Marketplace Technical Foundation - [Car Rental Ecommerce Website]

1. System Architecture Overview

Architecture Diagram

```
[Frontend (Next.js)]
      |
      V
[Sanity CMS] -----> [Product Data API]
      |
      V
[Third-Party APIs] -----> [Shipment Tracking API]
      |
      V
[Payment Gateway]
```

Component Roles

- **Frontend (Next.js)**: Displays the user interface (UI) for browsing cars, placing rentals, and managing payments. Handles user input and API calls.
- **Sanity CMS**: Manages car data, user information, and rental records. Acts as the central database.
- **Third-Party APIs**: Provides real-time updates for shipment tracking and other functionalities.
- **Payment Gateway**: Processes payments securely and returns transaction statuses.

2. Key Workflows

Workflow 1: Browsing Cars

1. User navigates to the homepage.
2. Frontend sends a GET request to `/cars` API endpoint.
3. Sanity CMS fetches the car data and returns it.
4. Frontend displays car listings dynamically.

Workflow 2: Renting a Car

1. User selects a car and provides rental details (e.g., duration, pickup/drop-off location).
2. Frontend sends a POST request to `/rentals` with rental data.
3. Sanity CMS stores the rental record and returns a confirmation response.

Workflow 3: Tracking Rental Status

1. User views rental details on their profile.
2. Frontend sends a GET request to `/rentals/:orderId/status`.
3. Third-party API fetches the status and returns it to the frontend.

3. Category-Specific Instructions

Rental eCommerce

Focus areas for rentals include:

- **Rental Duration**: Allow users to select and modify rental periods.
- **Condition Reports**: Enable condition tracking for cars before and after rentals.
- **Return Management**: Provide a smooth workflow for vehicle returns.

Example Schema Field:

```
{
  "rentalDuration": "7 days",
  "depositAmount": 500,
  "conditionStatus": "Good"
}
```

4. API Endpoints

Endpoint	Method	Purpose	Response Example
`/cars`	GET	Fetches all available cars	`{ "id": 123, "name": "Sedan X", "rentPerDay": 70 }`
`/cars/:id`	GET	Fetches details of a specific car	`{ "id": 123, "name": "Sedan X", "fuelCapacity": 40, "transmissionType": "Automatic" }`
`/rentals`	POST	Creates a rental order	`{ "orderId": "o456", "status": "Success" }`
`/rentals/:orderId`	GET	Fetches rental details	`{ "orderId": "o456", "status": "In Progress" }`
`/rentals/:orderId/status`	GET	Tracks rental status	`{ "status": "In Progress", "expectedDropoffDate": "2025-01-21" }`
`/payments`	POST	Processes rental payments	`{ "paymentId": "p789", "confirmation": "Success" }`

5. Sanity Schema Example

Car Schema

```
export default {
  name: 'car',
  type: 'document',
  fields: [
    { name: 'name', type: 'string', title: 'Car Name' },
    { name: 'category', type: 'string', title: 'Category' },
    { name: 'rentPerDay', type: 'number', title: 'Rent Per Day' },
    { name: 'fuelCapacity', type: 'number', title: 'Fuel Capacity' },
    { name: 'seats', type: 'number', title: 'Seats' },
    { name: 'transmissionType', type: 'string', title: 'Transmission Type'
},
    { name: 'ratings', type: 'number', title: 'Ratings' },
    { name: 'image', type: 'image', title: 'Car Image' }
  ]
};
```

Rental Schema

```
export default {
  name: 'rental',
  type: 'document',
  fields: [
    { name: 'orderId', type: 'string', title: 'Order ID' },
    { name: 'userId', type: 'string', title: 'User ID' },
    { name: 'carId', type: 'string', title: 'Car ID' },
    { name: 'pickupLocation', type: 'string', title: 'Pickup Location' },
    { name: 'dropoffLocation', type: 'string', title: 'Dropoff Location' },
    { name: 'rentalDuration', type: 'string', title: 'Rental Duration' },
    { name: 'totalCost', type: 'number', title: 'Total Cost' },
    { name: 'status', type: 'string', title: 'Status' }
  ]
};
```

6. Technical Roadmap

Milestone	Details	Deadline
Create Sanity CMS Schemas	Implement schemas for cars, rentals, and users	Day 3
Develop Frontend UI	Build responsive pages for browsing and rentals	Day 4–5
Integrate APIs	Connect frontend with Sanity and third-party APIs	Day 5–6
Test System	Test all workflows end-to-end	Day 7