Hackathon Day 2: Planning the Technical Foundation

1. Technical Requirements

The primary objective for "**Rent a Car**" on Day 2 is to translate the business goals into actionable technical requirements.

The following are the defined technical requirements:

Frontend Requirements

- User-friendly interface for browsing and booking cars.
- Responsive design for both mobile and desktop platforms.
- Essential pages: Home, Car Listing, Car Details, Booking, Checkout, and Confirmation.

Backend Requirements

- Sanity CMS will manage data for cars, users, bookings, and providers.
- Integration of APIs for payment processing, booking management, and car availability.

APIs

- Third-party APIs will handle payment gateway, real-time tracking (if needed), and userauthentication.

2. System Architecture

The following is a high-level system architecture for "**Rent a Car":**

- **Frontend (Next.js)**: User interaction for browsing cars and managing bookings.

- **Sanity CMS**: Acts as the backend to manage car data, users, and bookings.
- **Third-Party APIs**
- Payment Gateway for secure transactions.
- Optional tracking services for car availability or delivery logistics.

3. Flowcharts and Workflow Explanation

The workflows for the platform include:

1. User Registration and Login

- User submits registration form -> Data stored in CMS -> Confirmation sent to the user.

2. Car Browsing

- User views car categories **->** CMS API fetches car data **->** Displayed dynamically.

3. Booking Process

- User selects a car -> Proceeds to checkout -> Booking details saved in CMS.

4. Payment and Confirmation

- Payment processed via Payment Gateway -> Confirmation sent -> Booking status updated**.**

Detailed flowcharts will visualize these workflows.

4. API Design

API endpoints required for the platform: you

1. Fetch Cars

- Endpoint: /cars
- Method: GET
- Response: { "CarID": 1, "Model": "Toyota Prius", "PricePerDay": 50 }

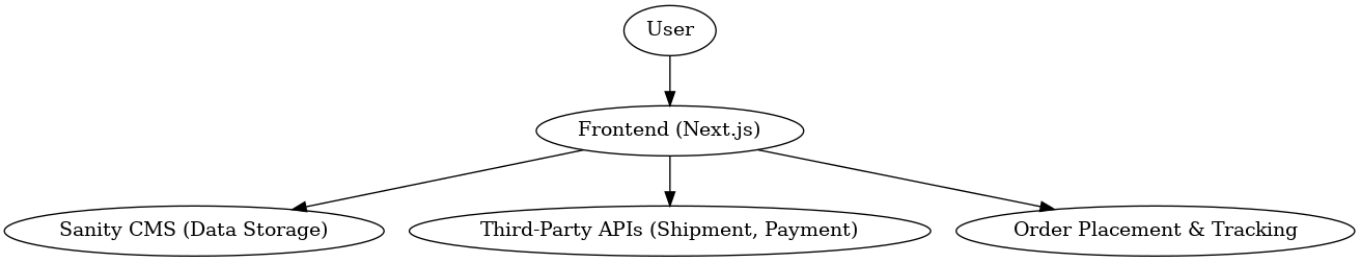
2. Create Booking

- Endpoint: /bookings
- Method: POST
- Payload: { "UserID": 1, "CarID": 2, "StartDate": "2025-01-20", "EndDate": "2025-01-25" }

3. Process Payment

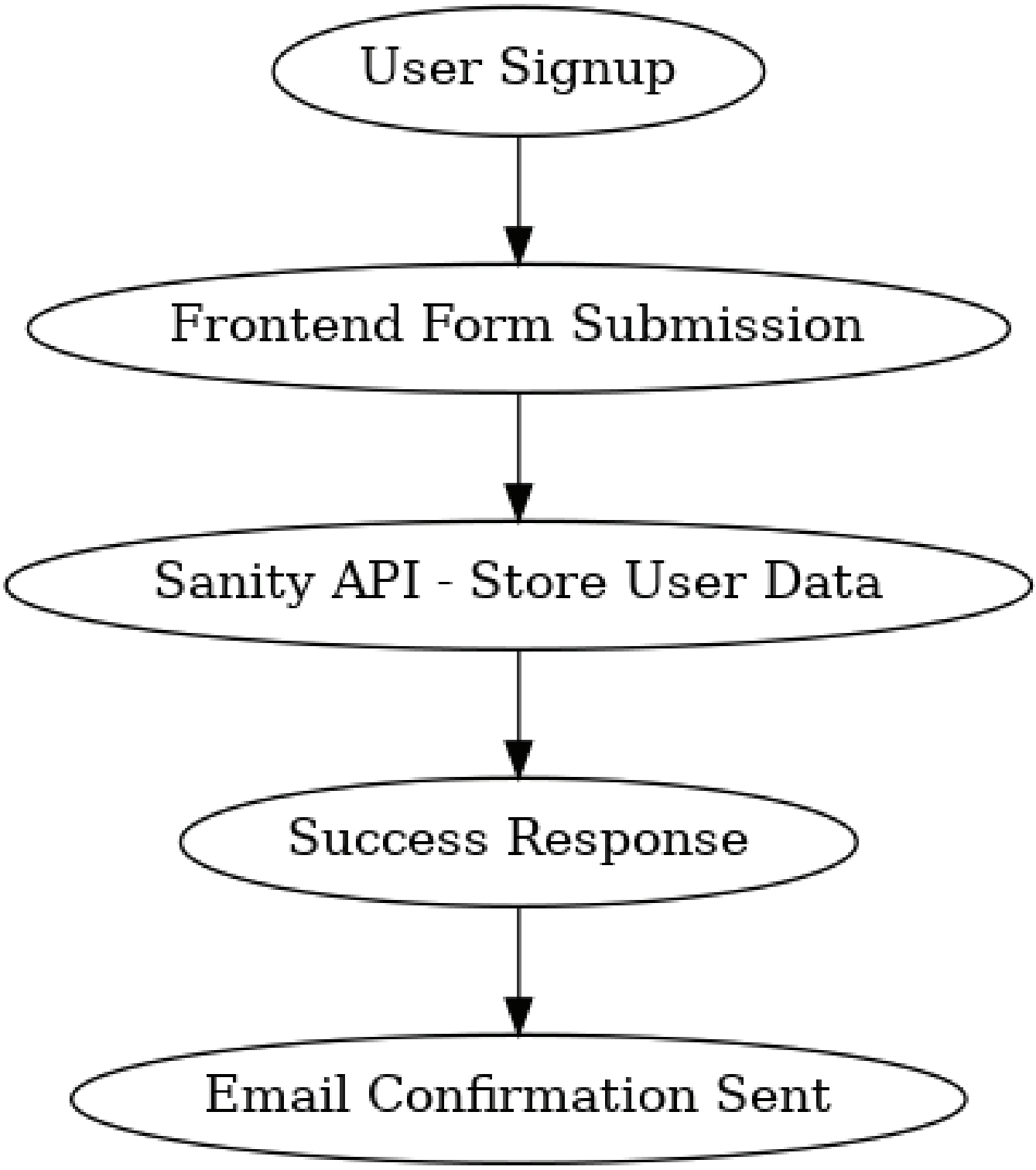
- Endpoint: /payment
- Method: POST
- Payload: { "BookingID": 101, "Amount": 250 }

System Architecture Flowchart



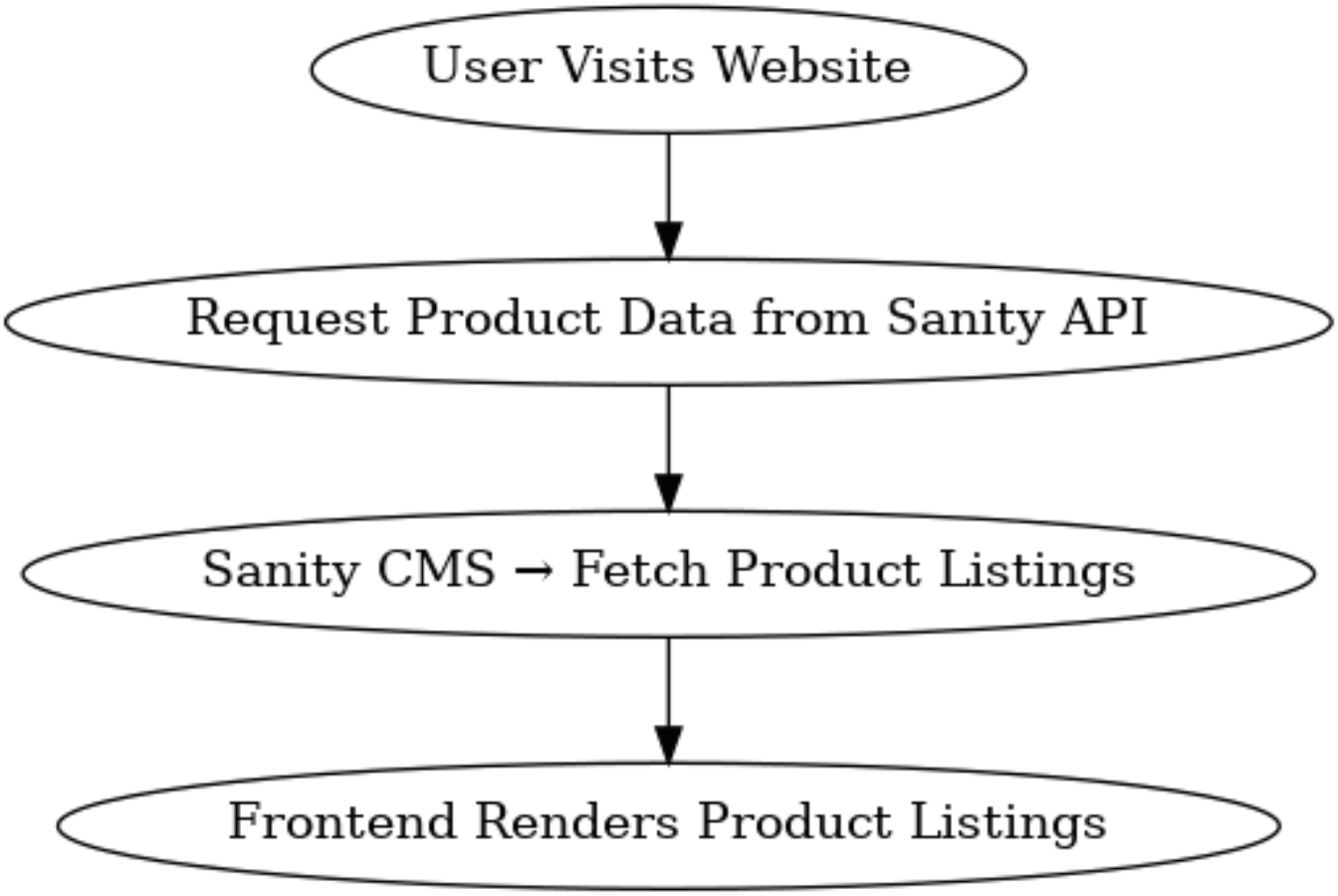
This flowchart represents the workflow for System Architecture Flowchart. Each step highlights interactions between components or processes in the Rent-A-Car system.

User Registration Workflow



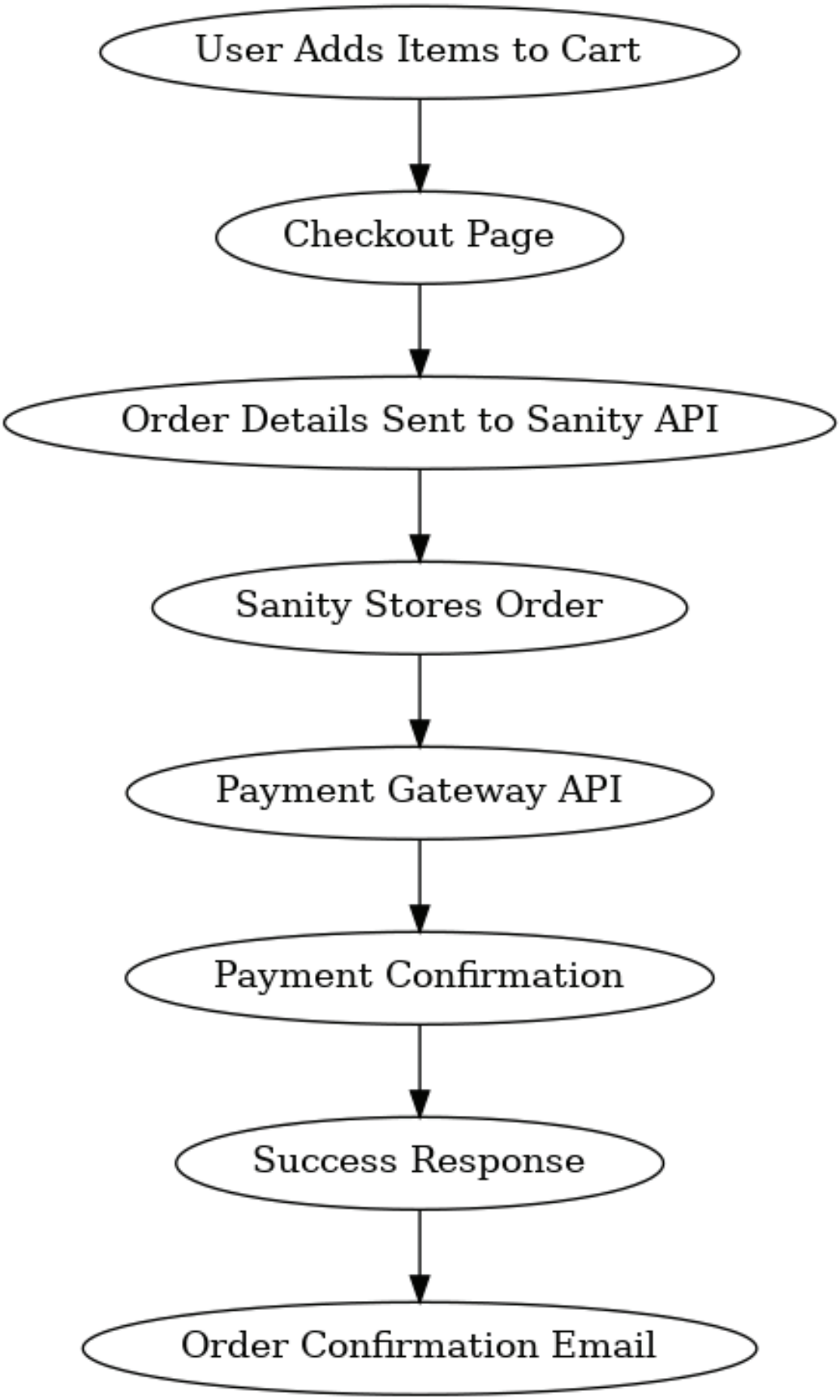
This flowchart represents the workflow for User Registration Workflow. Each step highlights interactions between components or processes in the Rent-A-Car system.

Product Browsing Workflow



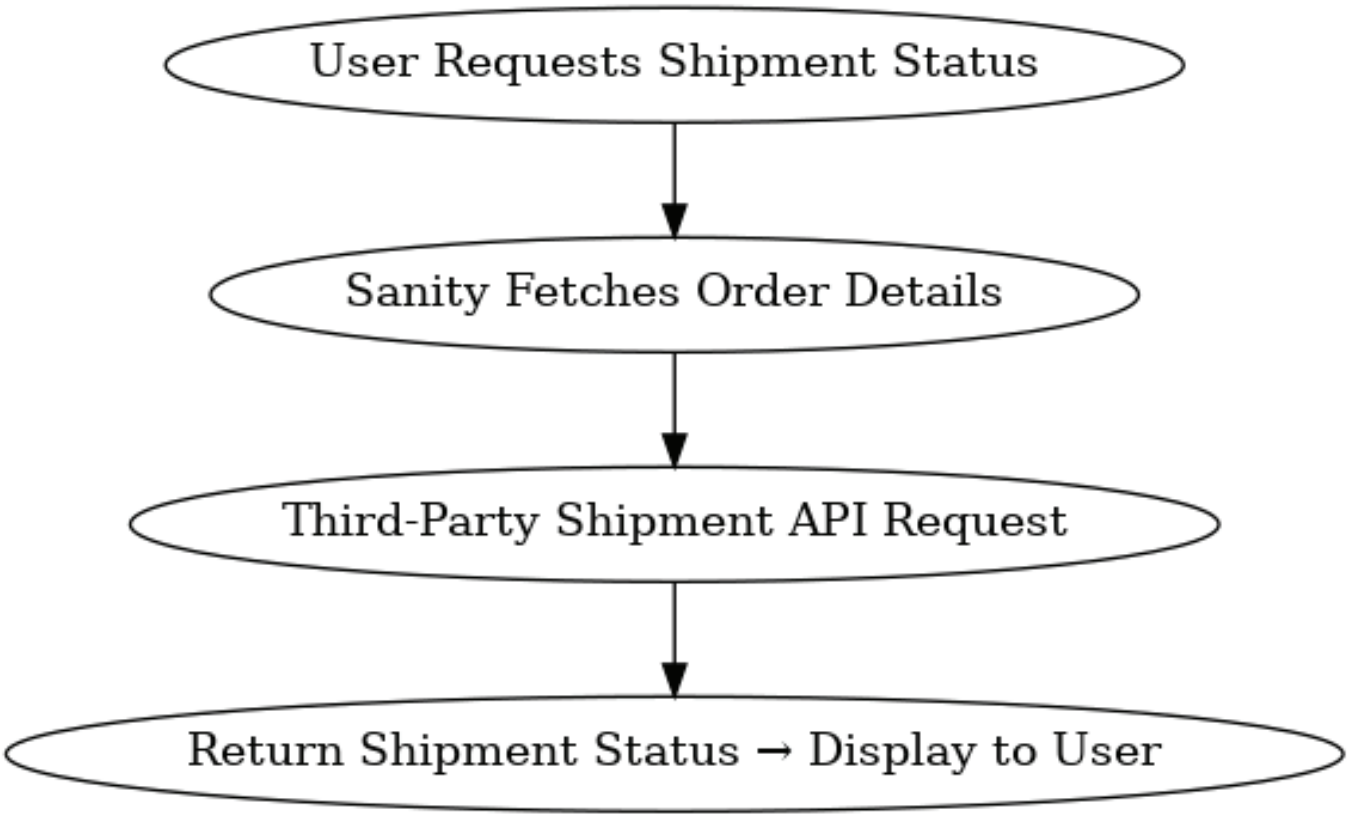
This flowchart represents the workflow for Product Browsing Workflow. Each step highlights interactions between components or processes in the Rent-A-Car system.

Order Placement Workflow



This flowchart represents the workflow for Order Placement Workflow. Each step highlights interactions between components or processes in the Rent-A-Car system.

Shipment Tracking Workflow



This flowchart represents the workflow for Shipment Tracking Workflow. Each step highlights interactions between components or processes in the Rent-A-Car system.