MARKETPLACE TECHNICAL FOUNDATION - CasaEncanto

Technical Planning Documentation

OVERVIEW

The Marketplace Project is an e-commerce platform where you can browse and buy products from different categories. Built with Next.js 14, Sanity CMS and various 3rd party API integrations this marketplace provides a smooth, responsive and user friendly experience for both buyers and sellers.

FUNCTIONAL SPECIFICATIONS

1. User Registration & Authentication:

- **Login & Registration:** Users can Signup/Login through Email or 3rd party auth (Google, Facebook etc.).
- **Session Management:** Secure user session management using JWT tokens or NextAuth.js.

- **Password Reset:** Users can Reset their passwords via email.

2. Product Management:

- **Product Listings:** Products can display in categories (Wall Decor, Furniture etc.) with details (name, price, stock, description).
- **Sanity CMS**: Manage and update product data easily (CRUD operations on products)

3. Shopping Cart & Checkout:

- **Cart Management:** Add/remove items, update quantities, view total price in cart
- **Checkout Process:** Enter shipping details and proceed to payment
- **Payment Gateway:** Integrate with Stripe or Easypaisa for secure payments

4. Order Management:

- **Order Confirmation:** After checkout user gets an order confirmation with order ID
- **Shipment Tracking:** Integrate with 3rd party APIs (e.g. Shippo,ShipEngine) for real time shipment tracking
- **Order History:** View previous orders and their status

5. Admin Features (Backend):

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

NON-FUNCTIONAL SPECIFICATIONS

1. Performance:

- 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:

- **Secure Authentication:** Use JWT or NextAuth.js for stateless user authentication.

3. Scalability:

- Platform should scale with more users and products as the marketplace grows.
- Use cloud hosting (e.g. Vercel, AWS) for scalability.

4. Usability:

- Mobile responsive, supports iOS and Android browsers.
- Users should be able to navigate and checkout in minimal steps.

TECHNICAL SPECIFICATIONS

1. Frontend:

- **Framework:** Next.js 14 (React-based framework for SSR/SSG).
- **Styling:** Tailwind CSS for rapid UI development.
- **State Management:** Use React Context or Redux for managing global state.
- **Components:** Use modular and reusable components with ShadCNUI, DaisyUI, TailBlock, for UI consistency.
- **Essential pages:** Home, Product Listing, Product Details, Cart, Checkout, and Order Confirmation, Login/Signup page.

2. Backend:

- **Backend Framework:** Next.js API Routes for handling server-side requests.
- **CMS:** Sanity CMS for managing content and product data (CRUD operations on products).
- **Database:** MongoDB serves as the backend database to store user's profile, orders cartand transactions etc..
- **Authentication:** NextAuth.js or Firebase Auth for handling user authentication and sessions.

3. Integrations:

- **Payment Gateway:** Stripe or EasyPaisa, JazzCash for handling payments.
- **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:** { **"id":** 1, **"name":** "Timbler Craft", **"price":** 2000, **"stock":** 50, **"image":** "url_to_image" }

2. Order API:

- **Endpoint:** `/api/orders`
- **Method**: **POST**
- **Description:** Place a new order
- **Request Payload:**`{ "userId": 1, "products": [{"id": 1, "quantity": 2}], "paymentStatus": "pending" }`

3. Shipment API:

- **Endpoint:** `/api/shipment/{orders}`
- **Method**: **GET**
- **Description:** Track the shipment for a specific order
- **Response:** `{ "orderId":123, "status":"In Transit", "ETA":"2 days" }`

User Authentication APIs

ENDPOINTS	METHOD	PURPOSE	REQUEST PAYLOAD	RESPONSE
/api/auth/signup	POST	Register a new user	{ "name": "John", "email": "john@example.com", "password": "123456" }	{ "message": "User registered", "userId": "abc123" }
/api/auth/login	POST	User login & token generation	{ "email": "john@example.com", "password": "123456" }	{ "token": "jwt-token", "user": { "id": "abc123", "name": "John" } }
/api/auth/logout	POST	User logout	{ "token": "jwt-token" }	{ "message": "Logged out successfully" }

/api/auth/session	GET	Get current user session	Authorization: Bearer token	{ "user": { "id": "abc123", "name": "John" } }

Product Management APIs

ENDPOINTS	METHOD	PURPOSE	REQUEST PAYLOAD	RESPONSE
/api/products	GET	Fetch all products	-	[{"id":1, "name":"Shirt", "price":100}]
/api/products/:id	GET	Fetch single product	-	{ "id": 1, "name": "Shirt", "price": 100 }
/api/products	GET	Add a new product (Admin Only)	{ "name": "Shirt", "price": 100, "stock": 50 }	{ "message": "Product added" }
/api/products/:id	PUT	Update a product (Admin Only)	{ "name": "Updated Shirt", "price": 120 }	{ "message": "Product updated" }
/api/products/:id	DELETE	Delete a product (Admin Only)	-	{ "message": "Product deleted" }

Cart Management APIs

ENDPOINTS	METHOD	PURPOSE	REQUEST PAYLOAD	RESPONSE
/api/cart	GET	Fetch user cart	-	{ "cart": [{ "productId": 1, "quantity": 2 }] }
/api/cart	POST	Add product to cart	{ "productId": 1, "quantity": 2 }	{ "message": "Product added to cart" }

/api/cart/:id	DELETE	Remove product from cart	-	{ "message": "Product removed" }

Order Management APIs

ENDPOINTS	METHOD	PURPOSE	REQUEST PAYLOAD	RESPONSE
/api/orders	GET	Get all orders for a user	Authorization: Bearer token	[{"id":1, "status":"Pending "}]
/api/orders/:id	GET	Fetch single order details	-	{ "id": 1, "status": "Pending", "items": [...] }
/api/orders	POST	Place a new order	{ "userId": "abc123", "items": [{ "productId": 1, "quantity": 2 }] }	{ "message": "Order placed", "orderId": "xyz789" }
/api/orders/:id/status	PUT	Update order status (Admin Only)	{ "status": "Shipped" }	{ "message": "Order status updated" }

Shipment & Tracking APIs

ENDPOINTS	METHODS	PURPOSE	REQUEST PAYLOAD	REPONSE
/api/shipments	GET	Get all shipments	-	[{"id":1, "status":"In Transit"}]
/api/shipments/:id	GET	Track a shipment	-	{ "id": 1, "status": "Delivered", "tracking": "XYZ12345" }

/api/shipments	POST	Create a new shipment (Admin Only)	{ "orderId": "xyz789", "courier": "DHL" }	{ "message": "Shipment created" }

Payment APIs

ENDPOINTS	METHODS	PURPOSE	REQUEST PAYLOAD	REPONSE
/api/payments	POST	Process a payment	{ "orderId": "xyz789", "amount": 200, "method": "Credit Card" }	{ "status": "Success", "transactionId": "TXN12345" }
/api/payments/:id	GET	Fetch payment details	-	{ "transactionId": "TXN12345", "status": "Success" }

User Profile APIs

ENDPOINTS	METHODS	PURPOSE	REQUEST PAYLOAD	RESPONSE
/api/users/:id	GET	Fetch user profile	-	{ "id": "abc123", "name": "John", "email": "john@example.com" }
/api/users/:id	PUT	Update user profile	{ "name": "Updated Name" }	{ "message": "Profile updated" }

SANITY SCHEMA FOR MARKETPLACE:

1. Product Schema

```
export default {
 name: 'product',
```

```
type: 'document',
 title: 'Product',
fields: [ {
        name: 'name', type: 'string', title: 'Product Name' },
        { name: 'description', type: 'text', title: 'Description' },
         { name: 'price', type: 'number', title: 'Price' },
         { name: 'stock', type: 'number', title: 'Stock Level' },
         { name: 'category', type: 'reference', to: [{ type: 'category' }], title: 'Category' },
        { name: 'image', type: 'image', title: 'Product Image' }, ], };
```

2. Category Schema

```
export default {
name: 'category',
 type: 'document',
title: 'Category',
fields: [
   { name: 'title', type: 'string', title: 'Category Name' },
   { name: 'description', type: 'text', title: 'Description' }, ], };
```

3. Order Schema

```
export default
 { name: 'order',
 type: 'document',
 title: 'Order',
fields: [
    { name: 'user', type: 'reference', to: [{ type: 'user' }], title: 'User' },
    { name: 'items', type: 'array', of: [{ type: 'reference', to: [{ type: 'product' }] }],
     title: 'Ordered Items' },
    { name: 'totalPrice', type: 'number', title: 'Total Price' },
    { name: 'status', type: 'string', title: 'Order Status', options: { list: ['Pending', 'Shipped', 'Delivered',
'Cancelled'] } },
    { name: 'createdAt', type: 'datetime', title: 'Order Date' }, ], };
```

4. User Schema

```
export default {
name: 'user',
type: 'document',
```

```
title: 'User',
fields: [
    { name: 'name', type: 'string', title: 'Name' },
    { name: 'email', type: 'string', title: 'Email' },
    { name: 'address', type: 'text', title: 'Address' },
    { name: 'orders', type: 'array', of: [{ type: 'reference', to: [{ type: 'order' }] }], title: 'Order History' },
], };
```

KEY WORKFLOWS:

User Registration & Authentication Workflow

1. User Registration & Authentication Workflow
2. NextAuth.js handles authentication and session creation.
3. User details are stored in MongoDB.
4. JWT token is issued for authentication.

Product Browsing Workflow

1. Frontend requests product data from Sanity API.
2. Sanity fetches product data and sends a response.
3. Frontend displays products dynamically.

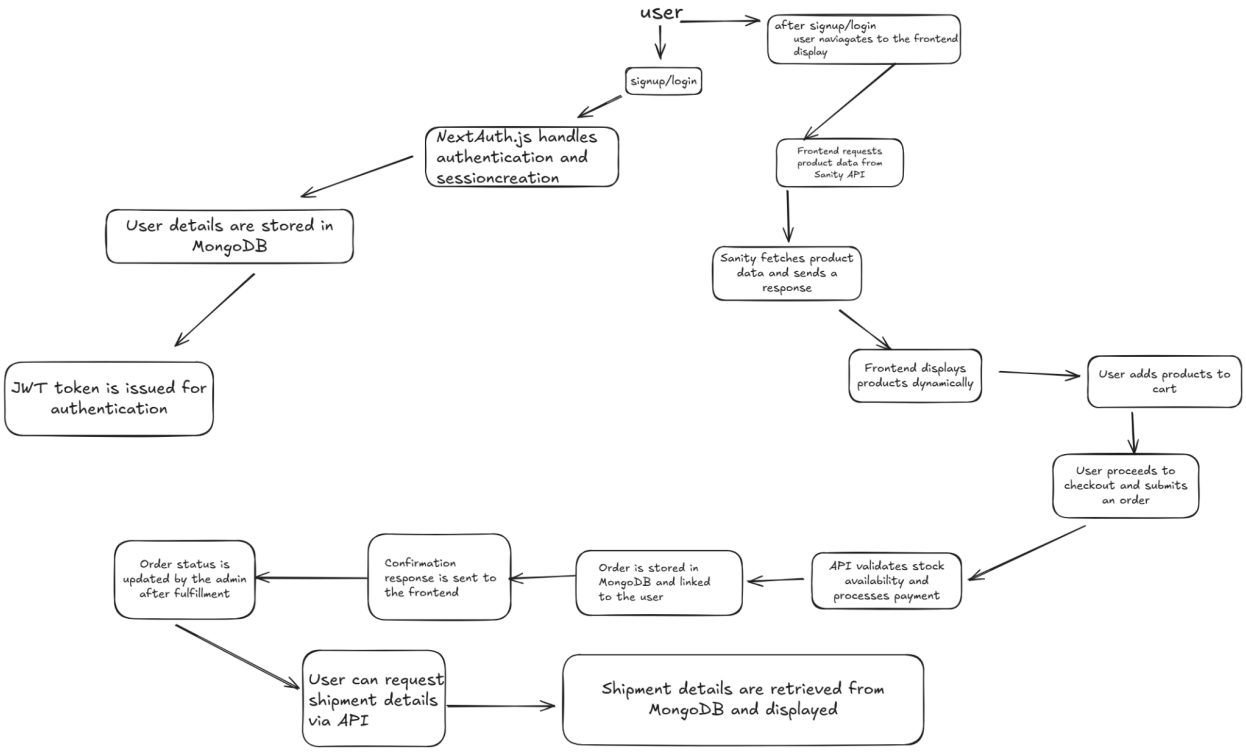
Order Placement Workflow

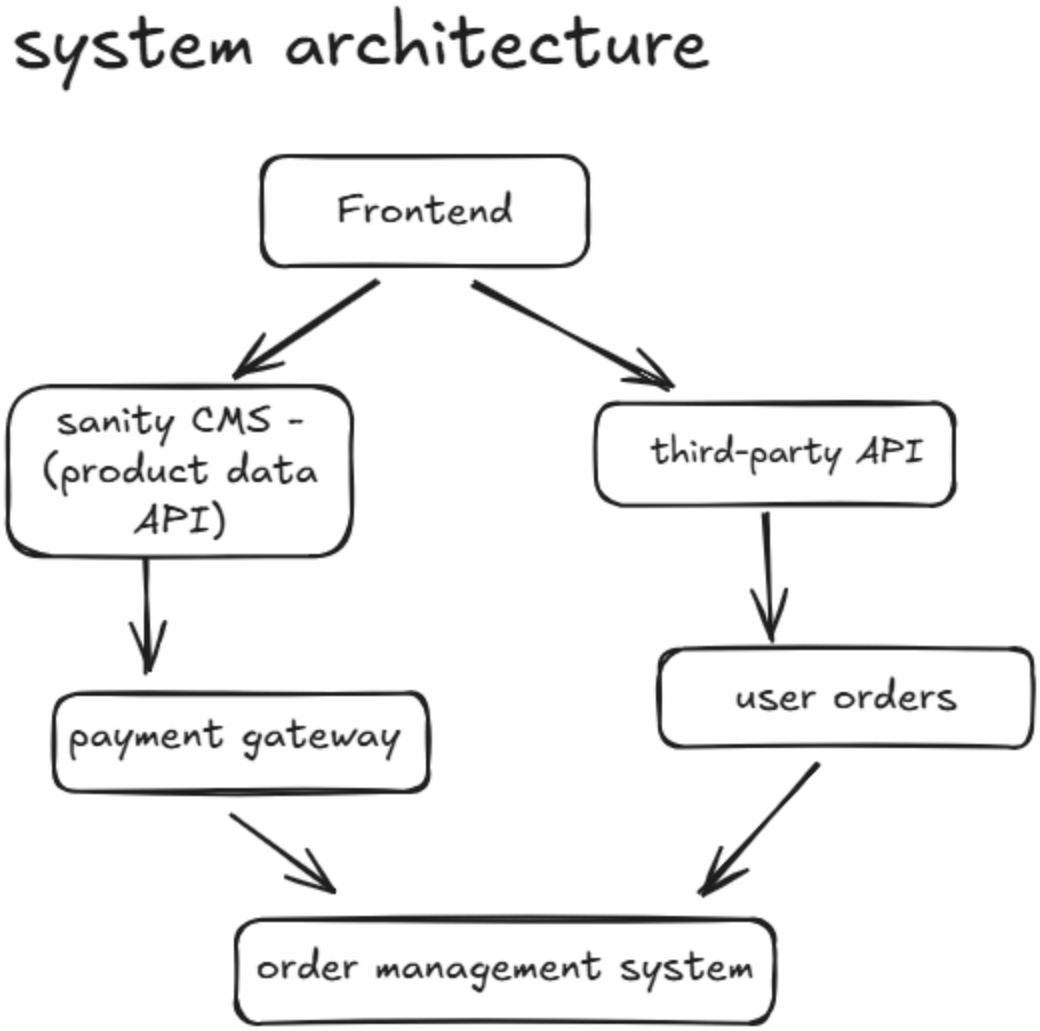
1. User adds products to cart.
2. User proceeds to checkout and submits an order.
3. API validates stock availability and processes payment.
4. Order is stored in MongoDB and linked to the user.
5. Confirmation response is sent to the frontend.

Shipment Tracking Workflow

1. Order status is updated by the admin after fulfillment.
2. User can request shipment details via API.
3. Shipment details are retrieved from MongoDB and displayed.

WORKFLOW ARCHITECTURE





Document prepared By : Laiba Khan