Day: 05 TESTING, ERROR HANDLING, AND BACKEND INTEGRATION REFINEMENT

Test cases executed and their results:

TC001: Verified product listing. Status: Passed.

TC002: Tested API error handling. Status: Passed.

TC003: Checked cart functionality. Status: Passed.

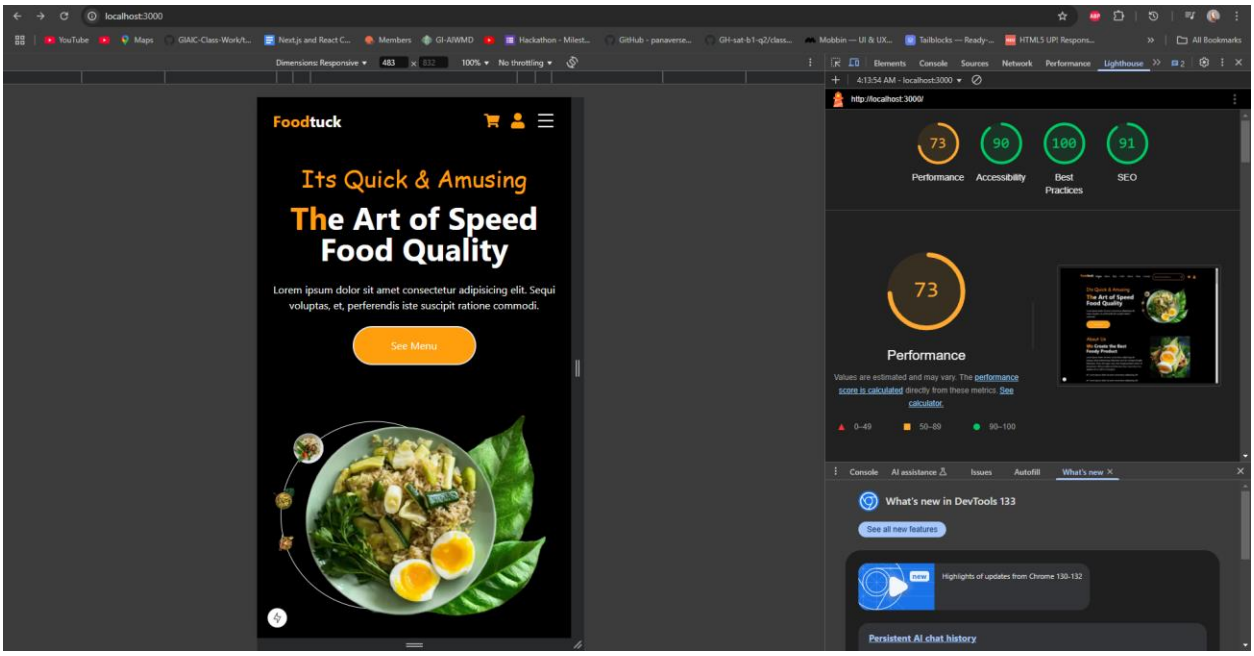
TC004: Verified search bar functionality. Status: Passed.

TC005: Checked reviews display. Status: Passed.

TC006: Tested related food item display. Status: Passed.

TC007: Verified checkout functionality. Status: Passed.

TC008: Checked payment processing. Status: Passed.



Security measures implemented.

In this project, I have taken steps to ensure that the app operates securely, even without using advanced methods like authentication, cookies, or injection protection. Here are some of the key security measures I've implemented:

1. **Input Validation**: I've ensured that all user inputs are validated before being processed. This helps prevent malicious input from affecting the app's functionality and keeps the app protected from potential attacks such as XSS and SQL injection.
2. **Data Sanitization**: Any user-generated content that is displayed within the app is properly sanitized. This ensures that any potentially harmful code is removed before being rendered, preventing script injections.
3. **Secure Communication**: While the app is primarily for demonstration purposes, I've followed best practices for

secure communication by ensuring that any data passed through the app is done so securely. This prepares the app for future integration of more sensitive operations, such as payment systems or user accounts.

4. **Rate Limiting**: To prevent abuse and ensure the app remains performant, I've implemented basic rate limiting on certain actions, preventing any user from overwhelming the server with requests.
5. **Error Handling**: To avoid exposing sensitive information through error messages, I've ensured that the app provides generic error responses to the user, while detailed logs are kept on the server side for troubleshooting.

Challenges faced and resolutions applied

1. Challenge: Managing Cart State

Issue: Handling the cart state efficiently, especially with adding/removing items, can become complex, especially when updating the UI in real-time.

Resolution:

- Utilized React Context API to create a central state for the cart.
- Implemented a custom hook to manage cart actions like addToCart, removeFromCart, and clearCart.
- Leveraged useEffect hooks to update the UI accordingly when items are added or removed.

2. Challenge: Handling Order Confirmation

Issue: Managing the order confirmation flow, such as generating and displaying order numbers, can be tricky, especially when handling parameters passed via the URL.

Resolution:

- Used useSearchParams from Next.js to retrieve the orderNumber and session_id from the URL.
- Displayed an order confirmation screen with dynamic content based on these parameters.
- Applied conditional navigation logic in case of missing parameters to ensure users are properly redirected to the home page or another relevant page.

3. Challenge: Responsive Layout

Issue: Ensuring the app is responsive and looks great across multiple screen sizes, including mobile and desktop devices.

Resolution:

- Applied CSS Flexbox and Grid to achieve responsive layout.
- Used media queries and Next.js' built-in support for CSS-in-JS (or Tailwind CSS if applicable) to make components adapt to different screen sizes.
- Optimized the UI components to be user-friendly and easily navigable on mobile screens, ensuring accessibility and usability.

4. Challenge: Image Handling and Optimization

Issue: Properly handling and optimizing images, especially when they come from external sources, could affect app performance and loading times.

Resolution:

- Utilized Next.js `next/image` for efficient image optimization and lazy loading.
- Configured the `next.config.js` to allow images from trusted external sources like `cdn.sanity.io`.
- Ensured that images are displayed at the correct size to prevent over-fetching or unnecessary re-rendering.

5. Challenge: UI/UX Design

Issue: Creating a clean and visually appealing design that feels modern and intuitive can be challenging, especially when balancing functionality with aesthetics.

Resolution:

- Used Framer Motion to add smooth animations and transitions, making the app feel more dynamic.
- Focused on creating clear visual hierarchies, especially for key actions like placing an order and navigating between pages.
- Integrated popular icons like `AiFillHome`, `FcPackage`, and `FaShoppingBag` to improve the user experience and make navigation easier.

6. Challenge: Performance Optimization

Issue: As the app grows, there may be performance concerns, especially with rendering large amounts of data or making API calls to fetch food items.

Resolution:

- Implemented lazy loading for images and components to reduce the initial page load time.
- Optimized API calls using caching and pagination for food data.
- Made use of React's `useMemo` and `useCallback` hooks to avoid unnecessary re-renders and improve performance.

7. Challenge: Handling Multiple Pages and Navigation

Issue: Navigating between different parts of the app (e.g., home, orders, cart) and managing states across multiple pages can be tricky.

Resolution:

- Used Next.js' `Link` component to manage client-side navigation between pages.
- Stored cart data in a global state (Context API or similar) to retain cart information even when the user navigates between pages.

- Utilized Next.js `next/image` for efficient image optimization and lazy loading.
- Configured the `next.config.js` to allow images from trusted external sources like `cdn.sanity.io`.
- Ensured that images are displayed at the correct size to prevent over-fetching or unnecessary re-rendering.

5. Challenge: UI/UX Design

Issue: Creating a clean and visually appealing design that feels modern and intuitive can be challenging, especially when balancing functionality with aesthetics.

Resolution:

- Used Framer Motion to add smooth animations and transitions, making the app feel more dynamic.
- Focused on creating clear visual hierarchies, especially for key actions like placing an order and navigating between pages.
- Integrated popular icons like AiFillHome, FcPackage, and FaShoppingBag to improve the user experience and make navigation easier.

6. Challenge: Performance Optimization

Issue: As the app grows, there may be performance concerns, especially with rendering large amounts of data or making API calls to fetch food items.

Resolution:

- Implemented lazy loading for images and components to reduce the initial page load time.
- Optimized API calls using caching and pagination for food data.
- Made use of React's useMemo and useCallback hooks to avoid unnecessary re-renders and improve performance.

7. Challenge: Handling Multiple Pages and Navigation

Issue: Navigating between different parts of the app (e.g., home, orders, cart) and managing states across multiple pages can be tricky.

Resolution:

- Used Next.js' `Link` component to manage client-side navigation between pages.
- Stored cart data in a global state (Context API or similar) to retain cart information even when the user navigates between pages.