



Rental E-commerce

06.02.2025

Name : Ubaid Alam

Summary of Day 1 Task

1. Marketplace Type:

- Developed concept for an online rental platform for luxury items
- Primary focus on car rentals with flexible duration option

2. Business Goals:

- Problem: Addressing vehicle breakdown during long trips
- Solution: Providing immediate replacement vehicles to ensure uninterrupted journeys

3. Target Audience:

- Event planners needing vehicles for weddings and picnics
- Families planning trips

4. Services Offered:

- Vehicle Rentals (sedans, SUVs, luxury cars)
- Roadside Assistance with replacement vehicles
- Prepaid fuel plans
- 24/7 customer support

5. Unique Selling Points:

- Quick delivery within promised timeframes
- Competitive pricing with high service quality
- Customizable vehicle selection based on preferences

6. Data Structure:

- Designed comprehensive database schema with five main entities
- Established relationships between Users, Cars, Bookings, Payments
- Included all necessary fields for each entity with appropriate data types

Summary of Day 2 Task

Objective:

Develop a scalable car rental platform with:

- Car browsing via **Sanity CMS**
- User authentication with **Clerk**
- Rental tracking through **custom APIs**

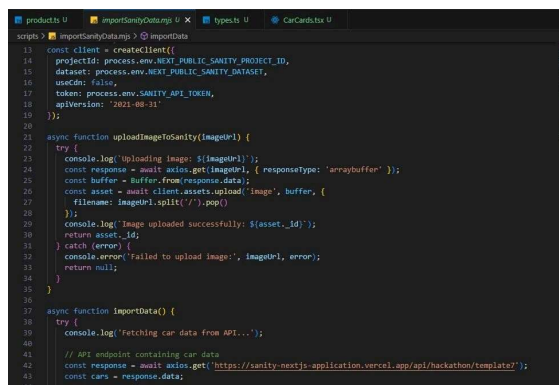
System Architecture:

- **Frontend (Next.js):** Fetches cars from Sanity, processes payments, and tracks orders.
- **Backend:** Handles data management and API endpoints.

Summary of Day 3 Task

API Understanding: Studied docs, identified endpoints, planned integration.

Schema Validation: Designed & validated schemas in Sanity CMS.

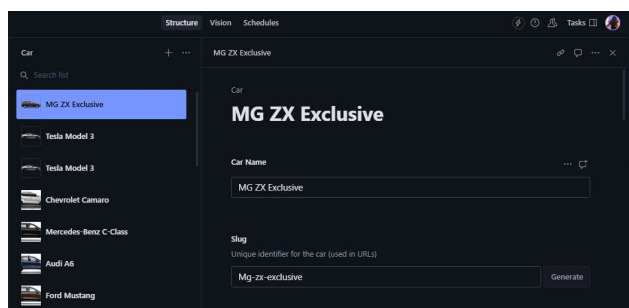


```

13 const client = createClient({
14   projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
15   dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
16   useCdn: false,
17   token: process.env.SANITY_API_TOKEN,
18   apiVersion: '2021-08-31'
19 });
20
21 async function uploadImageToSanity(imageUrl) {
22   try {
23     console.log('Uploading image: ${imageUrl}');
24     const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
25     const buffer = Buffer.from(response.data);
26     const asset = await client.assets.upload('image', buffer, {
27       filename: imageUrl.split('/').pop()
28     });
29     console.log('Image uploaded successfully: ${asset._id}');
30     return asset._id;
31   } catch (error) {
32     console.error('Failed to upload image:', imageUrl, error);
33     return null;
34   }
35 }
36
37 async function importData() {
38   try {
39     console.log('fetching car data from API...');
40
41     // API endpoint containing car data
42     const response = await axios.get('https://sanity-nextjs-application.vercel.app/api/hackathon/template');
43     const cars = response.data;
44   }

```

Data Migration: Extracted, transformed, and loaded car rental data into Sanity CMS.

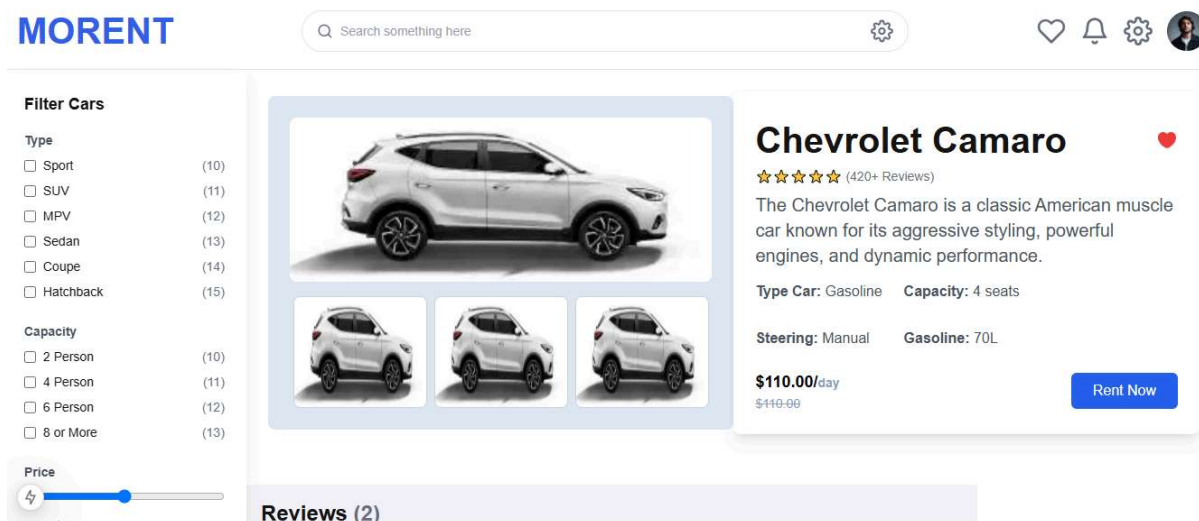


API Integration: Implemented API calls in Next.js, set up state management, tested performance

Summary of Day 4 Task

1. Dynamic Routing

- **Description:** Implemented dynamic routing for product details, allowing users to access specific product pages through URLs.



2. Search Bar Functionality

- **Description:** Added a search bar to help users easily find products with real-time results.

Recommendation Cars

[View all](#)

MG ZX Exclusive

Hatchback



Brand: **MG**

Transmission: **Manual**

Seating Capacity: **2 People**

Fuel Capacity: **90L**

Tags:

\$90.00/day

[Rent now](#)

3. Pagination

- Description:** Introduced pagination to handle large product lists, making them more manageable for users.

MORENT



Recommendation Cars

[View all](#)

Tesla Model 3

Electric



Brand: **Tesla**

Transmission: **Manual**

Seating Capacity: **5 seats**

Fuel Capacity: **100kWh**

Chevrolet Camaro

Gasoline



Brand: **Chevrolet**

Transmission: **Manual**

Seating Capacity: **4 seats**

Fuel Capacity: **70L**

Audi A6

Hybrid



Brand: **sedan**

Transmission: **Manual**

Seating Capacity: **5 seats**

Fuel Capacity: **60L**

Mercedes-Benz C-Class

Gasoline



Brand: **Mercedes-Benz**

Transmission: **Manual**

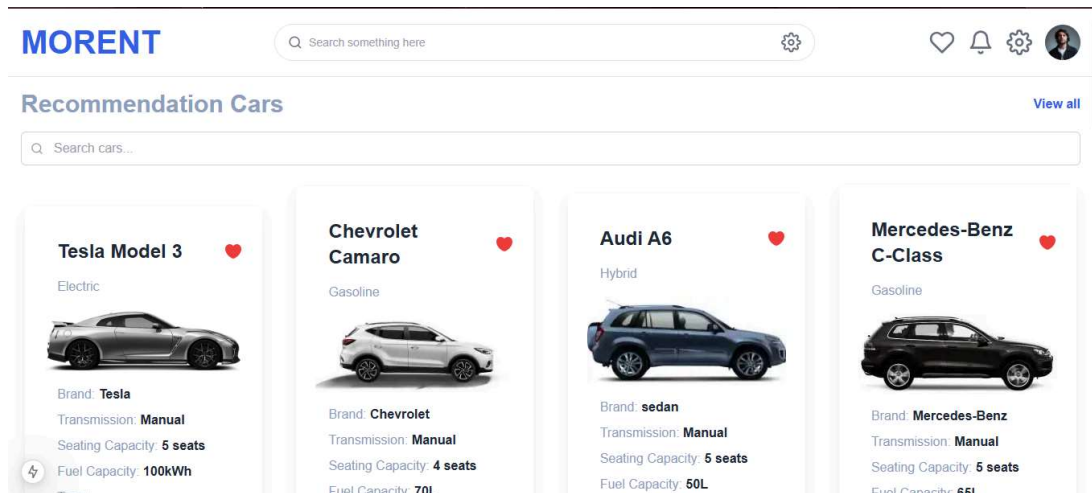
Seating Capacity: **5 seats**

Fuel Capacity: **65L**

Summary of Day 5 Task

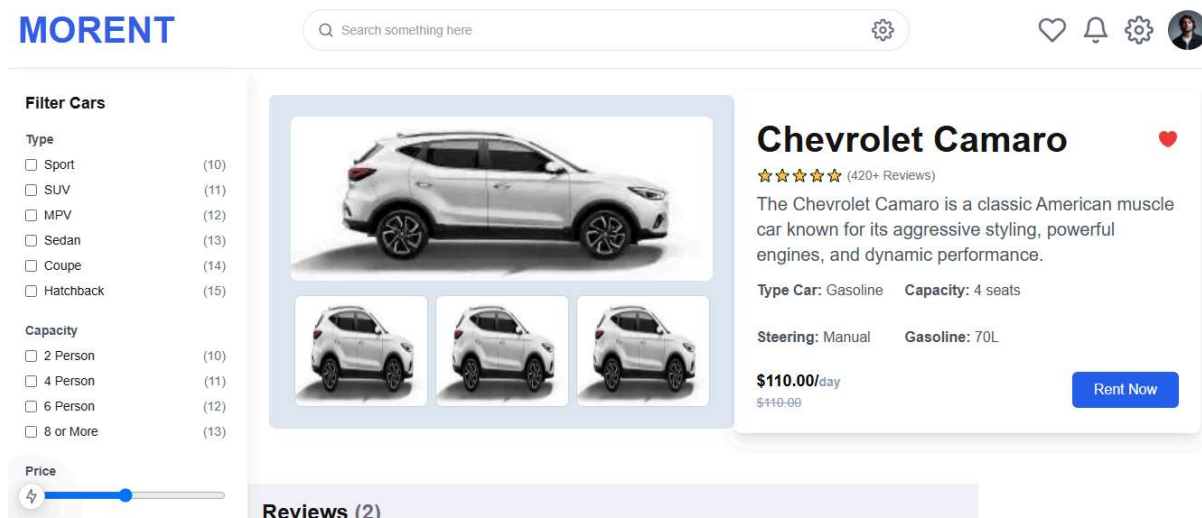
Step 1: Functional Testing

1. Product Listing Page



- **Description:** Displays all available rental products with filters for category, price, and availability.
- **Functionality:** Users can browse and filter rental items.

2. Product Detail Page

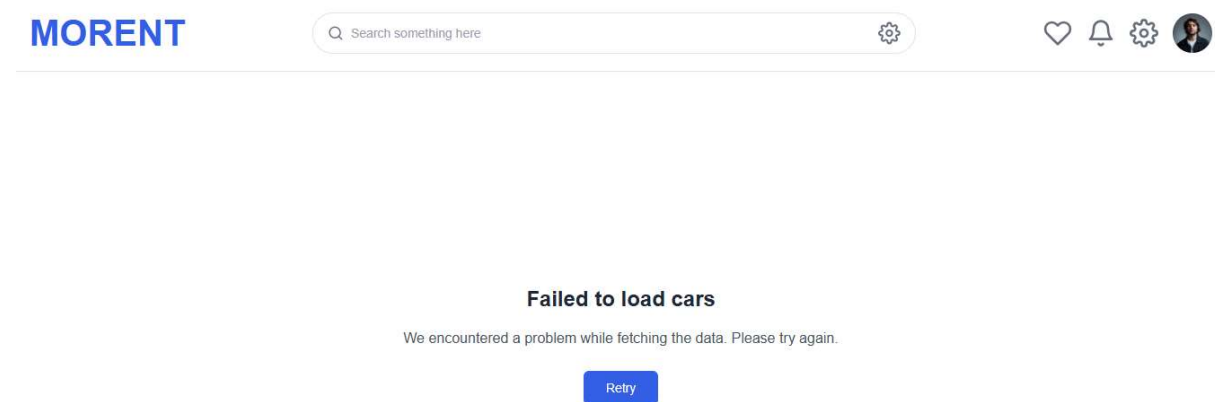


- **Description:** Shows detailed information about a specific rental product, including description, pricing, and availability.
- **Functionality:** Users can view product details and proceed to book the item directly.

Step 2: Error Handling

Error Handling Mechanisms

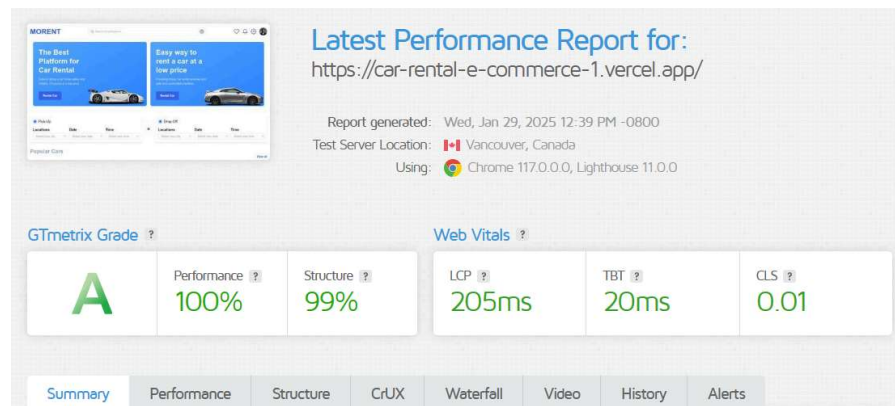
- **Network Failures:** If the API fails to load product data, a fallback UI is displayed with the message:



Step 3: Performance Optimization

Performance Testing Results

- **Lighthouse Report:**



Step 4: Cross-Browser and Device Testing

Description

- All cross-browser and device tests were executed successfully. The marketplace displayed consistent functionality and responsiveness across all tested browsers (Chrome, Firefox, Safari, Edge) and devices (desktop, tablet, mobile). No layout issues or functionality breaks were observed during testing.

Step 5: Security Testing

HTTPS Enforcement

1. **Description:**
 - All API calls and image requests are made over **HTTPS**, ensuring secure communication between the client and server.
 - The `images.remotePatterns` configuration in your `next.config.js` ensures that images are only loaded from trusted sources (e.g., `cdn.sanity.io`) over HTTPS.
2. **Testing:**
 - Verified that all requests to `cdn.sanity.io` and other APIs use **HTTPS**.
 - Confirmed that no mixed content (HTTP resources on an HTTPS page) is present in the application.

Step 6: User Acceptance Testing (UAT)

UAT Scenarios

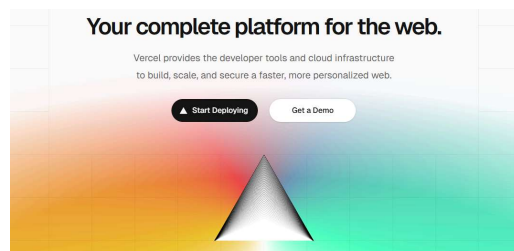
1. **Browsing Products:**
 - **Description:** Users successfully browsed rental products using filters and search functionality.
2. **Error Handling:** Users encountered and understood error messages during simulated network failures and server errors.

Step 7: CSV Based Testing Report

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Assigned To	Remarks
TC001	Validate product listing page	Open car page > Verify cars	car displayed correctly	Products displayed correctly	Passed	Low	-	No issues found
TC002	Test API error handling	Disconnect API > Refresh page	Show fallback UI with error message	Error message shown	Passed	Medium	-	Handled gracefully
TC003	Search functionality	Enter car name and brand > Check results	Relevant results displayed	Search works for name and brand	Passed	High	-	Implemented successfully
TC004	Payment page	Navigate to payment page > Verify UI elements	All payment UI elements present	Payment page UI complete	Passed	Medium	-	Payment processing not implemented yet

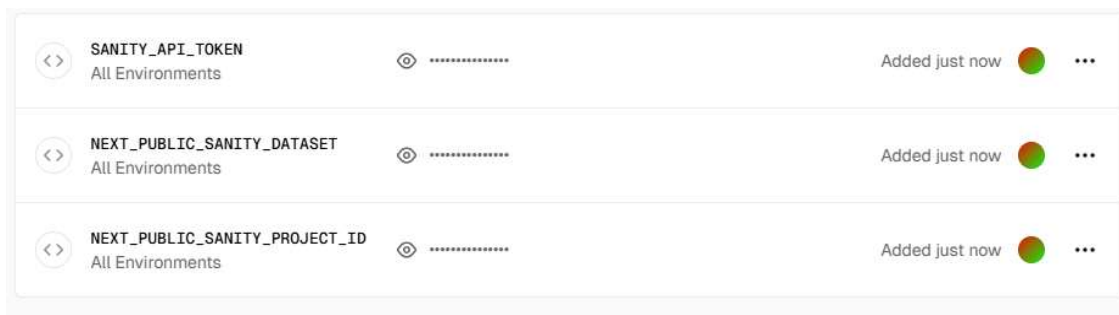
Summary of Day 6 Task

Step 1: Hosting Platform Setup



Step 2: Configure Environment Variables

Created a `.env` file with sensitive variables like API keys and tokens:

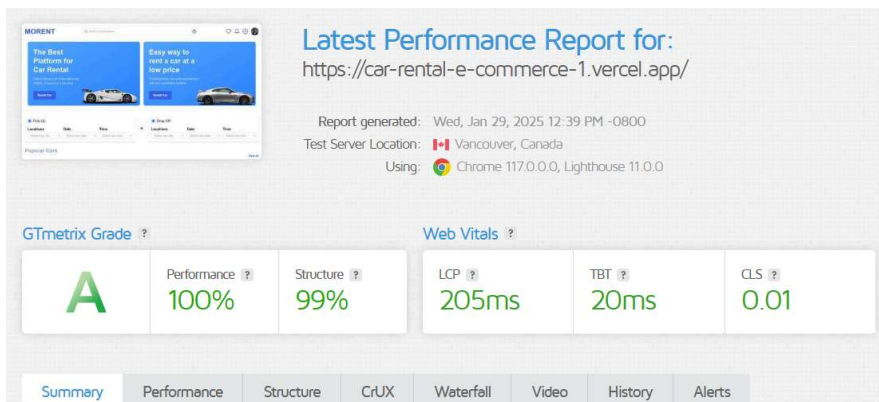


Step 3: Deploy to Staging

- Deployed the application to a staging environment through the hosting platform.
- Validated that the build process completed without errors.
- Verified basic functionality in the staging environment (e.g., product listing, search, cart operations).
- Outcome :
 - A fully deployed staging environment was created, and the application was tested for basic functionality.

Step 4: Staging Environment Testing

- Functional Testing :
 - Verified workflows such as product listing, search, and cart operations.
- Performance Testing :
 - Used tools like Lighthouse or GTmetrix to analyze speed, responsiveness, and load times



1. Security Testing :

- Ensured HTTPS was enabled and sensitive data (e.g., API keys) was handled securely.
2. Responsiveness Testing
 - Tested the application on different devices and screen sizes to ensure proper layout adjustments.
 3. Documented all test cases in a CSV file with the following fields

TC001	Validate product listing page	Open car page > Verify cars	car displayed correctly	Products displayed correctly	Passed	Low	-	No issues found
TC002	Test API error handling	Disconnect API > Refresh page	Show fallback UI with error message	Error message shown	Passed	Medium	-	Handled gracefully
TC003	Search functionality	Enter car name and brand > Check results	Relevant results displayed	Search works for name and brand	Passed	High	-	Implemented successfully
TC004	Payment page	Navigate to payment page > Verify UI elements	All payment UI elements present	Payment page UI complete	Passed	Medium	-	Payment processing not implemented yet
TC005	User payment and data storage	1. Fill payment form > 2. Submit payment > 3. Verify Sanity storage	Payment processed and user data stored in Sanity	Payment successful, data saved in Sanity CMS	Passed	High	-	Implemented successfully
TC006	Website Performance Testing	1. Run GTmetrix performance test > 2. Check Core Web Vitals > 3. Verify metrics	Performance grade A, LCP < 250ms, TBT < 30ms, CLS < 0.1	Grade A, Performance 100%, LCP 205ms, TBT 20ms, CLS 0.01	Passed	High	-	Excellent performance metrics achieved across all measures

Expected Output

By the end of Day 6, I successfully delivered the following:

1. A fully deployed staging environment for the marketplace.
2. Environment variables securely configured in the hosting platform.
3. Test case and performance reports documenting staging tests.
4. All project files and documentation organized in a GitHub repository.

Conclusion.

Over the course of six days, I successfully developed a scalable and secure rental e-commerce platform for luxury car rentals. The project progressed through well-structured phases, starting from conceptualization to deployment. Key achievements include designing a comprehensive database schema, implementing API integrations, and optimizing performance through functional and security testing. The platform ensures seamless user experience with features like dynamic routing, search functionality, and pagination.

Additionally, thorough testing, including cross-browser, security, and user acceptance testing, validated the system's reliability. A fully functional staging environment was deployed, ensuring all essential workflows operated correctly.