Online Medicine Ordering System - Development Plan

System Overview

This document outlines the development plan for an online medicine ordering system that allows users to upload prescriptions, buy medicines, and track orders. The system will include features such as prescription verification, teleconsultation, secure payment processing, order tracking, and health record management.

Architecture Design

The system will follow Object-Oriented Programming principles with a layered architecture:

1. **Presentation Layer**: User interfaces for different users (customers, doctors, administrators)
2. **Application Layer**: Business logic, validation, and process management
3. **Data Access Layer**: Database interactions and file management
4. **Security Layer**: Authentication, authorization, and data encryption

Class Structure

Core Classes (Minimum 6 Required Classes)

1. **User (Abstract)** - Base user functionality
   - Properties: userId, name, email, password, contactInfo
   - Methods: login(), register(), updateProfile()
2. **Customer** - Extends User
   - Properties: prescriptions, orders, paymentInfo
   - Methods: uploadPrescription(), orderMedicine(), trackOrder(), makePayment()
3. **Doctor** - Extends User
   - Properties: licenseNumber, specialization, availability
   - Methods: verifyPrescription(), consultPatient(), prescribeMedicine()
4. **Medicine**
   - Properties: medicineId, name, description, price, requiresPrescription, stock
   - Methods: checkStock(), getDetails(), updateStock()
5. **Order**
   - Properties: orderId, customer, items, totalAmount, status, paymentStatus
   - Methods: calculateTotal(), processPayment(), updateStatus(), generateInvoice()
6. **Prescription**
   - Properties: prescriptionId, patientId, doctorId, medicines, issuedDate, isVerified

- Methods: verify(), uploadImage(), getPrescriptionDetails()

Supporting Classes

7. **HealthRecordManager**
   - Properties: patientRecords, prescriptionHistory
   - Methods: storeRecord(), retrieveRecord(), updateRecord(), deleteRecord()
8. **PaymentProcessor (Interface)**
   - Methods: processPayment(), verifyTransaction(), generateReceipt()
9. **PaymentGateway** - Implements PaymentProcessor
   - Properties: gatewayId, supportedMethods, transactionLog
   - Methods: processPayment(), verifyTransaction(), generateReceipt()
10. **SecurityManager**
    - Properties: encryptionKeys, sessionTokens, auditLog
    - Methods: encryptData(), decryptData(), validateToken(), logActivity()
11. **NotificationService**
    - Properties: templates, notificationQueue
    - Methods: sendNotification(), scheduleReminder(), createTemplate()
12. **OrderItem**
    - Properties: medicine, quantity, price
    - Methods: calculateSubtotal(), updateQuantity()

Required OOP Concepts Implementation

1. Overloaded Methods (minimum 2)

- **User.login(String username, String password)**
- **User.login(String token)**
- **NotificationService.sendNotification(String userId, String message)**
- **NotificationService.sendNotification(String userId, String message, NotificationType type)**

2. Overloaded Constructors (minimum 2)

- **Order(String customerId)**
- **Order(String customerId, List<OrderItem> items)**
- **Medicine(String name, double price)**
- **Medicine(String name, double price, boolean requiresPrescription)**

3. Nested Interfaces (minimum 2)

- **Order.StatusUpdateListener**
- **User.AuthenticationProvider**
- **PaymentProcessor.TransactionValidator**

4. Static Classes (minimum 1, can be nested)

- **SecurityManager.EncryptionUtil**
- **HealthRecordManager.RecordValidator**
- **User.Validator**

5. Abstract Classes (minimum 1)

- **User** (base class for Customer and Doctor)
- **PaymentMethod** (base class for CreditCard, DebitCard, etc.)

6. Interface Hierarchy (minimum 1, can be nested interface or single level or multiple inheritance)

- **PaymentProcessor** interface
- **SecurePaymentProcessor** interface extends PaymentProcessor
- **NotificationHandler** interface

7. Hierarchical Inheritance (at least 1)

- **User** → **Customer** and **Doctor**
- **PaymentMethod** → **CreditCard**, **DebitCard**, **NetBanking**

8. Multiple Inheritance (at least 1, in addition to VI above)

- **PremiumCustomer** implements both **Discountable** and **PriorityService** interfaces
- **Admin** extends **User** implements **SystemAccess**

Additional Requirements

9. Wrappers

- **ResponseWrapper** - Encapsulates API responses
- **SecurityWrapper** - Wrapper for secure data handling

10. Package

- **com.medicineordering.user** - User-related classes
- **com.medicineordering.order** - Order processing classes
- **com.medicineordering.security** - Security-related classes

- **com.medicineordering.payment** - Payment processing classes
- **com.medicineordering.notification** - Notification handling classes

11. Exception Handling (at least two cases)

- **PrescriptionVerificationException** - When prescription verification fails
- **PaymentFailedException** - When payment processing fails
- **InvalidUserException** - When user authentication fails
- **MedicineOutOfStockException** - When ordered medicine is not in stock

12. I/O File Handling, Scanner Class etc.

- Prescription image upload and storage
- User profile image handling
- Import/export of medicine inventory
- Log file management

13. Multithreading

- **OrderProcessor** - Implements Runnable for concurrent order processing
- **NotificationDispatcher** - Extends Thread for asynchronous notification delivery
- **InventoryUpdater** - Background thread for inventory management

Development Phases

Phase 1: Core Structure

- Create basic class structure and implement inheritance hierarchy
- Set up project structure with proper packages
- Implement abstract classes and interfaces

Phase 2: Business Logic

- Implement core functionality for each class
- Develop authentication and authorization systems
- Create data models and database access

Phase 3: User Interface

- Develop user interfaces for different user types
- Implement responsive design for web and mobile access
- Create API endpoints for service access

Phase 4: Integration

- Integrate payment processing
- Implement security features
- Set up notification system
- Connect teleconsultation services

Phase 5: Testing and Optimization

- Unit testing of individual components
- Integration testing of the full system
- Performance optimization
- Security auditing

Phase 6: Deployment

- System documentation
- User manual creation
- Deployment planning
- Launch preparation

System Features Checklist

- [ ] User Registration and Authentication
- [ ] Prescription Upload and Verification
- [ ] Medicine Catalog and Search
- [ ] Shopping Cart Functionality
- [ ] Order Processing and Tracking
- [ ] Secure Payment Processing
- [ ] Health Record Management
- [ ] Teleconsultation Integration
- [ ] Automatic Refill Reminders
- [ ] Data Security and Compliance
- [ ] Admin Dashboard for System Management

Technical Requirements

- Java Development Kit (JDK) 17+
- Spring Framework for backend services
- React/Angular for frontend development
- RESTful API design

- MySQL/PostgreSQL for database
- AWS/Azure for cloud hosting
- SSL for secure communication
- JWT for authentication
- Docker for containerization