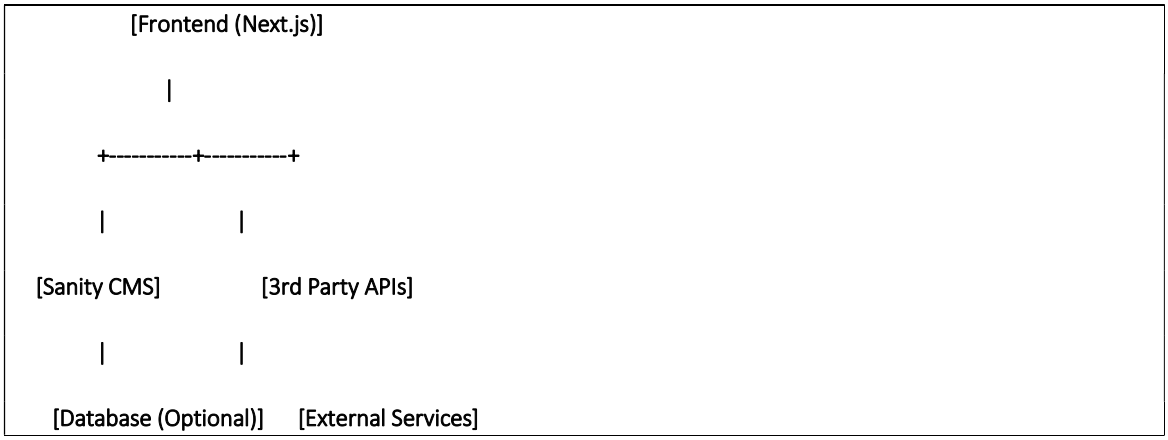
Marketplace Builder Hackathon 2025 (Day-2)



Frontend (Next.js)

The frontend is built using **React with Next.js** and includes features like routing, server-side rendering, and dynamic content fetching.

Key Features:

1. **File-based Routing:**
   - Each file in the pages/ directory corresponds to a route.
   - Example: pages/index.js serves as the homepage.
2. **Dynamic Pages and API Data:**
   - Integrates data from **Sanity CMS** or **3rd Party APIs** to dynamically render pages.
   - Example: Fetch blog posts or product listings dynamically.
3. **Styling:**
   - Custom styling using **CSS**, **Sass**, or styled-components.
4.
5. **Client-side Interactivity:**
   - Components are interactive and reusable, ensuring an excellent user experience.

Backhend (API and Data Management)

The backend processes data, manages external integrations, and handles server-side operations.

1. API Routes:

- Located in the pages/api/ directory, these routes act as serverless functions.
- Use cases:
  - Fetch data from Sanity CMS or external APIs.
  - Process user authentication.
  - Handle CRUD operations.

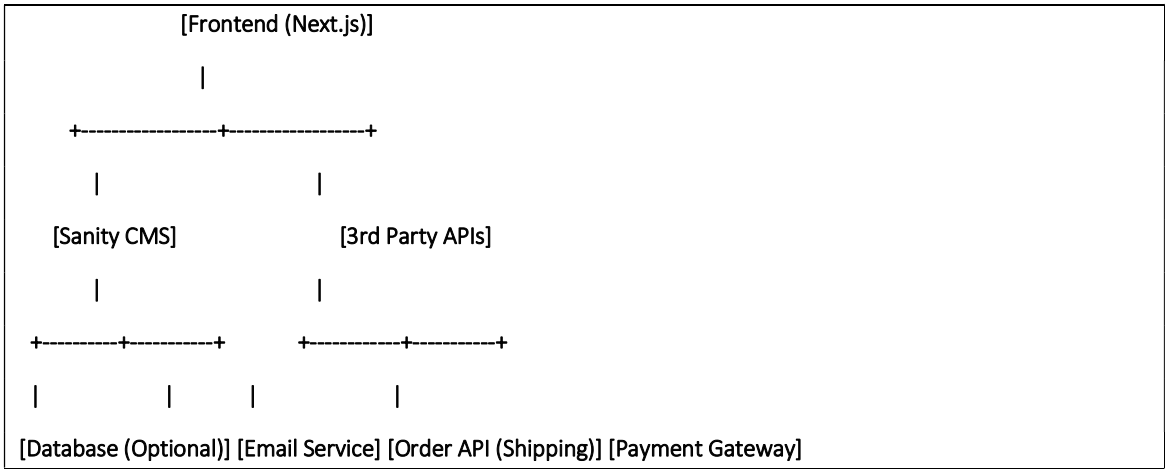
2. CMS Integration (Sanity CMS):

- Sanity CMS stores dynamic content like blog posts, product descriptions, or user-generated data.
- Next.js fetches this data during the build (SSG) or at runtime (SSR).

3. 3rd Party API Integration:

Example: Fetching data for external services such as payment gateways (e.g., Stripe), weather APIs, or mapping tools.

System Architecture with Key Workflows



Key Workflows Breakdown:

1. User Registration:
   - Action: User signs up (through frontend, i.e., a registration form).
   - Workflow:
     - User submits details (name, email, password, etc.).
     - Data is stored in Sanity (optional: in a user collection, or another storage if needed).
     - An email confirmation is sent to the user via an **Email Service** (e.g., SendGrid).
   - Related Systems: Frontend (Next.js), Sanity CMS, Email Service.
2. Product Browsing:
   - Action: User views product categories.
   - Workflow:
     - Frontend sends a request to **Sanity CMS** to fetch product data (product categories, details, etc.).
     - Data is processed and displayed on the frontend.
   - Related Systems: Frontend (Next.js), Sanity CMS.
3. Order Placement:
   - Action: User adds items to the cart and proceeds to checkout.
   - Workflow:
     - The frontend gathers order details (items, quantities, etc.) and submits the data to **Sanity CMS** (could be an "orders" collection).
     - Order is confirmed and the frontend displays confirmation.
     - Data about the order is stored in Sanity (optional, as a record of the order).
   - Related Systems: Frontend (Next.js), Sanity CMS.
4. Shipment Tracking:
   - Action: User tracks the status of their order.
   - Workflow:
     - The frontend sends a request to the **Order API (Shipping)** (could be a 3rd-party API like UPS, FedEx, or custom).
     - The order status is fetched and displayed on the frontend (e.g., "In Transit", "Delivered").
   - Related Systems: Frontend (Next.js), 3rd Party APIs (Shipping), optional **Order API**.

5. **Payment Processing (Optional):**
    - **Action:** User completes payment for the order.
    - **Workflow:**
    - The frontend sends payment details to a **Payment Gateway API** (e.g., Stripe, PayPal).
    - Payment is processed and confirmation is sent back to the frontend.
    - **Related Systems:** Frontend (Next.js), Payment Gateway API.

Key Technologies:

- **Sanity CMS:** A headless CMS to store product, user, and order data.
- **3rd Party APIs:** For shipment tracking, external data sources, or other services.
- **Email Service:** To send confirmation and communication emails (like SendGrid or Mailgun).
- **Database (Optional):** For storing user or order information, if required
- **Payment Gateway API:** For processing payments.

Customized Car Rental Marketplace Plan

Objective

To create a scalable, user-friendly car rental platform with modern features, including car browsing, user authentication, order tracking, and secure payments.

System Architecture

[User] -->|Sign Up/Sign In| Clerk

[User] -->|Browse Cars| Frontend (Next.js)

[Frontend] -->|Fetch Car Data| Sanity CMS

[Frontend] -->|Process Payment| Stripe API

[Frontend] -->|Track Orders| Custom APIs

[Admin] -->|Manage Cars & Orders| Sanity Studio

Key Features

Frontend Features

Authentication

User Login/Signup:

    - Users can register, log in, and manage accounts.
    - Session management is fully handled by Clerk.
    - **Social Login Options:** Enable Google, Facebook, or other OAuth providers for convenience.
    - **Custom Redirects:** Redirect users to specific pages after login (e.g., the car listing page).
2. **Car Browsing** *(Powered by Sanity CMS)*:
    - **Fetch Cars:**
        - Use **GROQ queries** to retrieve car listings from Sanity CMS.

        - Fetch fields such as car type, model, price per day, availability, and images.
    - **Filters:**
        - **Price Range:** Slider for minimum and maximum price.
        - **Car Type:** Dropdown or checkbox for car categories (e.g., SUV, Sedan, Luxury).
        - **Availability:** Show only available cars.
    - **Sorting Options:**
        - Sort by price (low to high, high to low).
        - Sort by popularity or newest additions.
3. **Checkout Process** *(Powered by Stripe)*:
    - Collect rental details:
        - Selected car, rental duration, and additional services (e.g., insurance).
    - Redirect users to **Stripe-hosted checkout pages** for secure payments.
    - Display an order confirmation page with:
        - Receipt details.
        - Rental information (e.g., start and end dates).
        - Booking reference number.
4. **Order Tracking** *(Via Custom APIs)*:
    - Allow users to view the status of their rental orders.
    - Display details such as:
        - Rental duration.
        - Current order status (e.g., confirmed, in progress, completed).

Backend Features

1. **Sanity CMS** *(Headless CMS for Content Management)*:
    - Manage car listings:
        - Schema includes fields like model, pricePerDay, carType, availability, images, and features.
    - Track orders:
        - Schema for rental orders includes userId, carId, startDate, endDate, totalPrice, and status.

Custom APIs *(Built for Core Workflows)*:

2. **/api/cars**:
3. **Method:** GET
4. **Description:** Fetch all available cars.
5. **Query Parameters:**
6. priceRange, carType, availability (optional).
7. **Response Example:**

```
[
  {
    "id": "123",
    "model": "Tesla Model 3",
    "pricePerDay": 150,
    "carType": "Electric",
    "availability": true,
    "images": ["https://example.com/tesla.jpg"]
  }
]
```

- **/api/rent**:
  - **Method:** POST
  - **Description:** Create a rental order.
  - **Payload:**

```
{
 "userId": "456",
 "carId": "123",
 "startDate": "2025-01-18",
 "endDate": "2025-01-25",
 "totalPrice": 1050
}
```

  - **Response Example:**

```
{
 "orderId": "789",
 "status": "Confirmed",
 "message": "Your rental order has been placed successfully."
}
```

- **/api/track-order**:
  - **Method:** GET
  - **Description:** Retrieve the status of a rental order.
  - **Query Parameters:**
    - orderId.
  - **Response Example:**

```
{
 "orderId": "789",
 "status": "In Progress",
 "rentalDuration": "7 days",
 "startDate": "2025-01-18",
 "endDate": "2025-01-25"
}
```

- **/api/checkout**:
  - **Method:** POST
  - **Description:** Process payments through Stripe.
  - **Payload:**

```
{
 "orderId": "789",
 "amount": 1050,
 "currency": "USD",
 "paymentMethod": "card"
}
```

- **Response Example:**

```
{
  "status": "Payment Successful",
  "receiptUrl": "https://stripe.com/receipt/12345"
}
```

8. **Admin Panel** *(Managed via Sanity Studio)*:
   - **Car Management:**
     - Add, update, or delete cars from the inventory.
     - Fields: model, pricePerDay, carType, availability, features, images.
   - **Order Management:**
     - View all rental orders.
     - Update order statuses (e.g., confirmed, in progress, completed).

Schema Design (Sanity CMS)

Car Schema

```
{
  "title": "Car",
  "type": "document",
  "fields": [
    { "name": "model", "type": "string" },
    { "name": "pricePerDay", "type": "number" },
    { "name": "carType", "type": "string", "options": ["SUV", "Sedan", "Luxury", "Electric"] },
    { "name": "availability", "type": "boolean" },
    { "name": "images", "type": "array", "of": [{ "type": "image" }] },
    { "name": "features", "type": "array", "of": [{ "type": "string" }] }
  ]
}
```

Order Schema

```
{
  "title": "Order",
  "type": "document",
  "fields": [
    { "name": "userId", "type": "string" },
    { "name": "carId", "type": "string" },
    { "name": "startDate", "type": "date" },
    { "name": "endDate", "type": "date" },
    { "name": "totalPrice", "type": "number" },
    { "name": "status", "type": "string", "options": ["Pending", "Confirmed", "In Progress",
"Completed"] }
  ]
}
```

Security Considerations

- Authentication: Use Clerk to secure endpoints with user roles (admin vs. customer).
- Payment: Use Stripe's PCI-compliant hosted checkout for secure transactions.
- API Rate Limits: Implement rate-limiting to prevent abuse.
- Validation: Validate all user input for APIs to avoid malicious payloads.

Schema Overview

Car Schema (Sanity CMS)

- **Fields:** Model, price per day, car type, availability, images, features.

Order Schema (Sanity CMS)

- **Fields:** User ID, car ID, start/end dates, total price, status (Pending, Confirmed, In Progress, Completed).