Name: M.Abdullah **Roll no: 00433137** **Slot: Friday 7 to 10**

Hackathon Day 2:

Planning the Technical Foundation for Car Rental E-commerce Platform

Business Focus:

On Day 1, I laid the groundwork for my **Car Rental Website** by focusing on its **foundation**. Here's what I achieved:

1. Defined the Mission:

- I identified the **key challenges** my platform tackles, such as high car ownership costs and inconvenient rental processes.
- I outlined how my platform offers a **better solution**—affordable, flexible, and hassle-free car rentals.

2. Understood the Audience:

- I determined who my platform serves:
  - Busy professionals needing quick rentals.
  - Tourists looking for reliable travel options.
  - Families and individuals needing cars for special occasions.
- I also considered businesses requiring short-term rentals for events or employees.

3. Highlighted What Makes My Platform Unique:

- I emphasized my platform's **standout features**:
  - Transparent pricing with no hidden fees.
  - Easy online booking with minimal paperwork.

- A wide range of cars, including eco-friendly options.
- 24/7 customer support for a seamless experience.

4. **Created a Data Blueprint:**

- I drafted a simple yet effective **data schema** to map out how key elements like **Cars**, **Customers**, and **Bookings** interact.

By focusing on these aspects, I ensured my marketplace is built on a **clear, purpose-driven foundation** that aligns with real-world needs.

Day 2:

Transitioning to Technical Planning For E-commerce Car Rental Platform

1. Define Technical Requirements

Technical Requirements outline the technical specifications for building and maintaining a system, including frontend design, backend logic & database management. They ensure alignment, reduce risks, and support scalable solutions.

My platform's technical requirements include a secure backend, responsive frontend, API database which are given below:

➢ Frontend Requirements :

- Intuitive and user-friendly interface for easy car exploration and booking.
- Fully responsive design, ensuring seamless performance on mobile and desktop.
- Key pages include: Homepage, Car Listings, Car Details, Reservation, Payment, and Booking Confirmation.

➢ Sanity CMS for Backend:

It will Manage Car Details

- **What it does :**

  Stores all information about the cars available for rent.

- **Example Schema:**

```
export default {
  name: 'car',
  type: 'document',
  fields: [
    { name: 'name', type: 'string', title: 'Car Name' },
    { name: 'type', type: 'string', title: 'Car Type' },
    { name: 'price', type: 'number', title: 'Price per Day' },
    { name: 'availability', type: 'boolean', title: 'Availability' },
    { name: 'image', type: 'image', title: 'Car Image' }
  ]
};
```

- **How it works:**

  When a user searches for cars, the frontend fetches data from this schema.

- **Example data:**

```
 {
"name": "Toyota Corolla",
 "type": "Sedan",
"price": 50,
 "availability": true,
"image": "https://example.com/car1.jpg"
 }
```

❖ Handle Customer Information

- **What it does:**

  Stores details about users who rent cars.

- **Example Schema:**

```
export default {
name: 'customer',
type: 'document',
fields: [
{ name: 'name', type: 'string', title: 'Full Name' },
{ name: 'email', type: 'string', title: 'Email' },
```

```
{ name: 'phone', type: 'string', title: 'Phone Number' },
{ name: 'address', type: 'string', title: 'Address' },
{ name: 'license', type: 'string', title: 'Driving License' }
]
};
```

- **How it works:**

  When a user signs up or books a car, their details are saved here.

- **Example data:**

```
{
  "name": "John Doe",
  "email": "john@example.com",
  "phone": "+123456789",
  "address": "123 Main St, City",
  "license": "DL123456789"
}
```

❖ Track Booking Records

- **What it does:**

  Stores all rental transactions.

- **Example Schema:**

```
export default {
 name: 'booking',
 type: 'document',
 fields: [
   { name: 'car', type: 'reference', to: [{ type: 'car' }], title: 'Car' },
   { name: 'customer', type: 'reference', to: [{ type: 'customer' }], title: 'Customer' },
   { name: 'startDate', type: 'date', title: 'Start Date' },
   { name: 'endDate', type: 'date', title: 'End Date' },
   { name: 'status', type: 'string', title: 'Booking Status' }
 ]
};
```

- **How it works:**

  When a user books a car, the details are saved here.

- **Example data:**

```
{
```

```
  "car": { "name": "Toyota Corolla" },
  "customer": { "name": "John Doe" },
  "startDate": "2025-01-20",
  "endDate": "2025-01-25",
  "status": "Confirmed"
}
```


Why Use Sanity CMS?

- **Flexible and Easy:** Sanity allows me to create custom schemas that fit my website's needs.
- **Real-Time Updates:** Any changes to car details, customer info, or bookings are updated instantly.
- **Scalable:** As my website grows, Sanity can handle more data without slowing down.

➢ Third Party APIs

For my car rental project **,** utilize APIs to enable to cars, booking , Payments & Shipment-tracking.

❖ User Sign-Up

- **What Happens:**
  - A new user creates an account on my platform.
  - Their information (name, email, phone, address, and driving license) is stored securely in **Sanity CMS**.
  - They receive a confirmation message to verify their account.

❖ Exploring Cars

- **What Happens:**
  - Users browse through the list of available cars on my website.
  - The frontend fetches car details (like name, type, price, and availability) from the **Car Data API** (powered by Sanity CMS).
  - The cars are displayed with filters for easy searching.

❖ Booking a Car

- **What Happens:**
    - A user selects a car, chooses rental dates, and provides their details.
    - The frontend sends this information to Sanity CMS via the **Booking API**.
    - The booking is saved, and the user gets a confirmation.

❖ Making Payments

- **What Happens:**
    - The user enters their payment details (e.g., card number, amount).
    - The frontend sends this data to the **Payment Gateway API** for processing.
    - Once the payment is successful, the user receives a confirmation.

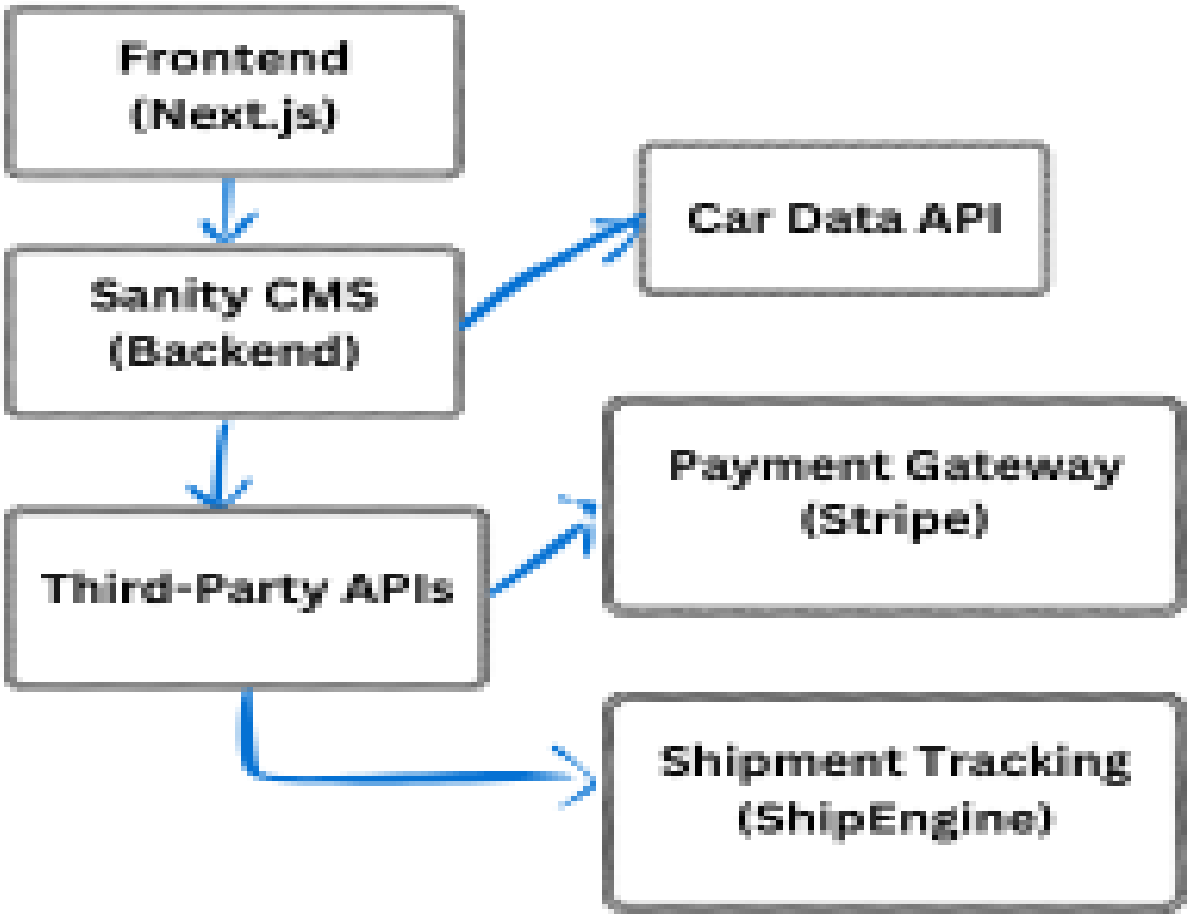
❖ Shipment Tracking

- **What Happens:**
    - If the car is being delivered, the frontend fetches real-time updates from the **Shipment Tracking API**.
    - The user can see the car's current status and estimated arrival time.

Why These Workflows Are Perfect for My Website?

- **Simple and Clear:** Every step is designed to be easy for users to follow.
- **Fast and Smooth:** Data moves quickly between the frontend, Sanity CMS, and third-party APIs.
- **Trustworthy:** All actions (sign-up, booking, payment, tracking) are handled securely and reliably.

2. System Architecture for My Car Rental Website



How It Works

1. **User Sign-Up:**
    - A new user creates an account.
    - Their details (name, email, phone, license) are saved in **Sanity CMS**.
    - They receive a confirmation message.
2. **Exploring Cars:**
    - Users browse available cars on the website.
    - The frontend fetches car details (e.g., name, price, availability) from **Sanity CMS**.
    - Cars are displayed with filters for easy searching.
3. **Booking a Car:**
    - A user selects a car and chooses rental dates.
    - Booking details are sent to **Sanity CMS** and saved.
    - The user gets a booking confirmation.
4. **Making Payments:**
    - The user enters payment details (e.g., card number, amount).
    - The frontend sends this data to the **Payment Gateway API** (Stripe).
    - Once the payment is successful, the user receives a confirmation.
5. **Tracking Delivery:**
    - If the car is being delivered, the frontend fetches real-time updates from the **Shipment Tracking API** (ShipEngine).
    - The user can see the car's current status and estimated arrival time.

Why This Works for My Website?

- **Simple and Clear:** Every step is easy for users to follow.
- **Fast and Smooth:** Data moves quickly between the frontend, Sanity CMS, and third-party APIs.
- **Trustworthy:** All actions (sign-up, booking, payment, tracking) are handled securely and reliably.

3. Plan APIs Requirements and Methods :

API Name	Method	Purpose Example	Request/Response
/cars	GET	Fetch all available cars.	**Request:** GET /cars **Response:** [{ "id": 1, "name": "Toyota Corolla", ... }]
/bookings	POST	Create a new booking.	**Request:** POST /bookings **Response:** { "bookingId": 123, "status": "Confirmed" }
/payments	POST	Process payment for a booking.	**Request:** POST /payments **Response:** { "paymentId": 789, "status": "Paid" }
/shipment-tracking	GET	Track the status of a booking.	**Request:** GET /shipment-tracking?bookingId=123 **Response:** { "status": "In Transit", "ETA": "30 mins" }