MARKETPLACE TECHNICAL FOUNDATION – PERFUMA

Name: Samiullah

Date: 18/1/2025

Time: 2:00 PM

Roll No: 00340796

OVERVIEW

The Marketplace Project is an e-commerce platform for browsing and purchasing luxury products across various categories. The platform is developed using **Next.js 14**, **Sanity CMS**, and several third-party API integrations to provide a seamless, responsive, and user-friendly experience for both buyers and sellers.

FUNCTIONAL SPECIFICATIONS

1. User Registration & Authentication:

- **Login & Registration:** Users can sign up/login via email or third-party services like Google, Facebook, etc.
- **Session Management:** Secure user session management using JWT tokens or NextAuth.js.
- **Password Reset:** Users can reset their password via email.

2. Product Management:

- **Product Listings:** Products can be categorized (e.g., Men, Women) with detailed information (name, price, stock, description).
- **Sanity CMS:** Enables easy management and updates of product data (CRUD operations on products).

3. Shopping Cart & Checkout:

- **Cart Management:** Users can add, remove, and update quantities of products, and view the total price.
- **Checkout Process:** Users enter shipping details and proceed to payment.
- **Payment Gateway Integration:** Integration with Stripe or EasyPaisa for secure payment processing.

4. Order Management:

- **Order Confirmation:** After checkout, users receive an order confirmation with an order ID.
- **Shipment Tracking:** Integration with third-party APIs (e.g., Shippo, ShipEngine) for real-time shipment tracking.
- **Order History:** Users can view previous orders and their status.

5. Admin Features (Backend):

- **Content Management:** Admins can add, update, or delete products via Sanity CMS.
- **Order Overview:** Admins can view, process, and manage orders.
- **User Management:** Admins can view and manage user profiles.

NON-FUNCTIONAL SPECIFICATIONS

1. Performance:

- The system should handle at least 1,000 concurrent users without performance degradation.
- Pages should load in 2-3 seconds under normal network conditions.

2. Security:

- **Authentication:** Secure authentication using JWT or NextAuth.js to ensure seamless and secure user logins.

3. Scalability:

- The platform should scale to accommodate more users and products as the marketplace grows.
- Utilize cloud hosting (e.g., Vercel, AWS) for scalability.

4. Usability:

- The platform should be mobile-responsive, supporting both iOS and Android browsers.
- Navigation and checkout should be intuitive and require minimal steps.

TECHNICAL SPECIFICATIONS

1. Frontend:

- **Framework:** Next.js 14 (React-based framework for SSR/SSG).
- **Styling:** TailwindCSS for rapid UI development.
- **State Management:** React Context or Redux for managing global state.
- **UI Components:** Modular and reusable components using ShadCN UI and React Query for UI consistency.
- **Essential Pages:** Homepage, Product Listing, Product Detail, Cart, Checkout, Order Confirmation, Login, and Sign-Up pages.

2. Backend:

- **Framework:** Next.js API Routes for handling server-side requests.
- **CMS:** Sanity CMS for managing product data (CRUD operations).
- **Database:** MongoDB for storing user profiles, orders, cart actions, and transactions.

- **Authentication:** NextAuth.js or Firebase Auth for handling user authentication and sessions.
- **Integrations:**
    - **Payment Gateway:** Integration with Stripe or EasyPaisa for payment processing.
    - **Shipment Tracking:** Integration with third-party shipment tracking APIs like Shippo or ShipEngine.

API SPECIFICATIONS

1. Product API:

- **Endpoint:** `/api/products`
- **Method:** GET
- **Description:** Fetch all products
- **Response Example:**

```
{
  "id": 1,
  "name": "Mueoen Jacket",
  "price": 2000,
  "stock": 50,
  "image": "url_to_image"
}
```

2. Order API:

- **Endpoint:** `/api/orders`
- **Method:** POST
- **Description:** Place a new order
- **Request Payload Example:**

```
{
  "userId": 1,
  "products": [{"id": 1, "quantity": 2}],
  "paymentStatus": "pending"
}
```

3. Shipment API:

- **Endpoint:** `/api/shipment/orders`
- **Method:** GET
- **Description:** Track the shipment for the specific order
- **Response Example:**

```
{
  "order": 123,
```

```
  "status": "Transit",
  "ETA": "2 days"
}
```

1.Detailed Schema for the Product:

```
export default {
  name: 'product',
  title: 'Product',
  type: 'document',
  fields: [
    {
      name: 'name',
      title: 'Product Name',
      type: 'string',
      validation: Rule => Rule.required().min(3).max(100),
    },
    {
      name: 'slug',
      title: 'Slug',
      type: 'slug',
      options: {
        source: 'name',
        maxLength: 96,
      },
      validation: Rule => Rule.required(),
    },
    {
      name: 'description',
      title: 'Description',
      type: 'text',
      validation: Rule => Rule.required().min(20),
    },
    {
      name: 'price',
      title: 'Price',
      type: 'number',
      validation: Rule => Rule.required().positive(),
    },
    {
      name: 'image',
      title: 'Main Image',
      type: 'image',
      options: {
        hotspot: true,
      },
```

```
      validation: Rule => Rule.required(),
    },
    {
      name: 'category',
      title: 'Category',
      type: 'reference',
      to: [{ type: 'category' }],
    },
    {
      name: 'isFeatured',
      title: 'Featured Product',
      type: 'boolean',
      description: 'Mark as featured on the homepage or special
section.',
      initialValue: false,
    },
    {
      name: 'brand',
      title: 'Brand',
      type: 'string',
      validation: Rule => Rule.required().min(2).max(50),
    },
    {
      name: 'colors',
      title: 'Colors',
      type: 'array',
      of: [{ type: 'string' }],
      options: {
        list: [
          { title: 'Red', value: 'Red' },
          { title: 'Blue', value: 'Blue' },
          { title: 'Black', value: 'Black' },
          { title: 'White', value: 'White' },
          { title: 'Green', value: 'Green' },
          { title: 'Yellow', value: 'Yellow' },
          { title: 'Gold', value: 'Gold' },
          { title: 'Silver', value: 'Silver' },
        ],
      },
      description: 'Select the colors available for this
product.',
    },
    {
      name: 'sizes',
      title: 'Sizes',
      type: 'array',
```

```
      of: [{ type: 'string' }],
      options: {
        list: [
          { title: 'S', value: 'S' },
          { title: 'M', value: 'M' },
          { title: 'L', value: 'L' },
          { title: 'XL', value: 'XL' },
          { title: 'XXL', value: 'XXL' },
        ],
      },
      description: 'Select the sizes available for this
product.',
    },
    {
      name: 'stockDetails',
      title: 'Stock Details',
      type: 'array',
      of: [
        {
          type: 'object',
          fields: [
            {
              name: 'size',
              title: 'Size',
              type: 'string',
              options: {
                list: [
                  { title: 'S', value: 'S' },
                  { title: 'M', value: 'M' },
                  { title: 'L', value: 'L' },
                  { title: 'XL', value: 'XL' },
                  { title: 'XXL', value: 'XXL' },
                ],
              },
              validation: Rule => Rule.required(),
            },
            {
              name: 'color',
              title: 'Color',
              type: 'string',
              options: {
                list: [
                  { title: 'Red', value: 'Red' },
                  { title: 'Blue', value: 'Blue' },
                  { title: 'Black', value: 'Black' },
                  { title: 'White', value: 'White' },
```

```
                { title: 'Green', value: 'Green' },
                { title: 'Yellow', value: 'Yellow' },
                { title: 'Gold', value: 'Gold' },
                { title: 'Silver', value: 'Silver' },
              ],
            },
            validation: Rule => Rule.required(),
          },
          {
            name: 'stockQuantity',
            title: 'Stock Quantity',
            type: 'number',
            validation: Rule => Rule.required().min(0),
          },
          {
            name: 'images',
            title: 'Images for Size/Color',
            type: 'array',
            of: [{ type: 'image' }],
            description: 'Upload multiple images for this
size/color combination.',
          },
        ],
      },
    ],
    description: 'Track the stock quantity and images for
each size and color combination.',
  },
 ],
};
```

2.Category Schema:

```
export default {
  name: 'category',
  title: 'Category',
  type: 'document',
  fields: [
    {
      name: 'name',
      title: 'Category Name',
      type: 'string',
      validation: Rule => Rule.required().min(2).max(50),
    },
    {
```

```
      name: 'slug',
      title: 'Slug',
      type: 'slug',
      options: {
        source: 'name',
        maxLength: 96,
      },
      validation: Rule => Rule.required(),
    },
  ],
};
```

3. User Schema

```
export default {
  name: 'user',
  title: 'User',
  type: 'document',
  fields: [
    {
      name: 'name',
      title: 'Name',
      type: 'string',
      validation: Rule => Rule.required().min(2).max(100),
    },
    {
      name: 'email',
      title: 'Email',
      type: 'string',
      validation: Rule => Rule.required().email(),
    },
    {
      name: 'profilePicture',
      title: 'Profile Picture',
      type: 'image',
      options: {
        hotspot: true,
      },
    },
    {
      name: 'address',
      title: 'Address',
      type: 'text',
    },
  ],
```

```
};
```

4. Order Schema

```
export default {
  name: 'order',
  title: 'Order',
  type: 'document',
  fields: [
    {
      name: 'orderId',
      title: 'Order ID',
      type: 'string',
      initialValue: () => `ORD-${Math.floor(Math.random() *
1000000)}`,
      validation: Rule => Rule.required(),
    },
    {
      name: 'user',
      title: 'User',
      type: 'reference',
      to: [{ type: 'user' }],
      validation: Rule => Rule.required(),
    },
    {
      name: 'products',
      title: 'Products',
      type: 'array',
      of: [
        {
          type: 'object',
          fields: [
            {
              name: 'product',
              title: 'Product',
              type: 'reference',
              to: [{ type: 'product' }],
            },
            {
              name: 'quantity',
              title: 'Quantity',
              type: 'number',
              validation: Rule => Rule.required().positive(),
            },
            {
```

```
              name: 'price',
              title: 'Price',
              type: 'number',
              validation: Rule => Rule.required().positive(),
            },
          ],
        },
      ],
    },
    {
      name: 'totalAmount',
      title: 'Total Amount',
      type: 'number',
      validation: Rule => Rule.required().positive(),
    },
    {
      name: 'paymentStatus',
      title: 'Payment Status',
      type: 'string',
      options: {
        list: ['pending', 'completed', 'failed'],
        layout: 'radio',
      },
      initialValue: 'pending',
    },
    {
      name: 'shippingStatus',
      title: 'Shipping Status',
      type: 'string',
      options: {
        list: ['not-shipped', 'shipped', 'delivered'],
        layout: 'radio',
      },
      initialValue: 'not-shipped',
    },
    {
      name: 'orderDate',
      title: 'Order Date',
      type: 'datetime',
      initialValue: () => new Date().toISOString(),
    },
  ],
};
```

5. Payment Schema

```
export default {
  name: 'payment',
  title: 'Payment',
  type: 'document',
  fields: [
    {
      name: 'order',
      title: 'Order',
      type: 'reference',
      to: [{ type: 'order' }],
      validation: Rule => Rule.required(),
    },
    {
      name: 'paymentMethod',
      title: 'Payment Method',
      type: 'string',
      options: {
        list: ['Credit Card', 'PayPal', 'Stripe', 'Cash on
Delivery'],
        layout: 'dropdown',
      },
      validation: Rule => Rule.required(),
    },
    {
      name: 'amount',
      title: 'Amount',
      type: 'number',
      validation: Rule => Rule.required().positive(),
    },
    {
      name: 'status',
      title: 'Payment Status',
      type: 'string',
      options: {
        list: ['completed', 'failed', 'pending'],
        layout: 'radio',
      },
      initialValue: 'pending',
    },
    {
      name: 'paymentDate',
      title: 'Payment Date',
      type: 'datetime',
      initialValue: () => new Date().toISOString(),
    },
  ],
```

```
};
```

Key Workflows

User Registration & Authentication Workflow:

- NextAuth.js handles authentication and session creation.
- User credentials are stored in MongoDB.
- A JWT token is issued for authentication.
- User session is maintained using the issued token.

Product Browsing Workflow:

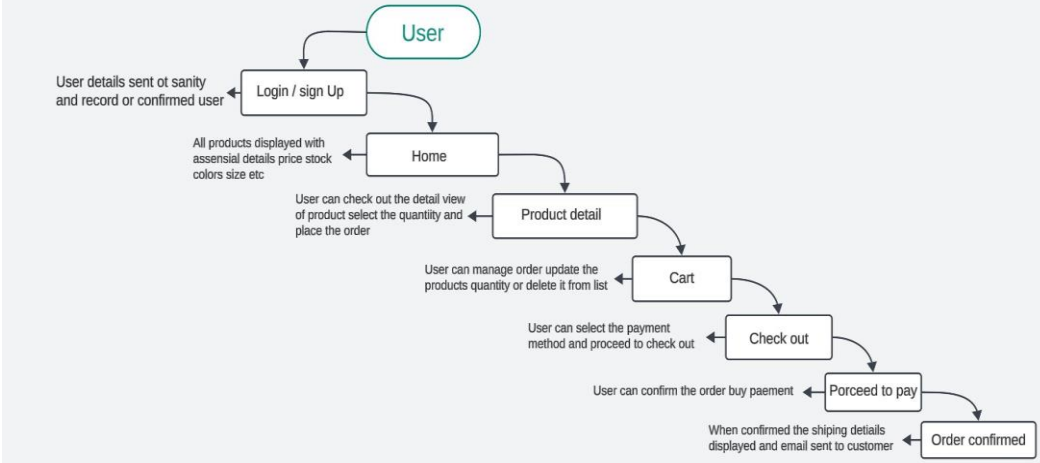
- The frontend requests product data from the Sanity API.
- Sanity fetches product data and sends a response.
- The frontend displays products dynamically based on the fetched data.

Order Placement Workflow:

- User browses products and adds them to the cart.
- User proceeds to checkout and submits the order.
- The API validates stock availability and processes payment.
- The order is stored in MongoDB and linked to the user.
- A confirmation response is sent to the frontend.

Shipment Tracking Workflow:

- The order status is updated by the admin after fulfillment.
- The user can request shipment details via the API.
- Shipment details are retrieved from MongoDB and displayed on the frontend.


User
Login / sign Up
User details sent ot sanity and record or confirmed user
Home
All products displayed with assensial details price stock colors size etc
Product detail
User can check out the detail view of product select the quantiity and place the order
Cart
User can manage order update the products quantity or delete it from list
Check out
User can select the payment method and proceed to check out
Porceed to pay
User can confirm the order buy paement
Order confirmed
When confirmed the shiping detiails displayed and email sent to customer