**Design Document for Online Delivery Application**

**1. Overview**

**Project Title:** Online Delivery Web-Application

**Authors:** Arkaprabha De

**Date:** 12.01.2025

**Version:** 1.0

**Abstract:**

This project deals with the design and implementation of an online delivery application that will enable users to order products from different vendors, track their orders, and manage deliveries efficiently. The system will be designed for four user roles: Admin, Vendor, Delivery Personnel, and Customer. The features to be provided include product management, order tracking, delivery updates, payment integration, and scalable architecture.

**Scope:**

The web-application aims to simplify and automate the process of managing online orders and deliveries while ensuring scalability, security, and ease of use for all stakeholders.

**2. Objectives**

* Enable users to order products seamlessly.
* Provide a secure and role-based authentication system.
* Offer real-time order and delivery tracking.
* Ensure scalability and responsiveness for high user traffic.

**3. Functional Requirements**

**User Roles and Authentication:**

* Role-based access for Admin, Vendor, Delivery Personnel, and Customer.
* Secure login and registration using encryption (e.g., BCrypt for passwords).
* Email verification during registration.

**Product Management:**

* Vendors can add, update, and delete products.
* Products categorized into categories and subcategories.
* Attributes include name, description, price, and availability status.

**Order Management:**

* Customers can browse, add to cart, and place orders.
* Order statuses: Order Placed, Processing, Shipped, Out for Delivery, Delivered, Cancelled.
* Order history view for customers.

**Delivery Management:**

* Assign orders to delivery personnel.
* Real-time delivery status updates.
* Notifications for delivery status changes.

**Payment Integration:**

* Multiple payment gateways (e.g., Razorpay, Stripe).
* Secure payment handling with encryption.
* Automated invoice generation.

**Notifications:**

* Email/SMS notifications for order updates.
* Push notifications for mobile users (Firebase).

**Admin Dashboard:**

* Manage users, products, orders, and deliveries.
* Generate reports on sales, user activity, and delivery performance.

**Search and Filters:**

* Product search functionality.
* Filters for categories, price range, and attributes.

**Reviews and Ratings:**

* Customers can leave product reviews and ratings.
* Display average ratings and reviews on product pages.

**4. Non-Functional Requirements**

* **Performance:** Responsive interface with minimal latency.
* **Scalability:** Handle up to 1 million users and transactions.
* **Security:** Implement SSL, data encryption, and secure APIs.
* **Usability:** Intuitive design for all user roles.

**5. System Architecture**

* **Frontend:** Angular for the web interface.
* **Backend:** Spring Boot framework for Java.
* **Database:** MySQL/PostgreSQL for relational data; Redis for caching.
* **APIs:** RESTful APIs for client-server communication.
* **Deployment:** AWS or Azure for hosting, CI/CD pipeline using Jenkins or GitHub Actions.

**6. Database Schema:**

**Buyers Table:**

|  |  |
| --- | --- |
| **Field** | **Type** |
| Email | Varchar (PK) |
| Password | Varchar |
| Name | Varchar |
| Phone | Number |
| Address | Varchar |
| Rating | Float |
| Premium | Boolean |
| Token | Varchar |
| Refresh\_Token | Varchar |

**Vendors Table:**

|  |  |
| --- | --- |
| **Field** | **Type** |
| Email | Varchar (PK) |
| Password | Varchar |
| Name | Varchar |
| Phone | Number |
| Address | Varchar |
| Rating | Float |
| Experience | Float |
| Token | Varchar |
| Refresh\_Token | Varchar |

**Delivery\_Boys Table:**

|  |  |
| --- | --- |
| **Field** | **Type** |
| Email | Varchar (PK) |
| Password | Varchar |
| Name | Varchar |
| Phone | Number |
| Rating | Float |
| Orders Delivered | Number |
| Token | Varchar |
| Refresh\_Token | Varchar |

**Products Table:**

|  |  |
| --- | --- |
| **Field** | **Type** |
| Name | Varchar |
| Description | Varchar |
| Price | Float |
| Availability | Number |
| Vendor | Vendor |
| Images | Varchar |
| Ordered\_No | Number |
| Tags | Array |

**Reviews:**

|  |  |
| --- | --- |
| **Field** | **Type** |
| Customer | Buyer |
| Product | Product |
| Review\_Date | Date |
| Rating | Float |
| Description | Varchar |

**Orders Table:**

|  |  |
| --- | --- |
| **Field** | **Type** |
| Customer | Buyer |
| Product | Product |
| Address | Varchar |
| Order\_date | Date |
| Delivery\_Date | Date |
| Price | Float |
| Status | Varchar |
| Delivery\_personnel | Delivery\_Boy |

**7. Technical Stack**

* **Backend:** Spring Boot.
* **Database:** MySQL or PostgreSQL.
* **Authentication:** JWT (JSON Web Tokens).
* **Notifications:** Firebase for push notifications.
* **Payments:** Stripe or Razorpay SDK.

**8. Module Design**

**Authentication Module:**

* **Inputs:** User credentials.
* **Processes:** Validate credentials, generate JWT.
* **Outputs:** Authentication token.

**Product Management Module:**

* **Inputs:** Product details from vendors.
* **Processes:** CRUD operations for products.
* **Outputs:** Updated product catalog.

**Order Management Module:**

* **Inputs:** Customer selections and payment details.
* **Processes:** Place orders, update statuses.
* **Outputs:** Order confirmations, real-time updates.

**9. Testing Plan**

* **Unit Testing:** Test individual modules (JUnit for Java).
* **Integration Testing:** Verify interaction between backend and postman.
* **System Testing:** End-to-end testing for workflows.

**10. Deployment Strategy**

* **Environment:** Tomcat, SQL for database.