Software Design Specifications

for

< Library Management System v1.0 >

Prepared by:

Team - 1
Students

Document Information

Title: Library Management System - SDD	**Document Version No: 1.0**
	Document Version Date: 08-04-2025
Prepared By: Team - 1	**Preparation Date: 08-04-2025**

Version History

Ver. No.	Ver. Date	Revised By	Description	Filename
1.0	08-04-2025	Team - 1	Initial Version	SDD_Library_Management_System.pdf

Table of Contents

1 Introduction

This Software Design Specification (SDS) document provides a detailed design for the system being developed. It serves as a guide for developers, testers, and stakeholders to understand the design structure and data handling of the system. The document includes the purpose, scope, definitions, acronyms, references, and overview of the software design.

1.1 Purpose

The purpose of this Software Design Specification document is to describe the internal design of the system in detail, focusing on how the requirements specified in the Software Requirements Specification (SRS) are transformed into a functioning software solution.

This document is intended for:

- Developers → To understand and implement the system design.
- Testers → To develop test cases based on the design.
- Project Managers → To ensure design meets the project requirements.
- Stakeholders → To review and approve the design.

1.2 Scope

This document focuses on the design of the system's architecture, data flow, domain models, and data models. It provides a blueprint for implementing the system and managing persistent data.

This document applies to:

- Functional and non-functional design of the software.
- Data storage and retrieval mechanisms.
- Relationships between different system entities

1.3 Definitions, Acronyms, and Abbreviations

1. SDS – Software Design Specification
2. SRS – Software Requirement Specification
3. UML - Unified Modelling Language
4. ERD - Entity Relationship Diagram
5. DTO – Data Transfer Object

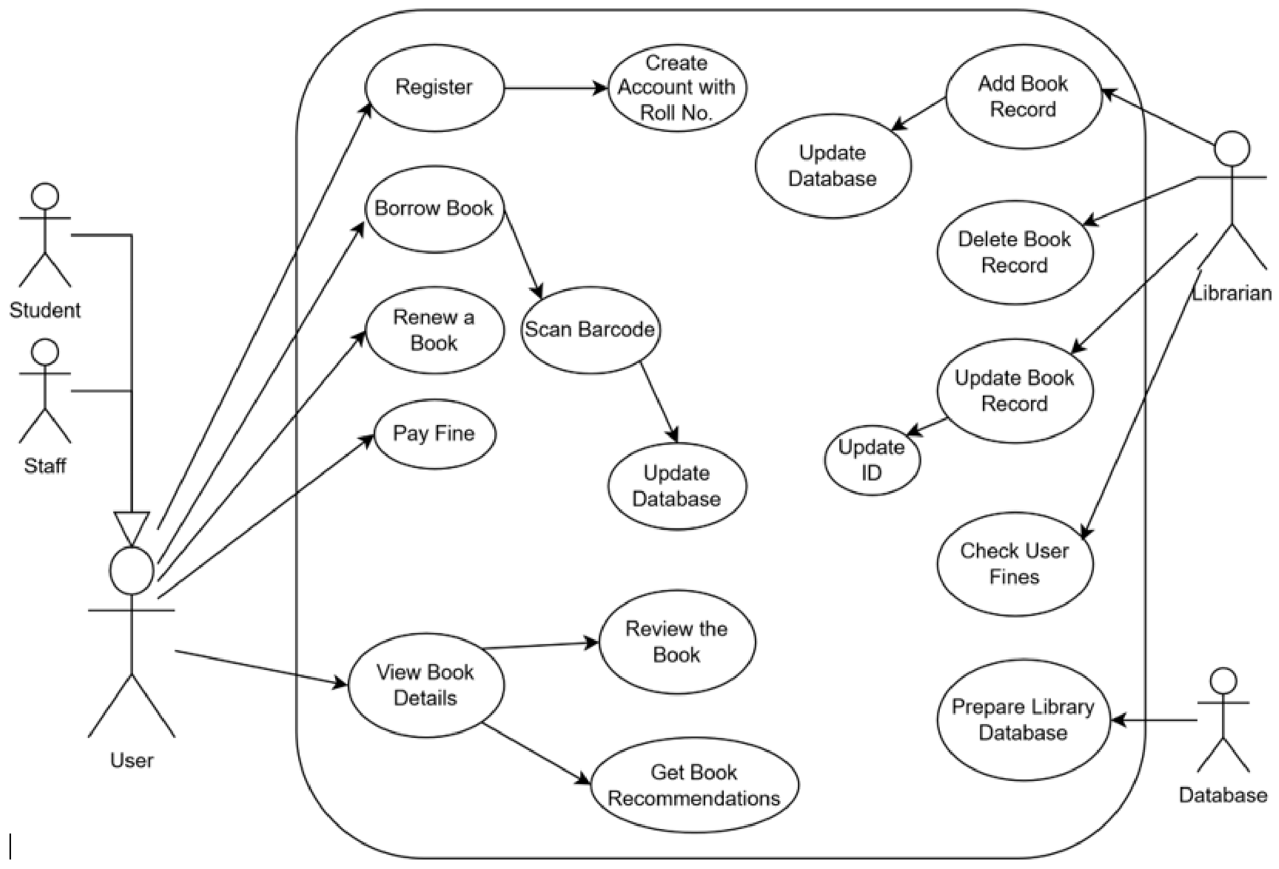
1.4 References

- Software Requirements Specification Document (SRS)
- IEEE 1016-2009 — IEEE Standard for Information Technology - Software Design Descriptions
- UML Diagrams Reference: https://www.uml-diagrams.org/
- ERD Tool: https://dbdiagram.io/
- Design Principles Documentation: https://refactoring.guru/design-patterns

2 Use Case View

This section identifies and describes the key use cases of the system based on the Software Requirements Specification (SRS). The use cases listed here represent the central functionalities of the system and provide insight into how the system interacts with users and other systems.

2.1 Use Case



2.1.1 Use Case: Manage Books

Description: *This use case allows the Librarian to perform operations like adding, updating, deleting, and viewing books in the library.*

Usage Steps:

1. Login as Librarian
2. Navigate to Book Management Section

3. Perform one of the following actions:
    a. Add New Book
    b. Update Book Details
    c. Delete Book
    d. View Book List

2.1.2 Use Case: Manage Members

Description: *This use case allows the Librarian to manage library members by adding, updating, deleting, or viewing member details.*

Usage Steps:

1. Login as Librarian
2. Navigate to Member Management Section
3. Perform one of the following actions:
    a. Add New Member
    b. Update Member Details
    c. Delete Member
    d. View Member List

2.1.3 Use Case: Issue/Return Books

Description: *This use case enables the Librarian to issue books to members and process book returns.*

Usage Steps:

1. Login as Librarian
2. Navigate to Transaction Section
3. Select Member
4. Select Book
5. Perform:
    a. Issue Book
    b. Return Book

2.1.4 Use Case: Manage Librarian

Description: *This use case allows the Admin to manage Librarian accounts within the system.*

Usage Steps:

1. Login as Admin
2. Navigate to Librarian Management Section
3. Perform one of the following actions:
    a. Add New Librarian
    b. Update Librarian Details
    c. Delete Librarian
    d. View Librarian List

2.1.5 Use Case: Search Books

Description: *This use case enables Members to search for books in the library database based on various filters.*

Usage Steps:

1. Login as Member
2. Navigate to Search Books Section
3. Enter Search Criteria
4. View Search Results

2.1.6 Use Case: Borrow/Return Books

Description: *This use case allows Members to view their borrowed books and request for book returns.*

Usage Steps:

1. Login as Member
2. Navigate to My Borrowed Books Section
3. Request for Return (if applicable)

2.1.7 Use Case: View Reports

Description: *This use case allows the Admin to view reports such as books inventory, transactions, and member activities.*

Usage Steps:

1. Login as Admin
2. Navigate to Reports Section
3. Select and View Required Report

3 Design Overview

3.1 Design Goals and Constraints

Goals / Constraints	Details
Performance Goals	The system should retrieve or update book records within 2 seconds.
Scalability Requirements	The system should support an increasing number of users (students/staff) and books without performance degradation.
Security Constraints	User authentication required for Admin and Librarian roles. Only authorized users can perform sensitive operations like book addition or member deletion.
Hardware/Software Constraints	Developed using Flutter, Firebase.
Regulatory or Compliance	Data privacy for user information, basic compliance with institutional IT policies.
Code Maintainability	Modular code structure for easy updates and feature addition.
Legacy System Considerations	No integration required with legacy systems.

3.2 Design Assumptions

- The system will run on local machines or college servers.
- Users will have stable network access to the database.
- The database used will be Firebase Realtime Database.
- Admin credentials are managed securely.
- No third-party APIs are used for core features.
- Maximum of 5,000 books and 1,000 users expected in the initial phase.
- Transaction records are limited to recent years for performance.

3.3 Significant Design Packages

Package Name	Description
User Management	Handles user registration, login, and role-based access (Admin, Librarian, Student).
Book Management	Manages book records: add, update, delete, search, and view book details.
Member Management	Manages library members: registration, profile update, and deletion.
Transaction Management	Manages issue and return of books, maintains transaction records.
Database Layer	Performs all CRUD operations interacting with Firebase.
Report Generation	Generates reports for issued books, returned books, defaulters, etc.

3.4 Dependent External Interfaces

External Module Using the Functionality/Application	Interface Name	Description and Interface Usage
Library Management System	Book Management Interface	Used by Admin/Librarian to Add, Update, Delete, and View Books. Helps manage book details in the library database.
Library Management System	Member Management Interface	Used by Admin/Librarian to Add, Update, Delete, and View Member details in the library database.
Library Management System	Transaction Interface	Used by Librarian to Issue and Return Books to/from members. Maintains transaction records.
Library Management System	Authentication Interface	Used by all Users (Admin, Librarian, Member) to Login and Logout securely.
Library Management System	Search Interface	Used by Members to search books by Title, Author, or Category.
Library Management System	Report Interface	Used by Admin to view system reports like total books, transactions, and member activities.

3.5 Implemented Application External Interfaces (and SOA app services)

The table below lists the implementation of public interfaces this design makes available for other applications.

Interface Name	Module Implementing the Interface	Functionality/Description
Book Management Interface	Book Management Module	This module implements CRUD (Create, Read, Update, Delete) operations for managing books in the system.
Member Management Interface	Member Management Module	This module implements functionality to manage member details and data.
Transaction Interface	Transaction Module	This module implements the functionality for issuing and returning books, managing book availability.
Authentication Interface	Authentication Module	This module implements login validation, password checking, and role-based access control for different users.
Search Interface	Search Module	This module implements search functionality for books using filters like Title, Author, or Category.
Report Interface	Report Module	This module generates various reports based on books, members, and transactions using stored data.

4 Logical View

This section provides the detailed design of the system. The design is represented in layers, starting from the interaction between application modules and drilling down to the interaction of classes within each module to implement the required functionality.

4.1 Design Model

This section provides the class design model, showing the decomposition of the system into modules and significant classes within each module.

Modules Overview:

Module Name	**Description**
User Management	Manages user registration, authentication, and user profiles.
Product Catalog	Manages product details, categories, and availability.
Order Management	Handles order creation, tracking, and status updates.
Payment Gateway	Interfaces with third-party payment systems to handle transactions.
Notification	Sends notifications (Email/SMS) to users regarding order or system updates.

Significant Classes & Responsibilities:

Class Name	**Responsibilities**	**Attributes**	**Operations**	**Relationships**
User	Manage user data	userID, name, email, password	register(), login(), updateProfile()	Aggregates Address class
Product	Store product details	productID, name, price, stock	addProduct(), updateStock(), getDetails()	Associated with Category class
Order	Handle order details	orderID, userID, status, items	createOrder(), cancelOrder(), trackOrder()	Aggregates User, Product
PaymentProcessor	Handle payment transactions	paymentID, orderID, amount	initiatePayment(), verifyPayment()	Associated with Order
NotificationService	Manage user notifications	notificationID, type, message	sendEmail(), sendSMS()	Associated with User

4.2 Use Case Realization

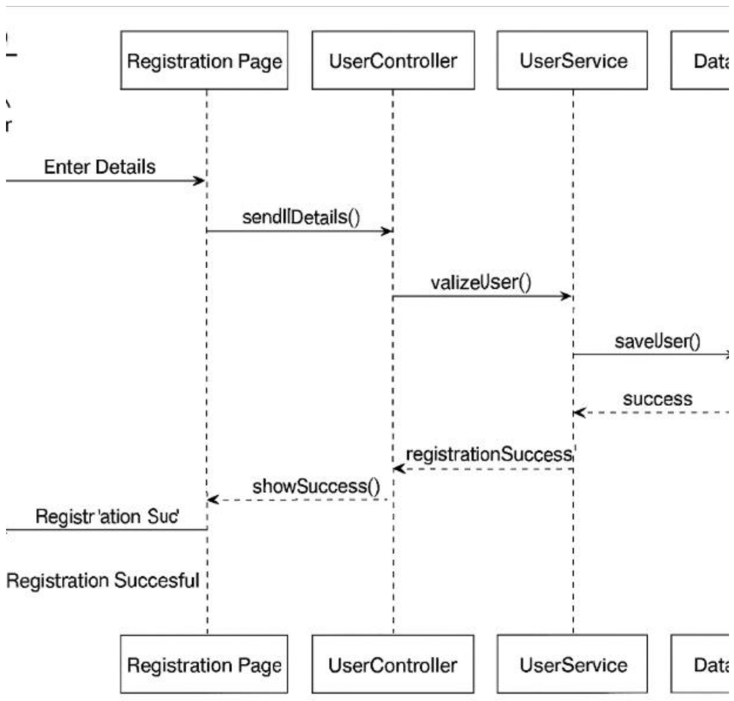
For each Use Case identified in Section 2, this section provides a detailed explanation of how the system modules and classes collaborate to implement the functionality.

Use Case 1: User Registration

Sequence of Interaction:

1. User submits registration form.
2. User Management module validates the input.
3. User object is created and stored in the database.
4. Notification module sends registration confirmation email.

Sequence Diagram

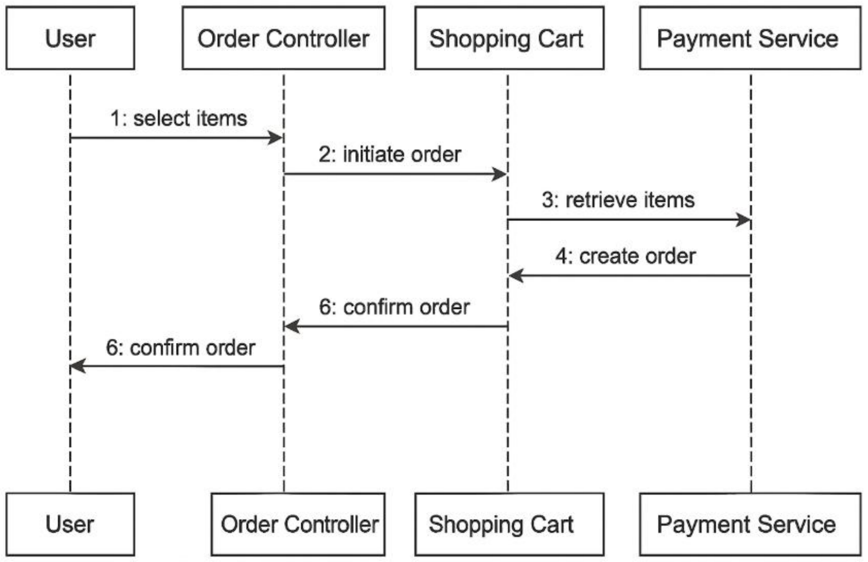


Use Case 2: Place an Order

Sequence of Interaction:

1. User adds products to the cart.
2. Order Management module creates a new order.
3. Payment Gateway processes payment.
4. On successful payment, order is confirmed.
5. Notification module sends order confirmation to the user.

Sequence Diagram

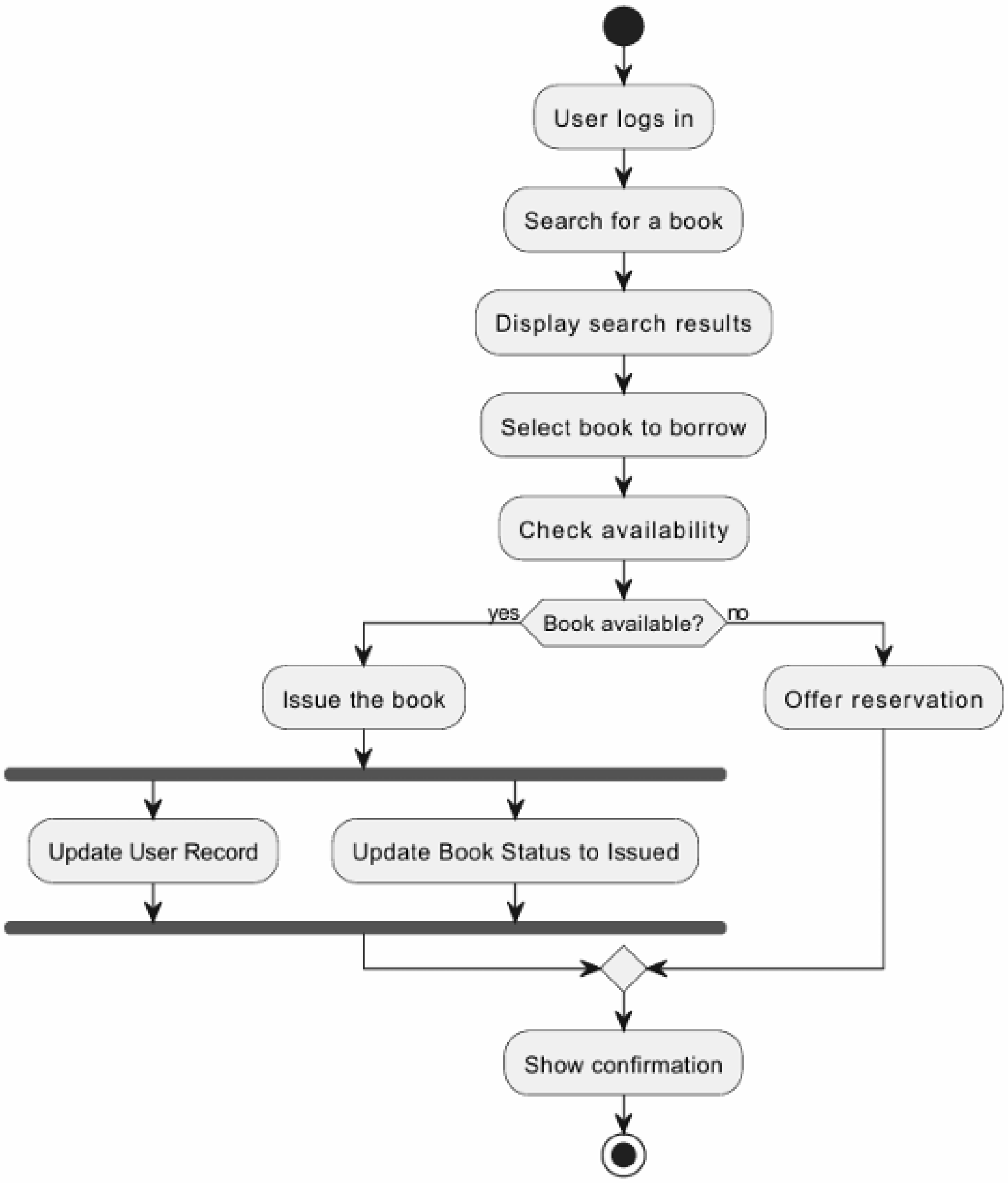


Use Case 3: Track Order

Sequence of Interaction:

1. User requests order status.
2. Order Management module fetches order details.
3. Response sent to user with current order status.

Activity Diagram



5 Data View

This section describes the persistent data storage perspective of the system. It provides details about the data models used in the system, their relationships, and how they represent the domain of the application.

5.1 Domain Model

The domain model represents the key entities (tables/objects) used in the application along with their relationships. These entities are directly mapped to the database and are used to transfer data between different layers of the application.

Entities and Relationships:

Entity	Attributes	Relationships	Description
User	user_id, name, email, password, address	One-to-Many (Order)	Stores user details for login and profile management
Product	product_id, name, description, price, stock	Many-to-Many (Order via OrderItem)	Stores information of products available in the store
Order	order_id, user_id, order_date, total_amount	One-to-Many (OrderItem)	Stores user order details
OrderItem	order_item_id, order_id, product_id, quantity, price	Many-to-One (Order, Product)	Stores individual product details in an order

5.2 Data Model (persistent data view)

```
User ————————< Order ————————< OrderItem >———————— Product

            1        *                 *                    1
```

Description:

- *User*: Can place multiple orders.
- *Order*: Contains order details made by a user.
- *OrderItem*: Contains details of products in each order.
- *Product*: Can belong to multiple orders via OrderItems

5.2.1 Data Dictionary

Table Name	Column Name	Data Type	Description	Constraints
User	user_id	INT	Unique identifier for user	Primary Key, Auto Increment
	name	VARCHAR(100)	Name of the user	Not Null
	email	VARCHAR(100)	Email address of the user	Unique, Not Null
	password	VARCHAR(100)	Encrypted password	Not Null
	address	VARCHAR(200)	Address of the user	-

Table Name	Column Name	Data Type	Description	Constraints
Product	product_id	INT	Unique identifier for product	Primary Key, Auto Increment
	name	VARCHAR(100)	Product name	Not Null
	description	TEXT	Product description	-
	price	DECIMAL(10,2)	Price of the product	Not Null
	stock	INT	Available quantity of the product	Not Null

Table Name	Column Name	Data Type	Description	Constraints
Order	order_id	INT	Unique identifier for order	Primary Key, Auto Increment
	user_id	INT	ID of user placing the order	Foreign Key (User)
	order_date	DATETIME	Date and time of order	Not Null
	total_amount	DECIMAL(10,2)	Total cost of the order	Not Null

Table Name	Column Name	Data Type	Description	Constraints
OrderItem	order_item_id	INT	Unique identifier for order item	Primary Key, Auto Increment
	order_id	INT	ID of related order	Foreign Key (Order)
	product_id	INT	ID of product ordered	Foreign Key (Product)
	quantity	INT	Number of units ordered	Not Null
	price	DECIMAL(10,2)	Price of each product at the time of order	Not Null

6 Exception Handling

This section describes various exceptions that can occur within the application, their causes, how they are handled, and actions to be taken.

6.1 User Registration Exceptions

Exception Name	Cause	Handling Mechanism	Follow-up Action
UserAlreadyExistsException	If email is already registered	Show error message: "User already exists."	Ask user to try with another email
InvalidEmailFormatException	If email format is invalid	Show error message: "Invalid Email Format."	Prompt user to correct email
WeakPasswordException	If password does not meet criteria	Show error message: "Weak Password."	Ask user to enter strong password

6.2 Login Exceptions

Exception Name	Cause	Handling Mechanism	Follow-up Action
UserNotFoundException	Email not found in database	Show error message: "User not found."	Suggest user to register
IncorrectPasswordException	Password does not match	Show error message: "Incorrect Password."	Prompt user to re-enter password

6.3 Product Exceptions

Exception Name	Cause	Handling Mechanism	Follow-up Action
ProductNotFoundException	Product ID not found	Show error message: "Product not available."	Return to product listing
OutOfStockException	Stock not available for requested quantity	Show error message: "Insufficient Stock."	Ask user to reduce quantity or select another product

6.4 Order Exceptionsz

Exception Name	Cause	Handling Mechanism	Follow-up Action
OrderFailedException	Database failure / Payment Failure	Show error message: "Order Failed."	Retry placing order
InvalidOrderDataException	Missing order details / Invalid input	Show error message: "Invalid Order Details."	Ask user to recheck inputs

6.5 Exception Logging

→ All exceptions are logged using centralized logging mechanism.

Log Entry Format: Timestamp | Exception Type | Message | User ID (if any) | Module Name

Logs are stored in:

- `error.log` file
- Admin panel (optional) for error tracking

6.6 Follow-up Actions

- Notify user with appropriate error message.
- Log the error for future debugging.
- If critical → Notify Admin automatically.
- Ensure application stability by redirecting user to safe page (like homepage or previous page).

7 Configurable Parameters

This section describes the configurable parameters used in the application.

7.1 Configuration Parameters Table

Configuration Parameter Name	Definition and Usage	Dynamic? (Can be changed without restart)
MAX_LOGIN_ATTEMPTS	Maximum number of login attempts allowed before locking the account.	Yes
PASSWORD_MIN_LENGTH	Minimum length of password for user registration.	Yes
SESSION_TIMEOUT_MINUTES	Duration of user inactivity after which session expires (in minutes).	Yes
DEFAULT_USER_ROLE	Role assigned to new users upon successful registration.	No
DATABASE_CONNECTION_STRING	Connection string for connecting to the database server.	No
PAYMENT_GATEWAY_API_KEY	API key used for connecting to the payment gateway.	No
EMAIL_SERVER_ADDRESS	SMTP server address used to send emails (like OTP, confirmation, etc.).	Yes
SUPPORT_EMAIL_ADDRESS	Email address where users can send their queries or complaints.	Yes
MAX_CART_ITEM_LIMIT	Maximum number of items allowed in user shopping cart.	Yes
ORDER_CONFIRMATION_TEMPLATE	Path of the template used for sending order confirmation emails.	No

8 Quality of Service

This section describes the design considerations related to application availability, security, performance, and monitoring.

8.1 Availability

- The system is designed for high availability with minimum downtime.
- The application supports 24/7 operation with planned downtime only during scheduled maintenance activities.
- Features to support availability:
- Database Backup & Recovery Mechanisms
- Auto-restart of failed services
- Load Balancer for distributing user requests
- Activities that may impact availability:
- Bulk Data Upload
- Database Maintenance
- Software Updates & Patch Installations

8.2 Security and Authorization

- Security is a primary focus to protect sensitive data and ensure authorized access only.
- User Authentication implemented using:
    - Username & Password
    - OTP Verification for critical actions
- Role-Based Access Control (RBAC) ensures users can only access features based on their assigned roles (Admin, Customer, Delivery Staff).
- Data Security Measures:
    - Password encryption using hashing algorithms
    - Secure database connections
    - API secured with tokens/keys
- User Management:
    - Admin has privilege to Add/Remove/Update User Roles
    - Password Reset feature for users
    - Account Lock after multiple failed logins

8.3 Load and Performance Implications

Component	Performance Expectation	Notes
User Login	< 2 seconds	Includes authentication & role verification
Place Order	< 3 seconds	Includes item validation & order creation
Payment Processing	< 5 seconds	Depends on third-party Payment Gateway
Search Product	< 2 seconds	Indexed database for faster search
Database Growth	10% per year	Proper archiving of old records to maintain performance
Concurrent Users Support	1000+ active users	Scalable architecture to handle load

Performance Testing will focus on:

- Stress Testing
- Load Testing
- Peak Hour Simulation

8.4 Monitoring and Control

Monitoring Features Implemented:

- Application Logs for all critical activities
- Error Logs for Exception Tracking
- Database Activity Monitoring
- Server Resource Monitoring (CPU, Memory, Disk Usage)

Controllable Processes:

- Automated Email Notifications for critical errors
- Alerts for Payment Failures
- Scheduled Job Monitoring (like daily reports, data cleanup)