Hackathon Day 2
Marketplace Technical Foundation

"Frontend Requirement: Flow Chart"

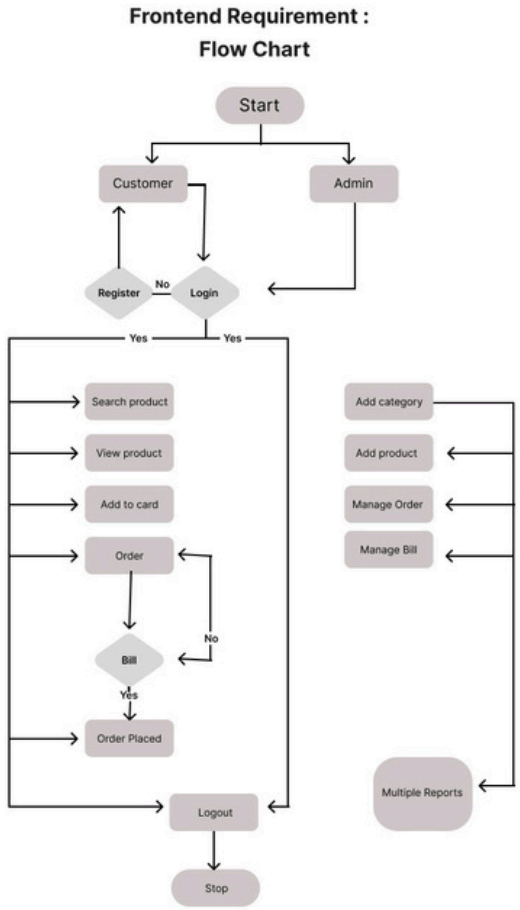
For Customers:

- **Register or Log In:** Customers can either register or log in.
- **Search and View Products:** Once logged in, customers can search for products and view them.
- **Add to Cart and Place Order:** Customers can add products to their cart and place an order.
- **Check Bill:** The order process involves checking the bill. If the bill is confirmed, the order is placed.
- **Log Out:** After placing the order, customers can log out, leading to the "Stop" node.

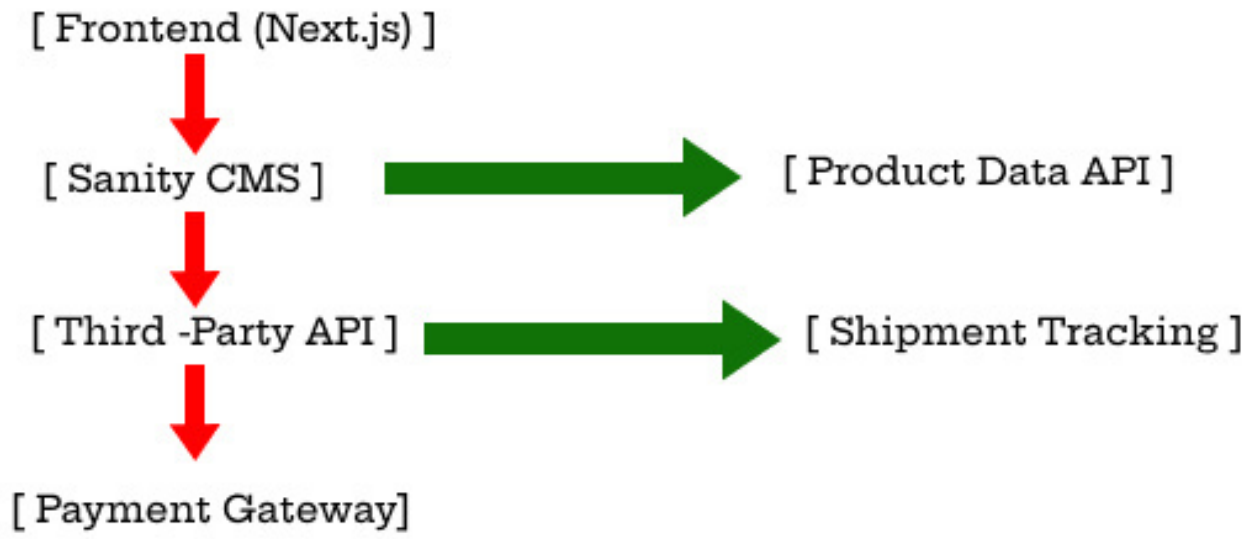
For Admins:

- **Add Categories and Products:** Admins can add categories and products.
- **Manage Orders and Bills:** Admins can manage orders and bills.
- **Generate Reports:** Admins can generate multiple reports.

This flow chart visually represents the steps and decision points in the frontend requirements for both customers and admins, making it easier to understand the process flow.



System Architecture Overview



Frontend (Next.js):

1. Displays the user interface for browsing products, managing the cart, and placing orders.

2. Handles user interactions and communicates with backend services via APIs.

Sanity CMS:

1. Acts as the primary backend to manage product data, customer details, and order records.

2. Provides APIs for the frontend to fetch and update data.

Product Data API:

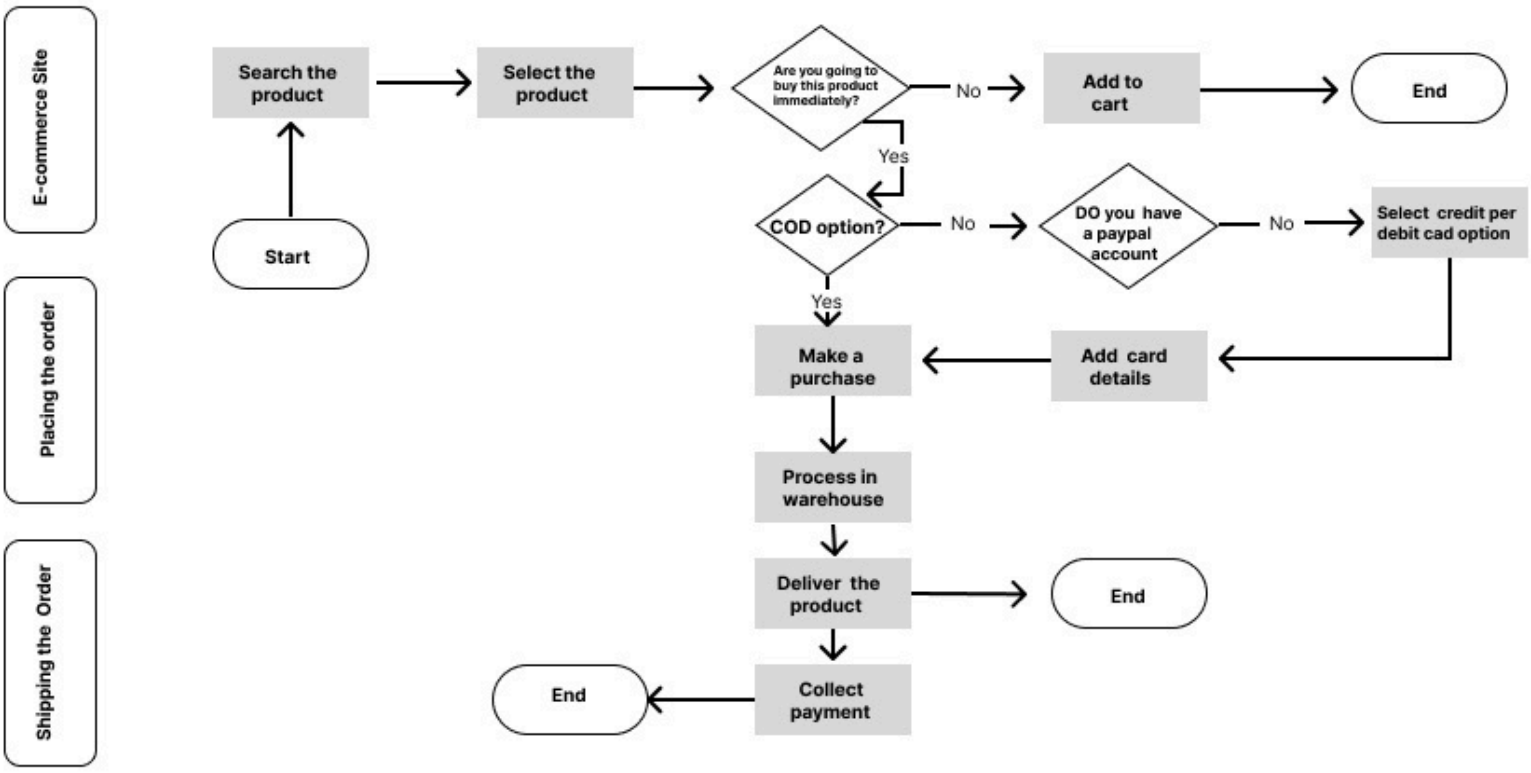
Provides endpoints to fetch product listings, details, and inventory status.

Third-Party APIs:

Integrates services like shipment tracking and payment processing.

Payment Gateway:

Processes user payments securely and provides transaction confirmation.



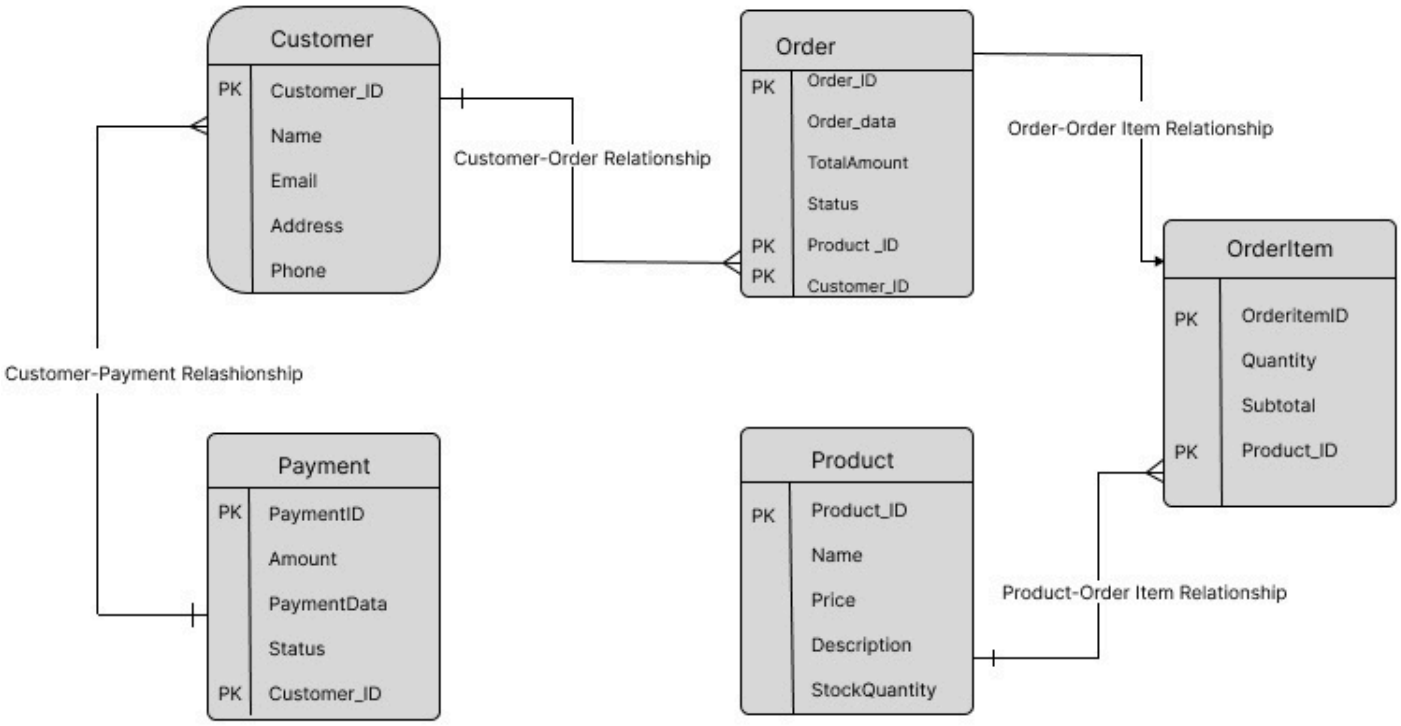
ER DIAGRAM

For Car Rental E-Commerce:

1. Cars:
   - Fields: id, name, price, description, image, categoryId.
2. Categories:
   - Fields: id, name.
3. Users:
   - Fields: id, name, email, password.
4. Bookings:
   - Fields: id, userId, bookingDate, status.

This schema represents the entities and their respective fields for a car rental e-commerce system. It includes cars, categories, users, and bookings, making it easier to manage and organize the data.

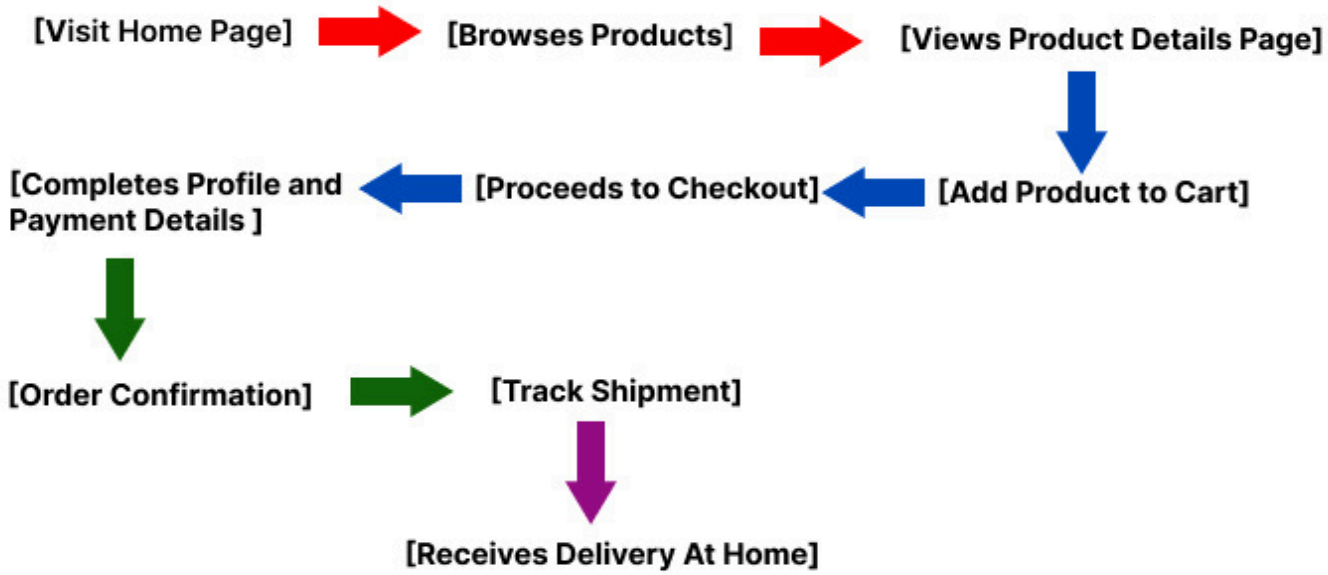
ER Diagram For E-Commerce Project



API Endpoint

Endpoint	Method	Description	Parameters	Response Example
/api/cars	GET	Fetch all available cars	None	{ id: 1, make: "Toyota", model: "Camry" }
/api/cars/:id	GET	Fetch details of a single	id (Path)	{ id: 1, make: "Toyota", model: "Camry" }
/api/cars	POST	Add a new car to the in	make, model, price, availability (Body)	{ success: true, id: 5 }
/api/cars/:id	PUT	Update car details	id (Path), make, model, price, availability (Body)	{ success: true }
/api/cars/:id	DELETE	Delete a car from the ir	id (Path)	{ success: true }
/api/bookings	GET	Fetch all bookings	None	{ bookings: ["Booking1", "Booking2"] }

Workflow Diagram Visual Representation:



sanitySchema.js

```
export default {
 name: 'carRental',
 title: 'Car Rental',
 type: 'document',
 fields: [
  {
   name: 'carName',
   title: 'Car Name',
   type: 'string',
   description: 'Name or identifier for the car (e.g., Toyota Camry).',
   validation: (Rule) => Rule.required().error('Car name is required.'),
  },
  {
   name: 'slug',
   title: 'Slug',
   type: 'slug',
   options: {
    source: 'carName',
    maxLength: 100,
   },
   description: 'URL-friendly identifier for the car.',
   validation: (Rule) =>
    Rule.required().error('Slug is required for car identification.'),
  },
  {
   name: 'description',
   title: 'Description',
   type: 'text',
   description: 'Detailed description of the car, including features and specifications.',
   validation: (Rule) => Rule.max(500).error('Description cannot exceed 500 characters.'),
  },
  {
   name: 'rentalPricePerDay',
   title: 'Rental Price Per Day',
   type: 'number',
   description: 'Daily rental price for the car in USD.',
   validation: (Rule) =>
    Rule.required()
     .positive()
     .error('Rental price must be a positive value.'),
  },
  {
   name: 'discount',
   title: 'Discount Percentage',
   type: 'number',
   description: 'Discount percentage (if applicable).',
   validation: (Rule) =>
    Rule.min(0)
     .max(100)
     .error('Discount must be between 0 and 100.'),
  },
  {
   name: 'availability',
   title: 'Availability',
   type: 'boolean',
   description: 'Indicates if the car is available for rent.',
  },
  {
   name: 'features',
   title: 'Features',
   type: 'array',
   of: [{ type: 'string' }],
   description: 'Key features of the car (e.g., GPS, Air Conditioning, Automatic Transmission).',
  },
  {
   name: 'numberOfSeats',
   title: 'Number of Seats',
   type: 'number',
   description: 'Number of seats in the car.',
   validation: (Rule) =>
    Rule.required()
     .positive()
     .integer()
     .error('Number of seats must be a positive integer.'),
  },
  {
   name: 'fuelType',
   title: 'Fuel Type',
   type: 'string',
   options: {
    list: [
     { title: 'Petrol', value: 'petrol' },
     { title: 'Diesel', value: 'diesel' },
     { title: 'Electric', value: 'electric' },
     { title: 'Hybrid', value: 'hybrid' },
    ],
   },
   description: 'Fuel type of the car.',
  },
  {
   name: 'transmission',
   title: 'Transmission',
   type: 'string',
   options: {
    list: [
     { title: 'Manual', value: 'manual' },
     { title: 'Automatic', value: 'automatic' },
    ],
   },
   description: 'Type of transmission (manual or automatic).',
  },
  {
   name: 'images',
   title: 'Images',
   type: 'array',
   of: [{ type: 'image' }],
   description: 'Images of the car.',
   validation: (Rule) =>
    Rule.required().error('At least one image of the car is required.'),
  },
  {
   name: 'pickupLocation',
   title: 'Pickup Location',
   type: 'string',
   description: 'Default pickup location for the car rental.',
  },
  {
   name: 'rentalPolicies',
   title: 'Rental Policies',
   type: 'text',
   description: 'Rental policies and terms for the car.',
   validation: (Rule) => Rule.max(1000).error('Rental policies cannot exceed 1000 characters.'),
  },
],
}
```