't chosen a database, recommend one for my use case and explain why in 2 sentences.
2. List each table (or collection) with its fields, data types, and a one-line explanation of each field.
3. Explain how the tables connect (e.g. "one user has many projects") and draw a simple text diagram.
4. Point out which fields need to be required, unique, or indexed, and why.
5. Give me the exact code or SQL to create it all.
6. Add a few rows of realistic sample data so I can test with it.
7. Point out one or two design mistakes beginners usually make for this kind of app, and how this design avoids them.



Tips to customize: Expecting the app to grow? Add "Design it so I can add [FUTURE FEATURE] later without rebuilding."

4.2 Connect My App to the Database (CRUD)

Use this when your UI uses fake data and you want to connect it to a real database. It's for making your app actually create, read, update, and delete real data.

Fill in: [TECH STACK], [DATABASE], [TABLE / COLLECTION NAME], [FIELDS], [SCREEN / FILE NAME], [CURRENTLY USING] (e.g. "a hard-coded array of fake data"), [PASTE OR TAG THE RELEVANT FILES].

Connect this part of my app to my real database so users can create, read, update, and delete data (CRUD).

Tech stack: [TECH STACK]

Database: [DATABASE]

The data: [TABLE / COLLECTION NAME] with fields [FIELDS]

The screen it's for: [SCREEN / FILE NAME]

Currently using: [CURRENTLY USING]

[PASTE OR TAG THE RELEVANT FILES]

Please:

1. Set up the database connection (if it isn't already), keeping keys in environment variables.
2. Write clear functions for: create, list all (for the current user), get one, update, and delete.
3. Wire them into my UI, replacing the fake data.
4. Add loading states (a spinner or skeleton), empty states ("No items yet - add your first one!"), and error messages.
5. Make the UI update straight away after adding, editing, or deleting, without needing a page refresh.
6. Ask "Are you sure?" before deleting.
7. Validate input before saving (required fields, max length) and show friendly messages.
8. Give me a quick test checklist.

4.3 Save Data Without a Backend

Use this when you're building a simple tool, prototype, or game and don't need accounts yet. It's for saving user progress, settings, or game data in the browser with no server or database.

Fill in: [APP IDEA], [TECH STACK], [WHAT TO SAVE] (e.g. "to-do list items", "game high score and unlocked levels", "theme setting"), [WHEN TO SAVE] (e.g. "every time something changes", "when the level ends").

I want my app to remember data between visits, without setting up a server or database.

App: [APP IDEA]

Tech stack: [TECH STACK]

What to save: [WHAT TO SAVE]

When to save: [WHEN TO SAVE]

Please:

1. Explain localStorage (and IndexedDB, if my data is large) in plain English, including its limits: it's per-browser, users can clear it, and it isn't secure for secrets.
2. Write a small, reusable save/load helper that handles JSON, missing data, and corrupted data without crashing.
3. Connect it to my app so the data loads on startup and saves automatically.
4. Add a "Reset / Clear data" button.
5. Add a version number to the saved data, so that if I change the structure later, old saves don't break the app.
6. Optional: add export/import of the save as a file, so users can back up their data.

 **Tips to customize:** Later, when you add accounts, use Prompt 4.2 and ask the AI to "migrate localStorage data into the database the first time a user logs in."

4.4 Add Search, Filters, and Sorting

Use this when your app shows a list that's getting long. It's for letting users find things quickly in lists of products, posts, tasks, and so on.

Fill in: [TECH STACK], [DATABASE OR "DATA IS STORED IN THE BROWSER"], [ITEMS] (e.g. "recipes"), [FIELDS], [SEARCH SHOULD MATCH] (e.g. "title and description"), [FILTERS I WANT] (e.g. "category dropdown, 'only favorites' checkbox, price range"), [SORT OPTIONS] (e.g. "newest, oldest, A-Z, most popular"), [ROUGHLY HOW MANY ITEMS] (e.g. "under 100" / "thousands"), [PASTE OR TAG THE LIST COMPONENT].

Add search, filtering, and sorting to a list in my app.

Tech stack: [TECH STACK]

Database: [DATABASE OR "DATA IS STORED IN THE BROWSER"]

The list shows: [ITEMS] with fields [FIELDS]

Search should match: [SEARCH SHOULD MATCH]

Filters I want: [FILTERS I WANT]

Sort options: [SORT OPTIONS]

Roughly how many items: [ROUGHLY HOW MANY ITEMS]

[PASTE OR TAG THE LIST COMPONENT]

Please:

1. Recommend whether to filter in the browser or in the database, based on the number of items, and explain why.
2. Build the search bar, filters, and sort dropdown with a clean, mobile-friendly layout.
3. Debounce the search so it doesn't run on every single keystroke.

4. Keep the filters in the URL (e.g. `?q=pasta&sort=new`) so results can be shared and survive a refresh.
5. Show a "No results" message with a "Clear filters" button.
6. Add pagination or "Load more" if there could be many items.

SECTION 5

Fixing Bugs & Errors

Get unstuck fast and actually understand what went wrong.

5.1 Fix This Error

Use this when you see a red error in your terminal, browser console, or on screen. It's for diagnosing and fixing an error message, and learning what caused it.

Fill in: [SKILL LEVEL], [TECH STACK], [ACTION] (e.g. "run npm run dev" / "click the save button"), [ERROR MESSAGE], [TERMINAL / BROWSER CONSOLE / ON THE PAGE / DEPLOY LOGS], [RECENT CHANGES, OR "NOT SURE"], [PASTE CODE OR TAG FILES] (e.g. FILE NAME and line number from the error).

I'm getting an error and need help fixing it. Please explain things simply, because I'm a [SKILL LEVEL].

Tech stack: [TECH STACK]

What I was trying to do: [ACTION]

The full error message (copied exactly):

[ERROR MESSAGE]

Where it appears: [TERMINAL / BROWSER CONSOLE / ON THE PAGE / DEPLOY LOGS]

What I changed right before it started: [RECENT CHANGES, OR "NOT SURE"]

Relevant code:

[PASTE CODE OR TAG FILES]

Please:

1. Explain what this error means in plain English (1-3 sentences).
2. Tell me the most likely cause in MY code, pointing to the exact file and line.
3. Give me the fix, showing just the code that changes, with before/after.
4. If you're not sure, list the top 3 possible causes and how to check each one, instead of guessing.
5. Tell me how to confirm it's fixed.
6. Give me one tip to avoid this kind of error in the future.

Don't rewrite unrelated parts of my code.



Tips to customize: If the first fix doesn't work, reply with "That didn't work. Here's the new error/behavior: [NEW ERROR]. What else could it be?" Don't just say "still broken".

5.2 It Runs, But Does the Wrong Thing

Use this when nothing crashes, but buttons don't work, data is wrong, or things don't update. It's for tracking down logic bugs where there's no error but the app behaves incorrectly.

Fill in: [TECH STACK], [FEATURE], [STEP], [EXPECTED], [ACTUAL], [ALWAYS / SOMETIMES / ONLY AFTER X], [PASTE OR TAG THE RELEVANT FILES].

My app doesn't show an error, but it's not behaving correctly. Help me debug it methodically.

Tech stack: [TECH STACK]

Feature with the problem: [FEATURE]

Steps to reproduce:

1. [STEP]
2. [STEP]
3. [STEP]

What I expected: [EXPECTED]

What actually happens: [ACTUAL]

Does it happen every time? [ALWAYS / SOMETIMES / ONLY AFTER X]

[PASTE OR TAG THE RELEVANT FILES]

Please:

1. Walk through how the code SHOULD flow for this feature, step by step, and point out where it likely goes wrong.
2. Add temporary console.log (or print) statements at the key points so I can see what's happening, and tell me what each log should show if things are working.
3. Once I share the log output, help me narrow it down to the exact cause.
4. Check for common culprits: state not updating, async code not awaited, wrong variable, stale data, event handlers not attached, and off-by-one errors.
5. Give me the fix and remind me to remove the debug logs afterwards.

5.3 The AI Broke My Code: Recover and Untangle

Use this when things worked before and now several things are broken, or the AI keeps going in circles. It's for getting back to a working state after an AI change made a mess, and then moving forward carefully.

Fill in: [TECH STACK], [LIST], [LIST + ERROR MESSAGES], [YES / NO], [SUMMARY].

My project was working, but after several AI-made changes, things are broken and I'm going in circles. Help me recover calmly.

Tech stack: [TECH STACK]

What was working before: [LIST]

What's broken now: [LIST + ERROR MESSAGES]

Do I use Git? [YES / NO]

Recent changes I asked for: [SUMMARY]

Please:

1. If I use Git, give me the exact commands to see what changed (git status, git diff) and how to safely go back to the last working commit WITHOUT losing anything, e.g. by saving my current work to a new branch first.
2. If I don't use Git, help me figure out which files were changed and how to undo the changes one by one.
3. Once I'm back to a working version, help me re-add the feature I wanted in small, tested steps.
4. Set up Git for me now if I don't have it, with a simple "commit every time something works" routine.
5. Give me 3 rules to prevent this from happening again (e.g. one change at a time, new chat for new features).

Be calm and step-by-step, because I'm stressed about losing my work.

 **Tips to customize:** In Cursor, you can also use the checkpoint/restore feature in the chat to roll back to before a specific AI edit.

5.4 Explain This Code to Me

Use this when you're looking at a file and have no idea what it does. It's for understanding code the AI wrote, so you can change it confidently.

Fill in: [SKILL LEVEL], [FILE NAME], [YOUR GUESS, OR "NO IDEA"], [SPECIFIC LINES / CONCEPTS, OR "ALL OF IT"], [PASTE CODE OR TAG FILE].

Explain this code to me like I'm a [SKILL LEVEL]. I want to understand it well enough to change it myself.

File: [FILE NAME]

What I think it's supposed to do: [YOUR GUESS, OR "NO IDEA"]

The part that confuses me most: [SPECIFIC LINES / CONCEPTS, OR "ALL OF IT"]

[PASTE CODE OR TAG FILE]

Please:

1. Give me a 2-3 sentence summary of what this file does and how it fits into the app.
2. Walk through it section by section in plain English, explaining any jargon the first time it appears.
3. Point out the parts I'm most likely to want to change (e.g. text, colors, limits, settings) and how.
4. Point out anything that looks buggy, insecure, or overcomplicated.
5. Give me one small, safe exercise to try (e.g. "change X to Y and see what happens"), so I learn by doing.

SECTION 6

Deploying & Going Live

Get your app off your laptop and onto the internet.

6.1 Deploy My App, Step by Step

Use this when your app works locally and you're ready to share it. It's for putting your app on the internet with a real, shareable link.

Fill in: [TECH STACK], [DATABASE / SERVICES USED] (e.g. Supabase, Stripe, OpenAI API, or "none"), [WHERE MY CODE IS] (e.g. GitHub repo / only on my computer), [PREFERRED HOST] (e.g. Vercel / Netlify / Render / Railway / GitHub Pages / "recommend a free one"), [YES - DOMAIN / NO / LATER].

Help me deploy my app so anyone can use it with a link. I've never done this before, so go step by step.

Tech stack: [TECH STACK]

Database / services used: [DATABASE / SERVICES USED]

Where my code is: [WHERE MY CODE IS]

Preferred host: [PREFERRED HOST]

Custom domain? [YES - DOMAIN / NO / LATER]

Please:

1. Recommend the best free or cheap host for my stack (if I haven't picked one) and explain why.
2. Give me a pre-deploy checklist: the app builds without errors (with the exact build command), there are no secret keys in the code, and .env is in .gitignore.
3. Walk me through pushing my code to GitHub, if it isn't there already.
4. Walk me through connecting the repo to the host, click by click, including where to add environment variables.
5. List every setting that needs updating for production (e.g. auth redirect URLs, API allowed origins, database URLs).
6. Tell me how to check that the live site works, and where to find the logs if it doesn't.
7. If I want a custom domain, explain how to connect it in simple terms.
8. Explain how future updates work (e.g. "push to GitHub and it redeploys automatically").

6.2 Fix My Failed Deploy

Use this when the deploy failed, or the live site is blank, broken, or erroring. It's for fixing build failures and "works locally, broken in production" problems.

Fill in: [TECH STACK], [HOST], [WHAT HAPPENS] (e.g. "build fails" / "blank white page" / "500 error" / "works locally but login fails live"), [ERROR MESSAGE / LOGS], [LIST NAMES ONLY - NEVER PASTE THE ACTUAL SECRET VALUES], [PASTE OR TAG RELEVANT CONFIG FILES] (e.g. package.json, next.config.js, vite.config.js).

My deploy is failing (or the live site is broken). Help me fix it.

Tech stack: [TECH STACK]

Host: [HOST]

What happens: [WHAT HAPPENS]

Build or runtime logs (copy the error part):

[ERROR MESSAGE / LOGS]

Environment variables I've set on the host: [LIST NAMES ONLY - NEVER PASTE THE ACTUAL SECRET VALUES]

[PASTE OR TAG RELEVANT CONFIG FILES]

Please:

1. Explain what the log is telling me in plain English.
2. Check the common deploy problems: missing environment variables, wrong build command or output folder, case-sensitive file names (works on Mac/Windows, fails on Linux), wrong Node version, code that only works in the browser running on the server, and hard-coded localhost URLs.
3. Give me the exact fix and the steps to redeploy.
4. Tell me how to run a production build locally, so I can catch these problems before deploying next time.

6.3 Launch Readiness Checklist

Use this when you're about to share your app publicly, launch on social media, or start charging money. It's for catching embarrassing or risky problems before real users show up.

Fill in: [APP IDEA], [URL], [TECH STACK], [USES] (e.g. login, database, payments, AI API, file uploads), [EXPECTED USERS AT LAUNCH] (e.g. "50 friends" / "a few thousand from a TikTok post"), [TAG KEY FILES OR SHARE THE REPO STRUCTURE].

I'm about to launch my app publicly. Review my project and give me a launch-readiness checklist tailored to it.

App: [APP IDEA]

Live URL: [URL]

Tech stack: [TECH STACK]

Uses: [USES]

Expected users at launch: [EXPECTED USERS AT LAUNCH]

[TAG KEY FILES OR SHARE THE REPO STRUCTURE]

Cover these areas, marking each item as Must fix / Should fix / Nice to have:

1. Security: exposed API keys, database rules, admin pages, rate limits on expensive actions (especially AI API calls).
2. Costs: anything that could run up a surprise bill if lots of people use it, and how to set spending limits.
3. Reliability: error handling, what users see when something fails, and a 404 page.
4. Basics: page titles, favicon, social share image, mobile check, loading speed.
5. Legal and trust: privacy policy and terms (and whether I need them), plus a contact or feedback link.
6. Analytics: a simple, privacy-friendly way to see how many people use the app.
7. Feedback: the easiest way for early users to report bugs.

Then give me the fixes for anything marked "Must fix".

SECTION 7

Making It Look Polished

Go from "a dev made this" to "this looks legit".

7.1 Design Glow-Up

Use this when your app works but looks amateur, and you're not a designer. It's for upgrading a plain or messy UI into something clean, modern, and consistent.

Fill in: [TECH STACK], [APP IDEA], [TARGET USER], [VIBE I WANT] (e.g. "clean like Linear", "friendly like Duolingo", "dark and techy", "cozy pixel-art game"), [COLORS OR "PICK FOR ME"], [PASTE OR TAG THE MAIN SCREENS / COMPONENTS, OR ATTACH A SCREENSHOT].

My app works, but it looks basic. Act as a senior product designer and give it a professional glow-up without changing how it works.

Tech stack: [TECH STACK]

App: [APP IDEA]

Target user: [TARGET USER]

Vibe I want: [VIBE I WANT]

Brand colors (if any): [COLORS OR "PICK FOR ME"]

[PASTE OR TAG THE MAIN SCREENS / COMPONENTS, OR ATTACH A SCREENSHOT]

Please:

1. Critique the current design: list the top 5 things making it look amateur (spacing, alignment, colors, typography, hierarchy, consistency).
2. Create a simple design system: a color palette (primary, background, surface, text, success, error), a font pairing, a type scale, a spacing scale, a border radius, and shadows. Set these up as CSS variables or theme config.
3. Apply it to my screens: improve spacing, alignment, and visual hierarchy, and make buttons and inputs look consistent.
4. Keep ALL existing functionality exactly the same.
5. Make sure the text contrast is accessible.
6. Explain each change briefly so I learn the design principle behind it.



Tips to customize: Add "Also create a dark mode version with a toggle that remembers the user's choice."

7.2 Microinteractions and Animations

Use this when the design is decent, but the app feels static or stiff. It's for adding smooth transitions, hover effects, and small touches of delight that make an app feel high quality.

Fill in: [TECH STACK], [ANIMATION LIBRARY PREFERENCE] (e.g. "CSS only" / "Framer Motion" / "recommend one"), [LIST], [PERSONALITY] (e.g. "subtle and professional" / "bouncy and playful" / "game-like and juicy").

Add tasteful animations and microinteractions to my app so it feels smooth and high quality, without being distracting.

Tech stack: [TECH STACK]

Animation library preference: [ANIMATION LIBRARY PREFERENCE]

Screens / components to enhance: [LIST]

Personality: [PERSONALITY]

Please add:

1. Hover, press, and focus effects on buttons, cards, and links.
2. Smooth page or section transitions.
3. Elements that fade or slide in as they appear.
4. Loading skeletons instead of blank screens or plain spinners.
5. Success feedback (e.g. a checkmark animation or toast notification) after actions like save or submit.
6. One "delight" moment that fits my app (e.g. confetti when a goal is completed).

Rules:

- Keep animations short (150-400ms) and smooth (animate transform/opacity).
- Respect "prefers-reduced-motion" for users who turn off animations.
- Don't change any functionality.

7.3 Empty, Loading, and Error States

Use this when your app looks broken or confusing when there's no data, while it's loading, or when something fails. It's for handling all the "in-between" moments that make apps feel finished and trustworthy.

Fill in: [TECH STACK], [APP IDEA], [TONE OF VOICE] (e.g. "friendly and encouraging" / "short and professional" / "playful"), [PASTE OR TAG THE MAIN SCREENS AND DATA-LOADING CODE].

Audit my app for missing "in-between" states and add them so it feels finished.

Tech stack: [TECH STACK]

App: [APP IDEA]

Tone of voice: [TONE OF VOICE]

[PASTE OR TAG THE MAIN SCREENS AND DATA-LOADING CODE]

For every screen and every action that loads or saves data, please:

1. List where these states are missing: loading, empty (no data yet), error (request failed), success, and offline / slow connection.
2. Design and add each missing state:
 - Loading: a skeleton or spinner that matches the layout.
 - Empty: a friendly message and a clear next-step button (e.g. "Create your first project").
 - Error: a human-friendly message (no raw error codes), a "Try again" button, and the technical error logged to the console.
 - Success: a short confirmation (toast or inline message).
3. Disable submit buttons while saving, so users can't double-submit.
4. Add a friendly 404 page and a fallback screen if the whole app crashes.
5. Write all the messages in my tone of voice.

SECTION 8

Adding AI Features & Agents

Put AI inside your own app, from chatbots to agents that take actions.

8.1 Add an AI Chat Feature

Use this when you want users to chat with AI inside your app. It's for adding a working AI chatbot or assistant to your app, safely connected to an AI API.

Fill in: [APP IDEA], [TECH STACK], [AI PROVIDER] (e.g. OpenAI / Anthropic / Google Gemini / xAI / OpenRouter / "recommend one"), [WHAT THE ASSISTANT DOES] (e.g. "helps users plan workouts", "answers questions about my product"), [ASSISTANT PERSONALITY] (e.g. "friendly coach, short answers, uses emojis sparingly"), [WHERE IT APPEARS] (e.g. "a chat page" / "a floating chat bubble in the corner").

Add an AI chat feature to my app.

App: [APP IDEA]

Tech stack: [TECH STACK]

AI provider: [AI PROVIDER]

What the assistant does: [WHAT THE ASSISTANT DOES]

Assistant personality: [ASSISTANT PERSONALITY]

Where it appears: [WHERE IT APPEARS]

Please:

1. Explain why the API key must ONLY be used on the server, never in the browser, and set up a server route / API endpoint to handle the AI calls.
2. Write a strong system prompt for the assistant based on my description, including what it should refuse or redirect.
3. Build the chat UI: a message list, input box, send button, auto-scroll, "AI is typing" indicator, and Markdown rendering for responses.
4. Stream the response word by word, if my stack supports it.
5. Keep the conversation history so it remembers earlier messages, with a sensible limit to control costs.
6. Add error handling (API down, rate limit hit) with friendly messages.
7. Add basic cost protection: a max message length, a per-user rate limit, and a max tokens setting.
8. Tell me roughly what it might cost per 100 conversations, and how to set a spending limit with the provider.

 **Tips to customize:** Want it to know about your own content? Add "Let the assistant answer from these documents: [DOCS]". For bigger document sets, see Prompt 8.3.

8.2 Build an AI Agent That Uses Tools

Use this when you want AI that *does things*, like booking, researching, managing tasks, or controlling parts of your app. It's for creating an AI agent that can take actions (search, look up data, call APIs, update your app), not just chat.

Fill in: [SKILL LEVEL], [WHAT THE AGENT DOES] (e.g. "a study buddy that can create flashcards, quiz me, and track my scores"), [TECH STACK], [AI PROVIDER / FRAMEWORK] (e.g. OpenAI function calling / Anthropic tool use / Vercel AI SDK / LangChain / "recommend the simplest option"), [TOOL 1] (e.g. "create_flashcard(question, answer)"), [TOOL 2] (e.g. "get_my_flashcards(topic)"), [TOOL 3] (e.g. "save_quiz_score(score)").

```
Help me build a simple AI agent that can use tools to complete tasks. Explain the
concepts as we go, because I'm a [SKILL LEVEL].
```

```
App / agent idea: [WHAT THE AGENT DOES]
```

```
Tech stack: [TECH STACK]
```

```
AI provider / framework: [AI PROVIDER / FRAMEWORK]
```

```
Tools the agent should have:
```

- [TOOL 1]
- [TOOL 2]
- [TOOL 3]

```
Please:
```

1. Explain in plain English how an agent loop works (the model decides, calls a tool, sees the result, then decides again).
2. Define each tool with a clear name, description, and input schema, since good descriptions are what help the AI pick the right tool.
3. Write the agent loop with a max-steps limit, so it can't run forever or burn money.
4. Write a system prompt that explains the agent's job, when to use each tool, and when to ask the user instead of guessing.
5. Add safety: require user confirmation before any destructive or costly action (deleting, sending, paying), and validate all tool inputs.
6. Log each step (thought / tool call / result) so I can see what the agent is doing while I debug.
7. Give me 5 test requests, from easy to tricky, to check that it behaves well.

8.3 Chat with My Documents (Simple RAG)

Use this when you want answers based on *your* information, not the AI's general knowledge. It's for letting AI answer questions using your own content: docs, notes, FAQs, game lore, product info.

Fill in: [APP IDEA], [TECH STACK], [MY CONTENT] (e.g. "30 Markdown help articles" / "PDFs" / "a game wiki" / "product FAQs"), [ROUGHLY HOW MUCH CONTENT] (e.g. "about 50 pages"), [AI PROVIDER], [VECTOR STORAGE] (e.g. Supabase pgvector / "recommend the simplest free option").

```
Build a "chat with my documents" feature (retrieval-augmented generation, or RAG)
for my app. Keep it as simple as possible.
```

```
App: [APP IDEA]
```

```
Tech stack: [TECH STACK]
```

```
My content: [MY CONTENT]
```

```
Roughly how much content: [ROUGHLY HOW MUCH CONTENT]
```

```
AI provider: [AI PROVIDER]
```

```
Vector storage: [VECTOR STORAGE]
```

```
Please:
```

1. Explain RAG in 3-4 plain sentences.
2. Tell me whether my content is small enough to skip RAG and just paste it into the prompt. If so, show me that simpler approach first.
3. If RAG is needed: write a script that splits my content into chunks, creates embeddings, and stores them.
4. Write the question-answering flow: embed the question, find the top matching chunks, and send them to the AI with instructions to answer **ONLY** from those chunks.
5. Make it cite which document each answer came from, and say "I don't know" when the answer isn't in the docs.
6. Show me how to re-run the script when my documents change.
7. Give me 5 test questions, including one that **ISN'T** in the docs, to check that it doesn't make things up.

8.4 Generate Content with AI (Structured Output)

Use this when you need AI to fill in your app's content automatically, not just chat. It's for using AI to generate structured content inside your app, like quizzes, game levels, product descriptions, characters, or plans, in a format your code can use reliably.

Fill in: [CONTENT TYPE] (e.g. "quiz questions" / "RPG characters" / "meal plans"), [APP IDEA], [TECH STACK], [AI PROVIDER], [USER INPUT] (e.g. "a topic and difficulty level"), [FIELDS] (e.g. "question, 4 options, correct answer index, explanation"), [NUMBER], [DATABASE / LOCALSTORAGE].

```
Add a feature where AI generates [CONTENT TYPE] inside my app, and returns it as
structured data my code can display.
```

App: [APP IDEA]
Tech stack: [TECH STACK]
AI provider: [AI PROVIDER]
User input: [USER INPUT]
Each generated item should have these fields: [FIELDS]
How many per request: [NUMBER]

Please:

1. Design a JSON schema for the output and use the provider's structured output / JSON mode, so the response always matches the schema.
2. Write a detailed prompt template that produces high-quality, varied results (not repetitive), with my user's inputs inserted.
3. Validate the response in code (e.g. with Zod or similar). If it's invalid, retry once, and show a friendly error after that.
4. Build the UI: an input form, a "Generate" button, a loading state, and nicely displayed results.
5. Add a "Regenerate" option, and save results to [DATABASE / LOCALSTORAGE] so users don't pay to generate the same thing twice.
6. Keep the API key on the server and add a simple rate limit.

SECTION 9

Simple Browser Games

Build fun, playable games that run right in the browser. No game engine needed to start.

9.1 Build a Complete Simple Game

Use this when you have a game idea and want something playable quickly. It's for creating a full, playable browser game from a concept, with a start screen, gameplay, scoring, and game over.

Fill in: [SKILL LEVEL], [GAME IDEA] (e.g. "a space shooter where you dodge asteroids and collect stars"), [GENRE / INSPIRATION] (e.g. "like Flappy Bird" / "endless runner" / "match-3" / "top-down arena"), [TECH] (e.g. "plain HTML5 Canvas + JavaScript in one file" / "Phaser 3" / "Kaboom.js" / "recommend the simplest"), [CONTROLS] (e.g. "arrow keys + space", "mouse click", "touch for mobile"), [ART STYLE] (e.g. "simple colored shapes for now" / "neon glow" / "pixel art").

Build a complete, playable browser game. I'm a [SKILL LEVEL], so keep the code organized and well commented.

Game concept: [GAME IDEA]

Genre / inspiration: [GENRE / INSPIRATION]

Tech: [TECH]

Controls: [CONTROLS]

Art style: [ART STYLE]

The game must include:

1. A start screen with the title, a "Play" button, and the controls explained.
2. The core gameplay loop running smoothly at 60fps, using requestAnimationFrame and delta time so the speed is the same on every computer.
3. A player, obstacles/enemies, collectibles, and collision detection.
4. A score and a high score that's saved between visits (localStorage).
5. Difficulty that slowly increases over time.
6. A game over screen with the score and a "Play again" button.
7. A pause option (P key or button).

Put all the tweakable numbers (speed, spawn rate, gravity, etc.) in one CONFIG object at the top, so I can tune the game feel easily. Tell me exactly how to run it.

 **Tips to customize:** Start with shapes, and once the game is fun, use Prompt 9.3 to add juice and art. Fun first, pretty later!

9.2 Add a Feature to My Game

Use this when your basic game works and you want to make it deeper or more fun. It's for adding new mechanics like power-ups, levels, enemies, or upgrades without breaking the game.

Fill in: [GAME DESCRIPTION], [GAME TECH], [FEATURE TO ADD] (e.g. "3 power-ups: shield, double score, slow-mo", "a level system with 5 levels", "a boss every 60 seconds", "a shop to buy upgrades with coins"), [HOW IT SHOULD FEEL] (e.g. "rewarding but rare", "challenging but fair"), [PASTE OR TAG GAME FILES].

Add a new feature to my browser game without breaking what already works.

Game: [GAME DESCRIPTION]

Tech: [GAME TECH]

Feature to add: [FEATURE TO ADD]

How it should feel: [HOW IT SHOULD FEEL]

[PASTE OR TAG GAME FILES]

Please:

1. Briefly explain how you'll add it and which parts of the code will change.
2. Build it in a modular way (e.g. separate objects or classes), so it's easy to add more later.
3. Add all the new tuning numbers to the CONFIG object.
4. Add clear visual and audio feedback hooks for when the feature triggers (even placeholders).
5. Update the start screen / UI to show the new feature if needed (e.g. a power-up timer bar).
6. Tell me what to test, and suggest the starting values for balancing.

9.3 Make My Game Feel Juicy

Use this when your game works but feels flat or boring to play. It's for adding "game feel" (screen shake, particles, sound, and animation), the small touches that make simple games feel great.

Fill in: [GAME DESCRIPTION], [GAME TECH], [MOOD] (e.g. "arcade and punchy" / "calm and cozy" / "spooky"), [PASTE OR TAG GAME FILES].

My game works, but it feels flat. Add "game juice" to make it feel satisfying and alive.

Game: [GAME DESCRIPTION]

Tech: [GAME TECH]

Mood: [MOOD]

[PASTE OR TAG GAME FILES]

Please add, with each effect easy to toggle or tune in CONFIG:

1. Screen shake on hits and explosions (subtle, and short).

2. Particle effects for collecting items, destroying enemies, and the player dying.
3. Squash-and-stretch or scale "pop" animations on jumps and pickups.
4. Hit flash (a brief white or red flash on damage) and a short freeze-frame on big hits.
5. Floating score numbers (+10) where points are earned.
6. Simple sound effects generated in code with the Web Audio API (no files needed), or tell me where to find free sound files and how to load them. Add a mute button.
7. Smooth transitions between the start, play, and game over screens.
8. A simple background effect that fits the mood (e.g. parallax stars, drifting clouds).

Keep performance smooth. Reuse particles instead of creating thousands of new objects.

9.4 Make My Game Work on Phones

Use this when your game only works with a keyboard on desktop, and you want to share it with friends on their phones. It's for adding touch controls and responsive scaling so your game plays well on mobile.

Fill in: [GAME DESCRIPTION], [GAME TECH], [CURRENT CONTROLS] (e.g. "arrow keys to move, space to shoot"), [PREFERRED MOBILE CONTROLS] (e.g. "on-screen joystick + fire button" / "tap to jump" / "swipe" / "recommend the best fit"), [PORTRAIT / LANDSCAPE / EITHER], [PASTE OR TAG GAME FILES].

Make my browser game fully playable on mobile phones as well as desktop.

Game: [GAME DESCRIPTION]

Tech: [GAME TECH]

Current controls: [CURRENT CONTROLS]

Preferred mobile controls: [PREFERRED MOBILE CONTROLS]

Orientation: [PORTRAIT / LANDSCAPE / EITHER]

[PASTE OR TAG GAME FILES]

Please:

1. Recommend the best touch control scheme for this game and explain why.
2. Add touch controls that only appear on touch devices. Keep the keyboard working on desktop.
3. Make the canvas scale to fit any screen while keeping the aspect ratio and staying sharp on high-DPI screens.
4. Prevent the usual mobile problems: page scrolling or zooming while playing, double-tap zoom, and the long-press menu.
5. Show a "rotate your device" message if the orientation is wrong.
6. Auto-pause when the player switches apps or tabs.
7. Make sure the game still runs smoothly on older phones, and suggest settings to lower if it's laggy.

8. Make it installable as a simple PWA (add to home screen), if that's easy for my setup.

Bonus: The "Magic Phrases" Cheat Sheet

Add these to the end of any prompt to improve the results:

- **"Plan first and wait for my go-ahead before writing code."** Stops the AI rushing off in the wrong direction.
- **"Only change what's needed. Don't rewrite unrelated code."** Protects your working features.
- **"Explain it like I'm a beginner."** Gets you plain-English explanations.
- **"If you're unsure, ask me instead of guessing."** Cuts down on made-up answers.
- **"List every file you changed."** Makes changes easy to review and undo.
- **"Give me a test checklist."** Makes sure you actually check that it works.
- **"What are 3 ways this could break?"** Catches problems early.
- **"Start fresh: here's my project brief."** For when a long chat gets confused (see Prompt 1.3).

You're Ready to Build

That's the whole Vibe Code Starter Pack: **33 prompts across 9 sections**, covering everything from the first idea to a live app with AI features, plus browser games.

Every expert was a beginner who just kept building. Pick a prompt, fill in the blanks, and ship something this week.

Happy vibe coding!