How I Used AI to Build the Cafe Fausse Website

Project: Cafe Fausse Restaurant Website
Course: Web Application & Interface Design
Date: 09 February 2026

1. Overview

For the Cafe Fausse website project, I used a couple of AI tools to help with the development process. They were useful for speeding things up, checking my code, and even giving me some design ideas. This paper explains which tools I used, what I used them for, and how they helped shape the final website.

2. Tools Used

I used two main AI tools:

Tool	Type	What I Used It For
Manus AI	AI coding assistant	Helping with everything from planning the project structure to writing code and fixing bugs.
GitHub Copilot	AI code completion	Getting in-editor suggestions and autocompleting code while I was writing it.

3. How I used the AI

3.1 Planning the Project

At the beginning of the project, I used AI to turn the project requirements into a solid plan. It helped me figure out how to structure the project, which libraries to use, and how to lay out the database. The AI gave me some good ideas for organizing the Flask backend, what information to put in the PostgreSQL database tables, and how the React front end should talk to the back end.

What the AI did: It suggested a folder structure for the project, recommended using Flask-CORS to handle requests between the front and back ends, and helped me design the three database tables (for customers, reservations, and newsletter subscribers) based on the assignment.

What I did: I looked over the AI's suggestions, tweaked them to fit the project's specific needs, and made the final calls on the technologies I would use.

3.2 Building the Backend and Database

The AI wrote the first draft of the PostgreSQL database structure, including the right data types and relationships between the tables. It also wrote the Flask code for the reservation system, which included the logic for checking if a table was free and then picking a random table for a specific time.

What the AI did: It generated the SQL code to create the tables, wrote the Flask functions for handling reservations and newsletter sign-ups, and implemented the random table assignment using Python's random.choice().

What I did: I tested everything by hand to make sure it worked, checked that the database was storing information correctly, and fixed a problem with column names when I switched from MySQL to PostgreSQL. I also made sure that the number of available tables went down after each booking.

3.3 Developing the React Frontend

The AI helped me create the React components for all five pages of the website: Home, Menu, Reservations, About, and Gallery. It wrote the JSX code, handled the state with React hooks, and made the calls to the Flask API. It also built the responsive navigation bar and footer.

What the AI did: It generated the starting code for each page, built the multi-step reservation form with a date picker and form validation, created the logic for filtering the menu, and built the photo gallery with filters.

What I did: I went through all the code the AI generated, adjusted the layout to match my design, and tested the reservation process from start to finish in my browser. I also fixed a few problems that came up during testing, like setting up the API proxy and dealing with conflicting port numbers.

3.4 UI/UX Design and Styling

The AI gave me the idea for the color scheme (burgundy, gold, and cream) and the fonts (Cormorant Garamond for headings, Montserrat for body text) to match the restaurant's fancy theme. It also generated the Tailwind CSS configuration and wrote the responsive styles.

What the AI did: It suggested the design theme, generated the CSS for the color scheme, wrote the responsive layouts using Tailwind, and created the hover effects for the interactive parts of the site.

What I did: I picked the final colors, chose the photos for the site, and adjusted the spacing and font sizes based on how everything looked in the browser. I also tested the site on different screen sizes to make sure it was responsive.

3.5 Writing the Documentation

The AI wrote the first drafts of the README file and this summary. It set up the README with instructions for getting the project running, a description of the project, and technical details.

What I did: I edited the documentation to make sure it was accurate, added some notes about my own setup process, and followed the instructions myself on my Mac to make sure they worked.

4. Code Contribution

This table shows roughly how much of the code was written by the AI versus how much I wrote or changed myself.

Component	**AI-Generated**	**Manually Written/Modified**
Database schema (SQL)	80%	20% (adjustments for PostgreSQL)
Flask backend (app.py)	75%	25% (debugging, fixing PostgreSQL column names, port configuration)
React components (JSX)	70%	30% (layout adjustments, design tweaks, fixing bugs)
CSS/Styling	65%	35% (fine-tuning colors, spacing, and responsive design)
Documentation	60%	40% (editing, checking for accuracy, adding personal notes)
Project configuration	50%	50% (setting up the environment, managing dependencies, local setup)

5. What I Learned

Working with AI coding tools taught me a few things. First, AI is great at writing boilerplate code and turning requirements into a first draft. The initial Flask and React code worked almost immediately, which saved me a lot of setup time.

Second, you still have to check the AI's code carefully. When I switched from MySQL to PostgreSQL for local development, the AI had used camelCase for column names in its SQL queries, but my PostgreSQL tables used snake_case. I had to read the error logs to figure this out and then recreate the tables with the correct names.

Third, AI is more of a partner than a replacement for knowing how to code. I needed to understand how Flask routing, React state, and PostgreSQL work to fix the problems that came up. Overall, the AI tools made the development process much faster, but I still had to use my own judgment for design, debugging, and making sure everything worked as it should.