Hackathon Day:02

Marketplace Technical Foundation

Introduction

This documentation outlines the technical details of the e-commerce furniture website. It covers the architecture, APIs, databases, and other technical aspects of the application.

Architecture

The application follows a microservices architecture, with separate services for products, orders, inventory, payment, and shipping. Each service communicates with the others through RESTful APIs.

Services

- _Product Service_: Responsible for managing product information, including product details, pricing, and inventory levels.

- _Order Service_: Handles order processing, including payment and shipping.

- _Inventory Service_: Tracks and manages product inventory levels.

- _Payment Service_: Processes payments for orders.

- _Shipping Service_: Provides shipping information for orders.

APIs

The application exposes RESTful APIs for interacting with the services. The APIs follow standard HTTP methods (GET, POST, PUT, DELETE) and return JSON responses.

API Endpoints

- _Product API_:

  - `GET /products`: Fetches all products

  - `GET /products/{id}`: Fetches a product by ID

  - `POST /products`: Creates a new product
  - `PUT /products/{id}`: Updates a product
  - `DELETE /products/{id}`: Deletes a product
- _Order API_:
  - `POST /orders`: Creates a new order
  - `GET /orders`: Fetches all orders
  - `GET /orders/{id}`: Fetches an order by ID
  - `PUT /orders/{id}`: Updates an order
  - `DELETE /orders/{id}`: Deletes an order

- _Inventory API_:
  - `GET /inventory`: Fetches all inventory items
  - `GET /inventory/{id}`: Fetches an inventory item by ID
  - `POST /inventory`: Creates a new inventory item
  - `PUT /inventory/{id}`: Updates an inventory item
  - `DELETE /inventory/{id}`: Deletes an inventory item

- _Payment API_:
  - `POST /payments`: Processes a payment
  - `GET /payments`: Fetches all payments
  - `GET /payments/{id}`: Fetches a payment by ID

- _Shipping API_:
  - `POST /shipments`: Creates a new shipment
  - `GET /shipments`: Fetches all shipments
  - `GET /shipments/{id}`: Fetches a shipment by ID

Data Fetching

Data will be fetched from Sanity CMS. Sanity CMS will store products, orders, inventory, and other data.

Database

The application uses a relational database management system (RDBMS) to store data. The database schema includes tables for products, orders, inventory, payments, and shipments.

Security

The application implements standard security measures to protect user data and prevent unauthorized access.

Authentication

The application uses OAuth 2.0 for authentication. Users can log in using their email and password.

🔹 Here's a finalized plan for implementing the e-commerce furniture website:

Implementation Plan

Frontend

1. Set up Next.js project with necessary dependencies.
2. Design and implement UI components for products, orders, and inventory.
3. Implement API calls to fetch data from Sanity CMS.
4. Set up payment gateway integration (e.g., Stripe).
5. Implement shipping provider integration (e.g., UPS).

Backend

1. Set up Sanity CMS project with necessary schemas.
2. Implement API endpoints for products, orders, and inventory.
3. Set up authentication and authorization using OAuth 2.0 and RBAC.
4. Implement data encryption using SSL/TLS.
5. Set up password hashing using bcrypt.

Testing

1. Write unit tests for frontend and backend components.
2. Write integration tests for API endpoints.
3. Perform end-to-end testing for user workflows.
4. Test payment gateway and shipping provider integrations.

Launch

1. Launch website and make it available to public.
2. Configure analytics and tracking tools.
3. Monitor website performance and fix any issues.
4. Continuously iterate and improve website based on user feedback.

Diagram

Date:

MARKETPLACE TECHNICAL FOUNDATION E-COMMIERCE WEBSITE