# Introduction:

The Pharmacy Management System takes a unique approach to streamlining pharmacy operations, emphasizing efficiency, accuracy, and scalability. This paper describes the system's core features, aims, and reasons for data structure selection.

# ADT Specification:

The Pharmacy Management System employs multiple Abstract Data Types (ADTs) to organize and manage data effectively. The final ADT specifications are as follows:

## User ADT (UserManagement.h):

**Attributes**: int userID, string username, string password, string role

**Functions**:

void userSignUp(vector<User>& users, const string& username, const string& password, const string& role)

bool userLogin(const vector<User>& users, const string& username, const string& password)

## Patient ADT (PatientManagement.h):

**Attributes**: int patientID, string Fname, string Lname, string contactDetails, string medicalHistory

**Functions**:

void addPatient(vector<Patient>& patients, int id, const string& Fname, const string& Lname, const string& contactDetails, const string& medicalHistory)

void updatePatient(vector<Patient>& patients, int patientID, const string& updatedFirstName, const string& updatedLastName, const string& updatedContact, const string& updatedDetails)

void removePatient(vector<Patient>& patients, int patientID)

void searchPatient(vector<Patient>& patients, int patientID)

bool patientExists(const vector<Patient>& patients, int patientID)

## Prescription ADT (PrescriptionManagement.h):

**Attributes**: int prescriptionID, int patientID, string dateIssued, string medicationDetails

**Functions**:

void createPrescription(vector<Prescription>& prescriptions, int patientID, const string& dateIssued, const string& medicationDetails)

void updatePrescription(vector<Prescription>& prescriptions, int prescriptionID, const string& updatedDetails)

void searchPrescription(vector<Prescription>& prescriptions, int patientID)

bool prescriptionExists(const vector<Prescription>& prescriptions, int prescriptionID)

## Medication ADT (MedicationManagement.h):

**Attributes**: int medicationID, string medicationName, int quantity, string issueDate, string expiryDate

**Functions**:

void addMedication(vector<Medication>& medications, int id, const string& medicationName, int quantity, const string& issueDate, const string& expiryDate)

void updateMedicationStock(vector<Medication>& medications, int medicationID, int newQuantity, const string& newName, const string& newIssueDate, const string& newExpiryDate)

void removeMedication(vector<Medication>& medications, int medicationID)

void searchMedication(vector<Medication>& medications, int medicationID)

bool medicationExists(const vector<Medication>& medications, int medicationID)

## Reporting ADT (Reporting.h):

**Functions**:

void generateInventoryReport(const vector<Medication>& medications)

void generatePatientReport(const vector<Patient>& patients)

# Implementation Details:

Justification for Data Structures and Algorithms is as follows:

## Data Structures:

**Stack** (for Prescription Management): Utilized to manage prescriptions in a Last-In-First-Out (LIFO) manner, reflecting real-world prescription handling.

**Queue** (for Reporting): Used to generate reports in a First-In-First-Out (FIFO) order, ensuring a chronological sequence in reporting.

**Linked List** (for Patient Management): Offers dynamic memory allocation and efficient insertion/deletion, crucial for patient data management.

**Vectors** (for User and Medication Management): Chosen for dynamic storage, easy traversal, and adaptability to varying data sizes.

Algorithms:

**Binary Search** (for Medication Search): Implemented for its efficiency on sorted data, which results in a quicker search than linear search. Because pharmaceutical data is intended to be sorted, binary search is appropriate.

**Linear Search** (for Patient and Medication Existence Check): Chosen for simplicity and efficiency in small-scale searches with manageable data sizes. It guarantees that patients and medications are accurately identified.

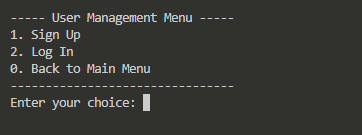
These decisions are made to improve data processing in accordance with each module's individual requirements, taking into account aspects such as data quantity, search efficiency, and real-world relevancy. The integration of several data formats and algorithms results in a flexible and effective Pharmacy Management System.

Conclusion:

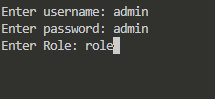
The Pharmacy Management System provides a comprehensive set of functionalities for managing user accounts, patients, prescriptions, medications, and generating reports. Users can sign up, log in, add, update, and remove patients, prescriptions, and medications. The reporting module facilitates the generation 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.

# PROGRAM SCREENSHOTS

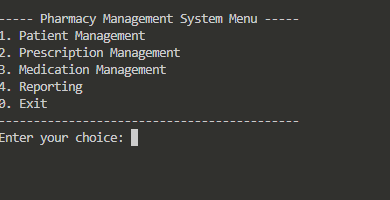
1. **Default Interface**



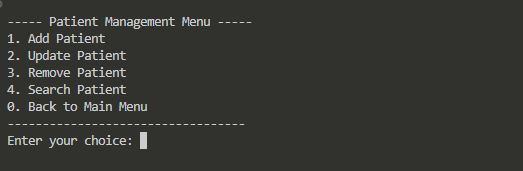
1. **signup inputs**



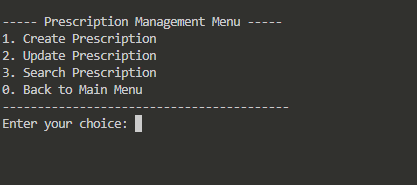
1. **Main Dashboard**



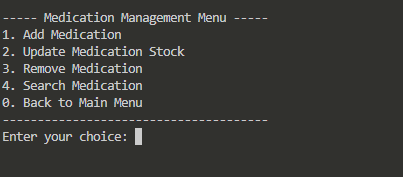
1. **Patient Management Menu**



1. **Prescription Management Menu**



1. **Medication Management Menu**



1. **Report Output Menu**

