Marketplace Technical Foundation - [RentalHub Marketplace]

Prepared By: Rizwan

Date: 16-1-2025 RollNo: 00289287

1. System Architecture Overview

The following architecture diagram illustrates how the components interact in the RentalHub Marketplace:

Components:

- Frontend (Next.js): User-facing interface for browsing rental products, managing orders, and tracking rentals.
- Sanity CMS: Acts as the backend for managing rental product data, order details, and user information.
- Third-Party APIs: Used for payment processing, shipment tracking, and notification services.

High-Level Architecture:

[Frontend (Next.js)]

| \

[Sanity CMS] [Third-Party APIs]

|

[Rental Data & Order Management]

Data Flow:

Rental Product Browsing: Frontend fetches data from Sanity CMS via API and displays rental products.

Order Placement: Frontend sends rental order details to Sanity CMS, where they are stored.

Payment Processing: Rental payment is processed via a third-party payment gateway.

Rental Tracking: Shipment status is fetched through a third-party API and displayed to the user.

2. Key Workflows

2.1 User Registration:

- Step 1: User signs up via the frontend form.
- Step 2: User details are stored in Sanity CMS under the "User" schema.
- Step 3: Third-Party APIs: Email confirmation is sent to the user via a third-party email API.

2.2 Product Browsing:

Step 1: User visits the product listing page.

Step 2: Frontend fetches available rental products from Sanity CMS.

Step 3: Products are displayed with details like price, rental duration, and availability.

2.3 Order Placement:

Step 1: User selects a product and specifies rental duration.

Step 2: Frontend sends order data to Sanity CMS.

Step 3: Payment is processed, and order status is updated to "Confirmed."

2.4 Rental Tracking:

Step 1: User checks rental order status on their dashboard.

Step 2: Shipment tracking details are fetched from a third-party API.

Step 3: Current rental status (e.g., "In Transit") and ETA are display

3. API Endpoints

Endpoint MethodPurpose Payload/Response Example

/products GET Fetch all available rental items Response: { "id": 1, "name": "Camera", "rentalPrice": 50, "available": true }

/rental-durationPOST Create a rental order Payload: { "productId": 1, "duration": "7 days", "deposit": 100 }

/rental-status GET Fetch status of a rental order Response: { "orderId": 123, "status": "In Use", "returnDate": "2025-02-01" }

/payment POST Process payment for the rental Payload: { "orderId": 123, "amount": 150 }

4. Sanity Schema Example

Here’s an example schema for the Rental Products and Rental Orders.

4.1 Rental Product Schema

```
export default {
  name: 'rentalProduct',
  type: 'document',
  fields: [
    { name: 'name', type: 'string', title: 'Product Name' },
    { name: 'description', type: 'text', title: 'Description' },
    { name: 'rentalPrice', type: 'number', title: 'Rental Price (per day)' },
    { name: 'deposit', type: 'number', title: 'Deposit Amount' },
    { name: 'available', type: 'boolean', title: 'Availability' },
    { name: 'condition', type: 'string', title: 'Condition (e.g., New, Good, Fair)' }
  ]
};
```

4.2 Rental Order Schema

```
export default {
  name: 'rentalOrder',
  type: 'document',
  fields: [
    { name: 'user', type: 'reference', to: [{ type: 'user' }], title: 'User' },
    { name: 'product', type: 'reference', to: [{ type: 'rentalProduct' }], title: 'Product' },
    { name: 'duration', type: 'string', title: 'Rental Duration' },
```

```
    { name: 'status', type: 'string', title: 'Order Status (e.g., Pending, In Use, Completed)' },
    { name: 'deposit', type: 'number', title: 'Deposit Amount' },
    { name: 'paymentStatus', type: 'string', title: 'Payment Status (e.g., Paid, Pending)' }
  ]
};
```

5. Collaborative Feedback

Feedback Incorporated: During the peer review, suggestions for improved schema modularity and additional fields like "Return Condition" in orders were added.

Scalability Suggestions: API endpoints were refined to ensure scalability by supporting batch operations (e.g., bulk fetching product data).

Key Outcomes

System Architecture: Visualized the interaction between components for a rental eCommerce system.

Workflows: Detailed steps for key processes, ensuring seamless user experience.

API Requirements: Clearly documented endpoints for core functionalities.

Sanity Schema: Drafted flexible schemas for rental products and orders.

Collaboration: Incorporated feedback to improve scalability and clarity.