Hackathon: Marketplace Builder 2025 (Day 2)

General E-Commerce Website

Name: Basit Khalil

REG ID :35473

Class: TUE 2-5 Batch 1-Q2

Marketplace Technical Foundation

-Define Technical Requirements

Frontend:

/Homepage:

Feature products, Product banners, Promotional banner, Categories.

/Product Page:

A list of product shows

Users are allowed to browse product through filters like best sellers, new arrivals etc.

/Product Detail Page:

Product details like

Name, description, price and reviews.

/Cart page:

Display products which are added to cart, total amount and quantity.

/Login page:

User Namer, email address, and contact number.

/Checkout

Display order id, tracking id, and estimate delivery date

BACKEND (Sanity CMS):

Sanity CMS will work as a backend to manage data like product, orders, tracking, shipment.

Implementation:

- Use sanity to create and test schemas
- Fetch data on fronted through GROQ queries.

DESIGN SYSTEM ARCHITECHTURE:

Work Flow:

User-interaction: User landing on a /**Homepage,** ---> click on products from navbar

Frontend: Display all products fetching data from API.

User-Interaction: User click on each product to show product detail page that is **/products/product[id],**

Frontend: Shows a product detail page which has a product name, price, image, description, stock and has 'add to cart' button on same page.

User-Interaction: when user click on Add to cart button then this redirect to **/cart page.**

Frontend: /**cart page** shows products, quantity of product, total amount, and promotion code which is optional and on bottom has a /checkout-Page.

User-Interaction: when user click on checkout button which will be redirect to **/customer-Login** page.

Frontend: /customer-Login which has a customer name, email address, phone number, address.

User-interaction: When user place order

Backend:

- Save the customer details in sanity.
- Save the order details in sanity.

After placing order a unique order id and tracking id will be generated and send to user,

User allows to track id through third-party logistics.

DATA SCHEMAS:

Product Schemas /products *sanity*

```
{
  "type": "object",
  "properties": {
    "product_id": {
      "type": "string",
      "description": "Each product has a unique product ID"
    },
    "name": {
      "type": "string",
```

```
    "description": "Product name"
  },
  "price": {
    "type": "number",
    "description": "Product price"
  },
  "description": {
    "type": "string",
    "description": "Product description"
  },
  "image": {
    "type": "string",
    "format": "uri",
    "description": "Product image (URL or path)"
  },
  "category": {
    "type": "string",
    "description": "Product category"
  },
  "stock": {
    "type": "integer",
    "description": "Product quantity in stock"
  }
},
```

Order Schema /orders ***Sanity***

```
{
 "type": "object",
 "properties": {
"order_id": {
"type": "string",
"description": "Unique order ID"
 },
"customer_id": {
 "type": "string",
"description": "Customer ID"
 }, "items":
{
 "type": "array",
 "description": "List of products in the order",
"items": {
 "type": "string",
 "description": "Product name or product ID"
 } },
"amount": {
"type": "number",
 "description": "Total order amount"
 },
```

"tracking_id":

{ "type": "string",

"description": "Unique tracking ID provided by logistics"

Customer Schema /customer *sanity*

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "name": {
      "type": "string",
      "description": "Customer name"
    },
    "address": {
      "type": "string",
      "description": "Customer address"
    },
    "phone_number": {
      "type": "string",
      "description": "Customer contact number"
    },
    "email": {
      "type": "string",
      "format": "email",
```

```
    "description": "Customer email address"
   }
  },
  "required": ["name", "address", "phone_number", "email"]
}
```


SANITY API ENDPOINTS:

Order schema (/customer):

- ☐ Create (Post) :Create a customer,
- ☐ Get (GET) : Get customer details,

Product Schema (/product):

- ☐ Get (GET): Get all products
- ☐ Post (POST):Post product
- ☐ Update (PUT): Update products
- ☐ Delete (DELETE): Delete product

Order Schema (/Order):

- ☐ Get (GET): Add a new order
- ☐ Update (PUT): Update order

☐ Delete (DELETE): Delete product

Cart Schema (/cart):

☐ Get (GET): add product in cart
☐ Post (POST):Post product
☐ Update (PUT): Update products
☐ Delete (DELETE): remove product

Fronted: Build a project fronted using Next.js, a framework that supports Server-side rendering, and static site generation.

Backend: Build Backend using Sanity (CMS) to fetch products data, restore customer data and orders, & support APIs.

Third-Party APIs: Using third party APIs to manage products data, shipment APIs etc.