Technical Implementation Plan

Creator: Ahsan Paracha Slot: Sunday 2-5 Roll No: 0088100
Premium Furniture E-Commerce Platform

Table Of Content

1. Frontend Architecture

Framework: Next.js 14 App Router with TypeScript
UI Library: Tailwind CSS

Core Pages:

- `app/account/page.tsx`: Clerk-authenticated profile management
- `app/contact/page.tsx`: Premium contact form
- `app/cart/page.tsx`: Persistent cart with local storage fallback
- `app/blog/page.tsx`: Inspiration content
- `app/checkout/page.tsx`: Multi-step checkout with Stripe integration
- `app/products/[productId]/page.tsx`: Product showcases

2. Backend Overview

1. **Customer Signs Up**
    - **Action: The user creates an account via Clerk.**
    - **Data Flow:**
        - **Clerk → Sanity CMS stores: `Customer ID`, `Name`, `Contact Info`.**
2. **Browsing Premium Furniture**
    - **Action: Users view curated products.**
    - **Data Flow:**
        - **Sanity CMS → Displays: `Product Images`, `Names`, `Prices`.**
3. **Adding to Cart**
    - **Action: The user selects a designer sofa.**
    - **Data Flow:**
        - **Website → Checks sanity for `Product Stock` and reserves if available.**
4. **Checkout Process**
    - **Action: The user completes the purchase.**
    - **Data Flow:**

        - Payment: Stripe → Saves: `Payment Method`, `Amount`, `Date` in Sanity.
        - Order: Creates a Sanity Order Record with `Customer ID`, `Product ID`, `Quantity`.
5. Shipping Preparation
    - Action: The warehouse team packs the order.
    - Data Flow:
        - Sanity → Informs team with `Delivery Zone`, `Customer Address`.
        - Updates Sanity with `Shipment ID` and `Status: Preparing`.
6. Delivery Tracking
    - Action: The customer tracks their package.
    - Data Flow:
        - Shipping Carrier (FedEx/UPS) → Updates Sanity with “Current Location”.
        - Display to customer: "Out for delivery to [Zone Name]".

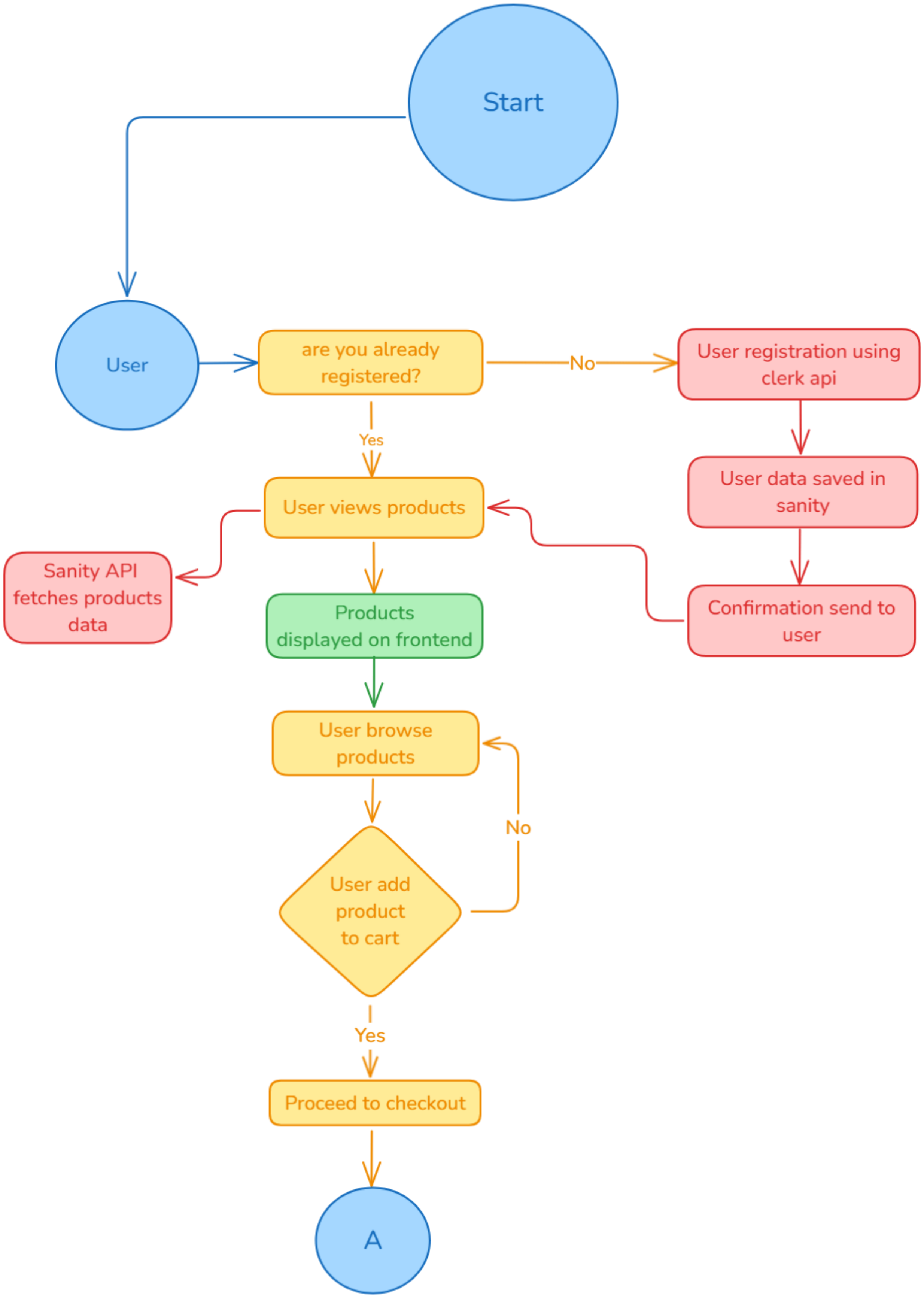
Key Relationships:

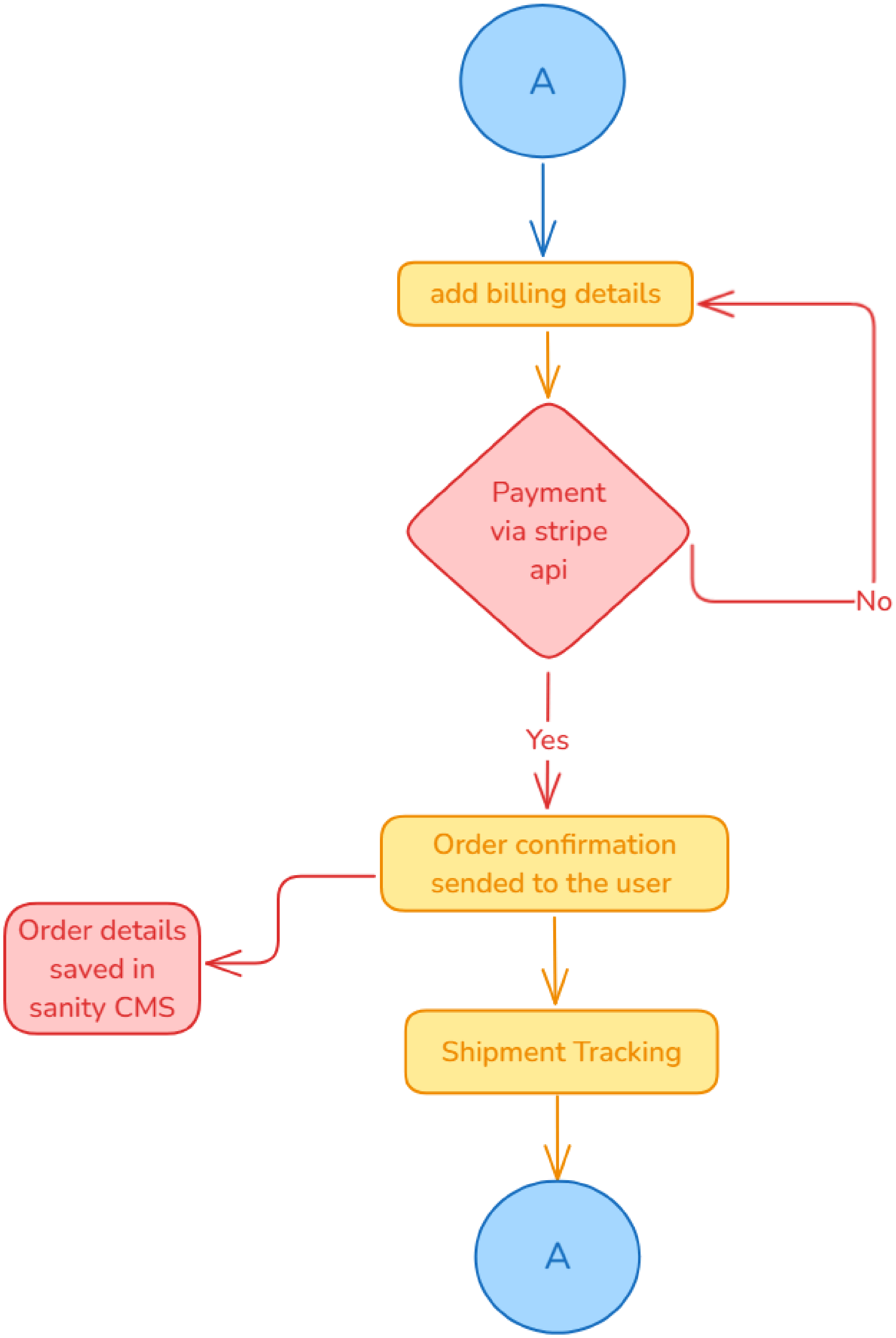
- Each order links to one Customer, one or more Products, one Payment, and one Shipment.
- Delivery Zones determine shipping costs and timelines.

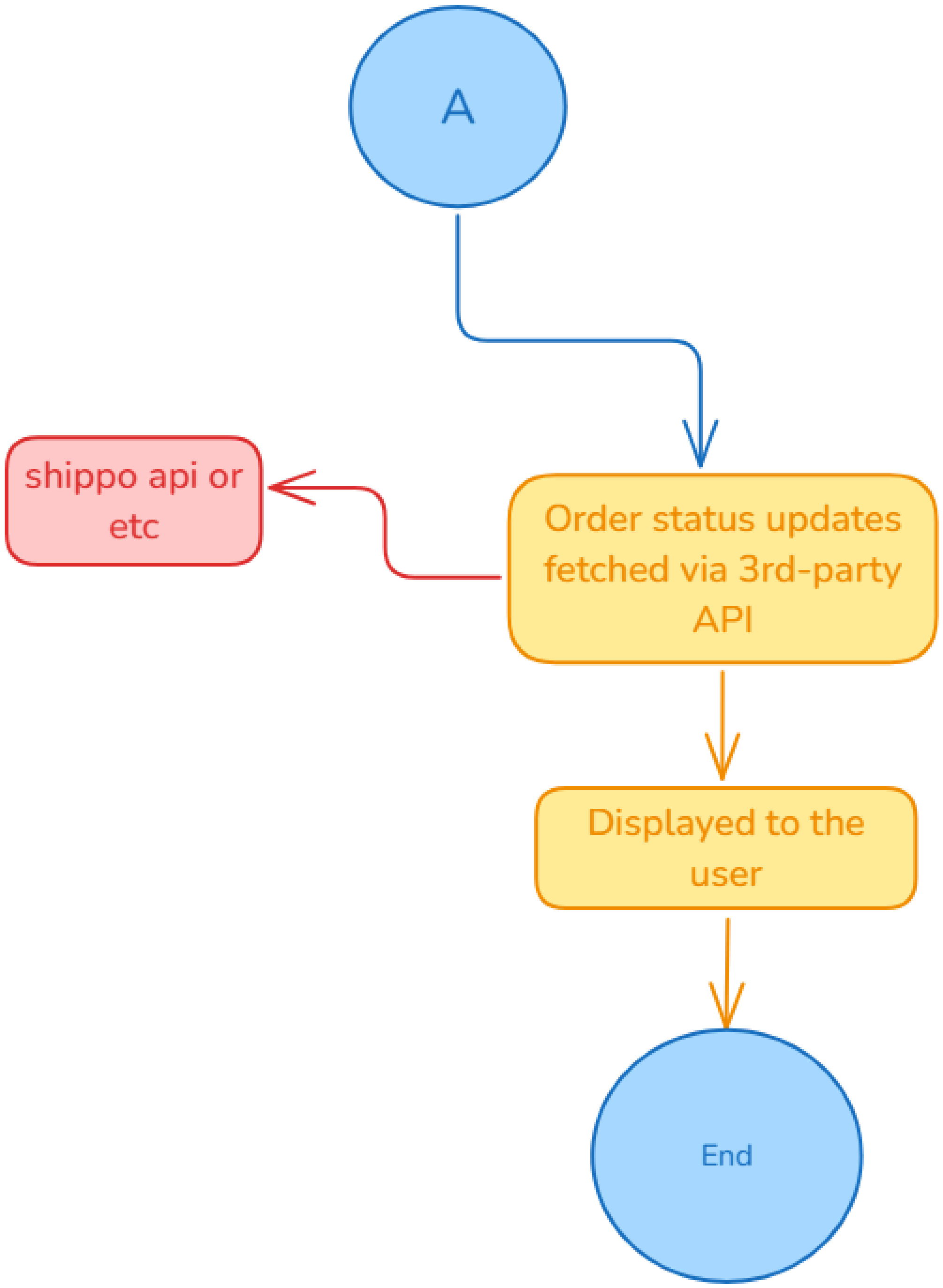
Benefits for Busy People:

- Sanity CMS centralizes all details.
- Automatic tracking reduces customer service workload.

System Architecture Overview


Start
User
are you already registered?
No
User registration using clerk api
User data saved in sanity
Confirmation send to user
Yes
User views products
Sanity API fetches products data
Products displayed on frontend
User browse products
User add product to cart
No
Yes
Proceed to checkout
A


A
add billing details
Payment via stripe api
No
Yes
Order confirmation sended to the user
Order details saved in sanity CMS
Shipment Tracking
A



API Endpoints

Endpoint	Method	Purpose	Response Example
/products	GET	Fetch products	`{ "id": "prod_001", "name": "Velvet Luxe Sofa", "price": 2499.99 }`
/orders	POST	Create a new order	`{ "orderId": "ord_456", "total": 2749.99 }`
/shipments	GET	Track shipment status	`{ "orderId": "ord_456", "status": "in_transit" }`
/customers	GET	Retrieve customer profile	`{ "id": "cust_123", "name": "Sarah Johnson" }`
/payments	POST	Process a payment	`{ "paymentId": "pay_789", "status": "completed" }`

Sanity Schema

```
export const product = {

  // Document configuration

  name: "product",

  type: "document",

  title: "Product",
// Fields definition
fields: [
  // 1. Product Image
  {
    name: "image",
    title: "Product Image",
    type: "image",
  },
  // 2. Product Title
  {
    name: "name",
    title: "Product Title",
    type: "string",
  },
  // 3. Product Price
  {
    name: "price",
    title: "Product Price",
    type: "number",
  },
  // 4. Stripe Price ID
  {
      name: "price_id",
      title: "Stripe Price ID",
      type: "string",
    },
    // 5. Product Description
    {
      name: "description",
      title: "Product Description",
      type: "text",
    },
  ],
};
```

Development Phase

1. Authentication

- User Registration & Login:
  Implement user registration and login functionalities using Clerk.
- Clerk Integration with Sanity CMS:
  Integrate Clerk with Sanity CMS to store and manage user data efficiently.

2. Product Management

- **Mock API:**
  Develop a mock API to handle product data management.

- **Store Products in Sanity CMS:**
  Save product details—such as name, price, and description—in Sanity CMS for centralized management.

- **Display Products:**
  Dynamically fetch and display product data on frontend pages.

3. Cart and Wishlist

- **Add-to-Cart Functionality:**

  - **Implement features that allow users to add products to their cart.**
  - **Utilize Redux to manage cart state, supporting multiple product additions and removals.**

- **Cart Summary:**
  Display a comprehensive cart summary including the total bill and a "Proceed to Checkout" button.

4. Payment Integration

- **Stripe Integration:**
  Integrate Stripe to enable secure payment processing.

- **Test Account:**
  Utilize a Stripe test account during development to simulate payment transactions.

- **Payment Handling:**
  Manage both successful and failed payment scenarios, providing appropriate feedback to users.

5. Shipment Tracking

- **ShipEngine Integration:**
  Connect the ShipEngine API to facilitate real-time shipment tracking.

- **Tracking Numbers:**
  Generate and display tracking numbers for each order.

- **Order Tracking:**
  Allow users to monitor the status of their shipments in real time.

Conclusion

This technical foundation outlines the architecture, workflows, and API endpoints for our eCommerce platform. The platform is designed to deliver a seamless online shopping experience by incorporating:

- **Robust Authentication:**
  Secure user registration and login through Clerk.

- **Real-Time Shipment Tracking:**
  Integration with ShipEngine to provide users with current shipment statuses.

- **Smooth Shopping Experience:**
  A comprehensive suite of features including cart and wishlist management, secure payment processing via Stripe, and a streamlined checkout process.