Avion General Ecommerce Marketplace Hackathon Full Documentation from (Day 1 to 6)

Day 1: Marketplace Concept & Design (paper & pencil work)

Key Achievements:

- Established the marketplace as a general e-commerce platform focused on furniture, chairs ans soafas.

Business Objectives:

- Support businesses and entrepreneurs by providing a digital selling space.
- Enable users to buy and sell furniture effortlessly through an online platform.

Data Structure Planning:

- Core Entities: Products, Orders, Customers, and Delivery Zones.
- Entity Relationships:
    - Customers place orders linked to specific products.
    - Delivery zones are assigned to drivers for efficient order fulfillment.

Day 2: Technical Planning (paper & pencil work)

Key Achievements:

- **Tech Stack Selection:**
    - Frontend: Built with Next.js for server-side rendering and optimized performance, styled using Tailwind CSS for a responsive and modern UI.
    - Backend: Integrated Sanity CMS for flexible and efficient content management, allowing dynamic updates without redeployment.
- **API Requirements:**
    - **User Management:**
        - /register → Handles new user signups.
        - /login → Manages authentication and session handling.
        - /verify-route → Verifies user identity and access permissions.
    - **Product Management:**
        - /products → Fetches all available products.
        - /product/:id → Retrieves specific product details.
    - **Order Processing & Shipment:**
        - /orders (POST) → Places new orders and processes transactions.
- **Deployment Strategy:**
    - Frontend: Deployed on Vercel for seamless CI/CD, fast load times, and automatic scaling.

Day 3: Sanity Integration & Data Migration

Key Achievements:

- Custom Data Migration Process:
    - Developed a custom migration script to seamlessly transfer data from Sanity CMS to Next.js.
    - Utilized GROQ queries to efficiently fetch structured data.
    - Example GROQ Query:
    - *[_type == "products"] {title, description, price, image}
- Database Schema Design:
    - Defined a structured product schema to ensure consistency in data storage.
    - Included key fields such as title, slug, description, price, and image, optimizing content retrieval.
- Frontend Data Integration:
    - Dynamically fetched migrated data and rendered it on the homepage.
    - Ensured a seamless user experience by efficiently displaying product details.

Day 4: Developing Dynamic Frontend Components (Next JS).

Key Achievements:

- Dynamic Product Listings page:
    - Developed a ProductList component that dynamically fetches and displays furniture products from Sanity CMS using GROQ queries.

    - Ensured real-time updates and smooth rendering for an optimized browsing experience.
    - Integrated lazy loading for images to improve page speed and performance.
- Used category for Filtering products:
    - Implemented category-based filtering to refine product searches.
- Core Reusable UI Components:
    - ProductCard: Displays product image, title, price, and quick actions like adding to cart or wishlist.
    - PaginationControls: Enables smooth navigation for browsing large product inventories.
    - ProductDetail:
        - Displays detailed product information, including images, descriptions, and pricing.
        - Integrated an "Add to Cart" and "Add to Wishlist" button for quick actions.
- Shopping Cart & Wishlist System:
    - Cart Component:
        - Displays items added by the user with an option to update quantity or remove items.
        - Shows a subtotal and estimated total price before checkout.
    - Wishlist Component:
        - Allows users to save favorite products for future purchases.
        - Integrated an "Move to Cart" option for a seamless transition to checkout.

- Checkout Flow & Order Processing:
    - Checkout Component:
        - Guides users through a step-by-step purchase process, including address input and payment selection.
    - Order Confirmation:
        - Displays a summary of the purchase with estimated delivery details.
        - Sends a confirmation email to the customer.

Day 5: Testing & Backend Refinement & CSV Report.

Key Achievements:

- Comprehensive Testing Process:
    - **Functional Testing:**
        - Ensured smooth operation of critical workflows, including product listings, cart functionality, and API interactions.
        - Tested user actions like adding/removing products from the cart and completing the checkout process.
    - **Performance Testing:**
        - Utilized Google Lighthouse to analyze page load times, responsiveness, and accessibility.
        - Optimized frontend rendering for faster interactions and improved Core Web Vitals scores.
    - **Security Testing:**

- Implemented input validation to prevent SQL injection and XSS attacks.
- Secured API keys and sensitive data using environment variables.
- Ensured HTTPS encryption for secure data transmission.

Csv report

Test Case ID	Description	Expected Result	Actual Result	Status	Severty Level	Remarks
TC001	Verify navigation links	Links navigate correctly	All links function correctly	passed	meduim	none
TC002	Product listing check	Products display as expected	Products displayed correctly	passes	low	none
TC003	Add to cart Checking	Items add, update, and remove	Cart functionality works as expected	passed	low	none
TC004	Add to Wishlist Checking	Items add and remove	Wishlist Items add and remove correctly	passed	high	none
TC005	Product details check	Products' details display expected	Products' details display correctly	passed	high	Addresses some issues
TC006	Verify Checkout Functionality	user's address form display	user's address form display correctly	passed	low	Minor bugs noted
TC007	Verify accessibility features	Accessibility score ≥ 90	Accessibility: 92	passed	meduim	none
TC008	Analyze performance metrics	Achieve Performance ≥ 90	Performance: 97	passed	meduim	none
TC009	Validate best practices	Best Practices score ≥ 90	Best Practices: 96	passed	low	none
TC010	Optimize SEO	SEO score ≥ 90	SEO: 100	passed	high	data validated successfully

Day 6: Deployment Preparation & Staging Environment Setup

Key Achievements:

1_Deployment Strategy

- ### Hosting on Vercel:
  - Deployed the application on Vercel, ensuring fast, scalable, and automatic deployments.

    Link : https://hackathone-quater-2.vercel.app/
  - Leveraged Vercel's serverless functions for backend operations, optimizing performance and reducing infrastructure costs.
- ### CI/CD Integration with GitHub:
  - Connected the GitHub repository to Vercel for continuous integration and deployment (CI/CD).

- Every push to the main branch triggers an automatic deployment, maintaining an up-to-date live version.

2- Secure Environment Variable Management

- .env Configuration:
  - Protected sensitive credentials like API keys and third-party service tokens.
  - Stored variables securely in Vercel's environment settings instead of exposing them in code.
  - Example .env.local file:

.env.local

NEXT_PUBLIC_SANITY_PROJECT_ID=your_project_id

NEXT_PUBLIC_SANITY_DATASET=production

API_KEY=your_api_key

3_Staging Environment Setup

- Staging Deployment:
  - Deployed a staging build to test the application in a production-like environment before going live.
  - Allowed team members to validate functionality and detect issues before the final release.
- Staging Testing Process:
  - **Functional Testing:**
    - Verified key workflows like product browsing, cart functionality, checkout process, and user authentication.
  - **Performance Testing:**

        - Used Lighthouse to analyze page speed, loading time, and responsiveness.
        - Optimized images, scripts, and caching strategies for better performance.
    - **Security Testing:**
        - Ensured all sensitive data was transmitted over HTTPS.
        - Checked input validation to prevent SQL injection and cross-site scripting (XSS) attacks.
        - Verified API security to prevent unauthorized access.

4_Documentation & Repository Organization

- ### README.md Documentation:
    - **Created a detailed README.md file to provide an overview of the project, including:**
        - Project structure
        - Tech stack
        - Setup and installation instructions
        - Deployment guidelines

- **Structure of my Hackathon App:**

```
QUARTER-02-HACKATHON
|
├── next
|   ├── app
|   |   ├── about
|   |   ├── cart
|   |   ├── ceramics
```

```
|   |   ├── chairs
|   |   ├── checkout
|   |   ├── fonts
|   |   ├── products
|   |   ├── register
|   |   ├── studio
|   |   ├── wishlist
|   |   ├── favicon.ico
|   |   ├── globals.css
|   |   ├── layout.tssx
|   |   ├── not-found.tssx
|   |   └── page.tssx
|   |
|   ├── components
|   ├── context
|   ├── lib
|   ├── node_modules
|   ├── public
|   ├── sanity
|   ├── scripts
|   ├── types
|   ├── .env.local
|   ├── .eslintr.c.json
|   ├── .gitattributes
|   ├── .gitignore
|   ├── components.json
|   └── next-env.d.ts
```