Pharmacy Portal System Report

1. Introduction

The Pharmacy Portal System is a web-based application designed to manage pharmaceutical operations, focusing on prescription handling, inventory, and sales. The system supports two user roles: pharmacists and patients, each with specific functionalities. The system is built using PHP for server-side logic, HTML and CSS for the front-end, and MySQL for the database.

2. System Architecture

The system follows a three-tier architecture:

- **Presentation Tier:** This layer consists of HTML and CSS files. It provides the user interface for interacting with the application.
- **Application Tier:** This layer is built using PHP. It handles the business logic, processes user requests, interacts with the database, and generates the appropriate responses.
- **Data Tier:** This layer consists of a MySQL database. It stores all the data related to users, medications, prescriptions, inventory, and sales.

3. Database Design

The database schema consists of the following tables:

- **Users:** Stores user information, including user ID, username, contact information, user type (pharmacist or patient), and password.
- **Medications:** Stores medication details, including medication ID, name, dosage, manufacturer, and price.
- **Prescriptions:** Stores prescription information, including prescription ID, user ID, medication ID, prescribed date, dosage instructions, quantity, and refill count.
- **Inventory:** Stores inventory details, including inventory ID, medication ID, quantity available, and last updated timestamp.
- **Sales:** Stores sales transaction information, including sale ID, prescription ID, sale date, quantity sold, and sale amount.

4. Functional Components

- **User Authentication:** The system provides a login functionality for both pharmacists and patients.
- **Patient Dashboard:** Patients can view their contact information and prescriptions.

- **Prescription Management:** Pharmacists can create and manage prescriptions for patients.
- **Inventory Management:** The system tracks the inventory of medications and updates it after sales or new stock arrivals.
- **Sales Processing:** The system allows processing sales transactions, updating inventory, and recording sale details.
- **Reports and Views:** The system provides views and stored procedures to generate reports, such as medication inventory.

5. Key Features and Functionalities

- **Login and Authentication:**
    - Users (pharmacists and patients) can log in to the system using their credentials.
    - The system validates the credentials against the data stored in the Users table.
    - Upon successful login, the system creates a session and redirects the user to their respective dashboard.
- **Patient Dashboard:**
    - Patients can view their profile information, including contact details.
    - Patients can view a list of their prescriptions, including medication details, dosage instructions, quantity, and refills.
    - Prescription Management
    - Pharmacists can create new prescriptions for patients.
    - Prescriptions are stored in the Prescriptions table, linked to the Users and Medications tables.
- **Inventory Management**
    - The Inventory table tracks the quantity of each medication in stock.
    - The system updates the inventory when a sale is made (using the ProcessSale stored procedure and AfterPrescriptionInsert trigger).
- **Sales Processing**
    - The ProcessSale stored procedure handles the sale of medications. It checks for sufficient stock, updates the Inventory table, and records the sale in the Sales table.

    - The saleAmount is calculated based on the quantity sold and the price of the medication.

- **Data Display**
    - The system retrieves data from the database and displays it on the web pages.
    - For example, the patient dashboard retrieves prescription information from the Prescriptions table and displays it in a user-friendly format.

- **Logout:**
    - Users can log out of the system, which ends their **session.**

6. Code Explanation (PHP)

- **PharmacyDatabase.php:** This file contains the PharmacyDatabase class, which encapsulates the database connection and provides methods for performing database operations.
    - **getConnection():** Establishes and returns a database connection.
    - **addPrescription():** Retrieves new prescription details to add the the prescriptions table in the database**.**
    - **getAllPrescriptions():** Retrieves all of the prescriptions details from the database.
    - **MedicationInventory():** Retrieves medication inventory data from a database and return it in a structured format.
    - **addUser():** Adds new users into the Users table in the database.
    - **verifyUserLogin():** Checks if the user's loging information is in the database and is created.
    - **addMedication():** Retrieves input to add to the Medication table in the database. It also validates the information before adding to the database.
    - **getUserDetails():** Retrieves user details (including prescriptions) from the database.
    - **getAllMedication():** Retrieves the data from the Medication table.
    - **addToInventory():** Retrieves input to add to the Inventory table.
    - **getPricePerBottle():** Calculates and return the price of one bottle.
    - **processSale():** Processes a sale transaction (updates inventory and creates a sale record).

- **PharamacyServer.php:** This file acts as a central controller for the Pharmacy Portal web application.
    - It uses the PharmacyDatabase class to interact with the database.
    - It checks the action parameter in the URL (e.g., ?action=viewInventory) to determine which function to run.
    - It supports multiple actions like adding/viewing prescriptions and inventory.
- **login.php:** This file handles user login.
    - It retrieves the username and password from the login form.
    - It queries the database to validate the credentials.
    - If the credentials are valid, it sets the session variables and redirects the user to the appropriate page (patient or pharmacist dashboard).
- **register.php:** This file adds new users into the database.
    - It receives the user details then calls the addUser() function to add the user into the database.
    - The function handleRegistration() checks if all of the necessary information is filled out before it calls the addUser() function.
- **patient_home.php:** This file displays the patient dashboard.
    - It retrieves the patient's details and prescriptions from the database.
    - It displays the information in an HTML format.
- **sales.php:** This file handles the processing of a sale.
    - It retrieves the prescription ID from the URL.
    - It displays the prescription details, including medication name, dosage, quantity, refills, and price.
    - It allows the user to enter the quantity of medication to buy.
    - Upon form submission, it validates the quantity, calculates the total price, and calls the processSale() method from the PharmacyDatabase class.
    - It displays a success or error message.
- **home.php:** This file displays the pharmacist dashboard.
    - It gives the pharmacist access to the neccessary files.

- **addPrescription.php:** This file allows pharmacists to add new prescriptions to patients through a form.
    - Collects and validates user input.
    - Calls the addPrescription() function from the PharmacyDatabase class.
    - Display success or error messages, depending if it was added to the database.
    - When the inventory of the medication prescribed is less than 10 the file won't run and the form won't display after slicking save.
- **viewPrescription.php**: This file allows pharmacists to view all the prescriptions that were given to which patient.
    - Calls the getAllPrescriptions() function from the PharamacyDatabase class to retrieve all of the prescriptions.
    - Displays all of the data given from the getAllPrescriptions() function.
- **addMedication.php**: This file allows logged in pharmacists to add new medication to the pharmacy database through a form.
    - It checks if the pharmacist is logged into their account, if not their taken to the login.php file.
    - It verifies that the entire form was filled out and is in the correct fomat.
    - It calls the addMedication() function from the PharmacyDatabase class to add the data into the Medication table.
- **addInventory.php:** This file allows logged in pharmacists to add inventory to the Inventory table in the database through a form.
    - It checks if the pharmacist is logged into their account, if not their taken to the login.php file.
    - From the PharmacyDatabase class, it calls the getAllMedication() function to retrieve all the medication names in the database.
    - Then it checks if the input given is a numeric value or a positive value.
    - If it comes out as true, the program will call the addToInventory() function to add the input into the inventory.
    - If successful a message will appear, as well as the form.
- **viewInventory.php:** This file allows logged in pharmacists to check view the inventory

- Like the other files, it first checks if the user is logged in, if not it will take them to the login.php file
- It calls the PharmacyDatabase class to use the MedicationInventory() function to retreieve data from the MedicationInventoryView from the database to get the necessary data.
- It shows the pharmacist what medication is in the inventory and how many tablets are available.

- **logout.php:** This file handles user logout. It ends the current session and redirects the user to the login page.

7. Code Explanation (SQL)

- **Database Creation and Table Setup:**
  - The SQL script creates the database pharmacy_portal_db and defines the tables: Users, Medications, Prescriptions, Inventory, and Sales.
  - The tables are designed to store information about users, medications, prescriptions, inventory levels, and sales transactions.
  - Appropriate data types, constraints (e.g., NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY), and relationships are defined to ensure data integrity and consistency.
- **Stored Procedures:**
  - AddOrUpdateUser: This stored procedure either adds a new user or updates an existing user's information.
  - ProcessSale: This stored procedure processes a sale transaction. It takes the prescriptionID and quantitySold as input. It checks if there is enough stock in the inventory, updates the inventory, and inserts a new record into the Sales table.
- **View:**
  - MedicationInventoryView: This view combines data from the Medications and Inventory tables to provide a consolidated view of medication information and their corresponding inventory levels.
- **Trigger:**
  - AfterPrescriptionInsert: This trigger is executed after a new prescription is inserted into the Prescriptions table. It updates the Inventory table to reflect the reduction in stock due to the new prescription. It also checks if the stock level is below 10 and raises an error if it is.

- **Data Population:**
    - The SQL script includes INSERT statements to populate the tables with initial data for users, medications, prescriptions, inventory, and sales. This provides a basic set of data for testing and demonstration purposes.

8. How to Run the Code

1. **Set up the Database:**
    1. Install MySQL on your system.
    2. Import the pharmarcy_portal_db.sql script into the phpMyAdmin to create the database.
2. **Set up the Web Server:**
    1. Place the PHP files (e.g., login.php, patient_home.php, process_sale.php, PharmacyDatabase.php) in the web server's document root directory (e.g., htdocs for XXAMP).
3. **Configure the Database Connection:**
    1. Open the PharmacyDatabase.php file and modify the database connection parameters (host, username, password, database name) to match your MySQL setup.
4. **Access the Application:**
    1. Open a web browser and navigate to the URL where you have placed the PHP files (e.g., http://localhost/login.php).
    2. You should be able to see the login page.
5. **Login and Use the Application:**
    1. Resister as a new user to test the applications.
    2. You can also use an existing account:
        1. Data Population
            1. UserName: jane_doe, Password: 123, UserType: patient
            2. UserName: alice_brown, Password: 123, UserType: patient
            3. UserName: john_smith, Password: 123, UserType: pharmacist

9. Improvements and Future Work

- **Security:**

    - Implement password hashing (e.g., using bcrypt) to store passwords securely in the database.
    - Implement input validation and sanitization to prevent SQL injection and other security vulnerabilities.
    - Use prepared statements for all database queries.
- **User Interface:**
    - Improve the user interface using HTML and CSS frameworks to make it more modern and user-friendly.
- **Additional Features:**
    - Implement more advanced features, such as:
        - Pharmacist dashboard with more functionalities.
        - Ability for pharmacists to manage medications, inventory, and prescriptions.
        - Reporting features to generate sales reports, inventory reports, and other relevant data.
        - Ability for patients to request prescription refills online.
        - Email notifications for low stock or prescription updates.
- **Error Handling:**
    - Implement more robust error handling to gracefully handle database errors, invalid user input, and other potential issues.
- **Testing:**
    - Implement unit tests and integration tests to ensure the application's functionality and stability.

10. Conclusion

The Pharmacy Portal System provides a basic framework for managing pharmaceutical operations. It includes essential features for user authentication, prescription management, inventory control, and sales processing.