**Day 4 - Dynamic Frontend Components for Your Marketplace**

**Objective:**

The objective of Day 4 is to design and develop dynamic frontend components that display marketplace data fetched from Sanity CMS or APIs. This step ensures modular, reusable component design, promoting scalable and responsive web applications.

**Key Learning Outcomes**

1. Develop dynamic frontend components using Next.js.

2. Fetch and display data from Sanity CMS or APIs.

3. Implement state management techniques for interactive components.

4. Ensure responsive design using modern UI frameworks.

5. Replicate real-world workflows for client projects.

**Key Components Built**

2. Navbar.tsx

3. Footer.tsx

4. Headingbar.tsx

5. HeroCommon.tsx

6.Cheackout.tsx as slug component

**Implementation Steps:**

1. Setup: Connected the Next.js project with Sanity CMS and verified API data availability.

2. Developed Components: Implemented modular UI components and applied state management.

3. Integrated API Data: Fetched and displayed API responses dynamically.

4. Testing & Debugging: Used Postman and React DevTools for validation.

**Expected Output:**

1. Fully functional product listing page displaying data from Sanity CMS.

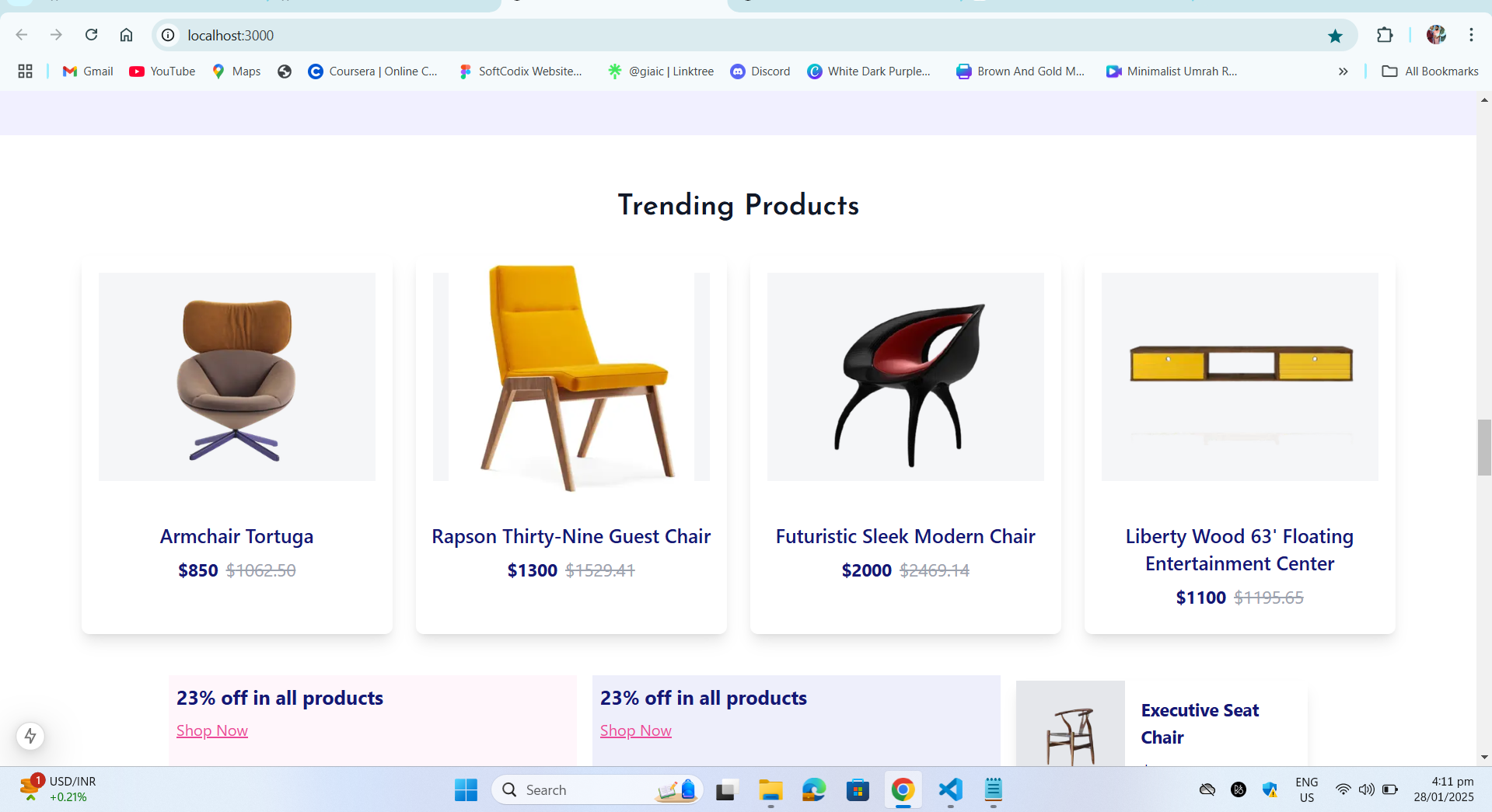
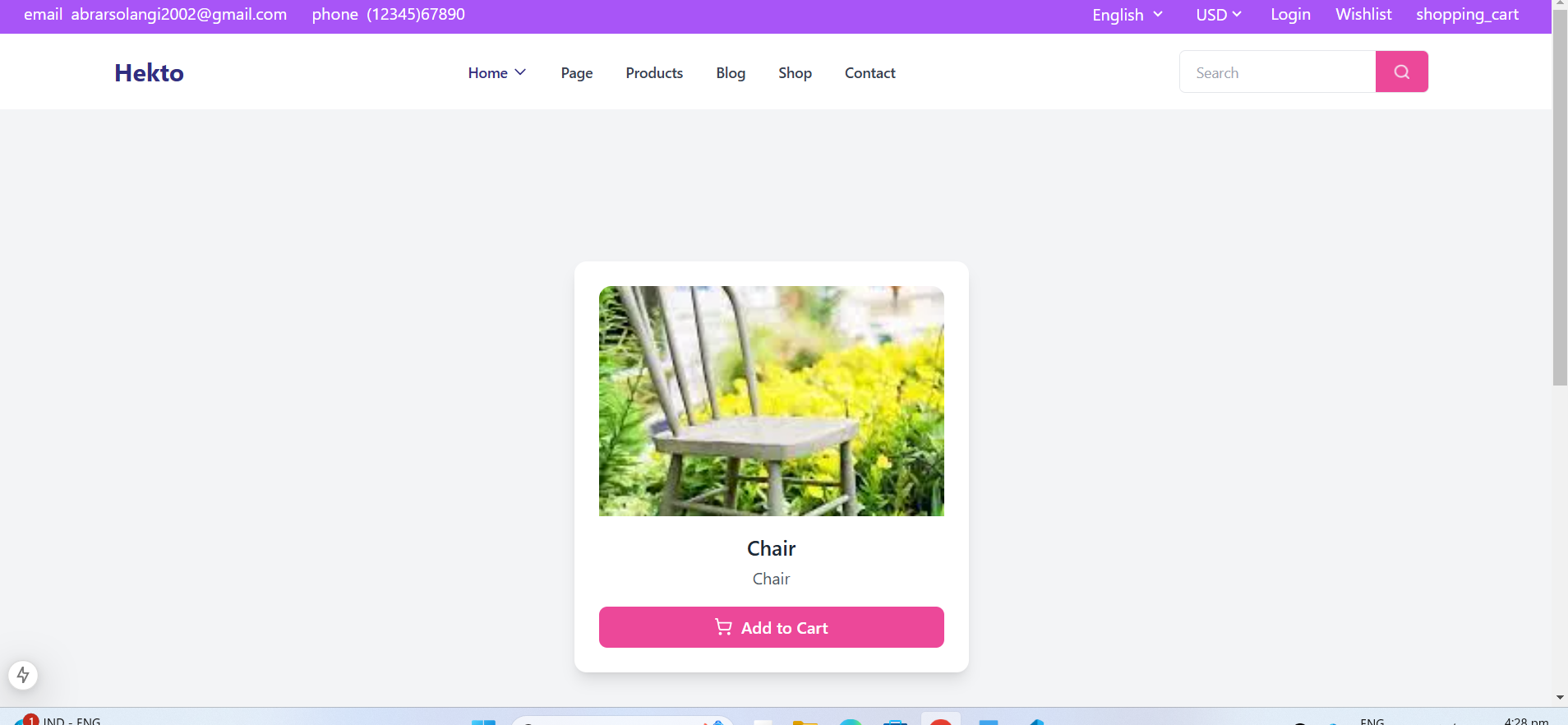
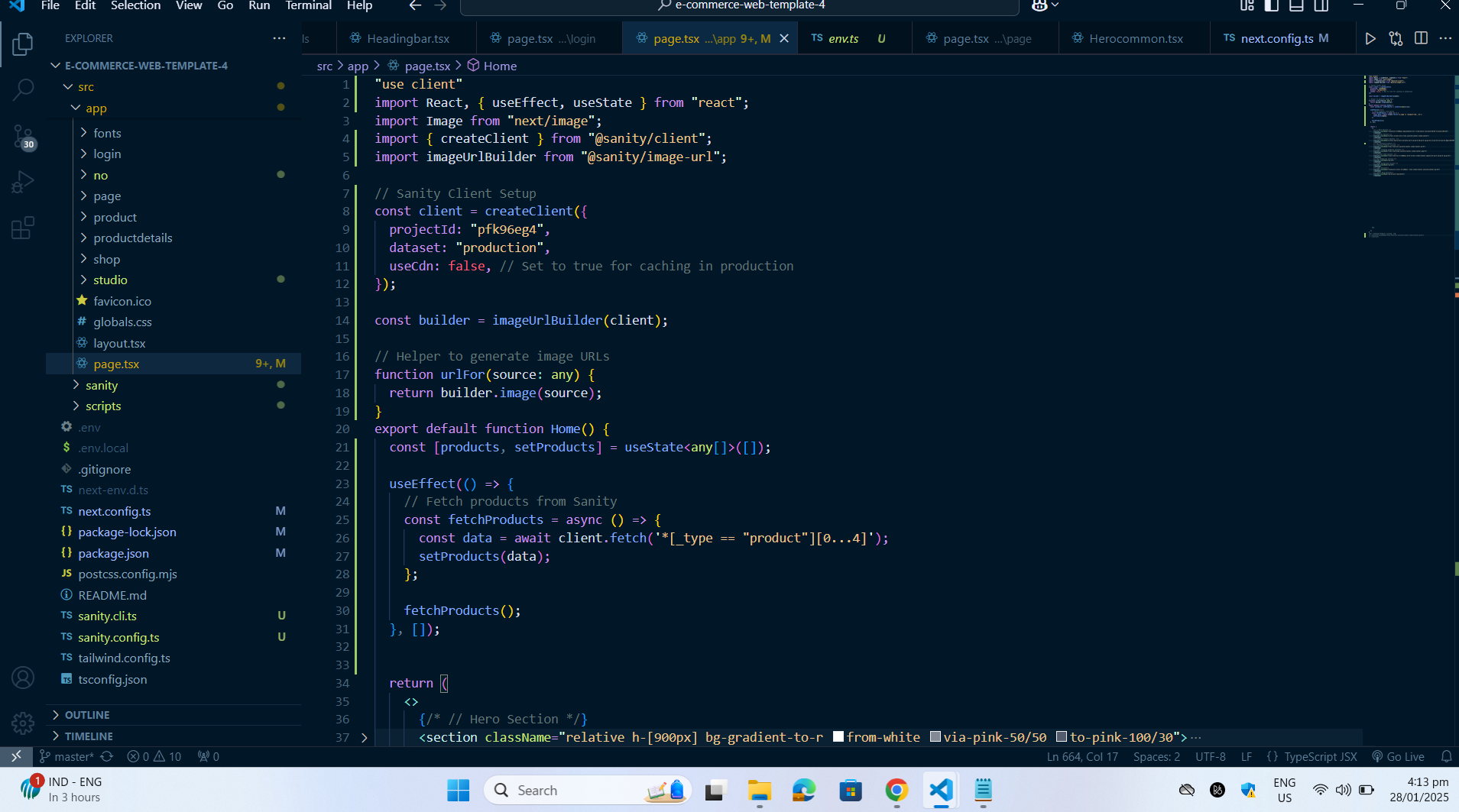
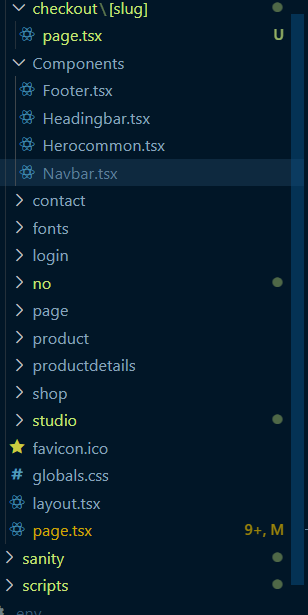
2. Dynamic product detail pages with accurate routing and content.

3. Advanced filtering options for better product discoverability.

4. A working search bar filtering products based on keywords.

5. Additional features like pagination, wishlist, and cart management.

**Screenshots:**

**Challenges & Solutions**

✔ Issue: API data inconsistencies in the product schema.

✔ Solution: Mapped API response fields to the Sanity schema.

✔ Issue: State management complexity for the cart system.

✔ Solution: Used useContext for centralized state management.

✔ Issue: Performance issues on large product listings.

✔ Solution: Implemented pagination and lazy loading for images.

**Best Practices Followed**

✔ Modular Component Design: Reused components for scalability.

✔ State Management: Utilized useContext for efficient state handling.

✔ Responsive UI: Ensured mobile and desktop compatibility.

✔ Performance Optimization: Used lazy loading and API caching.

✔ Code Quality: Followed clean coding practices with proper comments.

**Conclusion**

The completion of Day 4 resulted in a fully dynamic frontend powered by Sanity CMS and APIs. This implementation follows industry best practices and prepares the marketplace for a seamless shopping experience. Future improvements may include AI-powered recommendations and multi-language support.