Marketplace Technical Foundation - Food Restaurant Marketplace

Prepared by Muhammad Talha

Hackathon Day 2: Planning the Technical Foundation

Day 2 Activities: Transitioning to Technical Planning

1. Define Technical Requirements

Frontend Requirements:

- **User Interface:**
  - Intuitive design for easy restaurant and menu browsing.
  - Mobile and desktop responsive layout.
- **Essential Pages:**
  - Home, Restaurant Listings, Dish Details, Cart, Checkout, Order Status.

Backend Requirements:

- **Sanity CMS:**
  - Manage restaurant profiles, dish details, and order records.
  - Custom schemas for dynamic data management.

Third-Party APIs:

- **Payment Gateway:** Integrate Stripe, Easypaisa, Jazzcash or PayPal for secure transactions.
- **Delivery Tracking:** Utilize a real-time shipment tracking API.

---

2. Design System Architecture

A high-level architecture diagram:

```
[Frontend (Next.js)]
  |
[Sanity CMS] ---------> [Product Data API]
  |
[Third-Party API] -----> [Delivery Tracking API]
  |
[Payment Gateway]
```

Key Workflows:

1. **User Browsing Products:**
   - Fetch restaurant data and menu items via the Sanity CMS API.
   - Display dynamic content on the frontend.
2. **Order Placement:**
   - User adds items to the cart and checks out.
   - Order details are sent to Sanity CMS and payment is processed via the payment gateway.
3. **Delivery Tracking:**
   - Fetch real-time updates from the delivery tracking API.

---

3. Plan API Requirements

Endpoints:

Endpoint Name	Method	Purpose	Response Example
/restaurants	GET	Fetch list of restaurants	{ "id": 1, "name": "Pizza Palace" }
/dishes	GET	Fetch menu items for a restaurant	{ "id": 101, "name": "Margherita Pizza" }
/order	POST	Create a new order	{ "orderId": 123, "status": "Confirmed" }
/order-status	GET	Fetch real-time delivery updates	{ "orderId": 123, "status": "In Transit" }
/payment	POST	Process payment through gateway	{ "status": "Success" }

---

4. Write Technical Documentation

System Architecture Document:

- Diagram illustrating interactions between the frontend, CMS, and APIs.

API Specification Document:

- Endpoint details with methods, payloads, and expected responses.

Workflow Diagram:

- User journey from browsing to order completion.

Sanity Schema Example:

```
export default {
  name: 'dish',
  type: 'document',
  fields: [
    { name: 'name', type: 'string', title: 'Dish Name' },
    { name: 'price', type: 'number', title: 'Price' },
    { name: 'restaurant', type: 'reference', to: [{ type: 'restaurant' }] },
  ],
};
```