Day 2: Technical Planning

1. **System Architecture:**
    - Use **Next.js** to provide fast performance and SEO optimization
    - Use **Sanity CMS** for manage profile for admin, to **Add Product**, **Delete Product** and **Edit Product**. **Sanity** is use for **Content Management.**
    - Integrate a payment gateway API like **Stripe,** for secure transactions.
    - Also implement an **Authentication** when user is **Login** only can order the product.

2. **Frontend Requirements:**

- **User Interface:**
    - Create a user-friendly design for browsing products effortlessly.
    - Implement features for smooth navigation and intuitive user interactions.
- **Responsive Design:**
    - Ensure the marketplace works seamlessly on both mobile and desktop devices.
- **Essential Pages to Develop:**
    - **Home Page:** Display featured products or categories.
    - **About Page:** Display the about of our restruanent.
    - **Shop Listing Page:** All the available products with filtering options.
    - **Product Details Page:** Showcase product images, descriptions, prices, and a "Add to Cart" button.
    - **SignUp Page:** Create a sign and login page that can user login and can order product if user was not **Login** it can't order the product and not add the product into cart.
    - **Cart Page:** If user was login it can Display the selected products with options to update quantities or remove items.
    - **Checkout Page:** Handle payment, and information of customer and shiping.
    - **Order Confirmation Page:** Confirm that the order has been placed and show order details.

3. *Sanity CMS as Backend:*

- Use **Sanity CMS** to manage:
    - Product data (name, description, price, category, images).
    - Customer details (profile, orders).
    - Order records (order status, payment confirmation, and delivery details).
- **Schema Design:**
    - Create schemas within Sanity to organize data based on the marketplace's business goals (e.g., products data, order data, customer data and Delivery Zone).
- **Examples Of Sanity Data:**
    - ProductData{

        ProductId:131123,

        ProductName:"Chicken Pizza",

        Slug:"chicken-pizza,

Price:250

Stock:200

}

- OrderData{

  OrderId:131123,

  CustomerId:91314141, (Unique id of every customer)

  Slug:”chicken-pizza,

  TotalAmount : 500, (Total Amount Of cart items)

  Stock:200

  orderStaus: “pending || recived” ,}

  }

- CustomerData{

  CustomerId:91314141, (Unique id of every customer)

  CustomerName: ”Ahmed”,,

  CustomerEmail: abc@gmail.com,

  phoneNumber : 03xxxxxx,

  Address : ”etc ”

  }

- DeliveryZone{

  ZoneID: Unique identifier for the delivery zone..CustomerName: ”Ahmed”,,

  zoneName:[Korangi , saddar , north karachi, malir].

  coverageArea: All over the Karachi

  }

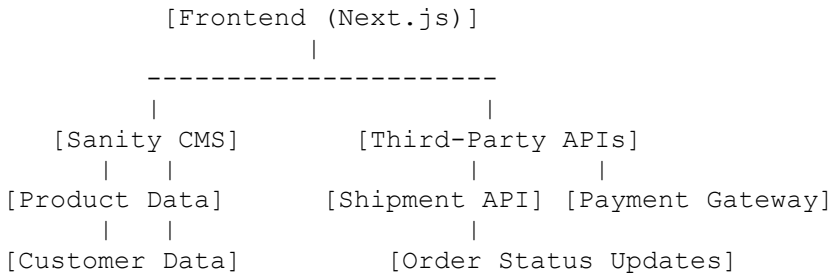
4. *Implement Third-Party APIs:*

- **Shipment Tracking API:** Allow customers to track the status of their orders.

- **Payment Gateway API (e.g., Stripe or Easypaisa):** Ensure secure and smooth payment processing.
- Ensure all APIs integrate seamlessly with both the frontend and backend, providing the necessary data for functionality.

Here's how you can design a **system architecture** for your marketplace, including high-level workflows and interactions:

System Architecture Diagram



Key Workflows to Include

1. User Registration Workflow

- **Flow**:
  User signs up → Form data is sent to Sanity CMS → User record is created → Confirmation email/message sent to the user.
- **Purpose**:
  Ensures seamless registration and account management for users.

2. Product Browsing Workflow:

- **Flow**:
  User navigates the marketplace → Frontend makes an API request to Sanity CMS → Product data (name, description, price, stock, images) is fetched → Data is dynamically displayed on the site.
- **Purpose**:
  Enables users to view product details and make purchasing decisions.

3. Order Placement Workflow

- **Steps**:
  1. User adds products to the cart on the frontend.
  2. At checkout, the order details (products, total amount, and user information) are sent to **Sanity CMS** via an API request.
  3. The order is recorded in Sanity CMS with a unique `orderId`.
  4. The user is redirected to the payment gateway for processing the payment.

5. Once payment is successful, a confirmation is sent to the user and updated in Sanity CMS.

- **Purpose**:
  Ensures orders are seamlessly recorded and payment is securely processed.

4. Shipment Tracking Workflow

- **Steps**:
  1. The user checks the order status on the frontend.
  2. The frontend fetches real-time order tracking details via a **Third-Party Shipment API**.
  3. The shipment details (current location, expected delivery time) are displayed to the user.
  4. Updates are logged back in **Sanity CMS** for record-keeping.
- **Purpose**:
  Keeps users informed about their order delivery status.

API Endpoints

1. Fetch All Products

- **Endpoint Name**: `/products`
- **Method**: GET
- **Description**: Fetch all available products from Sanity CMS, including perishable food items.
- **Response Example**:

JSON DATA :


```
[
 {
  "id": 1,
  "name": "Chicken Pizza",
  "price": 250,
  "stock": 200,
  "category": "Fast Food",
  "image": "pizza.jpg"
 },
]
```

2. Create a New Order

- **Endpoint Name**: `/orders`
- **Method**: POST
- **Description**: Create a new food order in Sanity CMS.
- **Payload Example**:

JSON DATA


```
{
 "customerId": "91314141",
 "customerName": "Ahmed",
 "address": "Karachi, Pakistan",
 "products": [
  {
   "productId": 1,
   "quantity": 2,
   "price": 250
  },
  {
   "productId": 2,
   "quantity": 1,
   "price": 180
  }
 ],
 "totalAmount": 680,
 "paymentStatus": "Paid"
}
```


3. Fetch Real-Time Delivery Updates

- **Endpoint Name**: `/express-delivery-status`

- **Method**: GET
- **Description**: Fetch real-time delivery updates for food items.
- **Response**

JSON DATA

```
{
  "orderId": 12345,
  "status": "Out for Delivery",
  "ETA": "20 mins"
}
```

Conclusion:

- This project provides an efficient and scalable platform for selling food
- We use **Next JS** for Frontend Development.
- **Sanity** is use for as a backend for manage the content of our website
- **Ship Engine** is for tracking of our order