Marketplace Technical Foundation – Aromas

Syed Muhammad Ahmed Khalique

00416629

GIAIC, Quarter 2

16th January 2025

Table of Content

Technical Requirements

Frontend Requirements:

User Interface

1. **Home Page**:
    - Showcase featured perfumes, bestsellers, and new arrivals.
    - Highlight ongoing promotions or discounts.
2. **Product Listing Page**:
    - Display a grid or list of perfumes with:
        - Name
        - Image
        - Price
        - Ratings/Reviews
        - Discount badges
3. **Product Details Page**:
    - Display detailed information for each perfume:
        - Name
        - Detailed description
        - Price and discounts
        - Product images
        - Add to cart button
        - Customer reviews and ratings
4. **Cart Page**:
    - List all selected perfumes with:
        - Quantity controls
        - Subtotal and total price
        - Remove item option
        - Proceed to checkout button
5. **Checkout Page**:
    - Collect user information:
        - Name, email, and phone.
        - Shipping address.
    - Payment options (card, COD).
    - Summary of items with total price.
6. **Order Confirmation Page**:
    - Display order details:
        - Order ID.
        - Estimated delivery date.
        - Option to track the order.

Search and Filter

1. **Search Bar**:
    - Keyword search for perfume names and brands.
    - Auto-suggestions based on input.
2. **Filters**:
    - Categories: Gender (Men, Women, Unisex), Concentration (EDP, EDT, etc.).
    - Price Range Slider.
    - Brands.
    - Ratings.
3. **Sorting Options**:
    - By price (low to high, high to low).
    - By popularity.
    - By new arrivals.

Responsive Design

1. **Mobile and Tablet Support**:
    - Adjust layout and images for smaller screens.
    - Collapsible filters and menu.
    - Mobile-optimized cart and checkout flow.
2. **Desktop Support**:
    - Full-screen layout with detailed views.

User Authentication

1. **Login/Signup**:
    - Social login (Google, Facebook).
    - Email-based signup.
2. **Profile Page**:
    - View order history.
    - Edit personal and shipping details.

Notifications and Alerts

1. **Toast Notifications**:
    - Item added to cart.
    - Successful login/signup.
    - Errors (e.g., invalid input).
2. **Email Subscription Popup**:
    - Prompt for promotions and discounts.

Other Features

1. **Wishlist**:
    - Allow users to save favorite perfumes.

2. **Reviews and Ratings**:
    - Allow users to submit reviews

Tools used to achieve Frontend requirements:

1.Framework

Next JS (Framework)

- Build the UI for the store with components like Home, Product Listing, Product Details, Cart, and Checkout.

Tailwind CSS (CSS Framework)

- Design a responsive and modern UI quickly using utility-first CSS classes.

Next App Router

- Manage navigation between pages (e.g., Hom e, Product Listing, Product Details, Cart, Checkout).

2. Component Libraries

ShadCN or Ant Design

- Use prebuilt components like buttons, modals, and forms.

3. State Management

React Context API

- Manage global state for features like the shopping cart, user authentication, and product filters.

4. API Integration

Axios or Fetch API

- Fetch data from backend APIs for products, reviews, and cart operations.

5. UI Enhancements

Swiper.js

- Add carousels for showcasing featured products or offers.

React Icons

- Add icons to enhance UI elements (e.g., cart, user profile, search).

Framer Motion

- Create smooth animations for UI elements like modals and page transitions.

6. Forms and Validation

Formik

- Build and validate forms for user registration, login, and checkout.

7. Development Utilities

VS Code Extensions:

- **ESLint**: Enforce code quality and consistent formatting.
- **Prettier**: Automatically format code.
- **Tailwind CSS IntelliSense**: Autocompletion for Tailwind classes.
- **React Developer Tools**: Debug React components.

Backend Requirements

Using Sanity CMS as Backend for this e-commerce

1. Schema Design in Sanity

Create the following schemas in Sanity to align with the business needs:

1. **Product Schema(perfume)**:
    - **Fields**:
        - name (string): Name of the perfume.
        - description (text): Detailed description of the perfume.
        - price (number): Price of the perfume.
        - discount (number): Discount percentage.
        - images (array of image): Product images.
        - category (string): E.g., "Men," "Women," or "Unisex."
        - size (array of numbers): volume in ml
        - tags (array of string): Tags like "floral," "citrus."
        - rating (number): Average rating.
        - stock_quantity (number): Available stock.
        - sku (string): sku of the product
        - slug (string): slug of the product
2. **User Schema**:
    - **Fields**:
        - name (string): Full name of the user.
        - email (string): Email address (unique).
        - password (string): Hashed password.
        - phone (string): Contact number.
        - address (array of objects): Multiple shipping addresses.
        - wishlist (array of references): Reference to saved perfumes.
        - order_history (array of orders): previous orders.
3. **Order Schema**:
    - **Fields**:
        - user (reference): Reference to the user placing the order.
        - items (array of objects): Each object includes:
            - perfume (reference): Reference to the perfume.
            - quantity (number): Quantity ordered.

        - price (number): Price at the time of purchase.
    - total_amount (number): Total price of the order.
    - order_status (string): Status (Pending, Shipped, Delivered, Cancelled).
    - order_date (datetime): Date of order placement.
    - tracking_id (string): Tracking ID for shipment.
4. **Review Schema**:
    - **Fields**:
        - perfume (reference): Reference to the perfume being reviewed.
        - user (reference): Reference to the user who wrote the review.
        - rating (number): Star rating (1–5).
        - review_text (text): Review content.
        - date (datetime): Date of review.
5. **Shipment Schema**:
    - **Fields**:
        - order (reference): Reference to the associated order.
        - carrier (string): Name of the shipping service.
        - tracking_id (string): Unique shipment tracking ID.
        - status (string): Shipment status (In Transit, Delivered, etc.).
        - destination (string): Shipment destination address.
        - estimated_delivery (datetime): Expected delivery date.
        - customer_contact (string): phone number of customer.

2. Data Management with Sanity

1. **Product Data**:
    - Store all perfume details in the Perfume Schema.
    - Update stock quantity after each purchase using Sanity mutations.
2. **Order Management**:
    - Use the Order Schema to track user purchases.
    - Update order status based on user actions or admin updates.
3. **User Authentication**:
    - Sanity can store user profiles, but passwords should be hashed and managed securely.
4. **Real-Time Updates**:
    - Utilize Sanity's real-time capabilities to sync stock levels, orders, and reviews.
5. **Image Management**:
    - Use Sanity's built-in **Image Asset Pipeline** for storing and optimizing product images.

3. API Endpoints with Sanity

1. **Perfumes**:
    - **GET**: Fetch all perfumes.

    - **GET**: Fetch a single perfume by ID.
    - **POST/PUT**: Add or update perfume details.
2. **Orders**:
    - **POST**: Create a new order.
    - **GET**: Fetch order details by user ID.
3. **Users**:
    - **POST**: Register a new user.
    - **POST**: Authenticate user login.
    - **GET**: Fetch user profile and order history.
4. **Reviews**:
    - **POST**: Add a new review.
    - **GET**: Fetch reviews for a perfume.
5. **Shipments**:
    - **GET**: Track shipment by tracking ID.

Tools used to achieve Backend requirements:

Sanity CMSwill be used to manage the data it will be used as a backend and database

- **Sanity Studio**to manage and update the data of the components
- **Schema** to design and create components schema
- **Sanity Query** write query to get all data or specific data from sanity using **GROQ** query language

Sanity Schema for the above components is in schema folder

Third Party API

1. Shipment Tracking API

To track orders and update customers about the status of their shipment.

Requirements:

- **API Integration**:
    - Choose a suitable tracking API like TrackingMore,UPS Developer Kit.
    - Obtain API keys and configure authentication.
- **Features**:
    - Retrieve shipment status.
    - Provide estimated delivery dates.
    - Fetch real-time tracking updates using tracking numbers.
- **Frontend Impact**:
    - Display shipment status on the "Order Details" page.
    - Add a "Track Your Order" button with a link to the carrier's tracking page.
- **Backend Implementation**:

    - Store shipment details in the database.
    - Use webhooks or periodic API calls to keep shipment statuses updated.

2. Payment Gateway API

To handle secure online payments for orders.

Requirements:

- **API Integration**:
    - Choose a payment provider likeJazzCash, Easypaisa.
    - Obtain API keys and configure secure payment workflows.
- **Features**:
    - Create and confirm payments.
    - Support multiple payment methods.
    - Refund processing and transaction history.
- **Security**:
    - Use HTTPS for secure communication.
    - Ensure PCI DSS compliance.
    - Encrypt sensitive customer payment information.
- **Frontend Impact**:
    - Integrate a payment form on the checkout page.
    - Display payment success or failure messages.
- **Backend Implementation**:
    - Validate payment before confirming the order.
    - Store transaction details securely for auditing purposes.

3. Address Validation API

To ensure valid customer addresses for deliveries.

Requirements:

- **API Integration**:
    - Use Google Maps API.
- **Features**:
    - Validate entered addresses to ensure they are deliverable.
- **Frontend Impact**:
    - Provide a seamless form-filling experience for customers during checkout.
- **Backend Implementation**:
    - Normalize and store validated addresses for shipment.

4. Notification API

To send email or SMS notifications for order confirmations and updates.

Requirements:

- **API Integration**:
    - Use providers like Twilio (for SMS) or SendGrid (for email).

- **Features**:
    - Send order confirmation, shipment updates, and promotional messages.
    - Support customizable templates for notifications.
- **Frontend Impact**:
    - Notify users about the success or failure of notifications.
- **Backend Implementation**:
    - Trigger notifications on specific actions.

7. Analytics API

To collect user activity and business insights.

Requirements:

- **API Integration**:
    - Use Google Analytics or Mixpanel.
- **Features**:
    - Track user interactions (e.g., product views, cart additions).
    - Provide sales and traffic reports.
- **Frontend Impact**:
    - Add tracking snippets to relevant pages.
- **Backend Implementation**:
    - Analyze collected data for business decision-making.

Error Handling and Security

When using third party API's you need to take some security measures and error handling, handle errors for invalid request and other errors. Protect API keys by using environment variables.

System Architecture

The below diagram shows the System Architecture and data flow of the E-commerce site.

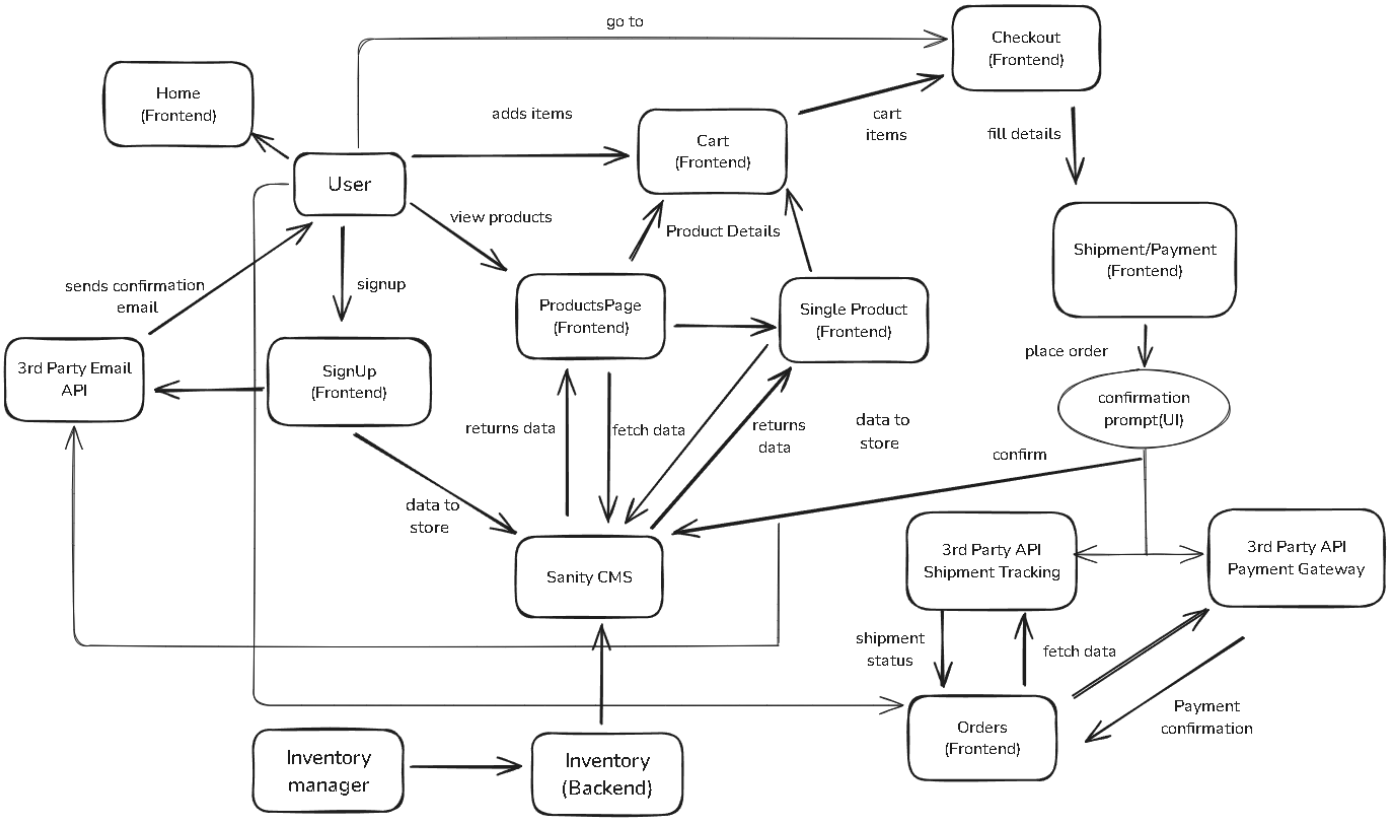

Figure 1Data Flow

In the architecture the data flow is:

1. The user visits the site and signs up via the Signup page. The signup details are sent to Sanity CMS for storage, and a confirmation email is sent using a 3rd Party Email API.
2. After signing up, the user can navigate between pages such as Home, Cart, Orders, and Products.
3. When the user visits the **Products Page**, the frontend requests product data from Sanity CMS. The data is fetched and displayed dynamically on the UI.
4. The user can view details of a specific product by clicking on it. The Single Product page fetches detailed product information from Sanity CMS and displays it.
5. The user adds items to the cart from the Single Product page. Cart items are managed on the frontend (local storage).
6. The user navigates to the Cart page to review their items.
7. The user proceeds to the **Checkout page**, fills in their details, and places the order.
8. The order details are sent to Sanity CMS for storage, and the inventory is updated through the backend Inventory Manager.

9. Payment details are processed securely via a 3rd Party Payment Gateway, and a confirmation is displayed to the user.
10. The user can track their shipment through the Orders page, where the Shipment Tracking API fetches real-time status updates, which are displayed in the UI.

API Requirements

API Endpoints Documentation

Task	Endpoint name	Method	Description	Request body example	Response example
Fetch All Products	/products	GET	Fetch all product details		[{ "id": 1, "name": "Perfume A", "description": "A fresh and floral scent.", "price": 100, "category": "Floral", "stock": 50, "rating": 4.5, "images": ["image1.jpg", "image2.jpg"] },{ },{ }]

Fetch Single Product	/products/:id	GET	Fetch details of a specific product by ID		{
					"id": 1,
					"name": "Perfume A",
					"description": "A fresh and floral scent.",
					"price": 100,
					"category": "Floral",
					"stock": 50,
					"rating": 4.5,
					"images": ["image1.jpg", "image2.jpg"]
					}

Place an Order	/orders	POST	Place an order for the items in the cart	{"userId": 123,
				"cartItems": [
				{
				"productId": 1,
				"quantity": 2,
				"price": 100
				},
				{
				"productId": 2,
				"quantity": 1,
				"price": 120
				}
				],
				"paymentMethod": "Credit Card",
				"shippingAddress": "123 Main Street, City"}
					{
					"success": true,
					"message": "Order placed successfully.",
					"orderId": 456
					}

Track Shipment	/orders/:id /shipment	GET	Fetch the shipment status of a specific order		{
					"orderId": 456,
					"shipmentStatus": "In Transit",
					"estimatedDelivery": "2025-01-20"
					}

Process Payment	/payments	POST	Process a payment for an order	{
				"orderId": 456,
				"paymentMethod": "Credit Card",
				"amount": 320
				}
					{
					"success": true,
					"message": "Payment processed successfully.",
					"transactionId": "TXN123456"
					}

Road map and Milestones