SOFTWARE DESIGN DOCUMENT(SDD)

Introduction

Purpose: This document provides a comprehensive architectural overview of the e-commerce system, outlining the software design and specifications required to meet the project's goals and objectives.

Scope: This document covers the design considerations, system architecture, component design, interface design, and data management for the e-commerce platform.

System Overview

System Architecture: The application is structured into four backend layers:

Controller Layer: Manages the receiving and responding of HTTP requests, and delegates operations to the Service Layer.

Service Layer: Contains business logic and communicates between the Controller and DAO layers.

DAO (Data Access Object) Layer: Responsible for direct database access and operations.

Database Layer: Manages data storage and retrieval.

Frontend Template: Utilizes responsive web design templates that are adaptable to various devices and screen sizes.

Technology Stack:

Backend: Spring Boot, used for creating microservices.

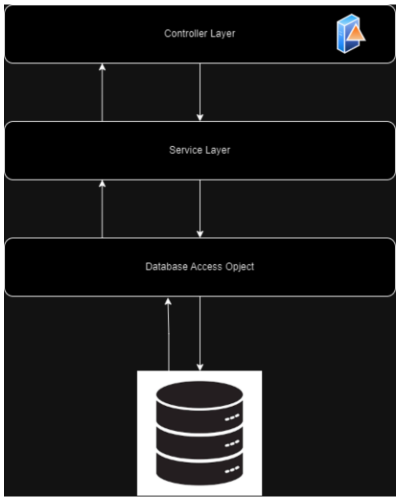
Database: (Specify the database, e.g., MySQL, MongoDB)

Cloud: Docker for containerization of the application components.

Frontend: (Specify frontend technologies, e.g., Angular, React)

System Architecture and Components Design

High-Level Architecture: Provide a detailed architecture diagram here showing the relationships between the Controller, Service, DAO layers, and the Database. Also, include how these interact with the frontend components.



Data Design

Detailed descriptions and structures for major entities in the system:

Category:

id: Unique identifier for the category.

name: Descriptive name of the category.

description: A brief description of what types of products fall under this category.

Product:

id: Unique identifier for the product.

name: Name of the product.

image: URL or path to the product's image.

price: Retail price of the product.

stock: Quantity of the product available.

Category Id: Foreign key linking to the Category table.

Admin:
id: Unique identifier for the admin.

name: Admin's full name.

image: URL or path to the admin's image.

email: Admin's email address.

Customer:
id: Unique identifier for the customer.

name: Customer's full name.

age: Customer's age.

image: URL or path to the customer's image.

email: Customer's email address.

Interface Design

API Endpoints:

Categories:
GET /categories - Retrieve all categories.

POST /categories - Create a new category.

PUT /categories/{id} - Update an existing category.

DELETE /categories/{id} - Delete a category.

Products:
GET /products - Retrieve all products.

GET /products/{id} - Retrieve a product by ID.

POST /products - Create a new product.

PUT /products/{id} - Update an existing product.

DELETE /products/{id} - Delete a product.

Admins and Customers:
User authentication and management endpoints.

Security Design

Authentication and Authorization: Implement JWT (JSON Web Tokens) for secure authentication and authorization of users.

Data Protection: Use HTTPS for secure data transmission. Encrypt sensitive data in the database.

User Interface Design

Mockups and Storyboards: Include detailed mockups of the user interface for both desktop and mobile views. Provide storyboards showing user interaction and flow through the application.

Navigation Paths: Describe typical user journeys, such as searching for a product, adding items to a cart, and checking out.

Appendices

Detailed Algorithms: Provide any complex business logic or algorithms used within the service layer.

Test Cases: Outline typical test cases for unit and integration testing of the application.

Version History

Document Revisions: Maintain a log of all changes, dates, and authors involved in the revisions.