🧠 AI-Powered App Workshop Guide

Build a front-end app prototype using ChatGPT + Vercel's v0.dev

---

Welcome!

Tools we'll use:

- ChatGPT (we'll use GPT-4o)
- Vercel's v0.dev
    - *GitHub is optional for this workshop*

Before we begin:

- Log in or sign up for free versions of both tools
- Get comfy, grab a drink—this is a hands-on, beginner-friendly session

Housekeeping + What to Expect

- Mics will stay on mute so we can keep a smooth pace through the workshop.
- Drop questions in the chat any time—we'll respond there or during a break if possible.
- We've got a lot to cover, so we'll keep things focused and hands-on.

Staying Current

- AI tools like ChatGPT and v0.dev are evolving fast.
- Don't be surprised if the UI or steps change over time.
- We recommend staying current with their latest docs, changelogs, or community threads.

Tip: If you're watching this later or reviewing the materials, always double-check what version you're using—new features or interface changes are common!

What We'll Do Today

Step-by-step flow:

1. Write a Product Requirements Doc (PRD) with ChatGPT
2. Use v0.dev to generate your app's front-end UI
3. Polish the UI using natural language prompts
4. (Optional) Connect a database to make it functional
5. Explore how your UI can evolve into an intelligent agent

ChatGPT Primer

Use GPT-4o (free or Pro)
Best for UI prompting, fast, and contextually sharp.

- See Appendix for more insight on Chat GPT models.

Features to know:

- **Projects** – Save your ideas + files together
- **Custom GPTs** – Create tools in your tone and voice
- **Tasks / Operator (Pro)** – See how the model "thinks" step-by-step

---

Vercel Overview

Key features:

Tool	What it does	Why it matters
v0.dev	Turns prompts into live UIs	Instant layout + styling
Preview Deploy	Shareable URLs	Easy testing + feedback
Forking	Save versions like "Save As"	Safe exploration
Vercel + Neon	Optional DB setup	Make it persistent

UI libraries used:

- shadcn/ui
- Tailwind CSS
- Radix UI

Write Your PRD in ChatGPT

Prompt to use:

```
Turn the following idea into a one-page PRD.
Include: 1. Problem 2. Target persona 3. Success metric
4. Happy path 5. Out-of-scope 6. What we can fake

Idea: “A recipe tracker that stores recipe URLs you give it, auto-tags by season & ingredients, lets you rate, and resurfaces in-season dishes. The app would also let you search by ingredient/season/dish.”
```

After generating it:

- Clean up hallucinations
- Cut extra features
- Clarify the success metric
- Highlight what we can fake (like search or rating)

After you Generate Your PRD Add This Line to the Top of Your Prompt for Vercel

```
Build a mobile-first front-end prototype using static props or mock JSON. No backend or real data calls—this is for design approval only.
```

This keeps v0’s focus on **UI only**. Save backend setup for later.

So it ends up looking like this (full prompt ready for Vercel):

Prompt:

Build a mobile-first front-end prototype using static props or mock JSON. No backend or real data calls—this is for design approval only.
Design a recipe tracker web app that lets users add and save recipes either by pasting a URL or manually entering details. Prioritize a clean, responsive UI with a fast, frictionless experience.
Core features:

- Homepage/dashboard showing saved and seasonal recipes.
- Sticky or floating "Add Recipe" button on mobile.
    - Opens two options: "Paste URL" or "Enter Manually."
- Paste URL flow:
    - Simple input form.
    - Simulate a loading state after submit.
    - Auto-fill fields: recipe name, ingredients (list), image, instructions.
    - If it fails, show a friendly message and the manual form.

Manual entry form:

- Recipe Name
- Ingredients (multi-line or tag-style)
- Instructions (multi-line)
- Optional image upload or link
- Season & Dish Type (dropdowns or buttons)

Saved recipes view:

- Recipe cards: image, name, rating, tags
- Filter/search bar (ingredient, season, dish type)
- Highlight "In Season Now" recipes

Extras:

- 1–5 star rating option
- Static tag mapping for seasonal ingredients
- Tooltip help for vague terms (e.g., "mixed vegetables")
- No login required; simulate all data
- UI should feel modern, friendly, fast, and touch-optimized

```
Okay to fake:

Saving recipes to local state only—no real database. Just simulate the action and visually confirm the recipe appears in the list.


Success metric:

A user should be able to sort through their recipes and find what they want using filters in no longer than 60 seconds.
```

Generate Your Base UI

Steps:

1. Go to v0.dev
2. Click **New Project**
3. Search by typing "**Blank Template"**
4. Paste your full prompt (include the line above + your PRD)
5. Choose model version `v0-1.5-md`
6. Let it build your layout (~2 min)

Save Your Work

- Click **Publish** → **Deploy to Vercel**
- Get a live, shareable URL
- Think of it like saving your work in Figma or Docs

Keep Iterating

To reopen or fork:

- Go to v0.dev

- Click **My Projects**
- Choose your project
- Hit **Fork** to create a copy you can safely experiment on

Test Your App

Look for:

- Missing features
- Extra features you didn't ask for
- UI bugs (unresponsive, weird spacing)
- Modals behaving oddly

Fixing flow:

- Hover → click **Design** → type your fix prompt
- OR type into the main prompt chat
- If stuck: ask v0 for a "complete audit" → paste it into ChatGPT → ask for a fix

Prompting tips:

- Vercel AI uses a large GPT-style context window (up to ~128k tokens), but complex UIs can still overflow it. Keep prompts scoped—one component or feature at a time. Once context degrades, fork the chat or start a new one to stay fast and reliable.

Examples of Fix Prompts

```
Make the Add Recipe button fixed at the bottom.
Make this card tappable but allow inner links.
Remove social sharing features.
Group tags into pill-style components.
```

Save a Snapshot Before Styling

- Hit **Fork**
- Name it something like `recipe-ui-polished`
- This is your “clean copy” before trying big layout changes

Polish Your UI

Prompt ideas:

- “Use soft drop shadows and generous spacing”
- “Style this like a wellness app”
- “Add mock recipe images”
- “Use soft greens as the accent color”
- “Make it responsive and mobile-first”

Inspo:

- https://godly.website
- https://dribbble.com

Examples of UI Tweaks

- Lightened background
- Changed primary color to sage green
- Increased font size for accessibility
- Rounded recipe cards
- Hid stars if not rated
- Improved spacing between sections

Fork & Deploy Your Final UI

- Name your new fork something like `recipe-ready-for-db`
- Hit **Publish** → **Deploy to Vercel**

- You now have a **final live version** of your polished front end

---

Adding a Database (Optional)

When you're ready:

1. Fork your app
2. Tell v0 what you want to persist ("Save recipes, tags, ratings")
3. v0 will prompt you to connect a Neon DB (free)
4. Follow setup flow + test
5. Fix any server errors by clicking them → v0 will help debug

This step uses more credits and is where bugs usually start—save it for last.

Your Final Prototype

Features:

- Responsive layout
- Recipe cards
- Add Recipe modal
- Tag and rating UX
- Deployed and clickable with no code written

Sharing a v0 Prototype (with Chat History):

Useful for when you want to show how the UI was built—great for teaching, feedback, or collaboration. Anyone with the link can see your UI and the full prompt/chat history used to generate it.

- Open your project in v0.dev → Click the **Share** button (top-right corner).
- Set visibility to **Unlisted** or **Everyone in the team**.
- Copy the link and share it.

Sharing a v0 Prototype (without Chat History):

Use this when you want to show just the UI—no prompts, no behind-the-scenes. Ideal for user testing, demos, or executive reviews. This gives you a polished, stand-alone front-end—clean and professional. Great for sharing with non-technical audiences.

- Fork your project (to create a clean copy).
- Edit or strip any visible chat content if needed.
- Go to **Export → Deploy to Vercel**
- Share the hosted link (e.g. `your-prototype.vercel.app`).

Example from Workshop:
https://v0-mobile-recipe-tracker.vercel.app/

Agent Graduation Roadmap

Stage	Capability	What It Enables
v1	Smart Tagger	Auto-labels ingredients + seasons
v2	Ingredient Scaler	Adjusts recipes for 2 or 6 servings
v4	Grocery Export	Sends shopping list to Instacart or Tasks

From Prototype to Agent

Next up: Cody will demo a **different agent**—not from this app, but to show how your prototype could evolve into something intelligent and helpful.

Watch her flow, take notes, and start imagining where you could go from here.

📎 Appendix

A. Credit Tips

- Free = 5 credits/day
- v0 generations = 1 credit
- Fork instead of regenerating

B. Prompting Tips

- Clear, plain English wins
- One intent per prompt
- “Make this look like Substack” works!
- Use emotional language for the UI
- Use screenshots if you have an exact UI you want to match

C. Common Bugs

- Layout broken? Prompt “fix responsive layout”
- Modal janky? Prompt “fix modal close logic”
- No save? Add a database.

D. Design Resources

- godly.website
- mobbin.com
- dribbble.com
- land-book.com
-

E. Choosing the Right ChatGPT Model

- **GPT-4o:** Fast, smart, and great at handling design + product prompts. Works well for UI feedback, PRD generation, and natural-language brainstorming. This is what we’re using today—and it’s free for most tasks.

- **GPT-4:** Stronger reasoning and context retention, great for debugging or helping with trickier prompts. It’s slower than 4o and only available on the Pro plan, but can be useful

when you want the AI to “think deeper.”

- **GPT-4 Mini / Mini High:** Faster and cheaper alternatives to full GPT-4 if you're on Pro. These are good for repeated structured tasks (like generating multiple PRDs or audits) where speed matters more than nuance.

- **GPT-3.5:** Free and quick, but struggles with complex UX, design prompts, or multi-step planning. Okay for basic writing tasks or early brainstorming, but not recommended for UI-heavy workflows.

F. Glossary

- PRD = Product Requirements Doc (What we want to build)
- Prompt = Instruction to the AI
- Fork = “Save As”
- Deploy = “Publish”
- Frontend = UI
- Backend = Logic/data (optional today)