Shoes Store Website - Technical Report

1. Frontend

Pages

- Home Page: Displays featured products and categories.
- All Products Page: Lists all available products fetched from Sanity CMS.
- Single Product Page: Shows detailed product information based on the product ID from Sanity CMS.
- Cart Page: Allows users to view and manage their selected products.
- User Authentication Pages: Login and Register functionalities.

Design

- Responsive design optimized for mobile and desktop.
- User-friendly navigation with a clean UI.
- Dynamic rendering using Next.js for fast performance.

2. Sanity CMS (Backend)

Purpose

- Acts as a content management system to store and manage product data.
- Provides an API to retrieve products dynamically.
- Manages user-generated content, such as comments from trusted users.

3. Third-Party API

- Payment Gateway API: Handles secure payment transactions.
- Shipping API: Provides real-time tracking for orders.

4. System Architecture

Flow Diagram

1. User navigates to the website.
2. Next.js fetches product data from Sanity CMS.
3. User selects a product and is redirected to the product details page.
4. The product ID is used to fetch detailed information from Sanity.

5. User adds the product to the cart.

6. Checkout process initiates order creation.

7. Payment gateway processes the transaction.

8. Order details are stored in Sanity CMS.

9. Shipping API provides tracking updates.

Workflow Example

- User Browsing: Fetches products from Sanity CMS and displays them dynamically.
- Adding to Cart: Stores selected items in session storage or state management.
- Order Processing: Sends order data to Sanity CMS and payment API.
- Tracking Orders: Fetches order status using third-party shipping API.

5. API Requirements

Fetch All Products

- Endpoint: /api/products
- Method: GET
- Response: Returns a list of all products with details (name, price, image, stock availability).

Create Order

- API Reference: Template-03 API (https://template-03-api.vercel.app/api/products)
- Endpoint: /api/orders
- Method: POST
- Request Body:

```
{
  "userId": "12345",
  "products": [{ "productId": "abc123", "quantity": 2 }],
  "totalPrice": 100,
  "status": "pending"
}
```

- Response:

```
{
  "orderId": "67890",
  "status": "confirmed"
}
```