Marketplace Technical Foundation - LuxeWalk

1. Introduction

1.1 Project Overview

LuxeWalk is a general e-commerce marketplace designed to cater to a diverse audience aged 18-50. It offers products for all genders, ensuring a seamless shopping experience. This document outlines the technical foundation for LuxeWalk, providing a blueprint for implementation and aligning with the business goals.

1.2 Purpose of the Document

This document serves as a guide for the technical development of LuxeWalk, detailing system architecture, workflows, API specifications, and data schemas. It ensures scalability, user-friendliness, and alignment with the marketplace's objectives.

---

2. System Architecture

2.1 High-Level Architecture

- **Frontend**: Built with Next.js and Tailwind for a responsive and dynamic user interface.
- **Authentication**: Integrated with Clerk for secure user authentication.
- **Backend**: Sanity CMS is used for content management and data handling.
- **APIs**: Integrated with third-party services for payments (Stripe) and shipping (ShipEngine).

2.2 Architecture Diagram

Example Workflow:

1. Users log in or sign up by entering their credentials.
2. Users browse products on the frontend.
3. Product data is fetched from Sanity CMS.
4. Orders are recorded in Sanity CMS.
5. Payment and shipment updates are handled via third-party APIs.

---

3. Technical Requirements

3.1 Frontend Requirements

- User-friendly interface with pages:
  - Home
  - Category
  - Product Details (dynamically generated using slugs)
  - Cart
  - Order Tracking
- Responsive design for mobile and desktop users.

3.2 Backend Requirements

- Sanity CMS for managing:
  - Product data
  - Customer records
  - Order details

3.3 API Integrations

- Payment Gateways: Stripe for secure transactions.
- Shipping APIs: ShipEngine for order shipping, tracking, and rate management.

---

4. API Specifications

Endpoint	Method	Description	Response Example
`/products`	GET/POST	Fetch all products or update product data like reviews, stock, etc.	`{ "_key": "8773ad04693ea74fcca6283c35dc9d4e", "name": "T-SHIRT WITH TAPE DETAILS", "price": 120, ... }`
`/orders`	POST/GET	Create a new order or fetch order history.	`[ { "OrderId": "ndiunsiu", "customer": { "CustomerId": "bn8899ndn888ns", "Name": "Hammad", ... } } ]`
`/shipment`	GET	Fetch shipment details.	`{ "orderId": 123, "status": "In Transit" }`

---

5. Workflows

5.1 User Registration

1. User signs up via the frontend using Clerk.
2. Data is stored in Sanity CMS.
3. A confirmation is sent to the user.

5.2 Product Browsing

1. Users view product categories.
2. Product data is fetched from Sanity CMS.
3. Products are dynamically displayed on the frontend.

5.3 Order Placement

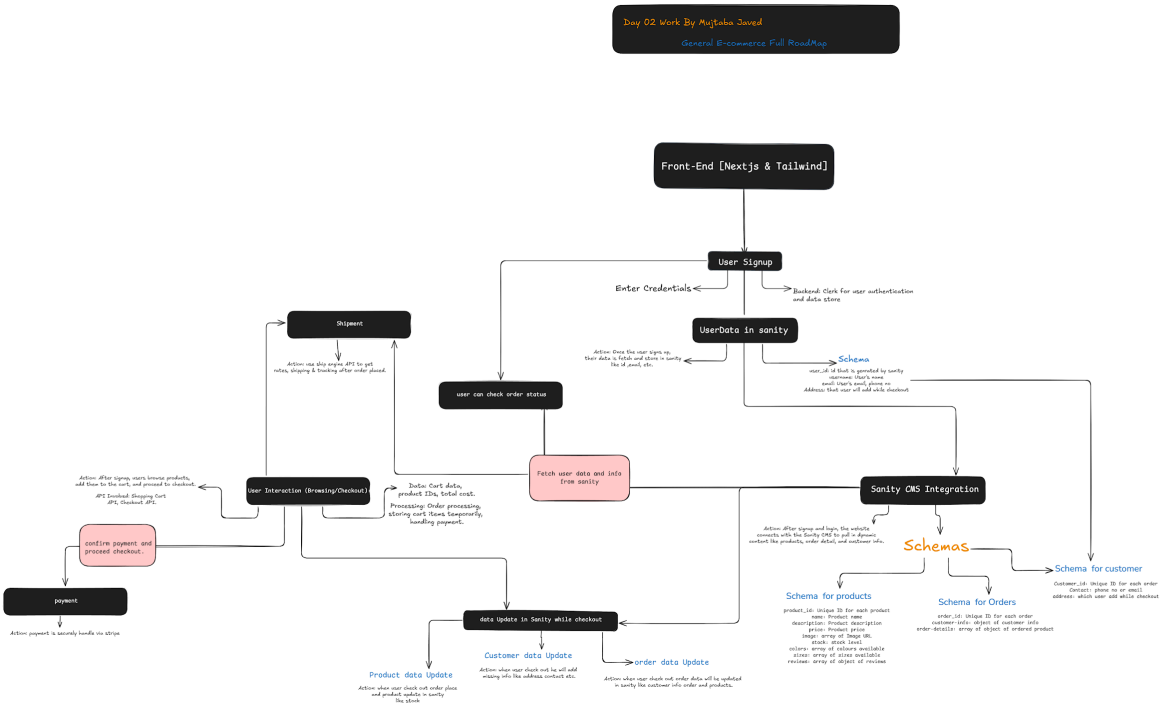
1. Users add items to the cart and proceed to checkout.
2. Order details are saved in Sanity CMS.
3. Payment is processed through Stripe, and a confirmation is sent to the user.

5.4 Shipment Tracking

1. Shipment status is fetched via ShipEngine APIs.
2. Updates are displayed to the user in real-time.

---

6. Data Schema Design



6.1 Sanity CMS Schemas

Product Schema

```
import { Rule } from 'sanity';
import { createClient } from '@sanity/client';

// Create Sanity client
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: 'production',
  useCdn: false,
  apiVersion: '2021-08-31',
  token: process.env.SANITY_API_TOKEN,
});

export default {
  name: 'product',
  title: 'Product',
  type: 'document',
  fields: [
    {
      name: 'name',
      title: 'Product Name',
      type: 'string',
    },
    {
      name: 'slug',
      title: 'Slug',
      type: 'slug',
      options: {
        source: (doc: { name: string; _id: string }) => `${doc.name}-${doc._id}`,
        maxLength: 96,
        slugify: (input: string) => input
          .toLowerCase()
          .replace(/\s+/g, '-')
          .replace(/[^\w\-]+/g, '')
          .slice(0, 96),
      },
    },
    {
      name: 'description',
      title: 'Description',
      type: 'text',
    },
    {
      name: 'price',
      title: 'Price',
```

```
  type: 'number',
},
{
  name: 'priceWithoutDiscount',
  title: 'Price Without Discount',
  type: 'number',
},
{
  name: 'discountPercentage',
  title: 'Discount Percentage',
  type: 'number',
},
{
  name: 'rating',
  title: 'Rating',
  type: 'number',
},
{
  name: 'stockLevel',
  title: 'Stock Level',
  type: 'number',
},
{
  name: 'tags',
  title: 'Tags',
  type: 'array',
  of: [{ type: 'string' }],
},
{
  name: 'sizes',
  title: 'Available Sizes',
  type: 'array',
  of: [{ type: 'string' }],
},
{
  name: 'colors',
  title: 'Available Colors',
  type: 'array',
  of: [{ type: 'string' }],
},
{
  name: 'images',
  title: 'Product Images',
  type: 'array',
  of: [
    {
      type: 'image',
      options: {
        hotspot: true,
```

```
        },
      },
    ],
  },
  {
    name: 'reviews',
    title: 'Customer Reviews',
    type: 'array',
    of: [
      {
        type: 'object',
        fields: [
          {
            name: 'id',
            title: 'ID',
            type: 'number',
          },
          {
            name: 'name',
            title: 'Reviewer Name',
            type: 'string',
          },
          {
            name: 'review',
            title: 'Review',
            type: 'text',
          },
          {
            name: 'rating',
            title: 'Rating',
            type: 'number',
          },
          {
            name: 'date',
            title: 'Review Date',
            type: 'date',
          },
        ],
      },
    ],
  },
],


  };
```

Order Schema

```
export default {
  name: 'order',
  type: 'document',
  fields: [
    { name: 'orderId', type: 'string', title: 'Order ID' },
    { name: 'customerId', type: 'string', title: 'Customer ID' },
    { name: 'products', type: 'array', of: [{ type: 'object', to: [{ type: 'product' }] }], title: 'Products' },
    { name: 'status', type: 'string', title: 'Order Status' },
    { name: 'createdAt', type: 'datetime', title: 'Order Created At' },
  ]



    };
```

Customer Schema

```
export default {
  name: 'order',
  type: 'document',
  fields: [
    { name: 'Customer_id', type: 'string', title: 'Customer_id' },
    { name: 'user_name', type: 'string', title: 'user name' },
    { name: 'email', type: 'string', title: 'email' },
    { name: 'Contact', type: 'string', title: 'Contact' },
    { name: 'address', type: 'string', title: 'Address' },
  ]



    };
```

---

8. Technical Roadmap

Phase 1: Frontend Development

- Create UI components with Next.js and Tailwind.
- Implement responsive design.

Phase 2: Backend Setup

- Configure Sanity CMS for managing data.
- Define schemas for products, customers, and orders.

Phase 3: API Integrations

- Integrate Stripe for payments.
- Integrate ShipEngine for shipping and tracking.

Phase 4: Testing and Deployment

- Conduct end-to-end testing.
- Deploy the application on a scalable hosting platform.