# **Next.js E-Commerce Storefront Scaffolding Exercise**

## **Scenario:**

You will create a simple e-commerce storefront using Next.js that displays products, allows users to add items to a shopping cart, and updates pricing dynamically. This project uses static JSON for product data, plain CSS for styling, and will be deployed to Vercel.

## **Technologies:**

* Next.js
* CSS (plain, no external framework)
* GitHub Copilot (for inline suggestions and Copilot Chat commands)
* Vercel (deployment)

## **Key Objectives:**

* Scaffold a Next.js project quickly using Copilot inline suggestions and Copilot Chat.
* Use commands like **@workspace, /start, /fix, /explain, #file, #git, and #terminalLastCommand** throughout the workflow.
* Build product listing, cart management, and page routing with minimal manual coding.
* Deploy your application on Vercel using Copilot to assist in terminal and Git commands.

|  |
| --- |
| **Note**: Below prompts are provided solely as examples of effective prompt construction. As participants, we must engage in an iterative process to refine these examples and develop the optimal prompt for the task at hand. |

## **Step 1: Project Setup**

**Objective:**Generate the entire Next.js project workspace and directory structure exclusively using GitHub Copilot Chat.

**Detailed Prompts**

**Initialize the Project Workspace**Use a detailed prompt that instructs Copilot Chat to create the entire project without manual commands. For example, enter:

|  |
| --- |
| @workspace: Initialize a new Next.js project named "next-ecommerce-store" using create-next-app. Ensure that the entire project is scaffolded and opened in VS Code automatically. |

|  |
| --- |
| **Note**: Above prompt would help you to generate the required commands. Once commands are created, hover on the commands and click on **Insert into Terminal** option |

1. **Inspect the Workspace Structure**Once the project is created, verify the structure with:

|  |
| --- |
| @workspace: Provide a detailed overview of my project structure, including key directories such as pages/, public/, styles/, and any other generated folders. |

|  |
| --- |
| **Note**: Above prompt would help you to generate the required commands. Once commands are created, hover on the commands and click on **Insert into Terminal** option |

1. **Refine the Directory Structure**If additional folders are needed, instruct Copilot Chat to add them. For instance:

|  |
| --- |
| /fix: Update the project structure to include a dedicated "components" folder for custom React components, a "public/assets" folder for additional static files, and a "docs" folder for documentation. |

1. **Confirm the Final Structure**Verify the updated structure:

|  |
| --- |
| @workspace: List all directories in my project. |

**Challenges for Project Setup**

* **Challenge 1:** Determine if any extra directories (e.g., configuration, scripts) would benefit your project. Use iterative Copilot Chat prompts to refine the workspace until it is perfectly organized for a scalable, multi-module Next.js project.

**Step 2: Initialize the Next.js Project Configuration**

**Objective:**Generate and refine all necessary configuration files (e.g., package.json, next.config.js) solely via GitHub Copilot Chat.

**Detailed Prompts**

**Create the Core package.json File**Use a detailed command that outlines all required settings:

|  |
| --- |
| /start: Generate a package.json file for the "next-ecommerce-store" project with the following details: - Name: next-ecommerce-store - Version: 1.0.0 - Scripts: Include "dev", "build", and "start" scripts as specified by Next.js. - Dependencies: Include next, react, and react-dom with the latest stable versions. |

|  |
| --- |
| **Note**: Above prompt would help you to generate the required commands. Once commands are created, hover on the commands and click on **Insert into Terminal** option |

1. **Enhance the package.json Configuration**After reviewing the initial output, refine it by instructing:

|  |
| --- |
| /fix: Enhance the package.json by adding custom environment variables, additional scripts for linting and testing, and any other settings that adhere to industry best practices. |

1. **Generate Additional Config Files**Create a custom Next.js configuration file by issuing:

|  |
| --- |
| /start: Generate a next.config.js file that optimizes images and sets up custom redirects, tailored for a production-ready Next.js storefront. |

1. **Request Explanations for Key Settings**To further understand the configuration, ask:

|  |
| --- |
| /explain: Provide a detailed explanation for each key setting in my package.json and next.config.js files. |

**Challenges for Configuration Files**

* **Challenge 1:** Modify the configuration files to support multiple environments (development, staging, production) using environment variables and custom scripts. Refine your prompts until the configuration is both comprehensive and flexible.
* **Challenge 2:** Experiment with adding advanced features like TypeScript support or progressive web app (PWA) configurations and refine the settings accordingly.

**Step 3: Create the Core Pages**

**Objective:**Generate the essential pages for your storefront (Home, About, Contact) exclusively through GitHub Copilot Chat commands.

**Detailed Prompts**

**Generate the Home Page**Use a detailed command such as:

|  |
| --- |
| /start: Generate a Next.js page component named Home in the pages/ directory that displays a welcome message and showcases featured products. |

|  |
| --- |
| **Note**: Above prompt would help you to generate the required commands. Once commands are created, hover on the commands and click on **Insert into Terminal** option |

1. **Generate the About Page**Instruct Copilot Chat with:

|  |
| --- |
| /start: Generate a Next.js page component named About in the pages/ directory that renders an <h1> element with "About Our Store" and includes a brief description of the company. |

|  |
| --- |
| **Note**: Above prompt would help you to generate the required commands. Once commands are created, hover on the commands and click on **Insert into Terminal** option |

1. **Generate the Contact Page**Create the Contact page with:

|  |
| --- |
| /start: Generate a Next.js page component named Contact in the pages/ directory that renders an <h1> element "Contact Us" and includes a form with fields for name, email, and message. |

|  |
| --- |
| **Note**: Above prompt would help you to generate the required commands. Once commands are created, hover on the commands and click on **Insert into Terminal** option |

1. **Refine and Verify Pages**If additional content or styling is needed, update your prompt accordingly (e.g., add further instructions for layout or content structure).

**Challenges for Core Pages**

* **Challenge 1:** Enhance the Contact page by adding form validation logic and dynamic error messages, all generated via Copilot Chat.
* **Challenge 2:** Refactor your pages to incorporate a shared layout (e.g., a common header and footer) by instructing Copilot Chat to extract and generate reusable components.

**Step 4: Build the Product Listing Page**

**Objective:**Create a product listing page that fetches product data from a JSON file and displays it in a grid layout, using only GitHub Copilot Chat.

**Detailed Prompts**

**Generate the Products JSON File**Instruct Copilot Chat to create a JSON file in the public folder:

|  |
| --- |
| /start: Generate a JSON array of 5 sample product objects in a file named products.json located in the public/ directory. Each product object must include the properties: id, name, price, and image. |

|  |
| --- |
| **Note**: Above prompt would help you to generate the required commands. Once commands are created, hover on the commands and click on **Insert into Terminal** option |

1. **Generate the Products Page Component**Use a detailed command to create the page:

|  |
| --- |
| /start: Generate a Next.js page component named Products in the pages/ directory. This component should fetch product data from /products.json using React hooks (e.g., useEffect) and display the products in a grid layout. |

|  |
| --- |
| **Note**: Above prompt would help you to generate the required commands. Once commands are created, hover on the commands and click on **Insert into Terminal** option |

1. **Iterate to Enhance Functionality**If you need to add additional features like filtering or sorting, update your prompt:

|  |
| --- |
| /fix: Enhance the Products page component to include functionality for filtering products by category. |

**Challenges for the Product Listing Page**

* **Challenge 1:** Add a pagination feature that limits the number of products displayed per page and allows users to navigate between pages.
* **Challenge 2:** Integrate an "Add to Cart" button into each product card, and design prompts to update the cart state dynamically.

**Step 5: Implement a Simple Cart System**

**Objective:**Develop a cart component and integrate persistent cart state management exclusively via GitHub Copilot Chat.

**Detailed Prompts  
Generate the Cart Component**Instruct Copilot Chat with:

|  |
| --- |
| /start: Generate a React component named Cart in the components/ directory. The component should display a list of added products and include a "Clear Cart" button. |

|  |
| --- |
| **Note**: Above prompt would help you to generate the required commands. Once commands are created, hover on the commands and click on **Insert into Terminal** option |

1. **Persist Cart State Using localStorage**Refine the cart functionality by updating state persistence:

|  |
| --- |
| /fix: Update the cart state logic within the Cart component to persist cart items in localStorage, ensuring they remain after a page reload. |

**Challenges for the Cart System**

* **Challenge 1:** Implement functionality to adjust the quantity of items in the cart (e.g., increment/decrement buttons) using detailed prompts to update the cart state.
* **Challenge 2:** Design a feature to remove individual items from the cart and verify that these changes persist using localStorage.

**Step 6: Apply CSS Styling**

**Objective:**Style your Next.js storefront entirely via GitHub Copilot Chat using plain CSS.

**Detailed Prompts  
Generate Global CSS**Command Copilot Chat to create a global CSS file:

|  |
| --- |
| /start: Generate a global CSS file named global.css with base styles for the body, headings, links, and general typography. Ensure that this file is imported in pages/\_app.js. |

|  |
| --- |
| **Note**: Above prompt would help you to generate the required commands. Once commands are created, hover on the commands and click on **Insert into Terminal** option |

1. **Style Specific Components**Use a prompt to generate CSS for the product grid and product cards:

|  |
| --- |
| /start: Generate CSS rules for a product grid layout. The rules should define a flex container that wraps, has gaps, and centers items, and include styles for product cards with borders, padding, border-radius, and a hover effect that scales the card. |

1. **Enhance Layout Responsiveness**To ensure responsiveness, update your CSS with:

|  |
| --- |
| /fix: Add media queries to the global CSS file to ensure that the product grid layout adapts smoothly to different screen sizes. |

**Challenges for CSS Styling**

* **Challenge 1:** Add subtle animations (e.g., smooth transitions on hover) to enhance user experience. Use Copilot Chat to generate and refine these CSS rules.
* **Challenge 2:** Ensure the entire layout is fully responsive, and test your design on multiple device sizes using iterative CSS prompts until the design is seamless.

**Step 7: Deploy the Application to Vercel**

**Objective:**  
Deploy your Next.js storefront to Vercel using only GitHub Copilot Chat commands.

**Detailed Prompts**

**Install Vercel CLI via Chat**Instruct Copilot Chat to install the Vercel CLI without manual intervention:

|  |
| --- |
| @terminal: Execute "npm install -g vercel" to install the Vercel CLI. |

1. **Deploy the Project**Deploy your application with a command:

|  |
| --- |
| @terminal: Execute "vercel" to deploy the application. Follow the on-screen prompts provided by Vercel through GitHub Copilot Chat. |

1. **Verify the Deployment**Once deployed, verify the outcome:

|  |
| --- |
| #terminalLastCommand: Explain the output of my last terminal command and confirm if the deployment was successful, including the live URL. |

**Challenges for Deployment**

* **Challenge 1:** Add a new page (for example, a Contact page) via GitHub Copilot Chat, then generate a descriptive commit message using a chat command (e.g., #git), and redeploy the application. Refine your workflow until deployment is fully automated and integrated.
* **Challenge 2:** Experiment with automating deployment using GitHub Actions. Generate a workflow file through Copilot Chat that builds and deploys your Next.js app to Vercel on every push to the main branch.

**Additional Advanced Challenges**

* **Implement Update Functionality:**Extend your API by generating a new API route (using Copilot Chat) that provides a PUT endpoint to simulate updating product details. Refine your commands until the endpoint behaves as expected.
* **Integrate a Global Error Boundary:**Use Copilot Chat to generate a custom React error boundary component that catches errors in the component tree and displays a fallback UI. Iterate until the component gracefully handles exceptions.
* **Refactor State Management with Context:**Move your cart state management into a React Context for global access. Generate detailed prompts to refactor your components and create a custom hook for managing cart data.
* **Automate Deployment with GitHub Actions:**Generate a GitHub Actions workflow file using detailed prompts that automatically builds and deploys your Next.js application to Vercel on every push. Refine the workflow until it meets your CI/CD requirements.

## **Conclusion**

In this exercise, you successfully built a Next.js e-commerce storefront **from scratch using GitHub Copilot** for assistance. You learned to scaffold a project quickly with Copilot’s help and created pages and components with minimal manual coding by leveraging AI suggestions.

By combining Next.js’s capabilities with Copilot’s AI-assisted development, you dramatically sped up the development process.

***Happy coding!***