**SHELBY (E-commerce Web Application)**

**LOW LEVEL DESIGN DOCUMENT**

**1.INTRODUCTION :**

Shelby is an application which is a web-based micro e-commerce platform, aims to provide a simplified and user-friendly interface for buying and selling niche products and services. Buyers can browse, search for items, read product reviews and make online shopping ease. Sellers can add their product and develop a business effectively.

**1.1 PURPOSE :**

This low-level design document provides an overview of the design on the component level. Input is always the High-level design(HLD) in case of LLD for SHELBY e-commerce platform. Basically, it is component level process that follows a step by step refinement process.

**1.2 SCOPE :**

Low-Level Design describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer directly code the program from the document.

**2. LOW LEVEL DESIGN :**

* Low-level design refers to the process of defining the detailed, functional design of a software system or component.
* It involves specifying the individual modules, data structures, algorithms, interfaces, and inputs/outputs of a system.
* The purpose of low-level design is to provide a clear and precise description of how the system should behave and how its different components will interact with each other.

**2.1 ENTITY RELATIONSHIP DIAGRAM:**

The ERD illustrates the structure of the application's classes and their relationships. It includes classes and essential entities such as user,admin,seller, product, order, cart, shipment and payment .

**Relationships:**

* Admins can have privileges to manage sellers, buyers, and products.
* Sellers can list multiple products for sale.
* Buyers can place multiple orders.

A diagram of a computer program

Description automatically generated

Based on these entities and attributes, we can now define the relationships between them:

1.**Seller-Product**: One-to-Many (Any Product can have only one Seller and One Seller can add multiple Products)

2. **User-Product:** One-to-Many (any user can have multiple order)

3. **Product-Cart**: Many-to-One (Multiple Products can be added into a single Cart)

4. **Cart-Order**: One-to-One (Each Cart can be confirmed into a single Order)

5.**Seller-Order** : Many-to-Many(Each seller can have multiple orders and each order can contain multiple sellers)

6.**Order-Payment**: One-to-One(One Order can have a single Payment)

7.**User-Payment**: One-to-many(One user can have multiple payments)

8**.Admin-User** : One-to-Many(Each Admin can have access to multiple Users Information)

9.**Admin-Product** :One-to-Many(Admin can have access to multiple Products Information)

10.**Admin-Seller**: One-to-Many(Admin can have access to multiple Sellers Information)

**3.SEQUENCE DIAGRAMS :**

**3.1 USER REGISTRATION :**

The diagram below shows all the processes required for the login or register feature. The user has to enter the URL of the website and click login/register. The user is then prompted to enter the fields required for registration or to log-in.

**A diagram of a software project

Description automatically generated**

**3.2PRODUCT SEARCHING :**

This sequence diagram illustrates the high-level flow of events involved in the product searching process in an e-commerce application, focusing on the interactions between the user, frontend, backend, and TABLE components.

**A screenshot of a search screen

Description automatically generated**

**3.3 PAYMENT PROCESS :**

The payment process in an e-commerce application involves the series of steps taken to securely and efficiently facilitate transactions between buyers and sellers. It begins with the user adding items to their cart and concludes with the successful transfer of funds from the buyer's account to the seller's account, typically mediated by a payment gateway.

Key steps in the payment process include validation of payment information, authorization of the transaction, processing of the payment, order confirmation, and settlement of funds. Throughout the process, security measures are implemented to protect sensitive information, and clear communication ensures transparency and trust between all parties involved.

**A diagram of a software flow

Description automatically generated**

**4.DATABASE SCHEMA :**

Below is a basic outline for the TABLE schema of the Micro E-commerce Application catering to both sellers and buyers using MySQL TABLE.

**1.USER TABLE :**

* userId INT PRIMARY KEY AUTO\_INCREMENT,
* username VARCHAR(50) NOT NULL,
* email VARCHAR(100) NOT NULL,
* password VARCHAR(100) NOT NULL,
* address VARCHAR(255),
* phone VARCHAR(20)

**2**.**SELLER TABLE :**

* sellerId INT PRIMARY KEY AUTO\_INCREMENT,
* name VARCHAR(255) NOT NULL,
* email VARCHAR(255) NOT NULL,
* password VARCHAR(255) NOT NULL,
* address VARCHAR(255),
* phone VARCHAR(20)

**3.ADMIN TABLE:**

* adminID INT PRIMARY KEY AUTO\_INCREMENT,
* name VARCHAR(50) NOT NULL,
* password VARCHAR(100) NOT NULL,
* email VARCHAR(100) NOT NULL,
* phone VARCHAR(20)

**4.PRODUCT TABLE :**

* productId INT PRIMARY KEY AUTO\_INCREMENT,
* p\_name VARCHAR(100) NOT NULL,
* description TEXT,
* p\_image BLOB,
* price DECIMAL(10, 2) NOT NULL,
* quantity INT NOT NULL,
* sellerId INT NOT NULL,
* FOREIGN KEY (sellerId) REFERENCES User(userId)\

**5.CART TABLE :**

* productId INT,
* p\_name VARHCHAR,
* p\_image BLOB,
* p\_description TEXT,
* price INT,
* quantity INT NOT NULL,
* PRIMARY KEY( productId),
* FOREIGN KEY (productId) REFERENCES Product(productId)

**6.ORDER TABLE :**

* orderId INT PRIMARY KEY AUTO\_INCREMENT,
* userId INT NOT NULL,
* product\_id,
* p\_name,
* p\_image BLOB,
* Amount DECIMAL(10, 2) NOT NULL,
* Quantity INT NOT NULL,
* orderDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,
* FOREIGN KEY (,product\_id,userId) REFERENCES User(userId) and Product(ProductId)

**7.PAYMENT TABLE :**

* paymentID INT PRIMARY KEY AUTO\_INCREMENT,
* orderID INT NOT NULL,
* Total\_amount DECIMAL(10, 2) NOT NULL,
* paymentDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,
* paymentMethod VARCHAR(50) NOT NULL,
* FOREIGN KEY (orderID) REFERENCES `Order`(orderID)

**8.SHIPMENT TABLE :**

* shipment\_id INT PRIMARY KEY AUTO\_INCREMENT,
* shipment\_dateTIMESTAMP DEFAULT CURRENT\_TIMESTAMP,
* address VARCHAR(255),
* user\_id INT,
* FOREIGN KEY (user\_id,address) REFERENCES `User`(user\_id,address)

**9.CATEGORY TABLE :**

* Category\_id INT PRIMARY KEY AUTO\_INCREMENT,
* Category\_name VARCHAR(25)

**5.COMPONENT DESIGN :**

**Frontend UI Components**

* User Interface: Developed using HTML, CSS, and JavaScript framework like Angular.
* Components: Modular components for user authentication, product listing, shopping cart, payment and checkout process.
* UI Design: Intuitive and user-friendly design with responsive layouts for various devices.

**Backend API Components**

* Routing Layer: Handles incoming HTTP requests and routes them to the appropriate controllers.
* Controllers: Contains business logic for user authentication, product management, order processing, and payment integration.
* Middleware: Implements middleware functions for request parsing, authentication, error handling, and logging.
* Data Access Layer: Interacts with the TABLE to perform CRUD operations on user data, product data, and order data.

**6. SECURITY CONSIDERATION**

* Encryption: Utilizes encryption algorithms to secure sensitive data such as user passwords and payment information.
* Authorization: Enforces role-based access control (RBAC) to restrict user access to authorized functionalities.
* Input Validation: Validates user input to prevent injection attacks and other security vulnerabilities.
* HTTPS: Ensures secure communication between the client and server using HTTPS protocol.

**7. ERROR HANDLING**

* Exception Handling: Implements robust error handling mechanisms to gracefully handle exceptions and errors.
* Error Logging: Logs errors and exceptions to facilitate troubleshooting and debugging.
* User Feedback: Provides informative error messages to users in case of invalid input or unexpected errors.

**8. PERFORMANCE OPTIMIZATION**

* Asynchronous Processing: Utilizes asynchronous processing techniques to offload time-consuming tasks and improve system responsiveness.
* TABLE Optimization: Optimizes TABLE queries and indexing strategies to minimize query execution time and improve overall TABLE performance.

**9. DEPLOYMENT ARCHITECTURE**

* Cloud Deployment: Hosts the application on cloud platforms such as AWS, Azure, or Google Cloud for scalability and reliability.
* Containerization: Uses containerization technologies like Docker for packaging and deploying the application components.
* Load Balancing: Implements load balancing mechanisms to distribute incoming traffic across multiple servers and improve scalability and fault tolerance.

This Low-Level Design (LLD) document provides a detailed overview of the e-commerce web application's design and implementation considerations. It covers various aspects such as class diagrams, sequence diagrams, TABLE schema, component design, security considerations, error handling, performance optimization, and deployment architecture.