DAY 2

PLANNING THE TECHNICAL FOUNDATION

Rental e-Commerce

Rental Car

Technical Requirements

1- Framework

- **Next.Js** – I choose next.js for building my **Rental Car Website** because it has great features specialy for E-commerce plateforms where SEO is crucial.

2- Frontend Simple and User friendly

- **Styling** – for style my user interface I choose,
  Tailwind CSS Utility-first CSS framework for quick and responsive UI design
- **Libraries** I use choose **shadcn ui** for hamburgermenu and **React** for different icons.
- **Pages and Components** In this website Home-page, Category page,Details Car Rent-page, Payment Car Rent-page, Admin-page. Components- Navbar, Hero,Card,Header, Footer, Review and more as per Web requirements.

3- Beckend

- **Sanity**: Headless CMS for managing content like car descriptions, images, and offers.
- **API integration**
  Stripe used For Secure Payment process.
  Custom APIs used for Tracking.
  Clerk used for Authentication.

4- Future Plan I want to incorporated AI features to my car rental website that gives it differentiate my business from my competitors. Below are some next-level AI-powered features

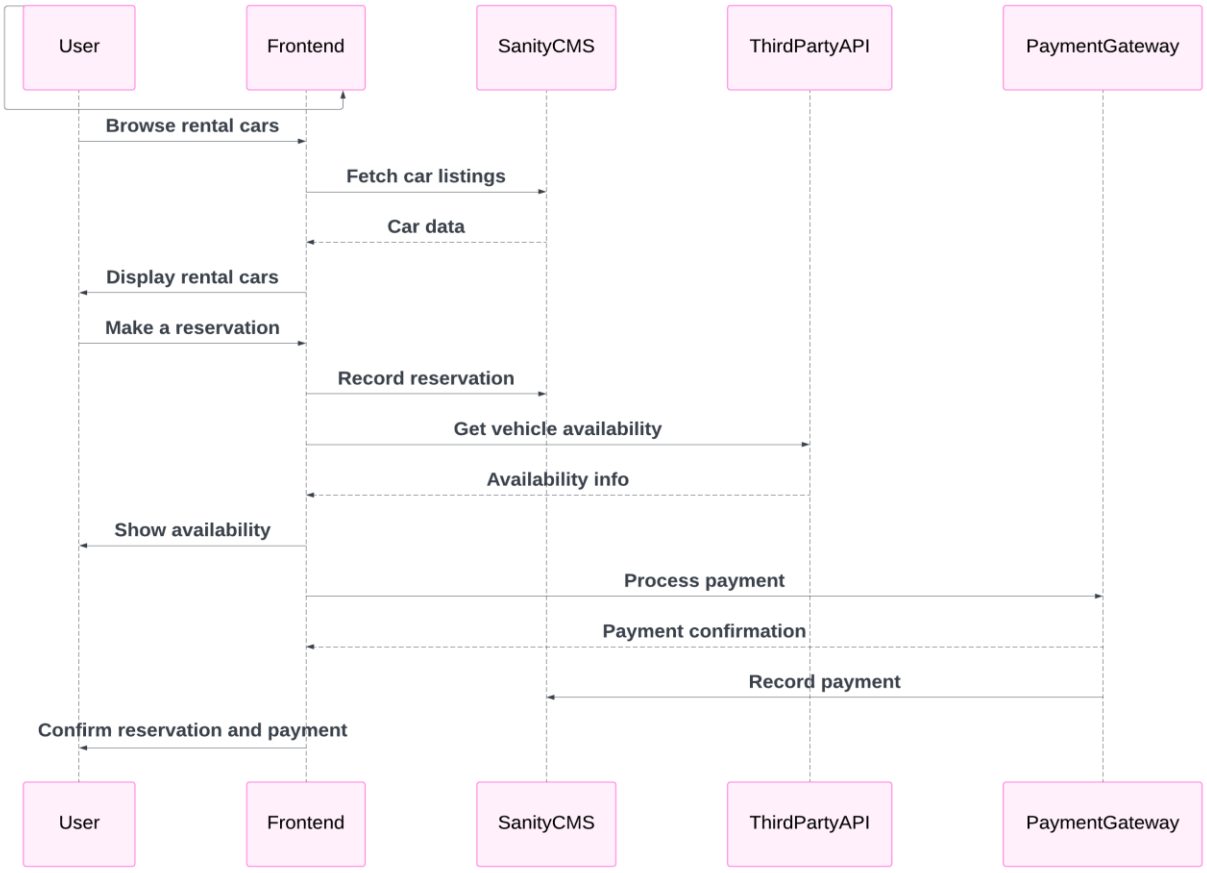
Intelligent Chatbot Provide instant customer support, answer FAQs, and guide users through booking.

- Integrate AI-powered chatbots like Dialogflow or OpenAI's GPT models.
- Train the bot on your car rental FAQs, policies, and workflows.

Smart Search Allow users to search for cars using natural language (native language)

- Use NLP (Natural Language Processing) models like OpenAI's GPT
- Combine it with your search functionality to parse user queries into actionable filters.

Design System Architecture



Workflow

1. **User Registration ->**The user creates an account by signing up with their name, email, and password**->** Saves the user's data in the database (e.g., Sanity CMS) **->**A confirmation email is sent, and the user can log in.

⬇

2. **Car Browsing ->**The user browses available cars by category (e.g., SUVs, Sedans). **->**Fetches car details from the database and displays them on the website. **->**The user sees the cars with pricing, availability, and features.

3. Booking a Car->The user selects a car, chooses rental dates, and proceeds to checkout. **->**System: Saves the booking details (car, dates, user info) in the database and processes payment via a payment gateway (e.g., Stripe)**->**Response: A booking confirmation is shown to the user.

4. Car Tracking->The user checks the status of their rental (e.g., "Confirmed", "In Progress").

-> Fetches tracking updates from the database or a 3rd-party API (e.g., for pickup and return details **->** The user sees the current status and updates.

Plan API Requirements

Endpoint	Method	Purpose	Response
/products	GET	Fetch all available products.	{"productId": 123, "name": "Car A", "type": "SUV", "pricePerDay": 50, "availability": true, "location": "New York"}
/product/:id	GET	Fetch details for a specific product.	"productId": 123, "name": "Car A", "type": "SUV", "pricePerDay": 50, "availability": true, "location": "New York", "features": ["Automatic", "Air Conditioning", "GPS"]
/product	POST	Add a new product to the catalog.	**Payload Example**: "name": "Car C", "type": "Truck", "pricePerDay": 60, "availability": true, "location": "Chicago", "features": ["Manual", "Air Conditioning"] **Response Example**: `{ "productId": 125, "status": "Created" }`
/rental-duration	POST	Add rental details for a specific product.	**Payload Example** "productId": 456, "duration": "7 days", "deposit": 500

			Response Example: "confirmationId": 789, "status": "Success"
/rental/:id	GET	Get details of a specific rental booking.	"rentalId": 789, "productId": 456, "userId": 101, "duration": "7 days", "totalCost": 350, "status": "Active"
/rental	PUT	Update rental details (e.g., duration or status).	**Payload Example** "rentalId": 789, "duration": "10 days", "status": "Extended" **Response Example**: "status": "Updated", "newTotalCost": 500
/users	POST	Register a new user.	"userId": 101, "status": "Created"
/payment	POST	Process a payment for a rental booking.	"rentalId": 789, "amount": 350, "paymentMethod": "credit_card", "cardDetails": { "number": "4242424242424242", "expiry": "12/25", "cvv": "123" { "paymentId": 456, "status": "Success" }
/search	GET	Search for products based on location, type, or price range.	"productId": 123, "name": "Car A", "type": "SUV", "pricePerDay": 50, "availability": true, "location": "New York"
/admin/rentals	GET	Fetch all rental bookings for admin management.	[{ "rentalId": 789, "productId": 456, "userId": 101, "status": "Active" }, { "rentalId": 790, "productId": 457, "userId": 102, "status": "Completed" }]

Sanity Schema for Rental Car

```
export default {

        name: 'rentalCar',
        type: 'document',
        title: 'Rental Car',
 fields: [

  {

    name: 'name',
    type: 'string',
```

```
  title: 'Car Name',
  description: 'The name of the car (e.g., Toyota Corolla)',
  validation: Rule => Rule.required(),
},

{

  name: 'type',
  type: 'string',
  title: 'Car Type',
  options: {

    list: [

      { title: 'SUV', value: 'suv' },
      { title: 'Sedan', value: 'sedan' },
      { title: 'Truck', value: 'truck' },
      { title: 'Hatchback', value: 'hatchback' },
    ],
  },

     validation: Rule => Rule.required(),

},

{

  name: 'pricePerDay',
  type: 'number',
  title: 'Price Per Day',
  description: 'Rental price per day in USD',
  validation: Rule => Rule.min(0).required(),
},

{

  name: 'availability',
  type: 'boolean',
  title: 'Availability',
  description: 'Is the car available for rent?',
},

{

  name: 'location',
  type: 'string',
  title: 'Location',
  description: 'Location where the car is available for rent',
},
```

```
  {
    name: 'features',
    type: 'array',
    title: 'Features',
    of: [{ type: 'string' }],
    description: 'Features of the car (e.g., Air Conditioning, GPS)',
  },
  {


    name: 'image',
    type: 'image',
    title: 'Car Image',
    description: 'Upload an image of the car',
    options: {
      hotspot: true,
    },
  },
  {
    name: 'description',
    type: 'text',
    title: 'Description',
    description: 'Detailed description of the car',
  },
 ],
};
```

Conclusion

The rental car website is built with Next.js for performance and SEO, styled using Tailwind CSS, and includes key pages like Home, Category, and Admin. The backend uses Sanity CMS for content management, Stripe for payments, and Clerk for user authentication. The workflow includes user registration, car browsing, booking, and tracking, with APIs designed for managing products, rentals, and payments efficiently.