

# Marketplace Builder Hackathon

## Day 2: Planning the Technical Foundation

**Student:** Imtiaz Ahmed Jahejo

**Timing:** Saturday (2 PM - 5 PM)

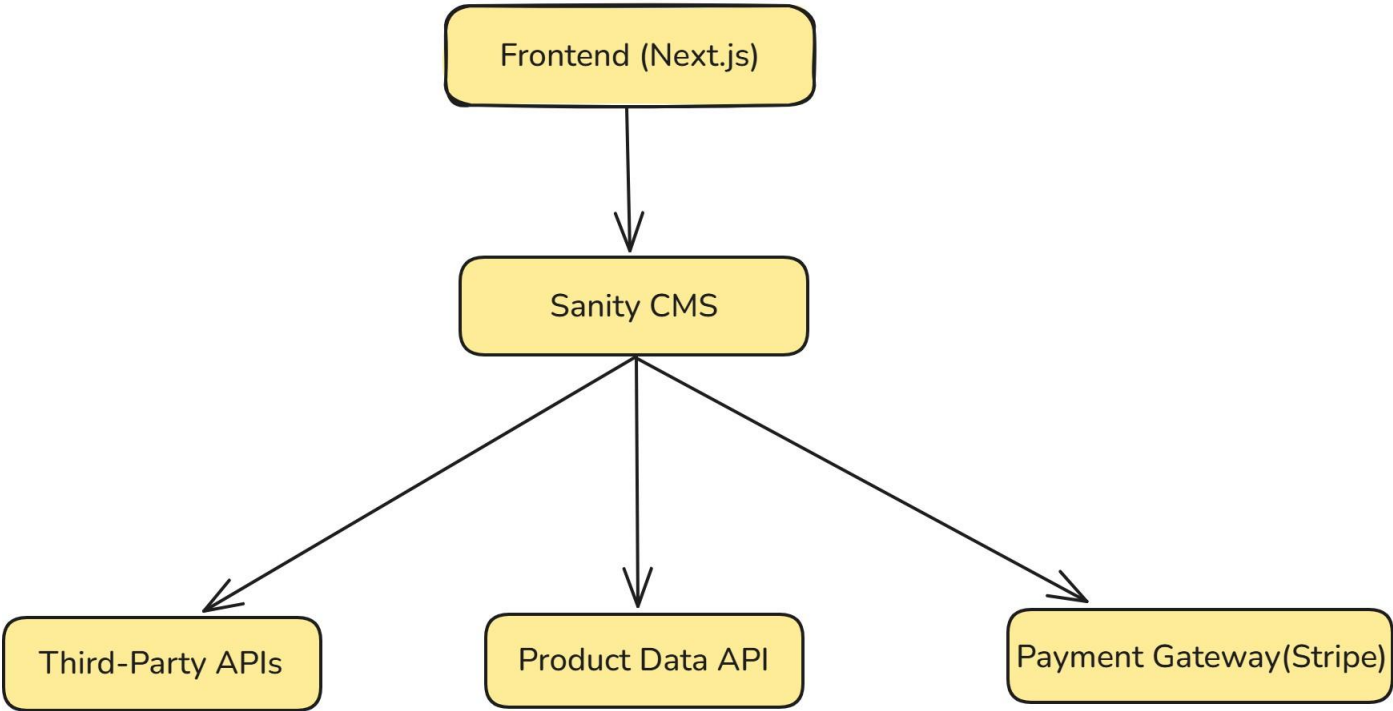
**Roll Number:** 00249533

## Comforty E-Commerce System Architecture

### 1. System Architecture Overview:

The architecture for the Comforty E-Commerce platform integrates multiple technologies to ensure scalability, user-friendliness, and reliability. Below is a high-level system architecture:

# Diagram



- **Components**

- Frontend (Next.js)
  - Dynamic and responsive UI for users.
  - Interacts with backend services to fetch and display data.

- **Sanity CMS:**

- Acts as the primary backend for managing product data, customer details, and order records.
- Provides APIs for CRUD operations on data.

- **Third-Party APIs:**

- Shipment tracking to update order delivery status.

- **Payment Gateway (Stripe):**

- Processes secure transactions.

## **Key Workflows**

### 1. Product Browsing:

- User visits the frontend to browse products.
- Frontend fetches product data via Sanity CMS API.

### 2. Order Placement:

- User adds items to the cart and checks out.
- Order details are stored in Sanity CMS.

### 3. Shipment Tracking:

- Order tracking updates are fetched via a third-party API.

### 4. Payment Processing:

- Stripe handles secure transactions, and payment confirmation is sent to Sanity CMS.

## 2.API Endpoints:

Endpoint	Method	Purpose	Payload	Response Example
/products	GET	Fetch all product details	N/A	{ "id": 1, "name": "Sofa", "price": 1000 }
/products/:id	GET	Fetch a specific product	N/A	{ "id": 1, "name": "Sofa", "price": 1000 }
/cart	POST	Add item to cart	{ "productId": 1, "quantity": 2 }	{ "cartId": 123, "status": "Added" }
/checkout	POST	Process checkout	{ "cartId": 123, "paymentInfo": {...} }	{ "orderId": 456, "status": "Confirmed" }
/shipment	GET	Get shipment tracking info	{ "orderId": 456 }	{ "trackingId": "789", "status": "In Transit" }
/login	POST	User login	{ "email": "user@example.com", "password": "..." }	{ "token": "jwt_token" }
/signup	POST	User registration	{ "email": "user@example.com", "password": "..." }	{ "status": "Registered" }

## 3. Sanity CMS Schema

### Product Schema

```
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: 'image', type: 'image', title: 'Product Image' }
  ]
};
```

### Order Schema

```
export default {
  name: 'order',
  type: 'document',
  fields: [
    { name: 'customer', type: 'reference', to: [{ type: 'customer' }] },
    { name: 'products', type: 'array', of: [{ type: 'product' }] },
    { name: 'status', type: 'string', title: 'Order Status' }
  ]
};
```

## 4. Technical Documentation

### Workflows

- **Login and Signup:**

- User registers or logs in via the frontend.
- Details are authenticated and stored in the Sanity CMS.

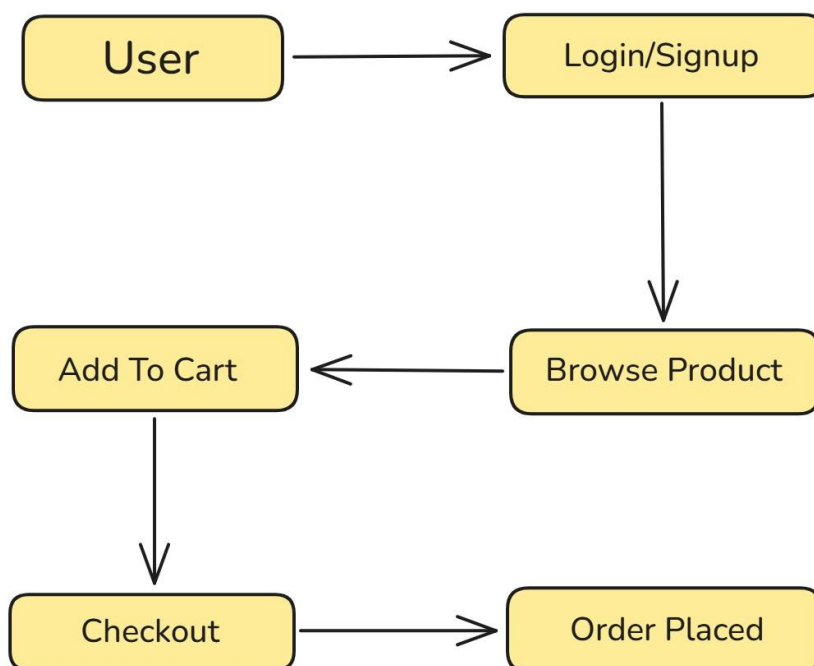
- **Cart Management:**

- Items added to the cart are tracked using the /cart endpoint.

- **Order Confirmation:**

- User checks out, and the order is stored in Sanity CMS with payment status.

### Work Flow Diagram



## 5. Submission

The document is structured to reflect the architecture, API specifications, and workflows. The PDF will include:

1. System Architecture Diagram.
2. Detailed API Endpoints.
3. Workflows and Key Features.
4. Sanity CMS Schemas.

## **Next Steps**

This document ensures the technical foundation is ready for implementation. Following these guidelines will enable smooth development and scalability.