1. **Introduction:**

The Pharmacy Management System is designed to streamline the management of patients, prescriptions, medications, and user accounts within a pharmacy setting. The system provides functionalities such as user registration, login, patient management, prescription management, medication management, and reporting.

1. **User Management Module (UserManagement.h):**

The user management module allows for user signup and login. Users can create an account by providing a username, password, and role. Roles can include pharmacist, admin, or other relevant roles. Once registered, users can log in to access the main functionalities of the system.

* + **struct User Definition:**
    - Defines a structure to store user information, including **userID**, **username**, **password**, and **role**.
  + **void userSignUp(vector<User>& users, const string& username, const string& password, const string& role) Function:**
    - **Arguments:**
      * **vector<User>& users**: Reference to the vector containing user information.
      * **const string& username**: Username entered during user registration.
      * **const string& password**: Password entered during user registration.
      * **const string& role**: Role assigned to the user during registration.
    - **Functionality:**
      * Checks for the uniqueness of the entered username.
      * Adds a new user to the vector if the username is unique.
      * Displays a message indicating the success or failure of the registration.
  + **bool userLogin(const vector<User>& users, const string& username, const string& password) Function:**
    - **Arguments:**
      * **const vector<User>& users**: Reference to the vector containing user information.
      * **const string& username**: Username entered during user login.
      * **const string& password**: Password entered during user login.
    - **Returns:**
      * **bool**: Returns **true** if login is successful, **false** otherwise.
    - **Functionality:**
      * Checks if the entered username and password match any user's credentials.
      * Returns the login status.

1. **Patient Management Module (PatientManagement.h):**

This module enables the management of patient information. It includes the ability to add, update, remove, and search for patients. Each patient is identified by a unique ID, and additional details such as first name, last name, phone number, and notes are stored.

* + **struct Patient Definition:**
    - Defines a structure to store patient information, including **patientID**, **Fname**, **Lname**, **contactDetails**, and **medicalHistory**.
  + **void addPatient(vector<Patient>& patients, int id, const string& Fname, const string& Lname, const string& contactDetails, const string& medicalHistory) Function:**
    - **Arguments:**
      * **vector<Patient>& patients**: Reference to the vector containing patient information.
      * **int id**: Patient ID entered during patient addition.
      * **const string& Fname**: First name of the patient.
      * **const string& Lname**: Last name of the patient.
      * **const string& contactDetails**: Contact details of the patient.
      * **const string& medicalHistory**: Medical history of the patient.
    - **Functionality:**
      * Adds a new patient to the vector with the provided information.
      * Displays a message indicating the success of the patient addition.
  + **void updatePatient(vector<Patient>& patients, int patientID, const string& updatedFirstName, const string& updatedLastName, const string& updatedContact, const string& updatedDetails) Function:**
    - **Arguments:**
      * **vector<Patient>& patients**: Reference to the vector containing patient information.
      * **int patientID**: ID of the patient to be updated.
      * **const string& updatedFirstName**: Updated first name of the patient.
      * **const string& updatedLastName**: Updated last name of the patient.
      * **const string& updatedContact**: Updated contact details of the patient.
      * **const string& updatedDetails**: Updated medical history/details of the patient.
    - **Functionality:**
      * Updates the information of an existing patient in the vector.
      * Displays a message indicating the success or failure of the update.
  + **void removePatient(vector<Patient>& patients, int patientID) Function:**
    - **Arguments:**
      * **vector<Patient>& patients**: Reference to the vector containing patient information.
      * **int patientID**: ID of the patient to be removed.
    - **Functionality:**
      * Removes a patient from the vector based on the provided patient ID.
      * Displays a message indicating the success or failure of the removal.
  + **void searchPatient(vector<Patient>& patients, int patientID) Function:**
    - **Arguments:**
      * **vector<Patient>& patients**: Reference to the vector containing patient information.
      * **int patientID**: ID of the patient to be searched.
    - **Functionality:**
      * Searches for a patient with the provided ID in the vector.
      * Displays the patient's information if found; otherwise, indicates that the patient was not found.
  + **bool patientExists(const vector<Patient>& patients, int patientID) Function:**
    - **Arguments:**
      * **const vector<Patient>& patients**: Reference to the vector containing patient information.
      * **int patientID**: ID of the patient to be checked for existence.
    - **Returns:**
      * **bool**: Returns **true** if a patient with the given ID exists, **false** otherwise.

1. **Prescription Management Module (PrescriptionManagement.h):**

Prescription management is closely tied to patient management. It facilitates the creation, updating, and searching of prescriptions associated with specific patients. Users can input the prescription date and details for efficient tracking of medications prescribed to patients.

* + **struct Prescription Definition:**
    - Defines a structure to store prescription information, including **prescriptionID**, **patientID**, **dateIssued**, and **medicationDetails**.
  + **void createPrescription(vector<Prescription>& prescriptions, int patientID, const string& dateIssued, const string& medicationDetails) Function:**
    - **Arguments:**
      * **vector<Prescription>& prescriptions**: Reference to the vector containing prescription information.
      * **int patientID**: ID of the patient for whom the prescription is created.
      * **const string& dateIssued**: Date on which the prescription is issued.
      * **const string& medicationDetails**: Details of the prescribed medication.
    - **Functionality:**
      * Creates a new prescription and adds it to the vector with the provided information.
      * Displays a message indicating the success of the prescription creation.
  + **void updatePrescription(vector<Prescription>& prescriptions, int prescriptionID, const string& updatedDetails) Function:**
    - **Arguments:**
      * **vector<Prescription>& prescriptions**: Reference to the vector containing prescription information.
      * **int prescriptionID**: ID of the prescription to be updated.
      * **const string& updatedDetails**: Updated details of the prescription.
    - **Functionality:**
      * Updates the details of an existing prescription in the vector.
      * Displays a message indicating the success or failure of the update.
  + **void searchPrescription(vector<Prescription>& prescriptions, int patientID) Function:**
    - **Arguments:**
      * **vector<Prescription>& prescriptions**: Reference to the vector containing prescription information.
      * **int patientID**: ID of the patient for whom prescriptions are to be searched.
    - **Functionality:**
      * Searches for prescriptions associated with the provided patient ID.
      * Displays the prescription information if found; otherwise, indicates that no prescriptions were found.
  + **bool prescriptionExists(const vector<Prescription>& prescriptions, int prescriptionID) Function:**
    - **Arguments:**
      * **const vector<Prescription>& prescriptions**: Reference to the vector containing prescription information.
      * **int prescriptionID**: ID of the prescription to be checked for existence.
    - **Returns:**
      * **bool**: Returns **true** if a prescription with the given ID exists, **false** otherwise.

1. **Medication Management Module (MedicationManagement.h):**

This module deals with the management of medication stock. It allows users to add new medications, update existing medication stock, remove medications, and search for specific medications. Key information such as medication ID, name, quantity, issue date, and expiry date is recorded.

* + **struct Medication Definition:**
    - Defines a structure to store medication information, including **medicationID**, **medicationName**, **quantity**, **issueDate**, and **expiryDate**.
  + **void addMedication(vector<Medication>& medications, int id, const string& medicationName, int quantity, const string& issueDate, const string& expiryDate) Function:**
    - **Arguments:**
      * **vector<Medication>& medications**: Reference to the vector containing medication information.
      * **int id**: Medication ID entered during medication addition.
      * **const string& medicationName**: Name of the medication.
      * **int quantity**: Quantity of the medication.
      * **const string& issueDate**: Date on which the medication is issued.
      * **const string& expiryDate**: Expiry date of the medication.
    - **Functionality:**
      * Adds a new medication to the vector with the provided information.
      * Displays a message indicating the success of the medication addition.
  + **void updateMedicationStock(vector<Medication>& medications, int medicationID, int newQuantity, const string& newName, const string& newIssueDate, const string& newExpiryDate) Function:**
    - **Arguments:**
      * **vector<Medication>& medications**: Reference to the vector containing medication information.
      * **int medicationID**: ID of the medication to be updated.
      * **int newQuantity**: Updated quantity of the medication.
      * **const string& newName**: Updated name of the medication.
      * **const string& newIssueDate**: Updated issue date of the medication.
      * **const string& newExpiryDate**: Updated expiry date of the medication.
    - **Functionality:**
      * Updates the stock details of an existing medication in the vector.
      * Displays a message indicating the success or failure of the update.
  + **void removeMedication(vector<Medication>& medications, int medicationID) Function:**
    - **Arguments:**
      * **vector<Medication>& medications**: Reference to the vector containing medication information.
      * **int medicationID**: ID of the medication to be removed.
    - **Functionality:**
      * Removes a medication from the vector based on the provided medication ID.
      * Displays a message indicating the success or failure of the removal.
  + **void searchMedication(vector<Medication>& medications, int medicationID) Function:**
    - **Arguments:**
      * **vector<Medication>& medications**: Reference to the vector containing medication information.
      * **int medicationID**: ID of the medication to be searched.
    - **Functionality:**
      * Searches for a medication with the provided ID in the vector.
      * Displays the medication's information if found; otherwise, indicates that the medication was not found.
  + **bool medicationExists(const vector<Medication>& medications, int medicationID) Function:**
    - **Arguments:**
      * **const vector<Medication>& medications**: Reference to the vector containing medication information.
      * **int medicationID**: ID of the medication to be checked for existence.
    - **Returns:**
      * **bool**: Returns **true** if a medication with the given ID exists, **false** otherwise.

1. **Reporting Module (Reporting.h):**

The reporting module provides insights into the pharmacy's inventory and patient information. Users can generate inventory reports and patient reports to assist in decision-making and analysis.

* + **void generateInventoryReport(const vector<Medication>& medications) Function:**
    - **Arguments:**
      * **const vector<Medication>& medications**: Reference to the vector containing medication information.
    - **Functionality:**
      * Generates an inventory report in a text file (**InventoryReport.txt**).
      * Displays medication details such as ID, name, quantity, issue date, and expiry date.
  + **void generatePatientReport(const vector<Patient>& patients) Function:**
    - **Arguments:**
      * **const vector<Patient>& patients**: Reference to the vector containing patient information.
    - **Functionality:**
      * Generates a patient report in a text file (**PatientReport.txt**).
      * Displays patient details such as ID, first name, last name, contact details, and medical history.

1. **Conclusion:**

The Pharmacy Management System provides a comprehensive set of functionalities for managing user accounts, patients, prescriptions, medications, and generating reports. Users can sign up, log in, add, update, and remove patients, prescriptions, and medications. The reporting module facilitates the generation of inventory and patient reports, offering valuable insights into the pharmacy's operations. The modular structure allows for easy extension and maintenance of the system making it a robust solution for pharmacies looking to enhance their management processes.