Marketplace Hackathon: Problems, Solutions, and Implementations

Problem 1: Sanity Client Error

Error: "Expected '}' following object body." Cause: Verified there were no missing brackets in the code, but the error persisted.

Solution:

- Switched from groq to defineQuery for defining queries in the Sanity project.
- Reviewed the query syntax to ensure proper structure.
- Resolved the issue by matching the Sanity documentation examples for defineQuery usage.

Implementation:

- Integrated defineQuery into the e-commerce website project.
- Created optimized queries to fetch and display product data effectively from Sanity.

Day 3: Data Fetching Challenges

Problem 1: Fetching Data from Sanity

Error: Difficulty in properly fetching and displaying data using Sanity.

Solution:

- Ensured correct dataset and project ID were configured in the Sanity client setup.
- Used defineQuery to structure efficient queries.
- Implemented server-side rendering with Next.js to fetch data before page load.

Implementation:

- Rendered dynamic product data on the e-commerce website.
- Added fallback content for scenarios where data fetching fails.

Day 4: API Integration Refinements

Problem 1: Display Issues in E-Commerce Website

Error: Data fetched from Sanity was not displaying correctly on the frontend.

Solution:

- Verified the structure of the fetched data using debugging tools.
- Updated the React component structure to map over the data correctly.
- Added TypeScript type definitions to ensure compatibility with the fetched data.

Implementation:

- Enhanced the product display page with dynamic rendering.
- Improved code readability and maintainability with TypeScript.

Day 5: User Interface Improvements

Problem 1: Styling Bugs

Error: UI inconsistencies when integrating Tailwind CSS.

Solution:

- Debugged the Tailwind CSS setup in the project.
- Ensured that the tailwind.config.js file included all necessary paths for purging unused styles.
- Used responsive design utilities to fix layout issues on different screen sizes.

Implementation:

- Polished the UI for a seamless shopping experience.
- Achieved consistent styling across all pages of the website

Day 6: Advanced Features

Problem 1: Adding Cart Functionality

Challenge: Implementing a shopping cart system for the e-commerce website.

Solution:

- Created a global state management system using Context API to manage cart items.
- Added functionality to add, update, and remove items from the cart.
- Implemented a persistent cart using local storage to save user selections.

Implementation:

- Users can now add products to the cart and view the total price dynamically.
- Enhanced the checkout experience with clear cart summaries and user-friendly navigation.

Day 7: Deployment Issues

Problem 1: Vercel Deployment Error

Error: Build errors while deploying the Next.js project to Vercel.

Solution:

- Reviewed the build logs to identify missing environment variables.
- Ensured all required API keys and variables were added to Vercel's environment settings.
- Adjusted configuration in the next.config.js file to handle dynamic imports correctly.

Implementation:

- Successfully deployed the project on Vercel.
- Conducted post-deployment testing to verify functionality.