Name:- SHOAIB TAHIR

Hackaton Day 2:

Planning The Technical Foundation

1) Define Technical Requirements

This document focusing on three key areas:-

1) Frontend Requirements:

• User-friendly interface & Responsive design:

Easy to use design for finding and booking cars which works well on both phones & Computers.

• Key Pages:

* Home Page: Welcome page with car options & promotions
* Car Listing Page: Show all available cars and allow users to search and filter cars by price, model or location.
* Car Details Page: Info about each car. Such as images, specifications, rental price and availability.
* Booking Page: Users to select rental duration & fill in their details
* Check out page: Enter payment & booking details.
* Order confirmation: Display success message and booking details.

*) Sanity CMS as Backend

• Use Sanity CMS to manage:

* Car Data: Manage all car details (such as model,

price, images & availability)
* Customer Details: Store user details like name, contact info, and past bookings.
* Booking Records: Record every rental transaction, including payment status.
· Design schemas in Sanity to match business goals.

Third-Party APIs (External Services)
* Payment gateway: Use a secure service to handle payments.
* Shipment Tracking: For any car delivery services.
* Map Services: Show rental locations and directions using Google Maps or similar.
* Notification: Send email or SMS updates for booking confirmation.
* Other services: Any additional backend needs.
· Ensure APIs provide necessary data for frontend

2) Design System Architecture:
· How system will work:

· Frontend (website)
* Built using Next.js to create a user-friendly design.
* It will request data from Sanity CMS or other APIs.

- Backend:
* Sanity CMS will store all data, like car details, users, and bookings.

- Third-Party APIs:
* Payment services to process money transaction.
* Map services to show car locations.

- Typical Data Flow:
* The user opens the website and browses cars.
* The website fetches data from Sanity CMS to show the car list and details.
* When the user books a car, the booking is saved in Sanity CMS via API request.
* Payment is processed securely, and the user receives a confirmation email or SMS.

- Key Workflows to include:
* User Registration:
- User signsup → Data stored in Sanity → Confirmation sent to the user.
* Car Browsing:
- User views cars categories → Sanity API fetches data → Cars displayed

* Booking Placement:
. User adds cars to Cart → Proceeds to checkout → Booking details saved in Sanity.
* Shipment Tracking:
Booking status updates fetched via 3rd-party API → Displayed to the user.

3) Plan API Requirements
• API End Points:
* End Point Name: /cars
* Method: GET
* Description: Fetch all available cars from Sanity.
* Response: Car details (ID, name, model, year, price per day, availability, image).
* e.g [{"id":1, "name": "Toyotta corolla", "model": 2020, "price": 5000, "availibity": "Available, "image": [image]]

* End Point Name: /bookings
* Method: POST
* Description: Create a new booking in Sanity.
* payload: customer info, car details, booking dates

* e.g. {"CarID": 1, "user ID": 701, "duration": "3 days", "price": 15000}

* e.g {"user ID": 101, "booking ID": 301, "amount": 15000}
{"transaction ID": 401, "status": "success"}

* Endpoint Name: /locations
* Method: Get
* Description: Fetches car rental locations
* e.g [{"Location ID": 1, "name": "city center", "distance": "5km"}],

* Endpoint Name: /payments
* Method: POST
* Description: Process payment
* e.g {"user ID": 101, "boking ID": 301 "amount": 15000}
{"transaction ID": 401, "status": "success"}

4) Write Technical Documentation:
- What to include in documentation:

* System Architecture: Create a simple diagram to show how the website, CMS, and APIs work together.
* API Details: Write down all API endpoints, how they work, and example responses.

* Work flows:
- Explain how users register, browse cars, book cars, and make payments
- Show these steps visually with arrows or flow charts.

* Data Design:
- Describe all data, like cars, users, and Bookings.
- Show how they connect (eg cars link to bookings through carsID).

* Plan the Steps:
- Step 1: Build the website
- Step 2: Setup data in sanity CMS
- Step 3: Connect APIs for payment and notifactions
- Step 4: Testing everything to make sure it works.

Sanity Schema:

```
export default {
  name: 'car',
  type: 'document',
  title: 'Car',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Car Name',
    },
    {
      name: 'brand',
      type: 'string',
      title: 'Brand',
      description: 'Brand of the car (e.g., Nissan, Tesla, etc.)',
    },
    {
      name: 'type',
      type: 'string',
      title: 'Car Type',
      description: 'Type of the car (e.g., Sport, Sedan, SUV, etc.)',
    },
    {
      name: 'fuelCapacity',
      type: 'string',
      title: 'Fuel Capacity',
      description: 'Fuel capacity or battery capacity (e.g., 90L, 100kWh)',
    },
    {
      name: 'transmission',
      type: 'string',
      title: 'Transmission',
      description: 'Type of transmission (e.g., Manual, Automatic)',
    },
    {
      name: 'seatingCapacity',
      type: 'string',
      title: 'Seating Capacity',
      description: 'Number of seats (e.g., 2 People, 4 seats)',
    },
    {
      name: 'pricePerDay',
      type: 'string',
      title: 'Price Per Day',
      description: 'Rental price per day',
    },
    {
      name: 'originalPrice',
      type: 'string',
      title: 'Original Price',
      description: 'Original price before discount (if applicable)',
    },

  {
      name: 'tags',
      type: 'array',
      title: 'Tags',
      of: [{ type: 'string' }],
      options: {
        layout: 'tags',
      },
      description: 'Tags for categorization (e.g., popular, recommended)',
    },
    {
      name: 'image',
      type: 'image',
      title: 'Car Image',
      options: {
        hotspot: true
      }
    }
  ],
};
```