16-1 2025 (Thursday) HackAthon Day # 02 Task

Day 2 Planning The Technical Foundation

Name: Farah Shabir

Roll No: 00006855

Day: Sunday

Time: 9:00 AM - 12:00 PM

Day 2 Activities: Transitioning to Technical Planning

1. Define Technical Requirements

- Frontend Requirements:

i) User-Friendly interface: The website should be simple and clear so users can easily find and browse products.

ii) Responsive Design: The design should look good and work smoothly on both mobile phones and computers.

iii) Essential Pages

* Home Page
* Product Listing
* Product details
* Cart Page
* Checkout Page
* Order confirmation Page

Fronted
Next.js tailwind

- **Sanity CMS as BackEnd:**

Sanity CMS is a headless content management system that helps you manage structured content

Product Schema (for managing products)

Fields:

- name : string (Product name)
- Price : Number (Product price)
- stock : Number (Quantity in stock)

schemas/Product.js

```
export default {
  name: 'product',
  title : 'product',
  type : 'document',
  fields: [
   {name: 'name', type: 'string'},
   {name : 'price', type: 'number'},
   {name : 'stock', type: 'number'},
  ]
}
```

Order Schema

Fields:

- OrderID
- product
- totalAmount
- Quantity

Schemas/order.js

```
export default {
  name : 'order',
  title : 'Order',
  type: 'document',
  fields : [
   { name : 'orderNumber', type: 'string'},
   { name : 'product', type: 'array', of :
    [{type : 'reference', to : [{ type: 'product'}] }] },
  {name: 'totalAmount', type: 'number'},
  {name: 'Quantity', type : 'number'},
```

Customer Schema

Fields

- name
- email
- address
- phone

Schemas/customer.js

```
export default {
  name : 'customer',
  title : 'Customer',
  type : 'document',
   fields = [
    { name : 'name', title : 'Name', type : 'string'},
   {name: 'email', title : 'Email', type: 'string'},
   {name: 'address', title : 'Address', type: 'text'},
```

```
{name: 'phone', title: 'Phone', type: 'string'},
],
};
```

Third-party APIs.

Third-party APIs are essential for extending your marketplace's functionality, such as shipment tracking, payment processing, and other backend services.

Third-API Integrations

- Shipment Tracking APIs
- Payment Gateway APIs

2. Design System Architecture

i) **Frontend**

- Built using Next.js (React framework)
- Handles user interactions and displays data fetched from APIs
- Key pages:
  - Home Page
  - Product Listing
  - Product Details
  - Cart & checkout
  - Order Tracking

ii) **Backend**

Uses Sanity CMS to store and manage products, customers, and order data.

- A Node.js/Express or Next.js API layer to process business logic and interact with third-party APIs
- API Endpoints: Fetch products, categories, and customer details.

iii) **Third-Party APIs**

- Payment Gateway: (e.g., Stripe/Paypal) for secure payment transactions.
- Shipment Tracking: (e.g., EasyPost/Shippo) to manage delivery statuses.
- Email/SMS Notifications: (e.g., Twilio/SendGrid) for order confirmations and updates.

iv **Database**

- Managed by Sanity CMS.
- Stores:

  Products

  Categories

  Orders

  Customer Information

v. **Delivery Management**

- Delivery zones and assigned drivers handled through backend logic or third-party APIs
- Integration with Google Map API

vi. **Authentication**

- User authentication through NextAuth.js or Firebase Auth

vii. **Cloud Hosting**

- Deploy on platforms like Vercel

Workflow

1. Frontend
   users browse products, add items to their cart, and proceed to checkout
2. Backend
   Processes API request from the frontend
3. Payment Gatway:
   Backend sends order and payment details to the gateway
4. Shipment Tracking:
   Backend interacts with tracking APIs to fetch the order status.
5. Delivery Zones
   Backend logic calculates delivery zones or fetches driver details based on customer location.
   uses Google Map API
6. Notifications
   Backend triggers notifications (email/SMS) upon order confirmation or shipment updates

3. Plan API Requirements

API endpoints required for the marketplace, based on the described schema and workflows.

1, Products : /products

(Fetch all data) (Response Example)

```
[
 {
  "id": 1,
  "name": "Product A"
  "price": 100,
  "stock": 50,
  "image": "url-to-image"
 },
 {
  "id": 2,
  "name": "Product B",
  "price": 200,
  "stock": 30,
  "image": "url-to-image"
 }
]
```

Endpoint Name: /products
Method : GET
Description: Fetch all available products from sanity CMS

ii, Products: /products/{id}

```
{
 "id": 1,
 "name": "Product A"
 "price": 100,
```

Endpoint Name: /products/{id}
Method : GET
Description: Fetch detailed information for a specific product by its ID

```
"stock": 50,
"description": "product detail",
"image": "url-to-image"
}
```

iii, Orders: /orders

```
{
"customer": {
  "name": "ABC",
  "email": "abc@gmail.com",
  "phone": "123456789"
},
"products": [
  { "id": 1 "quantity: 2 },
  { "id": 3, "quantity: 1 }
],
"paymentStatus": "Paid",
"totalAmount": 300
}
```

Endpoint Name: /orders
Method: POST
Description: Create a new order in sanity CMS

iv. Shipment Tracking: /shipment

```
{
"shipmentId": 'Ship123',
"orderID": '123',
"status": "process",
"delivery": 2025-1-31
```

Endpoint Name: /shipment
Method: GET
Description: Track order status via a third-party API
Query Parameters
OrderID

v. User Authentication

Endpoint Name: /auth/login

Method: POST

Description: Log in a user and return an authentication token

Response Example:

```
{
"token": "auth-token",
"message": "login"
}
```