

**WEB-BASED PHARMACY PRODUCT MANAGEMENT SYSTEM**

NAME:MOUNET NOTAM URIELLE MERVEILLE : ICTU20234379

LECTURER : ENG DANIEL MOUNE

COURSE : DISTRIBUTED SYSTEM AND CLOUD COMPUTING

# **TABLE OF CONTENTS**

1.0 Introduction

* + Project Vision & Objectives
  + Document Scope & Audience
  + Project Scope & Limitations

2.0 Overall System Description

* 2.1 System Overview & Value Proposition
* 2.2 User Characteristics & Roles
* 2.3 Operational Environment & Constraints

3.0 System Architecture & Design

* 3.1 Technology Stack
* 3.2 High-Level System Architecture
* 3.3 Use Case Model

4.0 Database Design

* 4.1 Entity-Relationship (ER) Diagram
* 4.2 Detailed Database Schema

5.0 Functional Requirements Specifications

* 5.1 Authentication & Authorization Module
* 5.2 Dashboard Module
* 5.3 Category Management Module
* 5.4 Supplier Management Module
* 5.5 Product Management Module
* 5.6 Stock Management Module
* 5.7 Sales Management Module
* 5.8 Reporting Module
* 5.9 System Administration Module
* 5.10 Note on Missing Deployment Functionality

6.0 Dynamic Modeling: Sequence Diagrams

* 6.1 User Login Process
* 6.2 Add a New Product Process
* 6.3 Execute a Sales Transaction Process
* 6.4 Restock Product Process

7.0 Behavioral Modeling: Activity Diagrams

* 7.1 Overall User Navigation Activity
* 7.2 Process a Sale Activity
* 7.3 Manage Product Inventory Activity

8.0 User Interface Design

* 8.1 Design Philosophy & Guidelines
* 8.2 User Interface Navigation Map

9.0 Conclusion & Future Enhancements

1.O References

# **1.0 Introduction**

# **1.1 Project Vision & Objectives**

The Web-based Pharmacy Product Management System is designed to be the operational backbone of a modern pharmacy. It aims to eliminate inefficiencies and errors associated with manual record-keeping by providing a centralized, secure, and intuitive platform. The core objectives are:

· Data Integrity: To maintain accurate and consistent information across all entities (products, suppliers, stock, sales).

· Operational Efficiency: To streamline key workflows, particularly the point-of-sale and inventory restocking processes.

· Decision Support: To provide actionable insights through dashboard analytics and sales reports, enabling data-driven restocking and business decisions.

· Security: To ensure that sensitive business data is accessible only to authorized personnel through robust authentication and role-based access control.

# **1.2 Document Scope & Audience**

This document provides a complete specification for the system's functionality, architecture, and behavior. It is intended for:

· Software Developers: As a blueprint for system construction, implementation, and unit testing.

· Quality Assurance Engineers: As a source for deriving test cases and validation criteria.

· Project Managers & Stakeholders: To verify feature completeness and project scope.

· End-Users & Pharmacists: To understand system capabilities and operational workflows.

· Maintenance Team: For future troubleshooting and system enhancements.

# **1.3 Project Scope & Limitations**

In-Scope Features:

· Secure user authentication and session management.

· Comprehensive CRUD operations for Categories, Suppliers, and Products.

· Manual stock management via explicit restocking actions.

· Integrated Point-of-Sale (POS) system with automatic, real-time stock deduction upon sale completion.

· Basic dashboard analytics and sales history reporting.

· User account management and database backup functionality.

Notable Limitation & Out-of-Scope Items:

· Deployment Module: The system, as specified, does not include a built-in functionality for deploying or migrating the database schema and initial data to a production server. Deployment is a manual process (see Section 9.2).

· Barcode or QR code integration.

· Expiry date tracking and alerts.

· Customer management and loyalty programs.

· Multi-branch or warehouse inventory management.

· Integration with external accounting systems.

# **2.0 Overall System Description**

# **2.1 System Overview & Value Proposition**

The system is a client-server web application. Users interact with the system via a web browser, which communicates with a backend server running PHP and MySQL. The primary value proposition is the seamless integration of sales and inventory management. When a product is sold through the POS interface, the system automatically and immediately deducts the quantity from the product's stock level, ensuring perpetual inventory accuracy without any manual intervention from the staff. This real-time update prevents stock-outs and overselling.

# **2.2 User Characteristics & Roles**

|  |  |  |
| --- | --- | --- |
| ROLE | RESPONSABILITIES | SYSTEMS ACCESS |
| Administrator | Full system oversight, user management, and data backup | Unrestricted access to all modules. |
| Pharmacy Manager | Oversee inventory, manage product catalog, analyze reports, process sales | Access to all modules except User Management. |
| Sales Clerk | Serve customers at the point of sale. | Access limited to the POS interface and product search functionality. |

# **2.3 Operational Environment & Constraints**

· Software Environment: The application requires a standard LAMP (Linux, Apache, MySQL, PHP) or WAMP (Windows, Apache, MySQL, PHP) stack to operate.

· Hardware Constraints: Can run on a standard desktop computer acting as a local server for a small pharmacy.

· Network Constraints: For single-store use, it can operate on a Local Area Network (LAN). A wider network would require appropriate network and security configuration.

· Browser Dependency: Requires a modern JavaScript-enabled web browser (Chrome, Firefox, Safari, Edge) on client machines.

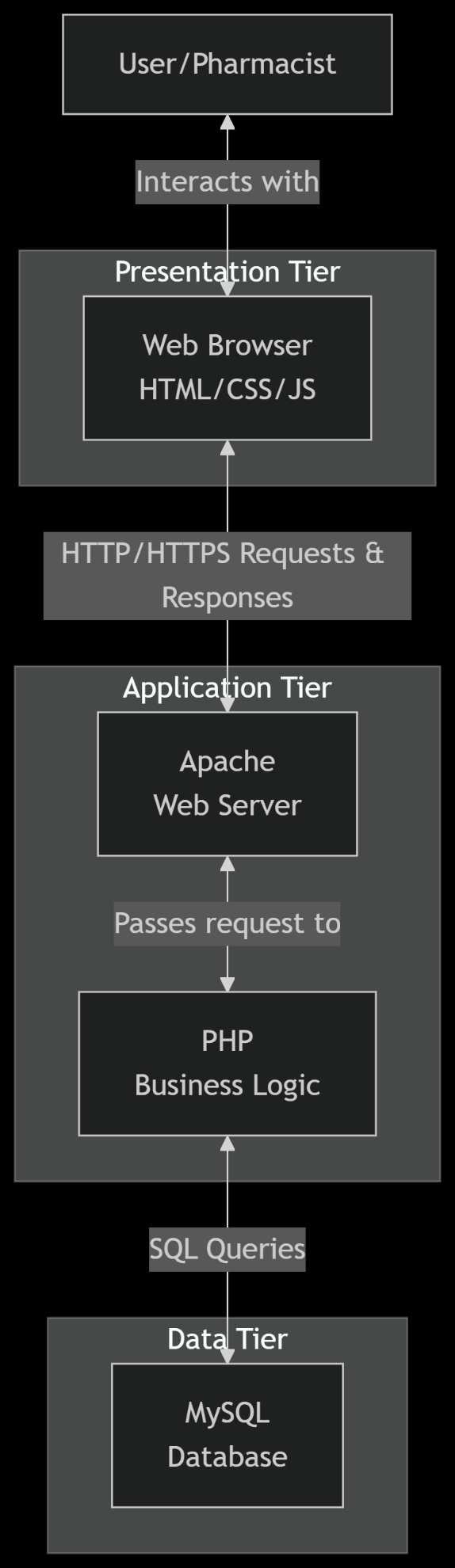
# **3.0 System Architecture & Design**

# **3.1 Technology Stack**

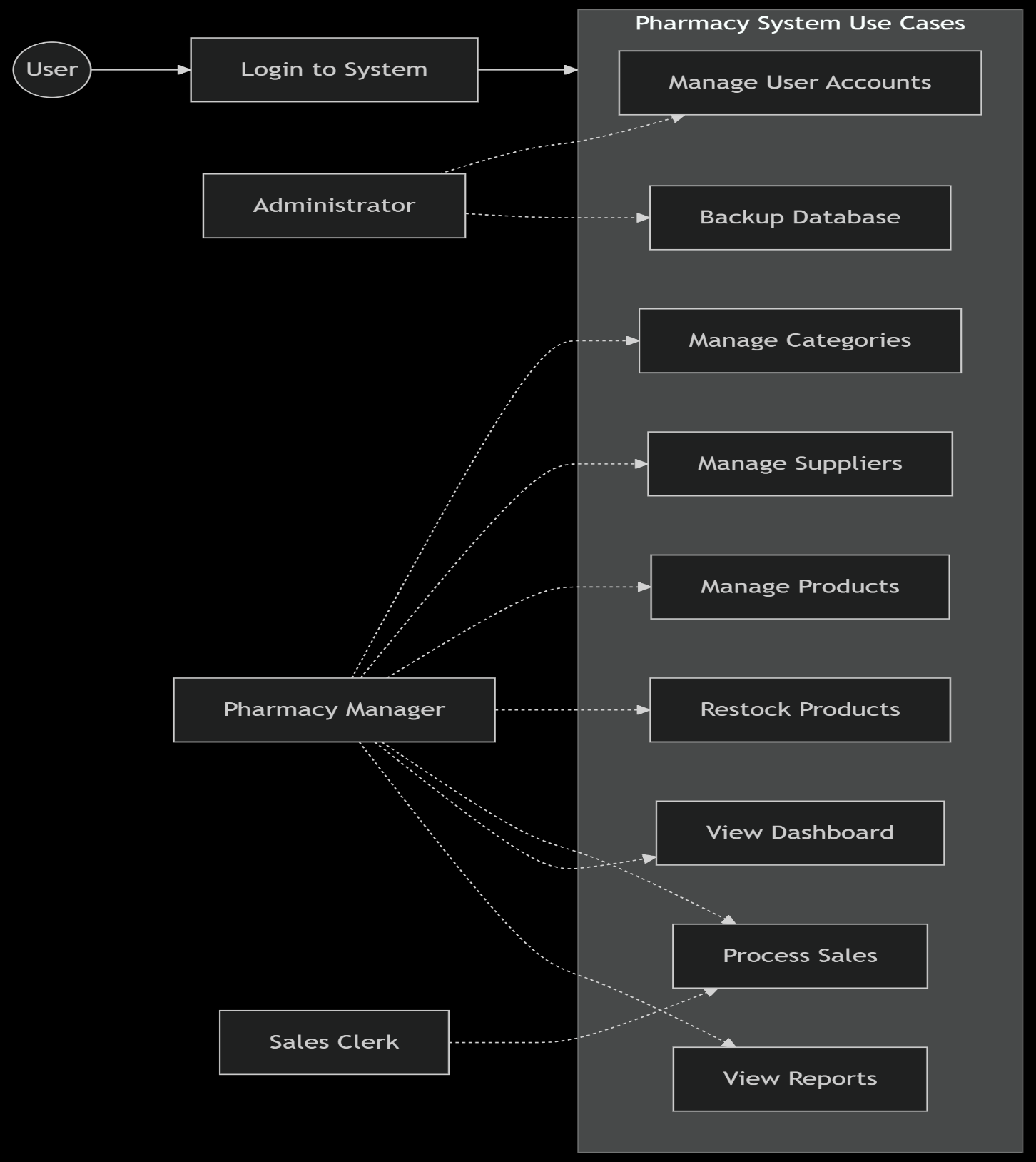
|  |  |  |
| --- | --- | --- |
| Layer | Technology | Description |
| Presentation Layer | HTML5, CSS3, JavaScript, Bootstrap | Creates a responsive, interactive, and user-friendly interface. |
| Application Layer | PHP | Implements all business logic, processes forms, manages sessions, and handles database communication. |
| Data Layer | MySQL | A relational database management system for persistent storage of all application data. |
| Server | Apache HTTP Server | Serves the web pages and processes PHP requests. |

# **3.2 High-Level System Architecture**

The system adheres to a multi-layered architecture pattern which separates concerns for better maintainability.

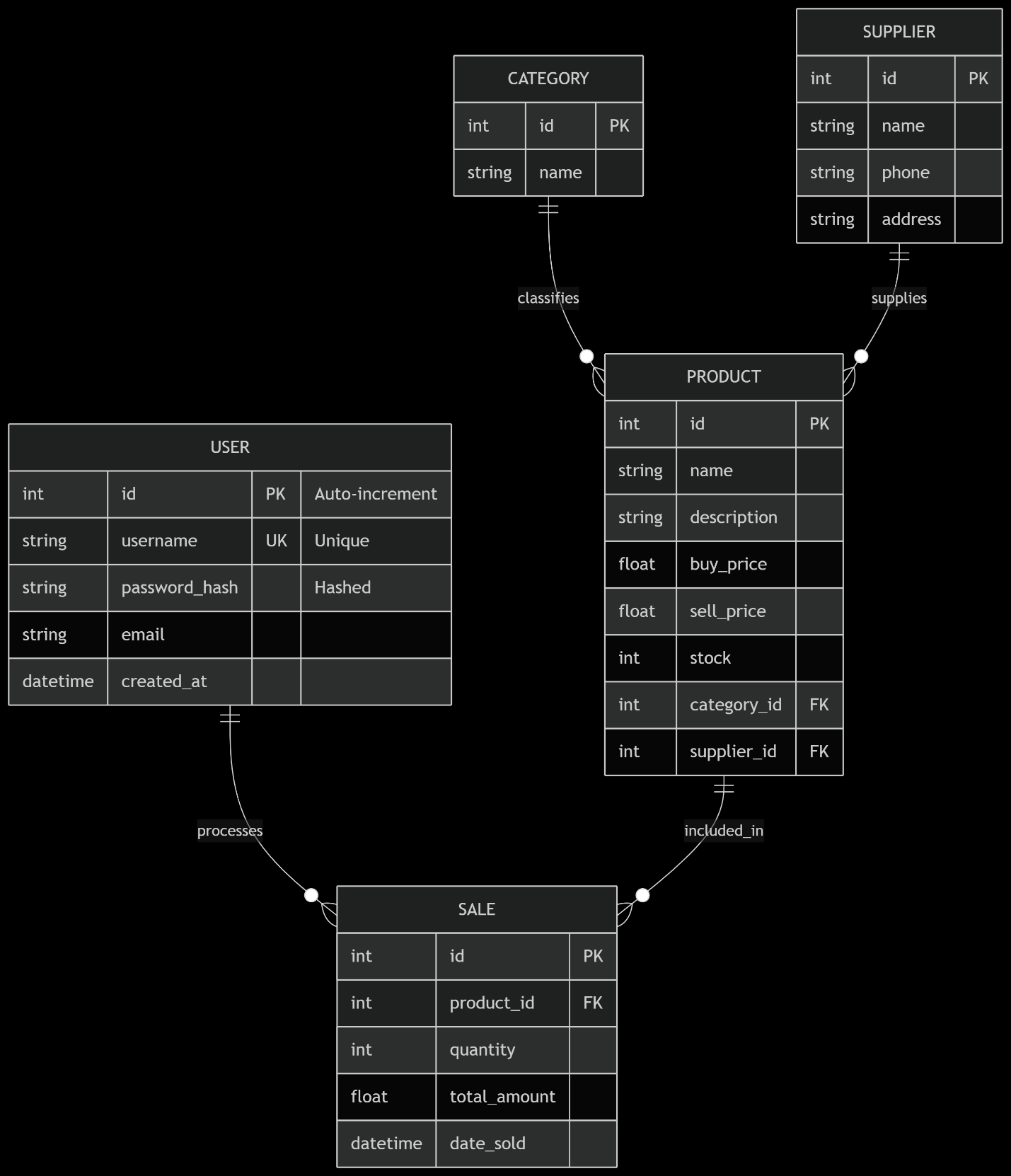


# **3.3 Use Case Model**



# **4.0 Database Design**

# **4.1 Entity-Relationship (ER) Diagram**



# **4.2 Detailed Database Schema**

· users: id (INT, PK, AI), username (VARCHAR(50), UNIQUE), password\_hash (VARCHAR(255)), email (VARCHAR(100)), created\_at (DATETIME)

· categories: id (INT, PK, AI), name (VARCHAR(100))

· suppliers: id (INT, PK, AI), name (VARCHAR(100)), phone (VARCHAR(20)), address (TEXT)

· products: id (INT, PK, AI), name (VARCHAR(100)), description (TEXT), category\_id (INT, FK), supplier\_id (INT, FK), buy\_price (DECIMAL(10,2)), sell\_price (DECIMAL(10,2)), stock (INT)

· sales: id (INT, PK, AI), product\_id (INT, FK), quantity (INT), total\_amount (DECIMAL(10,2)), date\_sold (DATETIME)

# **5.0 Functional Requirements Specifications**

(Detailed descriptions of each module's purpose, functions, and fields would be here.)

# **5.10 Note on Missing Deployment Functionality**

As identified in Section 1.3, a significant functional gap is the lack of a built-in deployment module. The system does not possess an automated setup wizard or an administrative interface to initialize the database (create tables, insert initial data) on a new server. This task must be performed manually by a technically proficient individual:

1. Creating the database via a tool like phpMyAdmin.

2. Executing the provided SQL schema script to create the tables.

3. Manually configuring the database connection parameters in the config.php file. This omission increases the technical barrier for initial setup and is a key candidate for a future enhancement.

# **6.0 Dynamic Modeling: Sequence Diagrams**

# **6.3 Execute a Sales Transaction Process**

Description: This sequence diagram captures the critical, multi-step interaction between actors and the system to finalize a sale, ensuring data consistency.

mermaid

sequenceDiagram

actor Clerk as Sales Clerk

participant Browser

participant Server as PHP Controller

participant DB as MySQL Database

Clerk->>Browser: 1. Initiates Checkout

Browser->>Server: 2. POST Request: process\_sale.php<br/>(JSON cart data)

Note over Server: Begin Transaction

loop For Each Cart Item

Server->>DB: 3. SQL: SELECT stock FROM products WHERE id = ?

DB-->>Server: 4. Current Stock Level

alt Stock >= Requested Quantity

Server->>DB: 5. SQL: UPDATE products SET stock = stock - ? WHERE id = ?

Server->>DB: 6. SQL: INSERT INTO sales (...)

else Insufficient Stock

Note over Server: Rollback Transaction

Server-->>Browser: 7. Error: Insufficient stock for [Product Name]

Browser-->>Clerk: 8. Display Error Message

break

end

end

Note over Server: Commit Transaction

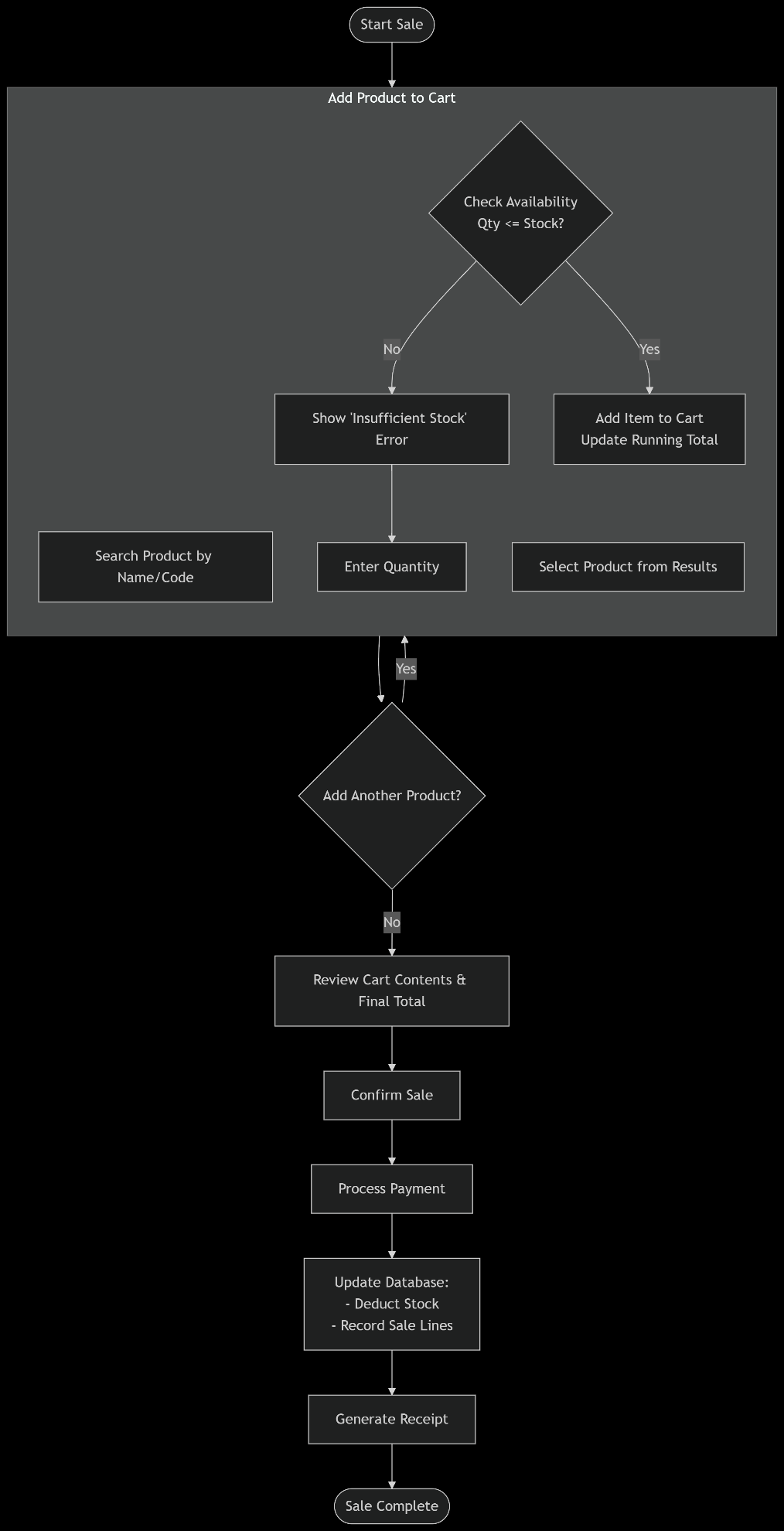
Server-->>Browser: 9. HTTP 200: Success & Receipt Data

Browser-->>Clerk: 10. Display Sale Confirmation & Receipt

# **7.0 Behavioral Modeling: Activity Diagrams**

# **7.2 Process a Sale Activity Diagram**

Description: This diagram illustrates the flow of activities, decisions, and user interactions involved in processing a single sales transaction from start to finish.



# **8.0 User Interface Design**

# **8.1 Design Philosophy & Guidelines**

The interface is built following a sidebar navigation layout, a standard and intuitive pattern for web applications.

· Consistency: The same header and sidebar are present on all pages, providing a stable navigation frame.

· Clarity: Forms are clearly labeled. Buttons are designed with descriptive text and appropriate colors (e.g., green for save, blue for neutral, red for delete).

· Feedback: Users receive immediate feedback for their actions through toast notifications or alert messages (e.g., "Product added successfully," "Sale completed.").

· Efficiency: The POS interface is designed for speed, with a prominent search bar and a minimal-click workflow to add products and complete the sale.

# **9.0 Installation, Deployment & Maintenance**

# **9.1 System Requirements**

· Server: A computer running Windows, Linux, or macOS.

· Software: Pre-installed XAMPP or WAMP stack, which includes:

· Apache 2.4+

· PHP 7.4+

· MySQL 5.7+ or MariaDB

· Browser: On client machines, any modern browser (Chrome, Firefox, Edge, Safari).

9.2 Manual Installation Guide

1. Install Prerequisites: Download and install XAMPP. Start the Apache and MySQL modules from the control panel.

2. Deploy Application Files: Extract the contents of the project ZIP file into the htdocs directory (e.g., C:\xampp\htdocs\pharmacy-system).

3. Create Database: Open a browser and go to http://localhost/phpmyadmin. Create a new database named pharmacy\_db.

4. Import Schema: Select the new database, go to the "Import" tab, and choose the provided database\_schema.sql file to create all necessary tables.

5. Configure Connection: Edit the config.php file in the project's includes/ folder. Update the database host, name, username, and password to match the local MySQL configuration.

6. Access System: Open a browser and navigate to http://localhost/pharmacy-system.

7. Login: Use the default administrator credentials (supplied in a separate readme file) to log in.

# **10.0 Conclusion & Future Enhancements**

The Web-based Pharmacy Product Management System is a robust, self-contained solution that effectively addresses the core operational needs of a pharmacy. Its integrated design ensures data integrity and provides tangible efficiency gains. The most significant identified gap is the lack of an automated deployment module, which is recommended for the next development cycle.

Proposed Future Enhancements:

1. Automated Deployment Setup Wizard: A web-based installer to eliminate the manual database setup process.

2. Barcode/QR Code Support: For rapid product identification during sales and restocking.

3. Expiry Date Management: Track product expiry dates and generate alerts for soon-to-expire stock.

4. Supplier Management Portal: Allow suppliers to view inventory levels and receive electronic orders.

5. Advanced Reporting Engine: Include trend analysis, profit margin reports, and forecasting.

# **11.0 References**

1. Ambler, S. W., & Sadalage, P. J. (2006). Refactoring Databases: Evolutionary Database Design. Addison-Wesley Professional.

2. Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd ed.). Addison-Wesley.

3. George, B., & Williams, L. (2004). A structured experiment of test-driven development. Information and Software Technology, 46(5), 337-342.

4. JavaScript MDN Web Docs. (2023). JavaScript reference. Mozilla Developer Network. Retrieved from https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

5. MySQL AB. (2023). MySQL 8.0 Reference Manual. Oracle Corporation. Retrieved from https://dev.mysql.com/doc/refman/8.0/en/

6. PHP Documentation Group. (2023). PHP Manual. The PHP Group. Retrieved from https://www.php.net/manual/en/

7. Sommerville, I. (2016). Software Engineering (10th ed.). Pearson.

8. W3Schools. (2023). HTML, CSS, and Bootstrap Tutorials. Retrieved from https://www.w3schools.com/

9. Whitten, J. L., & Bentley, L. D. (2007). Systems Analysis and Design Methods (7th ed.). McGraw-Hill/Irwin