Hackathon Day 2: Technical Foundation for General E-Commerce Marketplace

1. Technical Requirements

1. Frontend Requirements:

- **Tech Stack:** React or Next.js for dynamic and responsive pages.
- **Key Pages:**
  - Home Page
  - Product Listings
  - Product Details
  - Cart Management
  - Checkout Flow
  - Order Confirmation & Tracking
- Integration with **Sanity CMS** for fetching data (products, customers).

2. Backend Requirements:

- **Sanity CMS:** Centralized database for:
  - Products (name, price, stock, etc.)
  - Customers (ID, name, email)
  - Orders (order details, payment details).

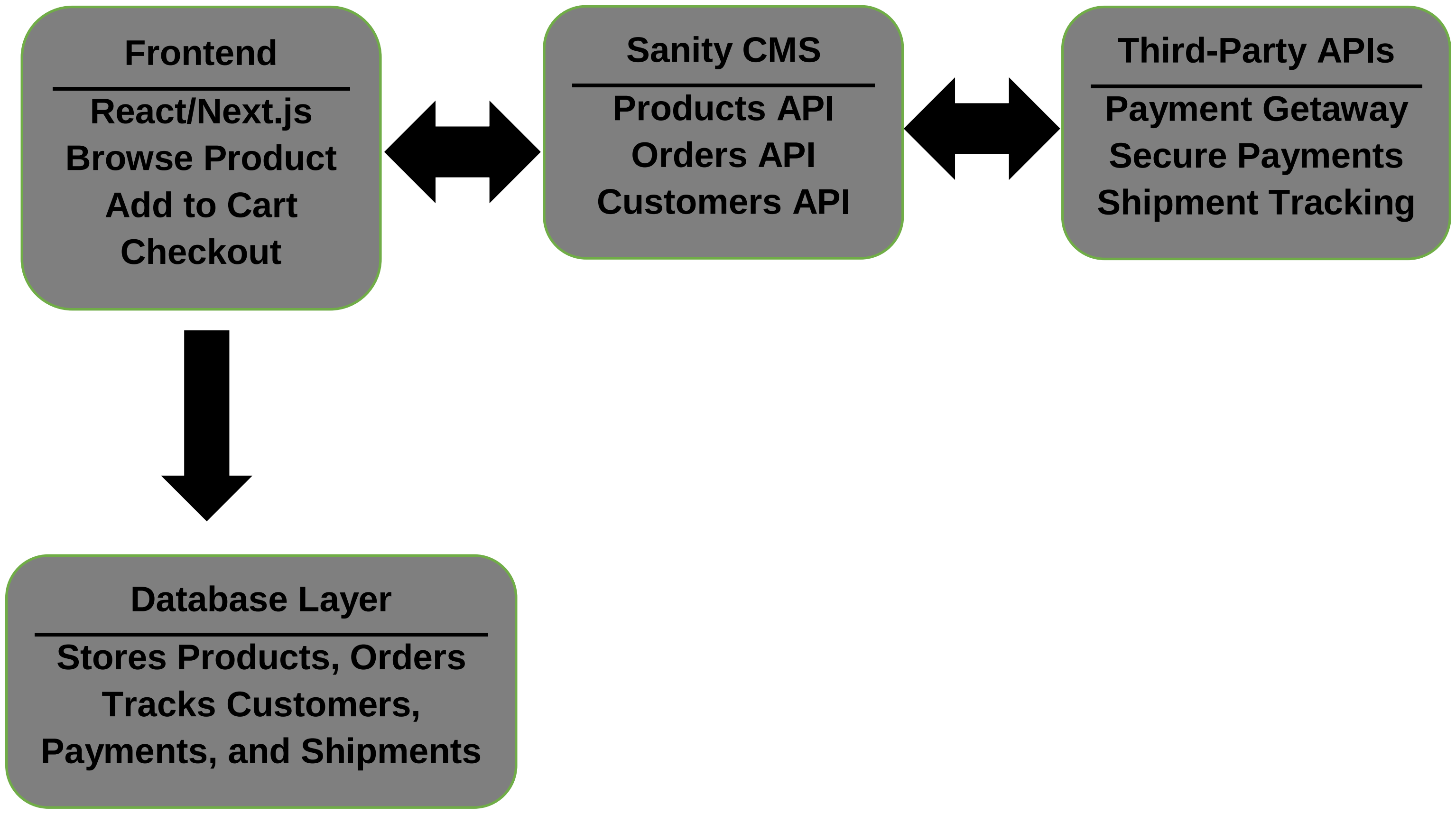
3. Third-Party API Integration:

- Payment Gateway (e.g., Stripe or PayPal) for secure payment processing.
- Shipment Tracking API for real-time delivery updates.

----

2. System Architecture

Here’s a clear system diagram to show how the components interact:



Explanation of Flow:

1. Frontend:

- Users browse and interact with the platform via the frontend.
- Product data is fetched via the **Sanity CMS API**.
- Checkout sends order details to the backend for processing.

2. Sanity CMS:

- Acts as the backend for managing all core data (products, customers, orders).

3. Third-Party APIs:

- Payment processing and shipment tracking data flow back to the frontend for updates.

----

3. API Documentation

Endpoint	Method	Purpose	Response Example
/products	Get	Fetches all available products.	{ "id": 1, "name": "Product A", "price": 100 }
/product/{id}	Get	Fetches details of a specific products.	{ "id": 1, "name": "Product A", "stock": 20 }
/orders	Post	Create a new order.	{ "orderId": 123, "status": "Order Placed"}
/shipment/{id}	Get	Fetches shipment tracking details for an order.	{ "shipmentId": 456, "status": "In Transit" }
/customers	Post	Adds a new customer to the database.	{ "customerId": 789, "status": "Customer Added" }

4. Sanity Schema Example

Here's a schema example for managing products in Sanity CMS:

```
export default {
  name: 'product',
  type: 'document',
  fields: [
    { name: 'name', type: 'string', title: 'Product Name' },
    { name: 'price', type: 'number', title: 'Price' },
    { name: 'stock', type: 'number', title: 'Stock Level' },
    { name: 'category', type: 'string', title: 'Category' },
    { name: 'image', type: 'image', title: 'Product Image' }
  ]
};
```

----

5. User Workflow

Key Flows to Implement:

1. Browsing Products:
- User visits the frontend → Frontend fetches product data via **/products** → Displays product listings.

2. Placing Orders:
- User adds items to cart → Proceeds to checkout → Frontend calls **/orders** API to store order details in Sanity CMS → Payment Gateway API processes payment.

3. Shipment Tracking:
- User checks order status → Frontend fetches tracking updates from **/shipment/{id}** API → Displays live tracking data.

----

6. Deliverables by Day 2

1. System Architecture:
- Clear diagram showing interactions between frontend, backend (Sanity CMS), and APIs.

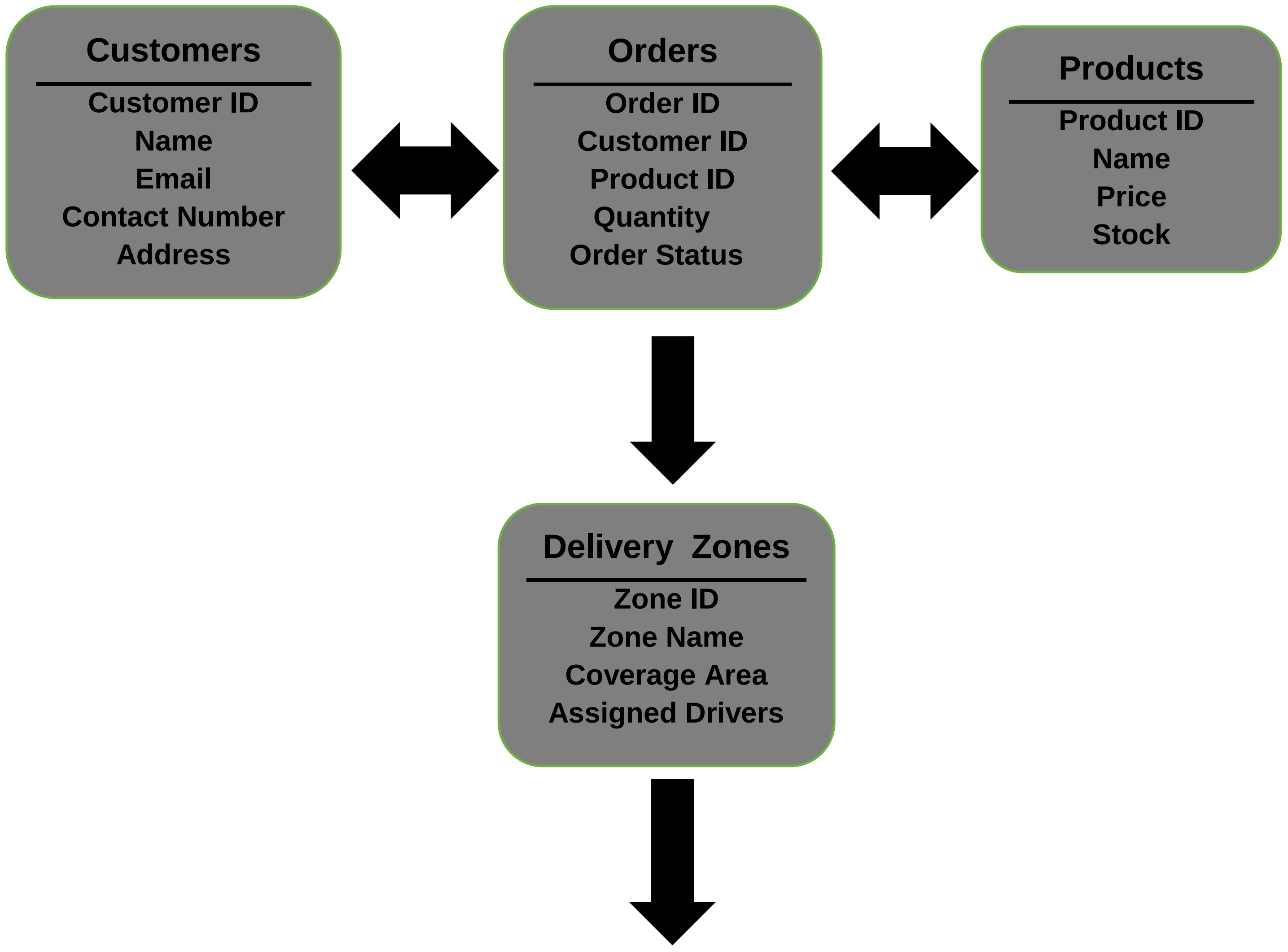
2. API Documentation:
- Endpoints for fetching products, creating orders, and shipment tracking.

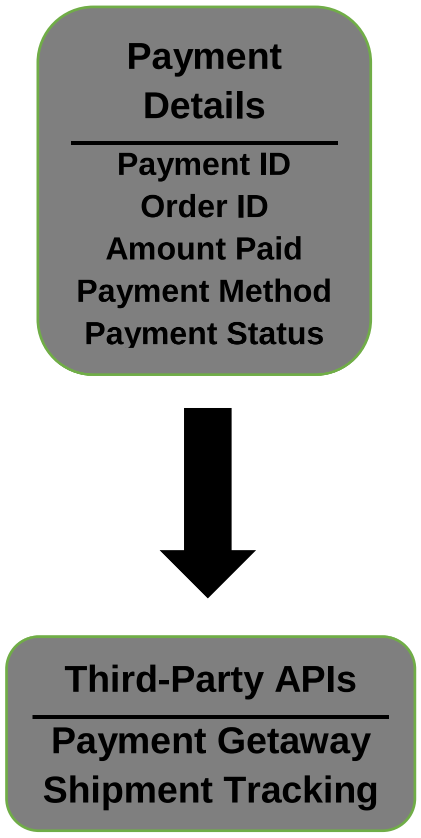
3. Sanity CMS Schema:
- Defined fields for managing product data.

4. Workflow Examples:
- Show how users browse, place orders, and track shipments.

Final Diagram

Here's the updated diagram based on today's requirements:




Payment Details
Payment ID
Order ID
Amount Paid
Payment Method
Payment Status
Third-Party APIs
Payment Getaway
Shipment Tracking
