Marketplace Technical Foundation - [General E-Commerce]

- This document outlines the **technical foundation** of an e-commerce marketplace, covering **frontend (Next.js, Tailwind, TypeScript)** and **backend (Sanity CMS, Stripe, APIs)**. It details the **user workflow** from registration to order completion and defines key **API endpoints** for product fetching, order creation, and shipment tracking.

1. Define Technical Requirements

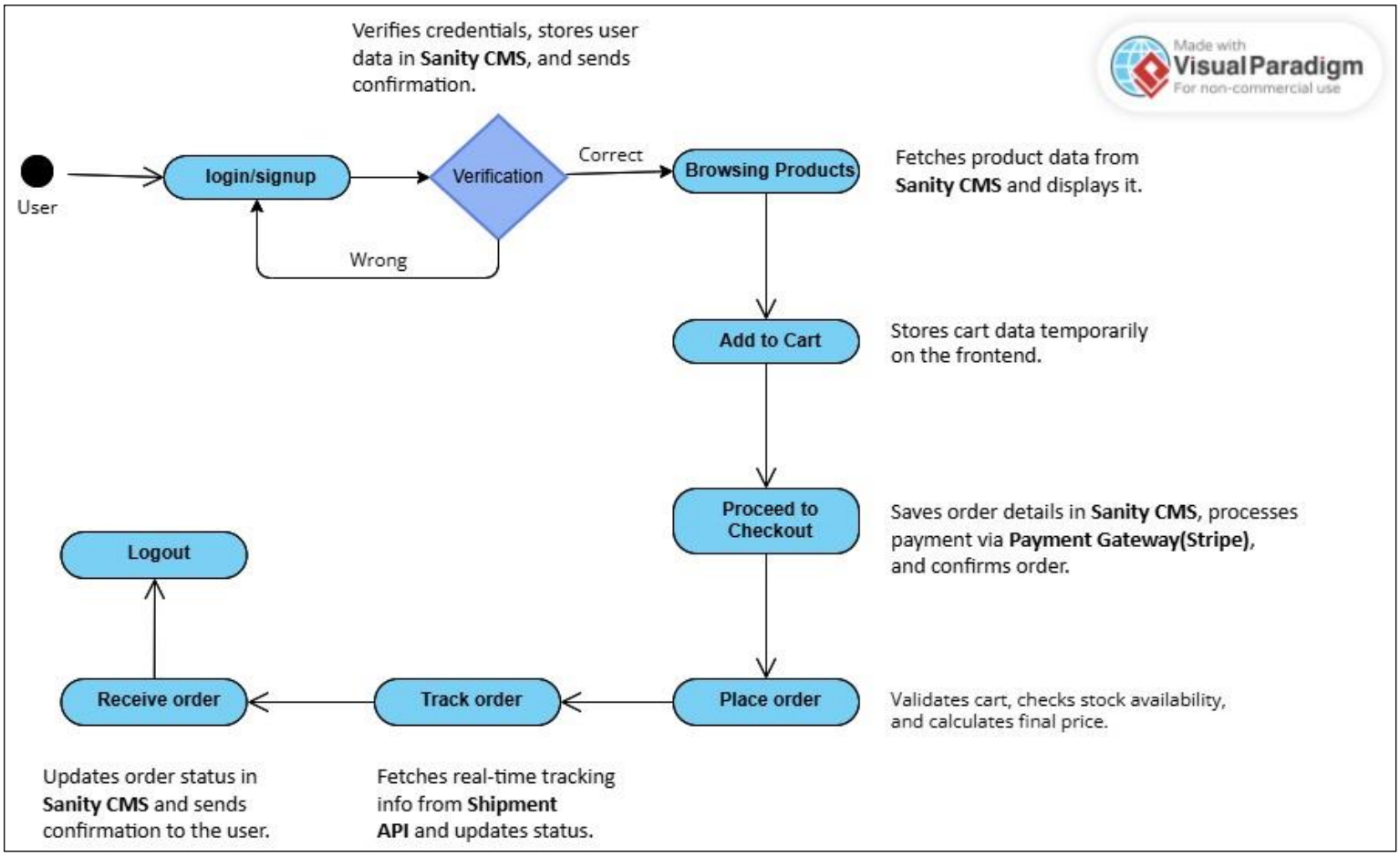
We will break down our business goals into technical requirements:

Frontend Development Requirements

- **Next.js** - Framework for building fast and dynamic React apps.
- **Tailwind CSS** - Utility-first CSS framework for styling.
- **TypeScript** - Strongly typed JavaScript for better code quality.
- **ShadCN/UI** - Pre-styled UI components for a modern look.
- **Font Awesome** - Icon library for UI elements.
- **Responsive Design** - Ensure mobile-first and cross-device compatibility.

Backend - Sanity CMS

- **Next.js API Routes** – Server-side API handling within Next.js.
- **Sanity CMS** – Headless CMS for managing products, orders, and users.
- **Third-Party APIs** – Integrations for shipping, tracking, and payments.
- **Stripe** – Payment processing integration.



User Workflow & Backend Process

1. User Registration & Login

- **User Action:** Signs up/logs in.
- **Backend:** Verifies credentials, stores user data in **Sanity CMS**, and sends confirmation.

2. Browsing Products

- **User Action:** Views product listings & details.
- **Backend:** Fetches product data from **Sanity CMS** and displays it.

3. Adding to Cart

- **User Action:** Adds products to the cart.
- **Backend:** Stores cart data temporarily on the frontend.

4. Checkout & Payment

- **User Action:** Enters shipping details & makes payment.
- **Backend:** Saves order details in **Sanity CMS**, processes payment via **Payment Gateway**, and confirms order.

5. Shipment Tracking

- **User Action:** Checks order status.
- **Backend:** Fetches real-time tracking info from **Shipment API** and updates status.

6. Order Completion

- **User Action:** Receives delivery.
- **Backend:** Updates order status in **Sanity CMS** and sends confirmation to the user. 🚀

3. Plan API Requirements

We need to define API endpoints to handle different tasks.

Example API Endpoints:

1. Get All Products

- **Endpoint:** `/products`
- **Method:** GET
- **Purpose:** Fetches all products from Sanity CMS.
- **Response Example:**

```
{ "id": 1, "name": "Product A", "price": 100, "stock": 10 }
```

2. Create a New Order

- **Endpoint:** `/orders`
- **Method:** POST
- **Purpose:** Saves a new order in Sanity CMS.
- **Payload Example:**

```
{
  "customerName": "John Doe",
  "products": [{ "id": 1, "name": "Product A", "quantity": 2 }],
  "paymentStatus": "Paid"
}
```

3. Track Shipment

- **Endpoint:** `/shipment`
- **Method:** GET
- **Purpose:** Fetches shipment status via a third-party API.
- **Response Example:**

```
{ "orderId": 123, "status": "In Transit", "ETA": "2 days" }
```

Created By Anas Khan