Table of Contents

1. Introduction

1.1 Purpose

This document defines the software requirements for an Online Pharmacy Store that sells only over-the-counter (OTC) medicines, cosmetics, and dietary supplements. It is intended for the project stakeholders: customers, pharmacist (admin), POS operators, developers, and the course instructor. Requirements, constraints, interfaces, and acceptance criteria are specified to guide design, implementation, and testing. Source materials used: feasibility study and requirements analysis.

1.2 Scope

The system will provide:

- Public-facing web storefront for browsing, searching, and ordering OTC products, cosmetics, and supplements.
- Secure user registration/login, shopping cart, checkout and payment integration.
- Admin dashboard for pharmacists to manage products, categories, inventory, prices, promotions, and orders.
- Integration (or simulation) with the physical store POS to synchronize inventory in real time.
- Reporting tools (sales, inventory, orders).
- No functionality for uploading prescriptions or selling prescription-only drugs.

1.3 Definitions, Acronyms, and Abbreviations

This section defines technical terms and abbreviations used throughout your document to ensure all readers (stakeholders, developers, TAs) understand them clearly.

- **OTC (Over-the-Counter):** Medicines that can be sold directly to a consumer without a requirement for a prescription from a healthcare professional.
- **POS (Point of Sale):** The physical system in the pharmacy where retail transactions are completed.

1.4 References

IEEE Recommended Practice for Software Requirements Specifications

1.5 Overview

This SRS document provides a comprehensive description of the requirements for the Online Pharmacy Store. Section 2 gives an overall description of the product, its users, and its operating environment. Section 3 details the specific functional and non-functional requirements, including use case models and system interfaces. The appendices provide additional details such as a glossary and data dictionary. This document is intended to serve as a guide for the developers and a validation reference for the stakeholders.

2. Overall Description

2.1 Product Perspective

The Online Pharmacy Store is a new web-based application designed to extend the reach of an existing physical pharmacy. It functions as a supplementary sales channel rather than a completely standalone system. A key characteristic of this product is its integration (or simulation for this project's scope) with the physical store's Point of Sale (POS) system to ensure real-time synchronization of inventory levels between online and in-store sales.

2.2 Product Functions

The major high-level functionalities of the system include:

- **Public Storefront:** Browsing and searching for Over-the-Counter (OTC) medicines, cosmetics, and dietary supplements.
- **User Account Management:** Secure registration, login, and profile management for customers.
- **E-commerce Features:** Shopping cart management, order placement, and simulated secure payment integration.
- **Administrative Dashboard:** Tools for pharmacists to manage product listings, categories, prices, and promotional discounts.
- **Order fulfillment:** Admin capabilities to monitor, approve, or cancel customer orders.
- **Inventory Synchronization:** Automated updates to stock levels based on both online and in-store (POS) sales.
- **Reporting:** Generation of basic sales, inventory, and order history reports

2.3 User Classes and Characteristics

1. Customers (Guest/Registered):

They use the website or mobile app to browse, search, and order medicines easily.

2. Pharmacists (Admin):

They manage products, check prescriptions, handle orders, and update inventory through the admin dashboard.

3. Store Staff (POS Operators):

Employees in physical stores who handle in-store sales. Their sales and stock data sync automatically with the online system.

2.4 Operating Environment

- **Client Side:** The web application must be accessible via standard modern web browsers (Chrome, Firefox, Edge, Safari) on both desktop and mobile devices.
- **Server Side:** The backend will run in a Python environment supporting the Flask framework.

2.5 Design and Implementation Constraints

Technical Constraints:

- The backend must be implemented using the Python Flask framework.
- The system must strictly adhere to the Model-View-Controller (MVC) architectural pattern.
- GitHub must be used for version control throughout development.

Domain & Legal Constraints:

- Sales are strictly limited to OTC products; prescriptions cannot be uploaded or sold.

- The system must comply with relevant national e-commerce and data privacy laws

2.6 User Documentation

- **Online FAQ:** A Frequently Asked Questions page for customers regarding ordering, payments, and returns.
- **Admin User Manual:** A simple guide for the Pharmacist/Admin explaining how to add products, manage inventory, and process orders via the dashboard

2.7 Assumptions and Dependencies

Assumptions:

The physical pharmacy maintains accurate and up-to-date stock records that can be accessed or integrated for synchronization with the online system.

Dependencies:

- **Payment Gateway API:** The system depends on a third-party service (simulated for this project) to handle secure credit card transactions.
- **Physical Store POS System:** The application depends on the existing Point of Sale system (or a simulation of it) to provide real-time inventory data from in-store sales.
- **Hosting Environment:** The system relies on a Flask-compatible hosting environment for deployment.

3. Specific Requirements

3.1 Functional Requirements

Customer Requirements:

- **FR-001: User Registration** The system shall allow a new customer to create a secure account. The registration process must require a unique email address (as a username), a password, and basic contact information. The system will perform field validation to ensure the email is in a valid format and that the password meets minimum complexity rules (e.g., minimum 8 characters). Upon successful registration, the user is automatically logged in and redirected to the home page.
- **FR-002: User Authentication** The system shall provide a secure login page for returning customers. A user must enter their registered email and password. The system will validate these credentials against the database. If successful, the user is granted access to their account-specific features (e.g., shopping cart, order history). If unsuccessful (e.g., incorrect password), a clear error message will be displayed. The system shall also provide a "Forgot Password" workflow.
- **FR-003: Product Browsing and Filtering** The system shall display all available products (OTC medicines, cosmetics, supplements) on a main storefront, organized by categories and sub-categories. A user must be able to browse these categories (e.g., "Vitamins & Supplements," "Skin Care") via a navigation menu. On a category page, the user shall be able to filter the product list by attributes such as brand and price range.
- **FR-004: Product Search** The system shall provide a prominent search bar on all pages. A user can enter keywords (e.g., product name, brand, or symptom like "cough") to find relevant products. The system will return a search results page displaying all products that match the query, including a product image, name, and price.

- **FR-005: Shopping Cart Management** A user (both guest and logged-in) shall be able to add a product to their shopping cart from a product details page or a product listing page. The user can view their cart at any time, which displays a summary of all items, their quantities, and the subtotal. From the cart view, the user must be able to update the quantity of any item, remove an item completely, or clear the entire cart.
- **FR-006: Checkout Process** The system shall guide a logged-in user through a checkout process to place an order. This process includes (1) confirming their shipping address and (2) viewing a final order summary (items, subtotal, simulated shipping). Upon confirming the order, the system will simulate a payment transaction. The system must validate that all required information is provided before placing the order.
- **FR-007: View Order History** A logged-in customer shall be able to access a "My Account" or "Order History" page. This page will display a list of all past orders placed by that user. For each order, the user can see the order number, date placed, total cost, items purchased, and the current order status (e.g., "Pending," "Processing," "Shipped," "Cancelled").

Admin (Pharmacist) Requirements:

- **FR-008: Admin Authentication** The system shall provide a separate, secure login portal for Admin (Pharmacist) users, inaccessible to regular customers. Admin credentials will be validated, and upon successful login, the admin will be directed to the main admin dashboard. Access to any admin-level page or function must be protected and require an active admin session.
- **FR-009: Product Management** The admin shall have full CRUD (Create, Read, Update, Delete) capabilities for all products in the system. When adding or editing a product, the admin must be able to specify its name, a detailed description, price, product images, its category, and its "availability" status (e.g., "In Stock," "Out of Stock").
- **FR-010: Inventory and Stock Management** The admin shall be able to view and manually update the stock level (quantity on hand) for any product. This stock level is the master count used to determine availability on the storefront. When an online order is placed, this stock level must be automatically decremented.
- **FR-011: Order Management** The admin shall be able to view a list of all incoming customer orders from the dashboard. The admin can click on an order to see its full details (customer, items, shipping address). From this view, the admin must be able to update the status of the order (e.g., change from "Pending" to "Processing" or "Shippe

System Requirements:

- **FR-012: POS Inventory Synchronization (Simulation)** The system shall simulate integration with a physical store's POS. When a (simulated) in-store sale occurs, the system's database for that item's stock quantity shall be automatically decremented in real-time. This ensures that the stock availability shown on the website is accurate and prevents selling out-of-stock items from either sales channel.

3.4 Non-Functional Requirements

- **NFR-001: Security (Authentication & Data Storage)**

Description: The system must protect user data and ensure secure access. All customer and admin passwords must not be stored in plain text. Instead, they must be stored in the database in a hashed and salted format. All data transmitted during login, registration, and checkout must be secure from casual interception.

- **NFR-002: Usability (Mobile Responsiveness)**

Description: The web application's user interface must be usable and aesthetically pleasing on small screens, such as mobile phones. Key pages (Home, Category, Product, and Cart) must re-format to fit smaller viewports without requiring horizontal scrolling.

- **NFR-003: Performance (Page Load Time)**

Description: The system must feel responsive to the user during normal browsing. Key user-facing pages, such as the Home page, Category pages, and Product Detail pages, should load quickly

3.2 Use Case Model

3.2.1 Use case diagram

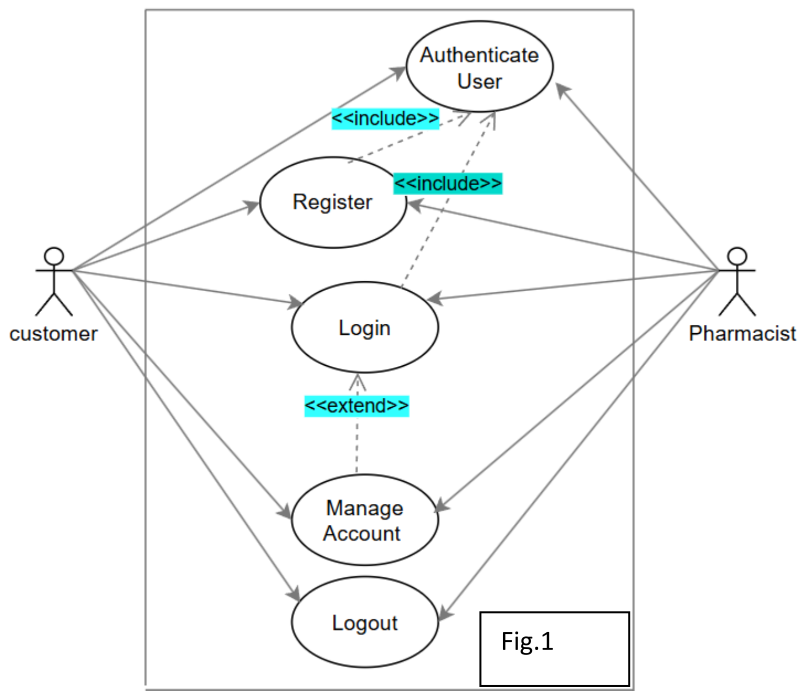


Fig.1

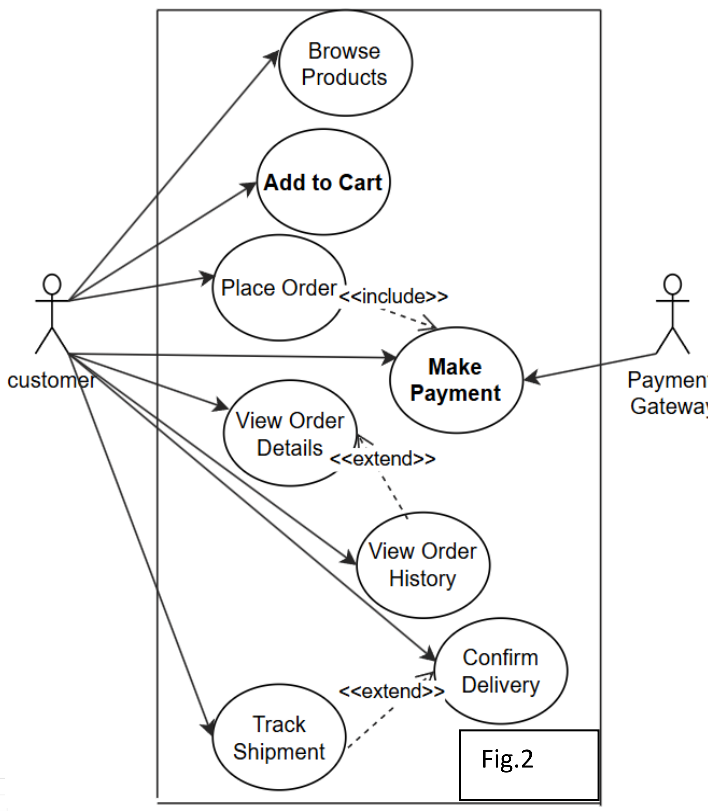


Fig.2

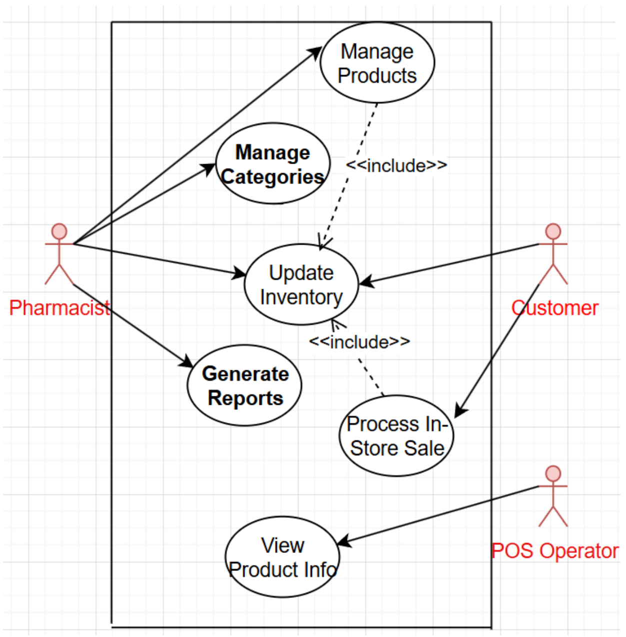


Fig.3

3.2.1 Use case description

Figure.1

Register

Use Case ID: UC-01
Use Case Name: Register
Primary Actor: Customer, Pharmacist
Stakeholders and Interests:
Customer: Wants to create an account to save details and track future orders.
System: Needs to collect accurate customer data and ensure generic accounts aren't duplicated.
Preconditions:
The customer does not currently have an active account with the provided email/username.
System is online.
Postconditions:
A new customer account is created and stored in the database (Success).
Customer is notified of successful registration.
Trigger:
Customer selects the "Register" or "Sign Up" option on the application
Main Success Scenario (Basic Flow):

1. Customer navigates to the registration page.
2. Customer enters required personal details (name, email, address, password).
3. System validates the input format (e.g., valid email format, strong password).
4. System checks for existing accounts with the same credentials.
5. System [Authenticates User] (verifies email/phone if required).
6. System creates the account and confirms success to the Customer

Special Requirements:
Passwords must be hashed and salted before storage.
Communication must be over HTTPS
Priority: High
Frequency of Use: Daily

Login

Use Case ID: UC-02
Use Case Name: Login
Primary Actor: Customer, Pharmacist
Stakeholders and Interests:
Customer/Pharmacist: Wants secure access to their specific features: ordering for customers, inventory management for pharmacists.

System: Ensures only authorized users access restricted areas.

Preconditions:
User is registered and has valid credentialsSystem is online.
Postconditions:
User is authenticated and redirected to their respective dashboard (Success).
Access is denied with an appropriate error message (Failure)
Trigger:
User attempts to access a restricted area or clicks the "Login" button.

Main Success Scenario (Basic Flow):

1. User navigates to the login screen.
2. User enters username/email and password.
3. User clicks "Submit".
4. System [Authenticates User] by validating credentials against the database.
5. System identifies the user role (Customer or Pharmacist).
6. System grants access to the appropriate dashboard.

Special Requirements:
Account should be temporarily locked after 5 failed login attempts.
Priority: High
Frequency of Use: High

Manage Account

Use Case ID: UC-03
Use Case Name: Manage Account
Primary Actor: Customer, Pharmacist
Stakeholders and Interests:
user: Needs to keep their personal information (address, phone, payment methods) up to date.
Preconditions:
User must be currently logged in (extends Login)..
Postconditions:
User profile is updated in the database.
Trigger:
Authenticated user selects "My Profile" or "Settings".
Main Success Scenario (Basic Flow):

1. User selects the option to edit their profile.
2. System displays current user details.
3. User updates one or more fields (e.g., new shipping address).
4. User clicks "Save".
5. System validates the new information.
6. System updates the record and displays a success message.

Special Requirements:
Critical changes (like password or email) may require re-authentication.
Priority: Medium
Frequency of Use: Low/Medium

Figure. 2

Browse Products

Use Case ID: UC-04
Use Case Name: Browse Products
Primary Actor: Customer, payment-gateway
Stakeholders and Interests:
Customer: Wants to find products, compare prices, and read descriptions easily.
Business: Wants to showcase products effectively to drive sales.
Preconditions:
User has accessed the website or mobile app.
Postconditions:

The customer has located one or more products of interest.
Trigger:
User visits the site, uses the search bar, or selects a product category
Main Success Scenario (Basic Flow):

1. User navigates to the "Products," "Shop," or a specific category page.
2. System displays a list of products with images, names, and prices.
3. User can optionally filter or sort the list (e.g., by price, brand).
4. User can optionally use the search bar to find a specific product.
5. User clicks on a product to view its detailed description, ingredients, and usage instructions.

Special Requirements:
Search functionality should be fast and handle common misspellings.
Priority: High
Frequency of Use: High

Add to Cart

Use Case ID: UC-05
Use Case Name: Add to Cart Primary
Primary Actor: Customer, payment-gateway
Stakeholders and Interests:
Customer: Wants to easily collect items they intend to purchase.
Preconditions:
Customer is browsing products
Customer has selected a specific product
Postconditions:
The selected product and quantity are added to the customer's virtual shopping cart.
The cart's subtotal is updated.
Trigger:
Customer clicks the "Add to Cart" button on a product page or product listing.

Main Success Scenario (Basic Flow):

1. User is on a product details page.
2. User (if applicable) selects a quantity.
3. User clicks "Add to Cart".
4. System validates the item (e.g., stock availability).
5. System adds the item to the cart session.
6. System provides visual feedback (e.g., "Item added to cart") and updates the cart icon/summary.

Special Requirements:
--
Priority: High
Frequency of Use: High

Place Order

Use Case ID: UC-06
Use Case Name: Place Order
Primary Actor: Customer, payment-gateway
Stakeholders and Interests:

Customer: Wants to finalize their purchase, confirm shipping, and proceed to payment.
Business: Wants to capture the sale and all necessary fulfillment details.
Preconditions:
Customer is logged in.
Customer has at least one item in their shopping cart
Postconditions:
An order record is created in a "Pending" state.
Payment is successfully processed.
Order status is updated to "Processing" or "Confirmed".
Inventory is updated.
Trigger:
Customer clicks the "Checkout" or "Proceed to Checkout" button from the shopping cart.
Main Success Scenario (Basic Flow):

1. User initiates checkout.
2. System displays steps for:
    a. Confirming/Entering Shipping Address
    b. Selecting Shipping Method
    c. Confirming/Entering Billing Address
3. System displays a final order summary (items, prices, taxes, shipping fee, total).
4. User confirms the order details and proceeds to payment.
5. System initiates the Make Payment use case (UC-13). (<<include>> relationship)
6. Upon successful payment, the system confirms the order.
7. System displays an "Order Confirmation" page and sends a confirmation email.

Special Requirements:
Must perform address validation.
Must clearly display all costs, including taxes and shipping, before payment.
Priority: High
Frequency of Use: medium

Make Payment

Use Case ID: UC-07
Use Case Name: Make payment
Primary Actor: Customer, payment-gateway
Stakeholders and Interests:
Customer: Wants to pay securely and receive confirmation of payment.

Business: Wants to securely receive payment for the order.

Payment Gateway: Provides the service to process the transaction.

Preconditions:
The **Place Order** (UC-6) use case has been initiated, and an order total has been calculated

Postconditions:
Payment is authorized and captured, OR payment is declined.

A transaction record is created.

Trigger:
This use case is automatically triggered (included) by the **Place Order** (UC-12) use case.

Main Success Scenario (Basic Flow):

1. System presents the user with payment options (e.g., Credit Card, PayPal).
2. Customer selects a payment method and enters their details.
3. System securely transmits the payment details and transaction amount to the Payment Gateway.
4. Payment Gateway processes the transaction and returns a success or failure response.
5. System receives the response.
6. System records the transaction status.
7. System returns control to the **Place Order** (UC-12) flow.

Special Requirements:
All communication must be over HTTPS.

No sensitive credit card information (full PAN, CVV) should be stored in the local database. Must be PCI-compliant.

Priority: High
Frequency of Use: medium

View Order Details

Use Case ID: UC-08
Use Case Name: view order details
Primary Actor: Customer, payment-gateway
Stakeholders and Interests:
Customer**:** Wants to review all specifics of a current or past order (items, cost, shipping address).

Preconditions:
Customer is logged in.

Customer has selected a specific order from their order history OR is in a process that allows viewing details (like payment).

Postconditions:

The customer has viewed the complete details for the selected order.

Trigger:
Customer clicks on an order from the **View Order History** list.

(As per diagram) This use case can optionally be triggered during the **Make Payment** use case.

Main Success Scenario (Basic Flow):

1. User selects an order to view.
2. System retrieves all information for that order.
3. System displays:
   - Order ID, Order Date, Order Status
   - List of items (name, quantity, price)
   - Shipping Address
   - Billing Address
   - Payment Method (partial/masked)
   - Cost summary (subtotal, shipping, tax, total) **Extension Scenario (Extending UC-13):**
4. While on the **Make Payment** (UC-13) screen, the user is worried they missed something.
5. User clicks a "Review Order Details" link (this is the extension point).

6. System displays the full order details (as in the Main Success Scenario).
7. User reviews the details and clicks "Back to Payment" to resume the payment process.

Special Requirements:
Must clearly show the order's *current* status (e.g., Pending, Processing, Shipped, Delivered).

Priority: medium
Frequency of Use: medium

View Order History

Use Case ID: UC-09
Use Case Name: View Order History
Primary Actor: Customer, payment-gateway
Stakeholders and Interests:

- **Customer:** Wants to see a list of all past and current orders.

Preconditions:

- Customer must be logged in.

Postconditions:

- The customer has viewed their list of orders.

Trigger:

- Authenticated user selects "My Orders" or "Order History" from their account menu.

Main Success Scenario (Basic Flow):

1. User clicks the "Order History" link.
2. System retrieves a list of all orders associated with the customer's account.
3. System displays a summarized list, showing (for each order):
    - Order ID
    - Order Date
    - Total Price
    - Current Status (e.g., "Shipped")
4. User can click on any order in this list to trigger **View Order Details**.

Special Requirements:

- Should provide pagination if the list of orders is long.

Priority: Medium

Frequency of Use: Medium

Track Shipment

Use Case ID: UC-10
Use Case Name: Track Shipment
Primary Actor: Customer, payment-gateway
Stakeholders and Interests:

- **Customer:** Wants to know the current location and estimated delivery date of their order.

Preconditions:

- Customer is logged in.
- The order has already been shipped and has an associated tracking number.

Postconditions:

- The customer has viewed the current shipping status.

Trigger:

- Customer clicks a "Track Shipment" link from their **View Order Details** page.

Main Success Scenario (Basic Flow):

1. User views the details for a shipped order
2. User clicks the "Track Shipment" button/link.
3. System retrieves the tracking number for the order.
4. System either: a. Displays the tracking status by calling the carrier's API. b. Redirects the user to the carrier's website with the tracking number pre-filled.
5. (Extension Point) If the carrier status is "Delivered," the system may present the option to **Confirm Delivery Special Requirements:**

- Must have integration with the shipping carrier's tracking system.

Priority: Medium

Frequency of Use: Low (per order)

Confirm Delivery

Use Case ID: UC-11
Use Case Name: Confirm Delivery
Primary Actor: Customer, payment-gateway
Stakeholders and Interests:

- **Customer:** Wants to confirm they have received their items.
- **Business:** Wants to get confirmation of receipt for their records, which can help resolve disputes.

Preconditions:

- Customer is logged in.

- This use case extends **Track Shipment**
- The carrier's system has marked the order as "Delivered".

Postconditions:

- The order status in the pharmacy's system is updated to "Delivery Confirmed by Customer".

Trigger:

- This use case is optionally triggered during the **Track Shipment** use case, when the order's status is "Delivered".

Main Success Scenario (Basic Flow):

1. User is tracking their shipment and the system shows a "Delivered" status.
2. System displays a button/prompt: "Have you received your order? Click to confirm."
3. User clicks the "Confirm Delivery" button.
4. System updates the order's status to "Delivery Confirmed".
5. System displays a "Thank you for confirming" message.

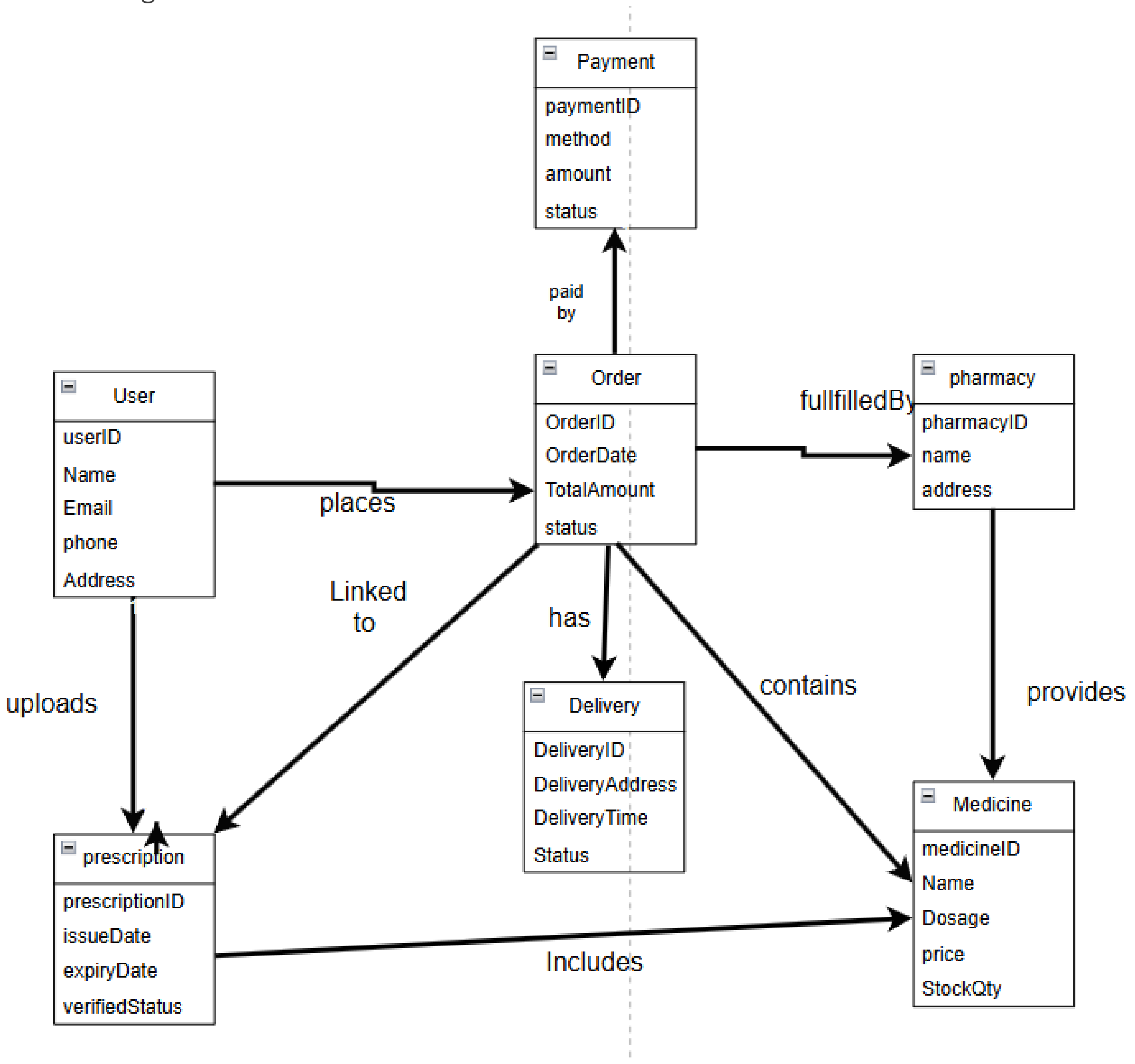
Special Requirements:

- This confirmation should be stored and timestamped.

Priority: Low **Frequency of Use:** Low

3.3 Domain Model

3.3.1 class diagram



3.3.2 class diagram description

User

Attributes: userID, name, email, phone, address

Responsibilities: Registers and authenticates; uploads prescriptions; places orders; receives deliveries.

Prescription

Attributes: prescriptionID, issueDate, expiryDate, verifiedStatus

Responsibilities: Represents a physician's authorization to dispense medicine; must be verified before controlled items are fulfilled.

Order

Attributes: orderID, orderDate, totalAmount (derived), status

Responsibilities: Captures a purchase event; aggregates line items; coordinates payment and delivery.

Payment

Attributes: paymentID, method, amount, status, transactionRef

Responsibilities: Records settlement information for an Order.

Delivery

Attributes: deliveryID. deliveryAddress, deliveryTime (ETA/actual), status, trackingCode

Responsibilities: Manages the logistics of getting an order to the user.

pharmacy

Attributes: pharmacyID, name, address, licenseNumber

Responsibilities: Supplies inventory; verifies prescriptions (where required); fulfills orders.

medicine

Attributes: medicineID, name, dosage, price, stockQty, isControlled

Responsibilities: A sellable item with safety and compliance constraints.

3.5 External Interface Requirements

3.5.1 User Interface

The user interface will be a web-based application accessible via standard web browsers (e.g., Chrome, Firefox, Edge). The interface will be functional and responsive, implemented primarily using standard HTML and CSS. The system will provide two main interface views:

- **Customer Storefront:** A public-facing interface allowing users to browse categories, search for over-the-counter (OTC) products, manage a shopping cart, and securely checkout.
- **Admin Dashboard:** A secured, authenticated interface for pharmacists to manage product listings, view sales reports, and update inventory.

3.5.2 Hardware Interface

The system is a web application and generally does not require specialized hardware on the client side beyond a standard device with internet access (computer, smartphone, tablet).

- **Server Side:** The application will run on a standard server capable of hosting Python Flask applications.
- **POS Integration (Simulation):** The system may interface with a simulated Point of Sale (POS) scanner input for the purpose of synchronizing physical store inventory with the online database.

3.5.3 Software Interface

Database Management System: The system will interface with a relational database (such as SQLite or MySQL) to store and retrieve centralized data regarding products, users, and orders.

Payment Gateway API: The system will interface with a third-party payment gateway API (e.g., Stripe or PayPal sandbox for testing) to securely process simulated online transactions.

Web Server Framework: The backend will interface with the Flask Python web framework to handle HTTP requests and application logic.

3.5.4 Communication Interface

Communication Protocols: The system will primarily use **HTTPS** (Hypertext Transfer Protocol Secure) to ensure encrypted and secure communication between the client browser and the web server, specifically protecting user login credentials and payment data.

Data Format: JSON (JavaScript Object Notation) will likely be used for asynchronous data transfer between the frontend and backend, and for communication with external APIs (like the payment gateway).