Day 02:

Planning the Technical Foundation

Marketplace Technical Foundation

Define Technical Requirements :

• Frontend Requirements :

- A clean and user-friendly design, using **Next.Js** optimized for browsing.
- Using **Tailwind CSS** for styling the website.
- Website is **fully responsive** and adapts to different screen sizes (mobile, tablet, and desktop).
- **General Pages:**
  - **Signup Page:** To create an account on the website.
  - **Login Page:** To access your existing account using email, and password.
  - **Home Page:** Highlights featured products, promotions, and categories.
  - **Products Page:** Display all products and enable filtering by category, size or price and sorting by popularity or price.
  - **About and Contact Page:** Business details and a contact form for customer inquiries.
  - **Product Details Page:** Show detailed information about each product (e.g., size guide, product description, customer reviews) and an "Add to Cart" button.
  - **Cart Page:** Allow users to view, update, or remove items in their cart.
  - **Checkout Page:** Enable easy and secure payment, address entry, and shipping options.
  - **Order Confirmation Page:** Display order summary, payment success, and tracking link.

• Backend :

- Using **sanity**, headless CMS, which will serve as a backend to handle data
- Manage product data, customer details, and order history.

1. Product Data Management

- **Objective:**
  - Manage product details (names, descriptions, prices, images, stock levels).
- **Implementation:**

- Use Sanity CMS to store and manage product information.
- Design product schemas to organize data (e.g., categories, sizes, colors, stock levels).

2. Customer Details Management

- **Objective:**
  - Manage customer information (name, contact, shipping address).
- **Implementation:**
  - Store customer data securely in Sanity CMS.
  - Ensure privacy and security by following data protection regulations.

3. Order Records Management

- **Objective**:
  - Track order details (customer info, items, quantities, prices, order status).
- **Implementation:**
  - Store and update order records in real-time as they move through the fulfilment process (e.g., "pending", "shipped", "delivered").

4. Schema Design

- **Objective:**
  - Create reusable and flexible schemas for products, customers, and orders.
- **Implementation:**
  - Design custom schemas to capture product, customers and order data
  - Ensure schemas are flexible for easy updates and additions.

➢ **Implementation of Sanity:**

1. **Set up Sanity CMS:** Install and configure **Sanity.io** for your e-commerce platform.
2. **Create Product, Customer, and Order Schemas:** Define schemas for each entity with relevant fields.
3. **Integrate with Frontend**: Fetch and manipulate data (products, customers, orders) on the frontend using **GROQ queries.**
4. **Manage Content:** Use the **Sanity Studio interface** for easy content updates (products, prices, customer data).

5. **Real-Time Updates:** Ensure order statuses are updated in real-time as they move through the fulfilment process.

• APIs :

- Integrate **third-party APIs** for tracking, payment gateways.
- Ensure APIs provide necessary data for frontend functionality.

➢ Third-party APIs Integration:

1. **Shipment Tracking API**
   - **Objective:** Track orders in real-time.
   - **Functionality:**
     - Fetch real-time shipment status (e.g., "In Transit", "Delivered").
     - Show expected delivery date.

2. **Payment Gateway API**
   - **Objective:** Secure payment processing.
   - **Functionality:**
     - Support multiple payment methods (credit/debit cards,...).
     - Secure handling of sensitive customer data.

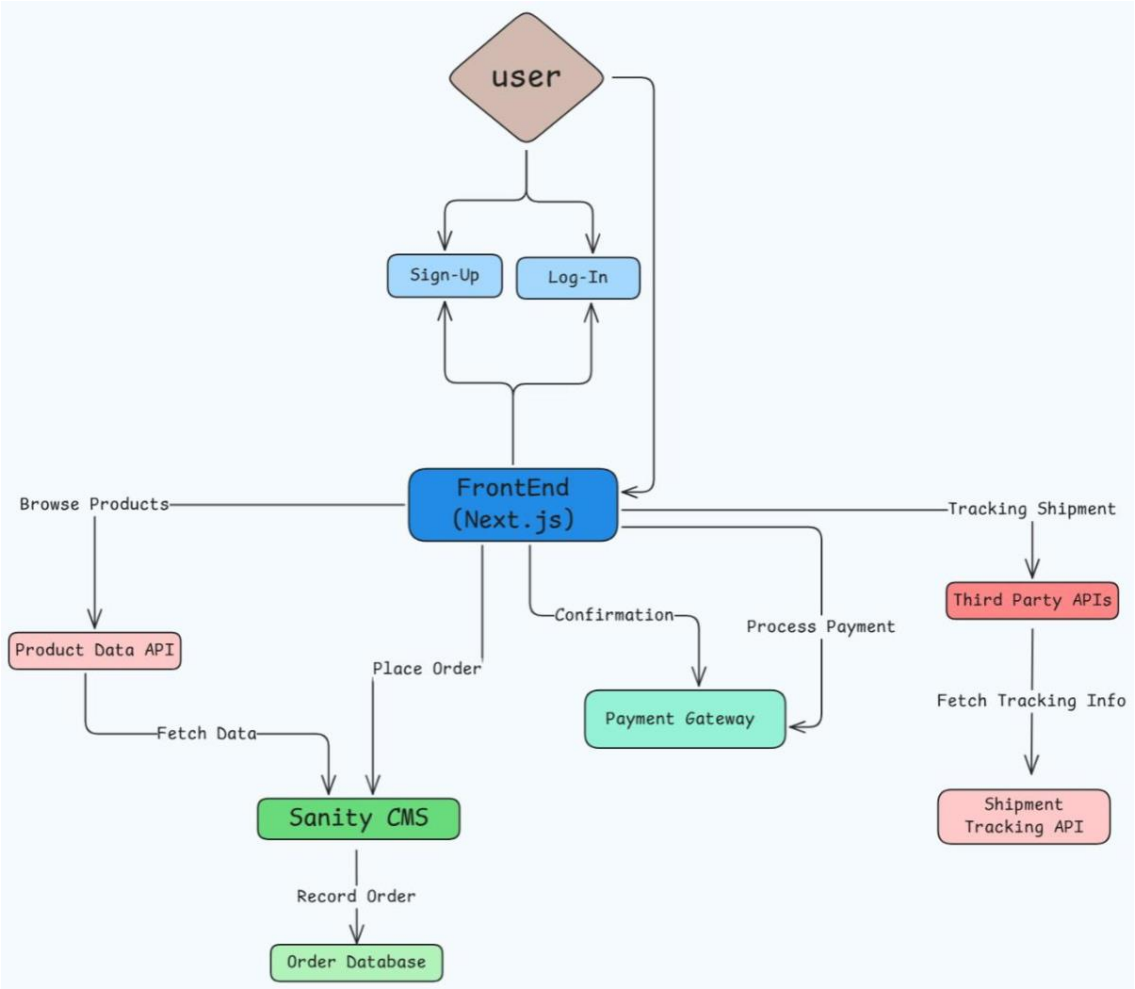
3. **Backend Services APIs**
   a. **Email Notifications**
      - **Functionality:** Send order confirmations, shipping updates.

   b. **Social Media Sharing**
      - **Functionality:** Allow customers to share products/orders.

   c. **Customer Support**
      - **Functionality:** Provide live chat support or chatbot services.

4. **Ensure API Compatibility**
   - **Objective:** Smooth integration with frontend features.
   - **Key Features:**
     - Ensure third-party APIs deliver required data.
     - Handle API failures with proper error management.

Design System Architecture:



1.User Browsing:

- A user visits the marketplace frontend to browse products.
- The frontend requests product listings from the Product Data API.

2.Product Display:

- The Product Data API fetches data from Sanity CMS.
- Product details are displayed dynamically on the site.

3.Order Placement:

- When the user places an order, the order details are sent to Sanity CMS via an API Request.
- The order is recorded in Sanity CMS.

4.Shipment Tracking:

- Shipment tracking information is fetched through a Third-Party API.
- Real-time tracking updates are displayed to the user.

5.Payment Processing:

- Payment details are securely processed through the Payment Gateway.
- A confirmation is sent back to the user and recorded in Sanity CMS.

Plan API Requirements

Endpoints	Method	Description
/users/signup	Post	Register a new user
/Products	Get	Fetches All Products
/Order	Get	Creates a new order
/Orders/ID	Get	Fetches a specific order
/cart/add	Post	Add a product to the users cart
/cart	Get	Retrieves the current state of the user's cart.
/checkout	Post	Processes payment and places an order.
/Shipment/Tracking/ID	Get	Tracks shipment status

Write Technical Documentation

● Key Workflows:

1 User Registration & Authentication

- **User Signup:** Users register by providing their email, password, and profile details. Data is stored in the database after validation.
- **Login**: Users enter their credentials to obtain a JWT token, enabling secure Session handling.
- **Password Recovery:** Users reset passwords via a token-based recovery System.

2 Product Browsing & Filtering

- Users view categories fetched from the CMS.
- Clicking a category triggers the /categories/{id}/products API to display.
- Users can filter products by attributes such as price, ratings, or availability.

3 Cart Management

- Users add products to their cart via the /cart/add endpoint.
- The cart updates dynamically, storing items in the database or local storage for guest users.
- The cart is displayed using the /cart endpoint.

4 Checkout & Payment

- The user proceeds to checkout, providing payment and shipping details.
- The /checkout endpoint processes the payment and creates an order.
- Users receive order confirmation via email.

- **Sanity Schema**

Products Schema

```
export const product = {
    name: "product",
    type: "document",
    title: "Product",
    fields: [
        { name: "productId",
            type: "string",
            title: "Product ID",
            validation: (Rule: any) =>
                Rule.required().custom(async (productId: string, context: any) => {
                    if (!productId) {
                        return "Product ID is required.";
                    }}),
        },
        { name: "name",
            type: "string",
            title: "Product Name",
            validation: (Rule: any) => Rule.required(),
        },
        { name: "price",
            type: "number",
            title: "Product Price",
            validation: (Rule: any) => Rule.required().positive(),
        },
        {   name: "tags",
            type: "array",
            title: "Product Tags",
            of: [{ type: "string" }],
            options: {layout: "tags",},
        },
        {  name: "discountedPrice",
            type: "number",
            title: "Discounted Price",
            validation: (Rule: any) => Rule.custom((discountedPrice: number, context: any) => {
                    const price = context.document?.price;
                    if (discountedPrice && discountedPrice >= price) {
                        return "Discounted price must be less than the original price.";}
                    return true;
                }),
        },
        {  name: "stock",
            type: "number",
            title: "Product Stock",
            validation: (Rule: any) => Rule.required().integer().min(0),
        },
        {   name: "sizes",
            type: "array",
            title: "Product Sizes",
            of: [{ type: "string" }],
            options: { list: ["S", "M", "L", "XL", "XXL"],},
        },
        {  name: "images",
            type: "array",
            title: "Product Images",
            of: [{ type: "image",
                    fields: [ {  name: "alt",
                            type: "string",
                            title: "Alternative Text",
                            validation: (Rule: any) => Rule.required(), }, ],
        },],
    },],
};
```

Customer Schema

```
export const customer = {
    name: "customer",
    type: "document",
    title: "Customer",
    fields: [
        {  name: "customerId",
            type: "string",
            title: "Customer ID",
            validation: (Rule: any) => Rule.required().error("Customer ID is required."),
        },
        {
            name: "name",
            type: "string",
            title: "Customer Name",
            validation: (Rule: any) =>
                Rule.required()
                    .min(2)
                    .error("Customer name must be at least 2 characters long."),
        },
        {
            name: "email",
            type: "string",
            title: "Email Address",
            validation: (Rule: any) =>
                Rule.required()
                    .email()
                    .error("Please provide a valid email address."),
        },
        {
            name: "contact",
            type: "string",
            title: "Contact Number",
            validation: (Rule: any) =>
                Rule.required()
                    .min(10)
                    .max(15)
                    .error("Contact number must be between 10 and 15 characters."),
        },
        {
            name: "address",
            type: "object",
            title: "Address",
            fields: [
                { name: "street", type: "string", title: "Street" },
                { name: "city", type: "string", title: "City" },
                { name: "state", type: "string", title: "State" },
                { name: "postalCode", type: "string", title: "Postal Code" },
                { name: "country", type: "string", title: "Country" },
            ],
        },
        {  name: "orderHistory",
            type: "array",
            title: "Order History",
            of: [
                {
                    type: "reference",
                    to: [{ type: "order" }],
                },
            ],
            description: "References to all orders placed by the customer.",
        },
    ],
};
```

Payment Schema

```
export const payment = {
    name: "payment",
    type: "document",
    title: "Payment",
    fields: [
        {
            name: "paymentId",
            type: "string",
            title: "Payment ID",
            validation: (Rule: any) =>
                Rule.required().error("Payment ID is required."),
        },
        {
            name: "orderId",
            type: "reference",
            title: "Order ID",
            to: [{ type: "order" }],
            validation: (Rule: any) =>
                Rule.required().error("Order ID is required."),
        },
        {
            name: "paymentMethod",
            type: "string",
            title: "Payment Method",
            options: {
                list: [
                    { title: "Credit Card", value: "credit_card" },
                    { title: "Debit Card", value: "debit_card" },
                    { title: "PayPal", value: "paypal" },
                    { title: "Cash on Delivery", value: "cash_on_delivery" },
                ],
            },
            validation: (Rule: any) =>
                Rule.required().error("Payment method is required."),
        },
        {
            name: "paymentStatus",
            type: "string",
            title: "Payment Status",
            options: {
                list: [
                    { title: "Pending", value: "pending" },
                    { title: "Completed", value: "completed" },
                    { title: "Failed", value: "failed" },
                    { title: "Refunded", value: "refunded" },
                ],
            },
            validation: (Rule: any) =>
                Rule.required().error("Payment status is required."),
        },
        {
            name: "amount",
            type: "number",
            title: "Payment Amount",
            validation: (Rule: any) =>
                Rule.required()
                    .positive()
                    .error("Payment amount must be a positive number."),
        },
    ],
};
```

Order Schema

```
export const order = {
    name: "order",
    type: "document",
    title: "Order",
    fields: [
        {
            name: "orderId",
            type: "string",
            title: "Order ID",
            validation: (Rule: any) => Rule.required().error("Order ID is required."),
        },
        {
            name: "productId",
            type: "reference",
            title: "Product ID",
            to: [{ type: "product" }],
            validation: (Rule: any) => Rule.required().error("Product ID is required."),
        },
        {
            name: "quantity",
            type: "number",
            title: "Quantity",
            validation: (Rule: any) =>
                Rule.required()
                    .integer()
                    .min(1)
                    .error("Quantity must be an integer and at least 1."),
        },
        {
            name: "totalPrice",
            type: "number",
            title: "Total Price",
            validation: (Rule: any) =>
                Rule.required()
                    .positive()
                    .error("Total price must be a positive number."),
        },
        {
            name: "status",
            type: "string",
            title: "Status",
            options: {
                list: [
                    { title: "Pending", value: "pending" },
                    { title: "Processing", value: "processing" },
                    { title: "Shipped", value: "shipped" },
                    { title: "Delivered", value: "delivered" },
                    { title: "Cancelled", value: "cancelled" },
                ],
            },
            validation: (Rule: any) =>
                Rule.required().error("Status is required."),
        },
        {
            name: "timestamp",
            type: "datetime",
            title: "Timestamp",
            validation: (Rule: any) =>
                Rule.required().error("Timestamp is required."),
        },
    ],
};
```

Delivery Schema

```
export const delivery = {
    name: "delivery",
    type: "document",
    title: "Delivery",
    fields: [
        {
            name: "zoneName",
            type: "string",
            title: "Zone Name",
            validation: (Rule: any) => Rule.required().min(3).error("Zone name must be at least 3 characters long."),
        },
        {
            name: "coverageArea",
            type: "array",
            title: "Coverage Area",
            of: [{ type: "string" }],
            description: "List of areas covered in this zone.",
            validation: (Rule: any) => Rule.required().min(1).error("At least one coverage area must be provided."),
        },
        {
            name: "assignedDriver",
            type: "object",
            title: "Assigned Driver",
            fields: [
                { name: "driverName",
                    type: "string",
                    title: "Driver Name",
                    validation: (Rule: any) =>
                        Rule.required().min(2).error("Driver name must be at least 2 characters long."),
                },
                {
                    name: "driverContact",
                    type: "string",
                    title: "Driver Contact",
                    validation: (Rule: any) =>
                        Rule.required()
                            .min(10)
                            .max(15)
                            .error("Driver contact must be between 10 and 15 characters."),
                },
                {
                    name: "vehicleDetails",
                    type: "object",
                    title: "Vehicle Details",
                    fields: [
                        { name: "vehicleType", type: "string", title: "Vehicle Type" },
                        { name: "vehicleNumber", type: "string", title: "Vehicle Number" },
                    ],
                },
            ],
        },
    ],
};
```

Shipment Schema

```
export const shipment = {
    name: "shipment",
    type: "document",
    title: "Shipment",
    fields: [
        {  name: "shipmentId",
           type: "string",
           title: "Shipment ID",
           validation: (Rule: any) => Rule.required().error("Shipment ID is required."),
        },
        {   name: "orderId",
           type: "reference",
           title: "Order ID",
           to: [{ type: "order" }],
           validation: (Rule:any) => Rule.required().error("Order ID is required."),
        },
        {  name: "trackingNumber",
           type: "string",
           title: "Tracking Number",
        },
        {  name: "deliveryAddress",
           type: "object",
           title: "Delivery Address",
           fields: [
               { name: "street", type: "string", title: "Street" },
               { name: "city", type: "string", title: "City" },
               { name: "state", type: "string", title: "State" },
               { name: "postalCode", type: "string", title: "Postal Code" },
               { name: "country", type: "string", title: "Country" },
           ],
        },
        { name: "status",
           type: "string",
           title: "Shipment Status",
           options: {
               list: [
                   { title: "Pending", value: "pending" },
                   { title: "Shipped", value: "shipped" },
                   { title: "In Transit", value: "in_transit" },
                   { title: "Delivered", value: "delivered" },
                   { title: "Canceled", value: "canceled" },
               ],
               layout: "radio", // Display as radio buttons
           },
           validation: (Rule: any) =>
               Rule.required().error("Shipment status must be specified."),
        },
        {  name: "deliveryDate",
           type: "datetime",
           title: "Delivery Date",
           validation: (Rule:any) =>
               Rule.custom((deliveryDate:any) => {
                   const currentDate = new Date();
                   if (deliveryDate && new Date(deliveryDate) < currentDate) {
                       return "Delivery date cannot be in the past.";
                   }
                   return true;
               }).error("Invalid delivery date."),
        },
    ],
};
```

Reviews Schema

```
export const review = {
    name: "review",
    type: "document",
    title: "Review",
    fields: [
        {
            name: "reviewId",
            type: "string",
            title: "Review ID",
            validation: (Rule: any) =>
                Rule.required().error("Review ID is required."),
        },
        {
            name: "productId",
            type: "reference",
            title: "Product ID",
            to: [{ type: "product" }],
            validation: (Rule: any) =>
                Rule.required().error("Product ID is required."),
        },
        {
            name: "customerId",
            type: "reference",
            title: "Customer ID",
            to: [{ type: "customer" }],
            validation: (Rule: any) =>
                Rule.required().error("Customer ID is required."),
        },
        {
            name: "rating",
            type: "number",
            title: "Rating",
            description: "Provide a rating between 1 and 5.",
            validation: (Rule: any) =>
                Rule.required()
                    .min(1)
                    .max(5)
                    .error("Rating must be between 1 and 5."),
        },
        {
            name: "comment",
            type: "text",
            title: "Comment",
            description: "Share your experience with the product.",
            validation: (Rule: any) =>
                Rule.required().min(10).error("Comment must be at least 10 characters long."),
        },
    ],
};
```