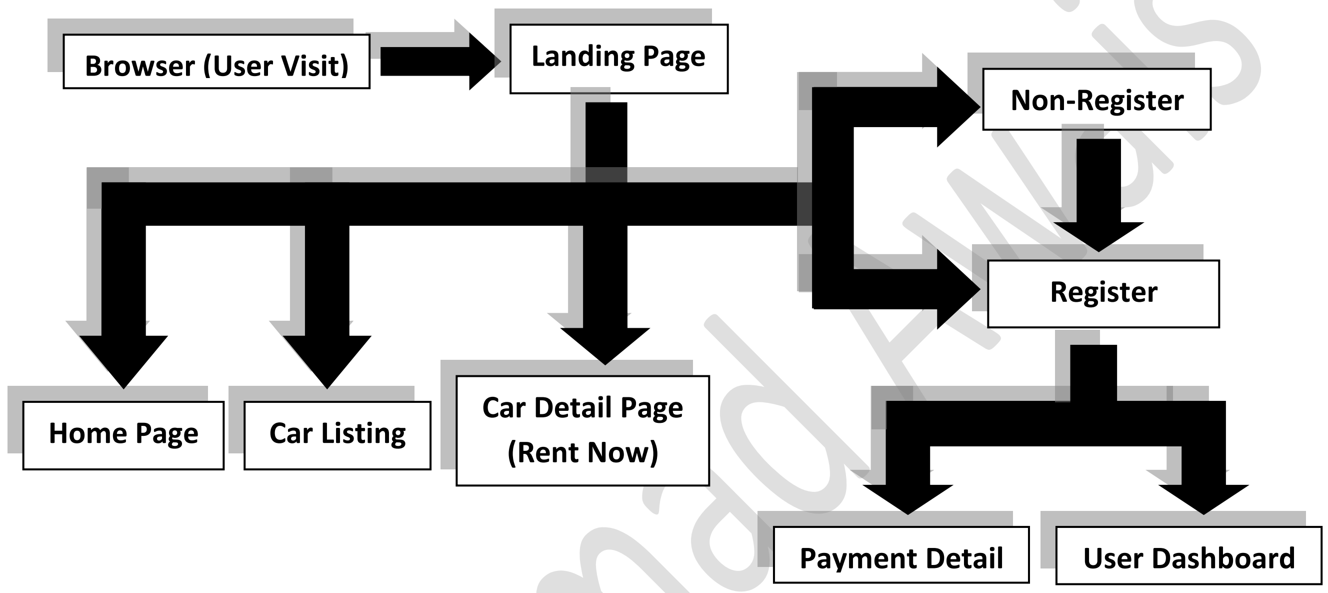
Hackathon Day 2: Planning the Technical Foundation

Step 01: Define Technical Requirements:

- **Frontend Requirements:**

Car Rental Website



Flow Chart:

The flowchart represents the **frontend requirements** for a **Car Rental Website**. Below are the points broken down:

1. Browser Visit

- The user accesses the website via a browser.

2. Landing Page

- The first page the user sees after visiting the website.

3. Navigation Options

- From the **Landing Page**, the user can navigate to the following:
  - **Home Page:** Serves as an introduction to the Car Rental Website.
  - **Car Listing**: Displays available cars for rent.
  - **Car Detail Page**: Provides detailed information about a specific car and a "Rent Now" option.

4. Non-Registered User Flow

- If the user is not registered:
  - They are prompted to **Register** before proceeding further.

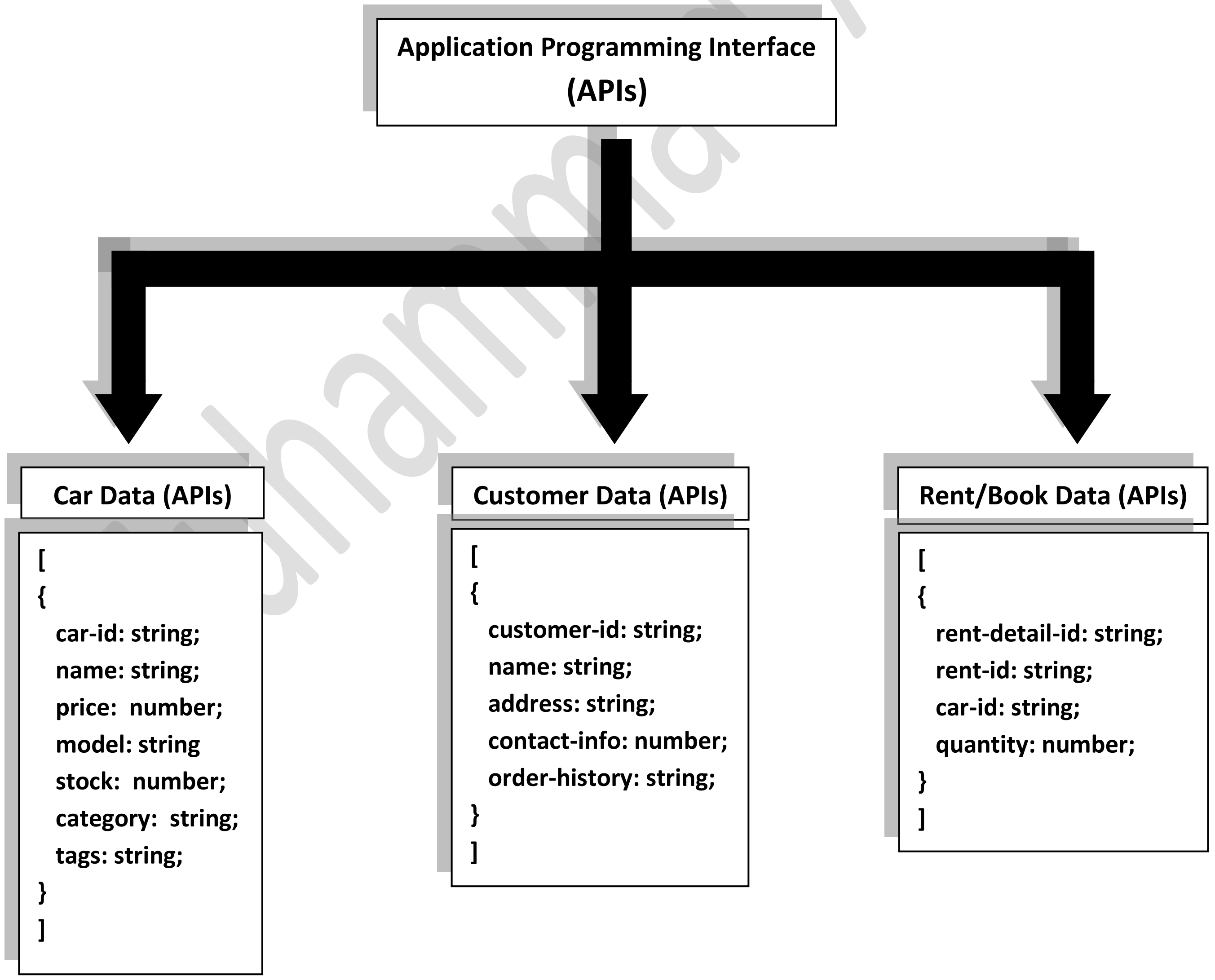
5. Post-Registration Options

- Once registered, users can access:
  - **Payment Detail Page**: To complete the car rental process.
  - **User Dashboard**: For managing user information, bookings, or preferences.

This diagram showcases the core navigation paths and interaction flow for a car rental website's frontend.

➢ <u>Sanity CMS as Backend:</u>

<u>Headless CMS Sanity</u>



```
[
{
  car-id: string;
  name: string;
  price:  number;
  model: string
  stock:  number;
  category:  string;
  tags: string;
}
]
```

```
[
{
  customer-id: string;
  name: string;
  address: string;
  contact-info: number;
  order-history: string;
}
]
```

```
[
{
  rent-detail-id: string;
  rent-id: string;
  car-id: string;
  quantity: number;
}
]
```

Description:

This diagram represents the Sanity CMS **architecture as a backend for managing data in a** Car Rental Website**. Here's the breakdown:**

1. Headless CMS - Sanity

- Sanity CMS is used as a backend to manage structured data.
- It communicates with the frontend through an **Application Programming Interface (API)**.

2. API Structure

The API organizes data into three main categories:

a. Car Data (APIs):

- Stores details about cars available for rent.
- Fields include:
    - Id: Car id(string)
    - name: Car name (string)
    - price per day: Rental price (number)
    - model: Car model (string)
    - category: Type of car (string)
    - tags: Keywords for search/filter (array of strings)

b. Customer Data (APIs):

- Stores information about registered customers.
- Fields include:
    - name: Customer name (string)
    - email: Customer email (string)
    - address: Address details (string)
    - contact-info: Phone number (string)
    - order-history: List of past bookings (array)

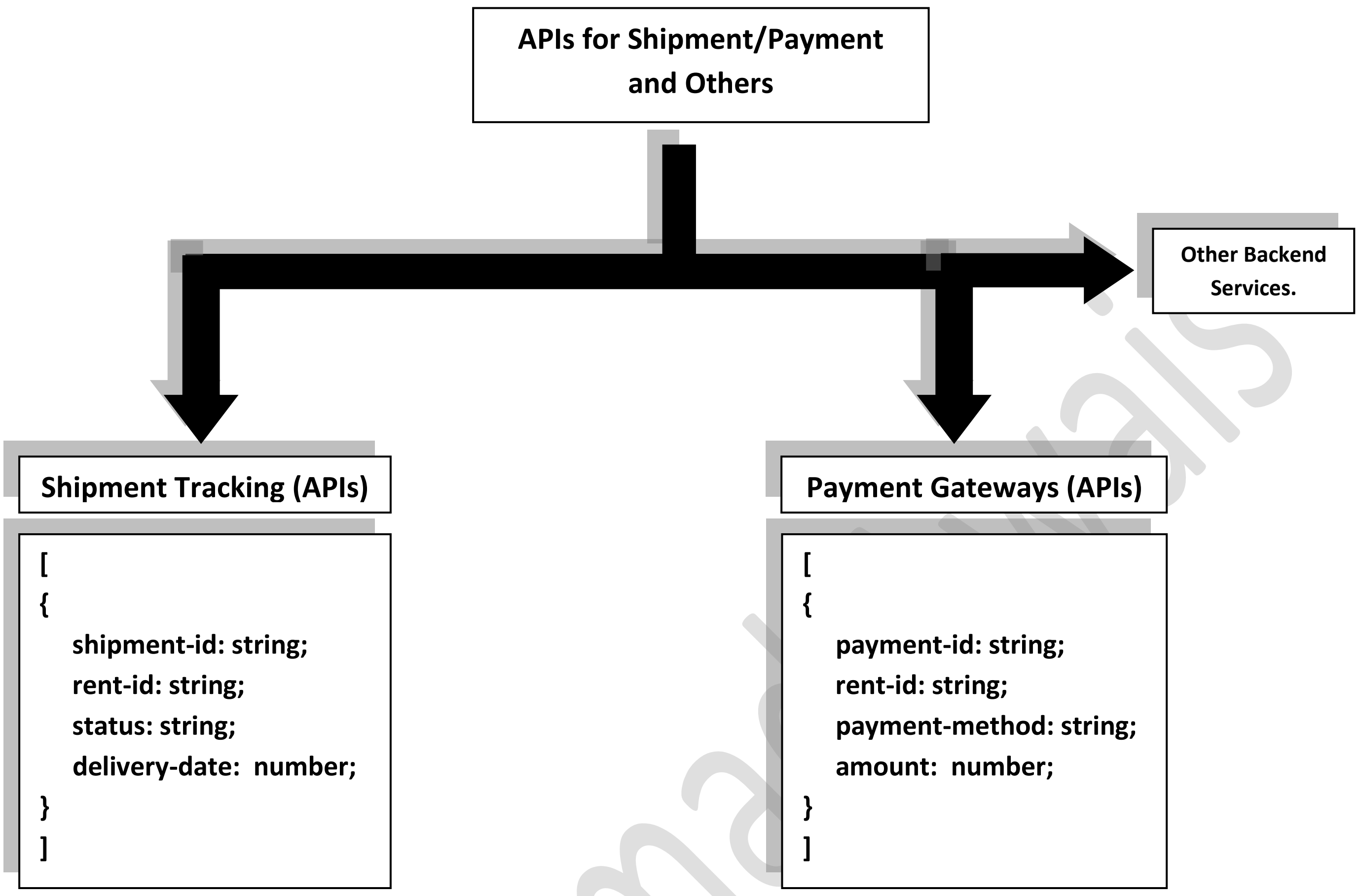
c. Rent/Book Data (APIs):

- Tracks rental and booking transactions.
- Fields include:
    - rent-id: Unique ID for the rental (string)
    - car-id: ID of the rented car (string)
    - quantity: Number of cars booked (number)
    - amount: Total rental cost (number)

3. Purpose of Each API:

- **Car Data API**: Helps populate car listings and detail pages.
- **Customer Data API**: Manages user registration, profiles, and history.
- **Rent/Book Data API**: Handles rental processes.

➢ Third-Party APIs:



```
[
{
    shipment-id: string;
    rent-id: string;
    status: string;
    delivery-date:  number;
}
]
```

```
[
{
    payment-id: string;
    rent-id: string;
    payment-method: string;
    amount:  number;
}
]
```

Description:

This diagram represents the integration of Third-Party APIs **for a car rental system, focusing on** Shipment Tracking **and** Payment Gateways**. Here's a detailed breakdown:**

Third-Party APIs

- These APIs handle external services such as shipment tracking, payment processing, and other backend services.

1. Shipment Tracking (APIs):

- Manages the delivery or shipment status for rented cars or related items.
- Key fields:
  - **shipment-id**: Unique identifier for the shipment (string).
  - **rent-id**: Links the shipment to the car rental transaction (string).
  - **status**: Current shipment status (e.g., "pending", "delivered") (string).
  - **delivery-date**: Expected delivery date for the shipment (number representing timestamp).

2. Payment Gateways (APIs):

- Facilitates secure payment processing.
- Key fields:
  - **payment-id**: Unique identifier for the payment transaction (string).
  - **rent-id**: Links the payment to the car rental transaction (string).
  - **payment-method**: Specifies the payment method used (e.g., "credit card", "PayPal") (string).
  - **amount**: Total payment amount (number).

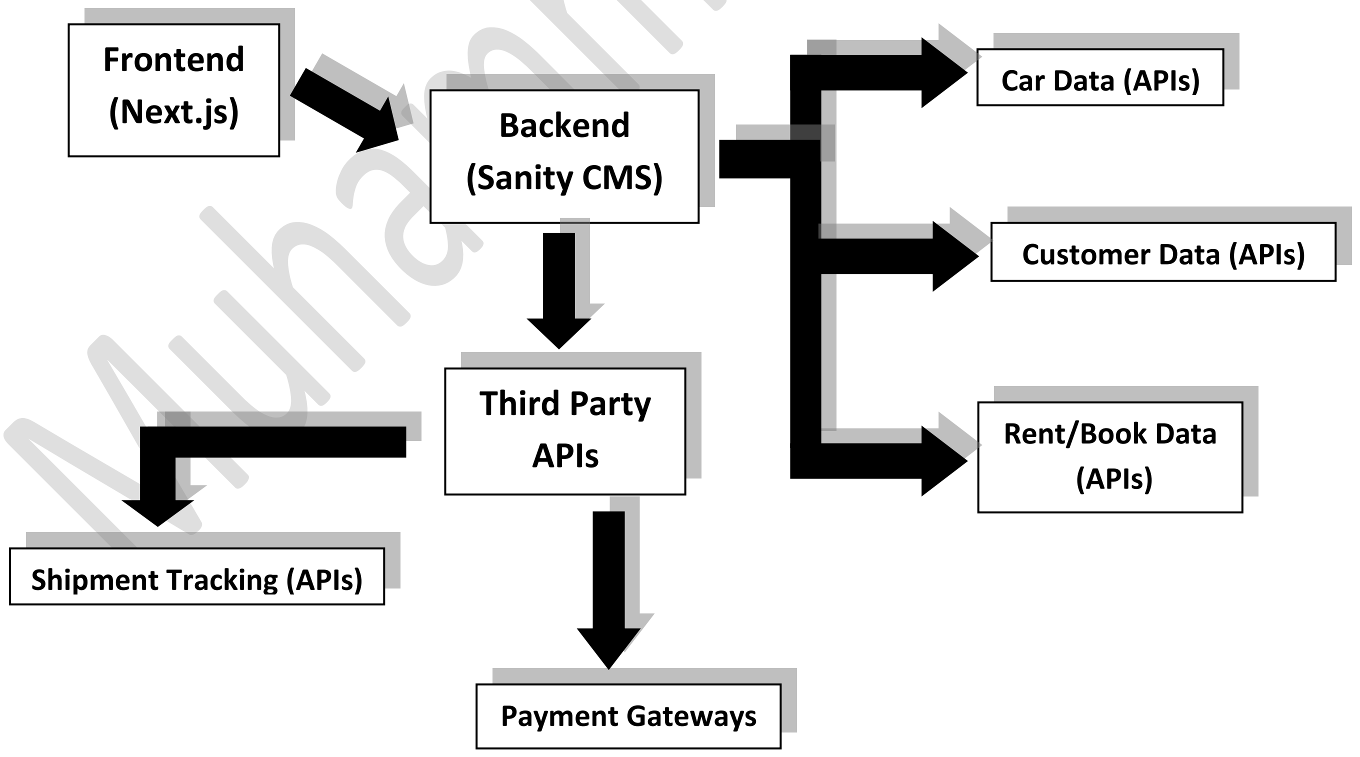
3. Other Backend Services:

- Placeholder for additional third-party services or APIs to be integrated in the future, such as:
  - Notification services (e.g., email/SMS updates).
  - Customer support integration.
  - Analytics and reporting tools.

Purpose of Integration

- **Shipment Tracking APIs** ensure transparency for users about delivery timelines.
- **Payment Gateways** offer secure, seamless payment experiences.
- Both services improve the user experience and operational efficiency.

Step 02: Design System Architecture:



Step 03: Plan for API Requirements:

The API endpoint structure tailored for **Car Rental Website**, aligning with the provided data schema and workflows:

API Endpoints

1. Fetch Cars.

- **Endpoint Name**: /categories
- **Method**: GET
- **Description**: Fetch the specific car category.
- **Response Example**:

```
{
   "id": 101,
   "name": "Audi",
   "price": 5000,
   "model": "2021",
   "category": "Sedans",
   “stock”: 2,
   "tags": ["fuel-efficient", "automatic"],
   "image": "https://example.com/car.jpg"

 }
```

2. Register a Customer

- **Endpoint Name**: /customer
- **Method**: POST
- **Description**: Register a new **Customer** on the platform
- **Response Example**:

```
{
   "customer-id": 202,
   "name": "Awais",
   "contact-info": 1234-56789,
   "address": "Karachi",
   "order-history": "none",
 }
```

3. Create a Rental Booking

- **Endpoint Name**: /rentals
- **Method**: POST
- **Description**: Create a new **Rental Order**.
- **Response Example**:

```
{
   "rent-detailed-id": CA-303,
   "rent-id": 303,
   "car-id":  101,
   "quantity": "2",
 }
```

4. Shipment Tracking

- **Endpoint Name**: /shipment/:rent-Id
- **Method**: GET
- **Description**: Track the shipment or pickup status of a car rental order via a third-party API.
- **Response Example**:

```
{
   "shipment-id": 404,
   "car-id": 101,
   "status":  "Active",
   "delivery-date": "2-Feb-2025",
 }
```

5. Payment Getaways

- **Endpoint Name**: /payment/payment-Id
- **Method**: POST
- **Description**: Process a rental payment via the **Payment Gateway.**
- **Response Example**:

```
{
   "payment-id": 505,
   "car-id": 101,
   "payment-method":  “Cash on Delivery”,
   "amount": 5000,
 }
```

Step 04: Sanity Schema Example:

```
// schemas/car.tsx

export default {
  name: 'car',
  title: 'Car',
  type: 'document',
  fields: [
    {
      name: 'id',
      title: 'ID',
      type: 'number',
   },
  {
      name: 'name',
      title: 'Car Name',
      type: 'string',
    },
   {
      name: 'price',
      title: 'Price',
      type: 'number',
    },
   {
      name: 'model',
      title: 'Model',
      type: 'string',
    },
```

```
 {
     name: 'category',
     title: 'Category',
     type: 'string',
   },
   {
     name: 'stock',
     title: 'Stock',
     type: 'number',
   },
   {
     name: 'tags',
     title: 'Tags',
     type: 'array',
     of: [{ type: 'string' }],
   },
   {
     name: 'image',
     title: 'Car Image',
     type: 'image',
   },
  ],
};
```

THANK YOU