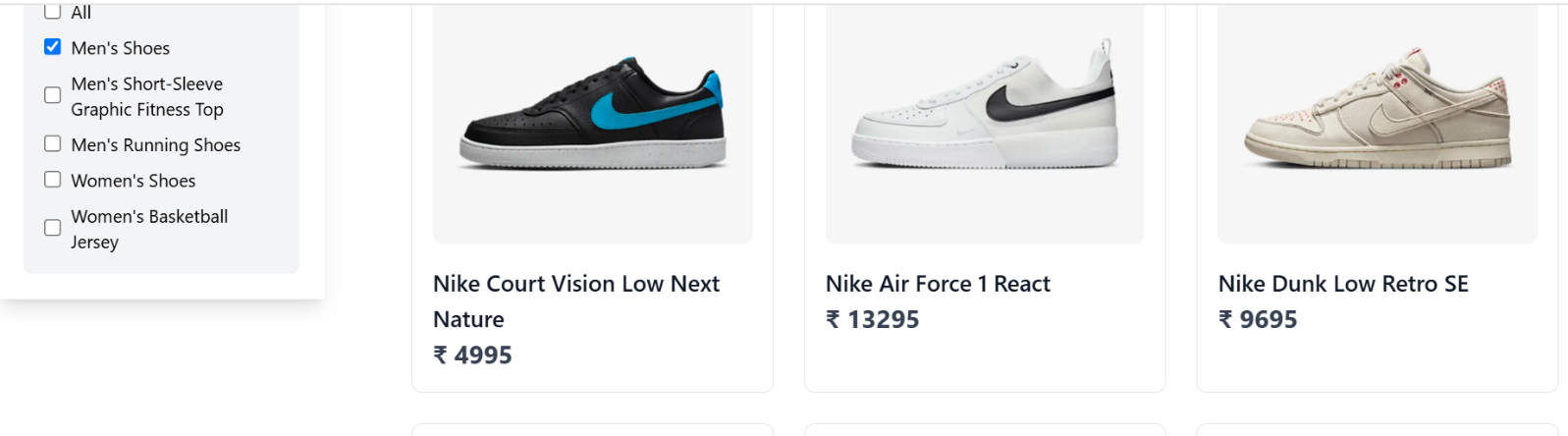
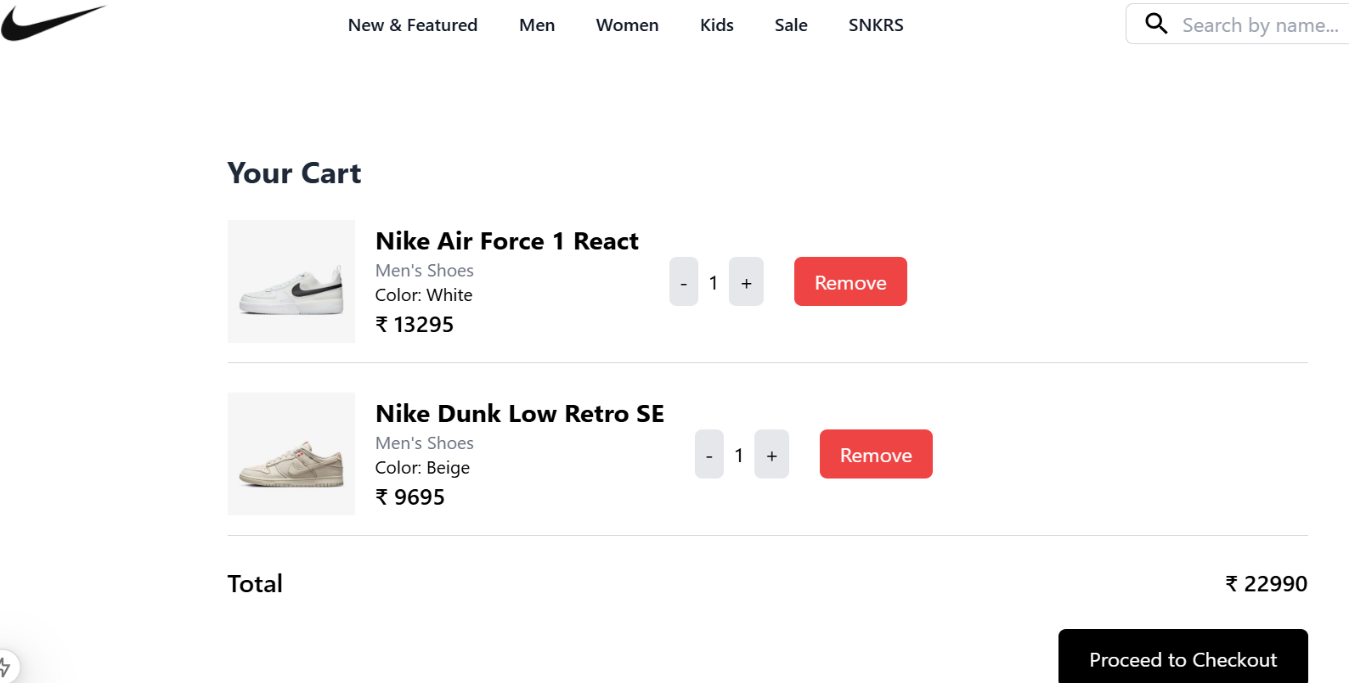
"Day 5 - Testing and Backend Refinement - [Nike store]"

Aqsa Iftikhar 00001663

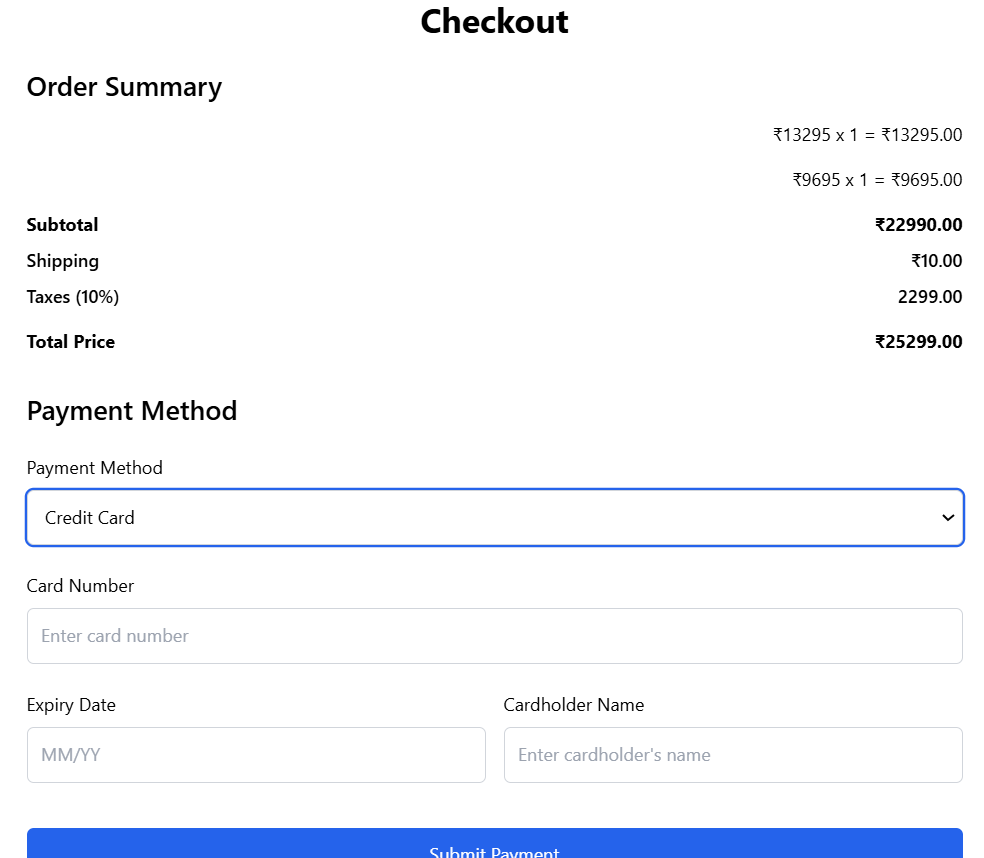
**1.Functional Deliverables:**

Functional and responsive components

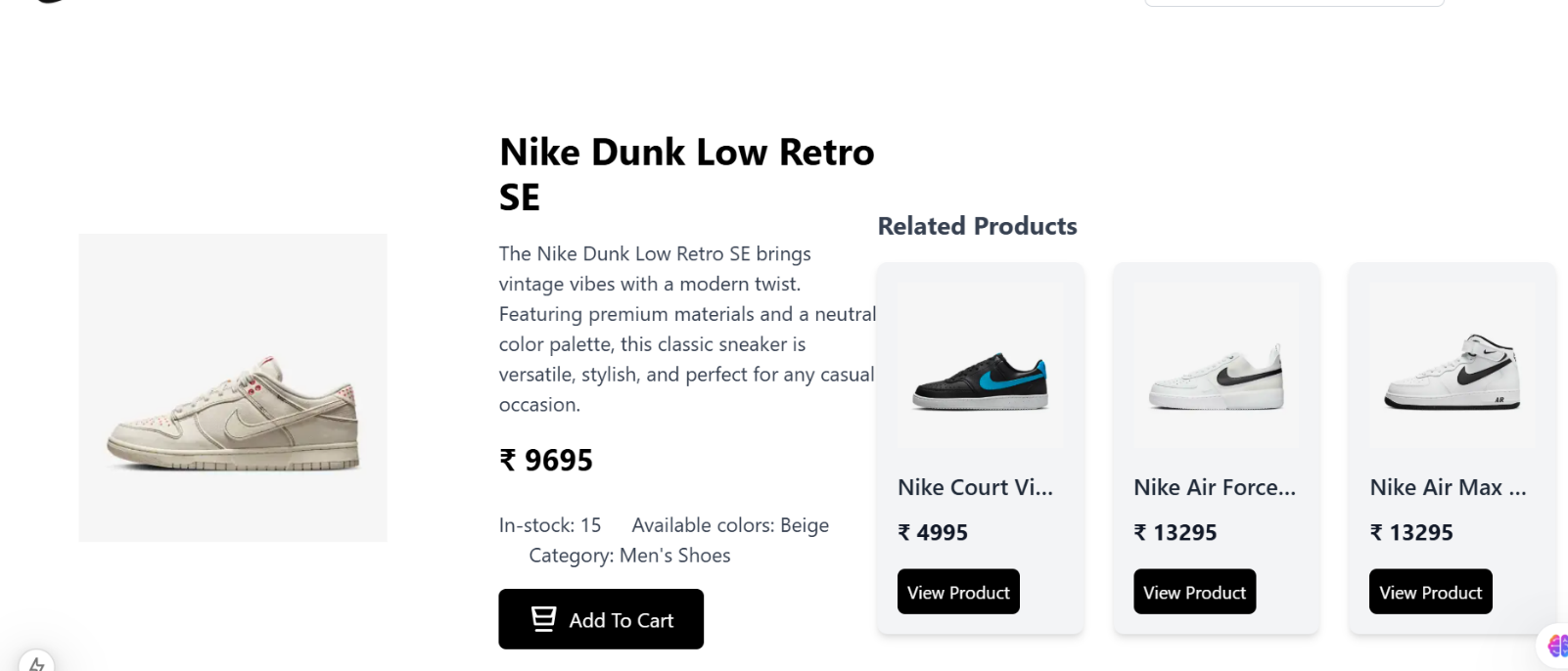
Filtered categories:

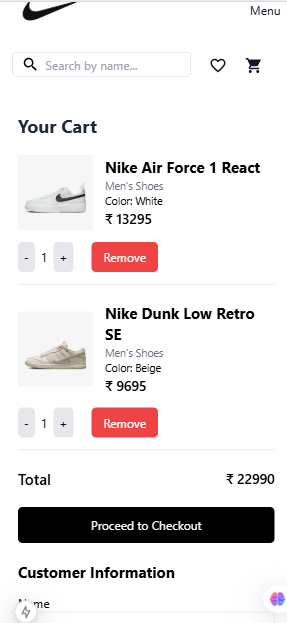


Functional cart

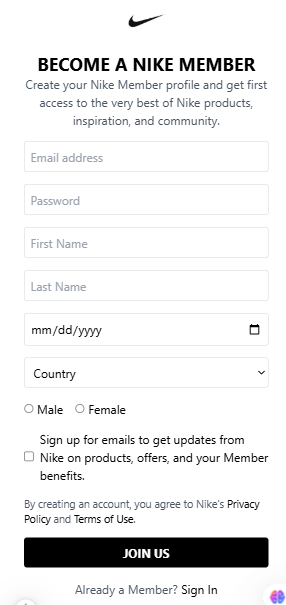


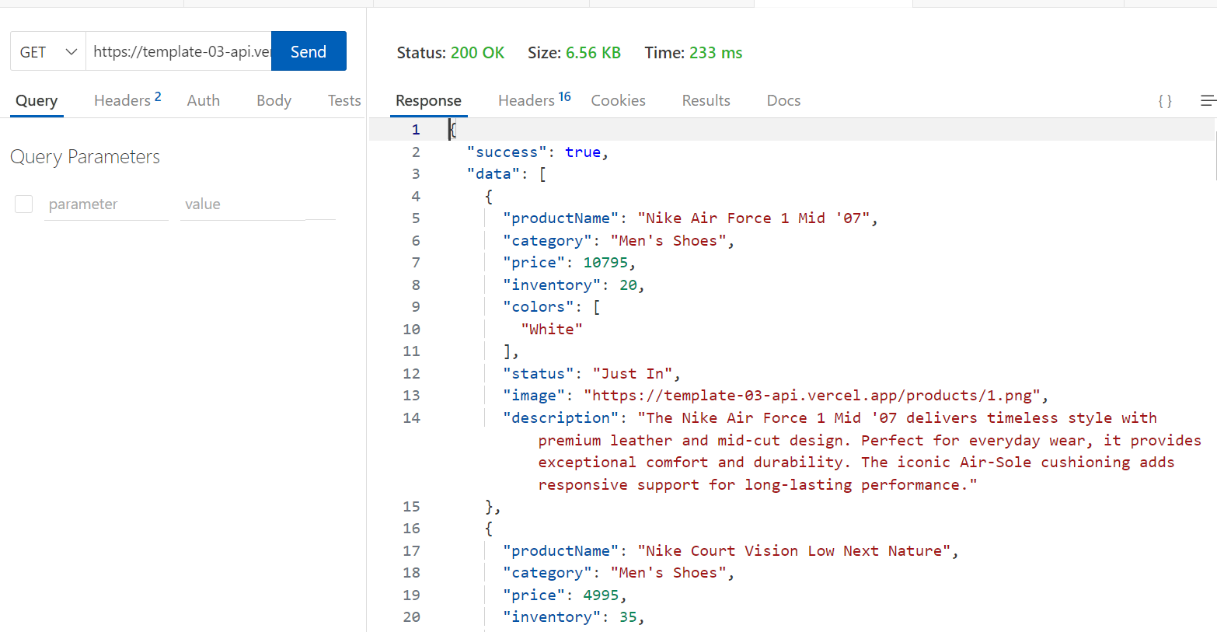
Checkout page

Product detail page with related products



Responsiveness guaranteed



**Reports from testing tools:**

Tool used: Thunder Client(vs code extension)

**2. Testing Report (CSV Format):**

********

**3. Challenges Faced and Resolutions Applied**

1. **Challenge: Handling Dynamic Cart Data in URL**:
   * **Problem**: Initially, I faced challenges with managing dynamic cart data in the URL query string, ensuring that the correct product details were passed and parsed.
   * **Resolution**: Implemented the useSearchParams hook from Next.js to easily retrieve query parameters like cartData and totalAmount. This enabled seamless cart data management via URL.
2. **Challenge: Implementing Responsive Checkout Form:**
   * **Problem**: Ensuring the checkout form is responsive and works seamlessly across different devices was initially tricky, especially with the form fields for card information.
   * **Resolution**: Used Tailwind CSS’s responsive utilities (e.g., sm:, md:) to create a responsive layout for the checkout form. This ensured that the form adapts to various screen sizes.
3. **Challenge: Securing Payment Information**:
   * **Problem**: Storing payment information securely and ensuring the data isn't exposed to unauthorized parties.
   * **Resolution**: For the current state, payment details like credit card numbers and personal info are not stored or processed. This functionality is mocked for now, but in the future, it will be integrated with a secure payment gateway like Stripe, which uses tokenization for data security.
4. **Challenge: State Management for Multiple Forms:**
   * **Problem**: Managing form state for multiple inputs (e.g., card number, expiry date) while ensuring that validation works correctly was challenging.
   * **Resolution**: Used React’s state hooks (useState) to manage each field individually, ensuring that form validation is easily applied and the state is updated accordingly.

### Optimization Steps and Their Implementation

**Lazy Loading of Images**  
Implemented lazy loading for images to load only when they enter the viewport, improving initial page load time. This has been successfully completed.

**Optimized Cart Data Handling**  
Used URL query parameters to pass cart data, reducing API calls and speeding up page rendering. This step has been completed.

**Code Splitting**  
Dynamic imports were used to load components only when necessary, reducing initial load time. This has been successfully completed.

**CSS Optimization**  
Reduced unused CSS and implemented Tailwind CSS utility classes, decreasing CSS file size and improving loading speed. This optimization has been completed.