PHARMACY MANAGEMENT SYSTEM

Mid-Term Sprint-Semester 3

Walid Jerjawi

Table of Contents

Table of Figures

User Documentation for Pharmacy Management System

1. Overview of the Application

Introduction

The Pharmacy Management System is a Java-based application that provides a comprehensive solution for managing a pharmacy's daily operations. It is designed to improve the efficiency of handling patients, doctors, medications, and prescriptions, reducing manual workload and ensuring accurate record-keeping.

The system enables pharmacists and healthcare providers to:

- Efficiently store and manage patient, doctor, and medication records.
- Process and track prescriptions issued by doctors to patients.
- Monitor medication stock levels and handle restocking when needed.
- Check for expired medications and take necessary action.
- Generate detailed reports to provide insights into the pharmacy's operations.

Key Features & Functionalities

The system provides ten core functionalities, each ensuring seamless pharmacy management:

1- Managing Patients, Doctors, and Medications

- Add, edit, search, and remove patients, doctors, and medications in the system.
- Each patient and doctor are uniquely identified by an ID.
- Medications have attributes such as dosage, stock quantity, and expiry date.

2- Processing and Tracking Prescriptions

- Doctors can issue prescriptions for patients.
- Each prescription contains a doctor, patient, medication, issue date, and expiry date (1 year from issue date).
- Prescriptions are linked to patients for easy tracking.

3- Checking for Expired Medications

- The system automatically generates a random expiry date for medications upon entry if not manually provided.
- Medications past their expiry date are flagged as expired, and a warning message is displayed.
- Pharmacists can take action to remove expired medications or notify suppliers.

4- Generating Reports on Patients, Doctors, and Prescriptions

- Users can generate reports summarizing all patients, doctors, and medications stored in the system.
- Reports include detailed structured tables with IDs, names, specializations, and status.
- Additional reports allow checking expired medications and listing all prescriptions issued by a specific doctor or assigned to a patient.

5- Restocking Medications in the Pharmacy

- Medications running low can be restocked automatically by a random quantity between 5 and 20 units.
- The pharmacist can also manually update stock levels.

2. Explanation of Classes & Their Functionality

1. Person (Super Class)

The Person class serves as the **base class** for both Patient and Doctor. It contains common attributes shared by both subclasses, ensuring code reusability.

Attributes:

- **id :** A String representing the unique identifier of the person.
- **name:** A String representing the full name of the person.
- **Age:** An int representing the person's age.
- **phoneNum :** A String representing the contact number of the person.

Methods:

- **getID():** Returns the person's ID.
- **getName():** Returns the person's name.
- **getAge():** Returns the person's age.
- **getPhoneNum():** Returns the person's phone number.
- **setID(String id):** Updates the person's ID.
- **setName(String name):** Updates the person's name.
- **setAge(int age) :** Updates the person's age.
- **setPhoneNum(String phoneNum):** Updates the person's phone number.
- **toString():** Returns a formatted string representation of the person.

2. Patient (Extends Person)

The Patient class represents a patient in the pharmacy system. Each patient has a list of medications and prescriptions.

Attributes:

- **medications:** A List<Medication> representing all medications prescribed to the patient.
- **Prescriptions:** A List<Prescription> representing all prescriptions issued for the patient.

Methods:

- **getMedications():** Returns the list of medications assigned to the patient.
- **getPrescriptions():** Returns the list of prescriptions assigned to the patient.
- **addMedication(Medication medication):** Adds medication to the patient's list. Prevents duplicates.
- **addPrescription(Prescription prescription):** Adds a prescription to the patient's records.
- **removeMedication(Medication medication):** Removes a medication from the patient's list.
- **removePrescription(Prescription prescription):** Removes a prescription from the patient's records.
- **toString():** Returns a formatted string representation of the patient.

3. Doctor (Extends Person)

The Doctor class represents a medical professional who manages patients and issues prescriptions.

Attributes:

- **specialization:** A String representing the doctor's field of expertise.
- **Patients:** A List<Patient> representing all patients assigned to the doctor.

Methods:

- **getSpecialization():** Returns the doctor's specialization.
- **getPatients():** Returns the list of patients under the doctor's care.
- **setSpecialization(String specialization):** Updates the doctor's specialization.
- **addPatient(Patient patient):** Assigns a patient to the doctor.
- **removePatient(Patient patient):** Removes a patient from the doctor's list.
- **toString():** Returns a formatted string representation of the doctor.

4. Medication

The Medication class represents a medicine available in the pharmacy. It contains details about dosage, stock, and expiry date.

Attributes:

- **medID:** A String representing the unique medication identifier.
- **medName:** A String representing the name of the medication.
- **Dose:** A String indicating the dosage (e.g., "500mg").
- **quantityStock :** An int representing the number of units available in stock.
- **expiryDate:** A LocalDate representing the expiration date of the medication.

Methods:

- **getMedID():** Returns the medication ID.
- **getMedName():** Returns the medication name.
- **getDose():** Returns the medication dosage.
- **getQuantityStock():** Returns the stock quantity.
- **getExpiryDate():** Returns the expiry date of the medication.
- **setQuantityStock(int amount) :** Updates the stock quantity.
- **isExpired():** Returns true if the medication has expired, otherwise false.
- **toString():** Returns a formatted string representation of the medication.

5. Prescription

The Prescription class represents a medical prescription issued by a doctor for a patient.

Attributes:

- **prescID:** A String representing the unique prescription identifier.
- **Doctor:** A Doctor object representing the prescribing doctor.
- **Patient:** A Patient object representing the recipient of the prescription.
- **Medication:** A Medication object representing the prescribed medication.
- **prescDate:** A LocalDate representing the date the prescription was issued.
- **prescExpiry:** A LocalDate representing the expiration date (defaults to 1 year after issuance).

Methods:

- **getPrescID():** Returns the prescription ID.
- **getDoctor():** Returns the prescribing doctor.
- **getPatient():** Returns the patient associated with the prescription.
- **getMedication():** Returns the prescribed medication.
- **getPrescDate():** Returns the prescription issue date.
- **getPrescExpiry():** Returns the prescription expiry date.
- **isExpired():** Returns true if the prescription has expired, otherwise false.
- **toString():** Returns a formatted string representation of the prescription.

6. MedicationTrackingSystem

The MedicationTrackingSystem class is the core system manager that handles all major functionalities, including managing patients, doctors, medications, prescriptions, reports, and restocking.

Attributes:

- **patients:** A List<Patient> storing all patients registered in the system.
- **Doctors:** A List<Doctor> storing all doctors registered in the system.
- **Medications:** A List<Medication> storing all medications available in the system.

Methods:

Managing Entities:

- **addPatient(Patient patient), addDoctor(Doctor doctor), addMedication(Medication medication):** Adds a new patient, doctor, or medication to the system.
- **removePatient(String patientID), removeDoctor(String doctorID), removeMedication(String medicationID):** Removes a patient, doctor, or medication based on ID.

Searching for Records:

- **searchPatient(String name), searchDoctor(String name), searchMedication(String name):** Searches and retrieves a patient, doctor, or medication by name.

Handling Prescriptions:

- **acceptPrescription(String prescID, String doctorName, String patientName, String medName, LocalDate prescDate):** Manually records a new prescription.
- **printPrescriptionsByDoctor(String doctorName):** Displays all prescriptions issued by a specific doctor.
- **printPrescriptionsByPatient(String patientName):** Displays all prescriptions assigned to a specific patient.

Generating Reports:

- **generateReport():** Generates a full system report, displaying all patients, doctors, and medications in the system.
- **checkForExpiredMedications():** Checks for expired medications and lists them for pharmacist action.

Restocking Medications:

- **restockAllMedications():** Automatically increases stock levels for all medications in the system.

3. How to Start & Access the Application

Before running the Pharmacy Management System, ensure that the following requirements are met:

3.1) Prerequisites

- Java Development Kit (JDK 17+) installed.
- Recommended IDE: Any Java-supported IDE such as:
    - **VS Code**
    - **IntelliJ IDEA**
    - **Eclipse**
- Git installed (Optional, for cloning the repository).

3.2) Running the Application

Option 1: Using VS Code / IntelliJ / Eclipse

1- Open the project in VS Code / IntelliJ / Eclipse.

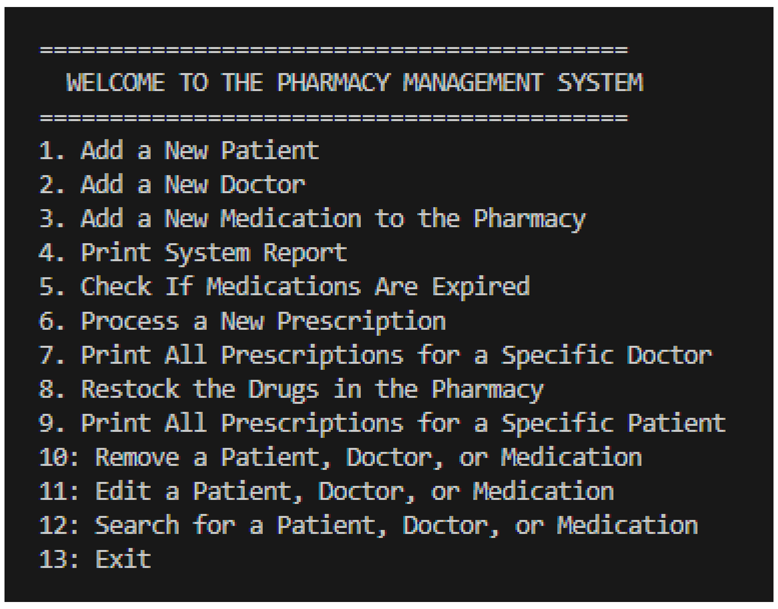
2- Locate MainMenu.java.

3- Click Run to start the application.

Option 2: Using Terminal If running manually from the terminal:

```
javac -d bin -sourcepath src src/ui/MainMenu.java

java -cp bin ui.MainMenu
```

3.3) Menu Options & Their Functions



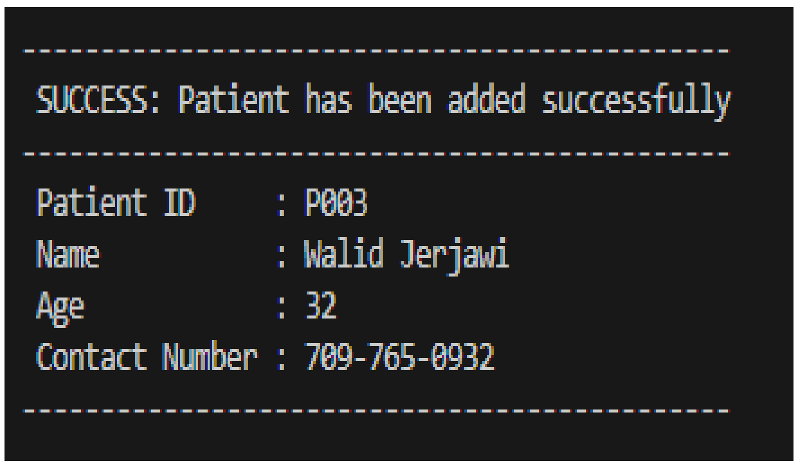
```
===========================================
  WELCOME TO THE PHARMACY MANAGEMENT SYSTEM
===========================================
1. Add a New Patient
2. Add a New Doctor
3. Add a New Medication to the Pharmacy
4. Print System Report
5. Check If Medications Are Expired
6. Process a New Prescription
7. Print All Prescriptions for a Specific Doctor
8. Restock the Drugs in the Pharmacy
9. Print All Prescriptions for a Specific Patient
10: Remove a Patient, Doctor, or Medication
11: Edit a Patient, Doctor, or Medication
12: Search for a Patient, Doctor, or Medication
13: Exit
```

Figure 1: Pharmacy Management System Menu options

Each option performs the following function:

1: Add A New Patient

- Allows adding a new patient by providing details such as ID, name, age, and phone number.
- The patient is stored in the system and available for prescription assignment.

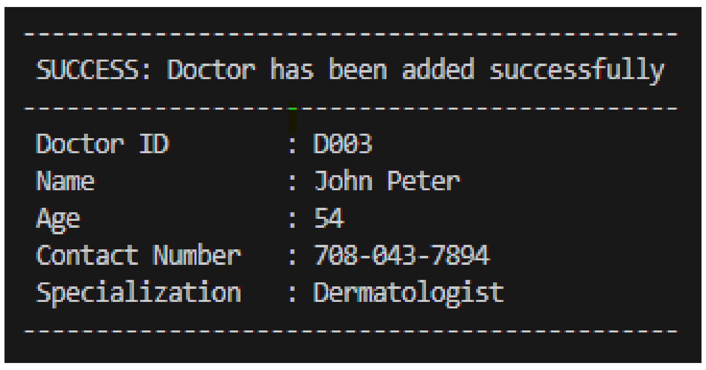


```
---------------------------------------------
SUCCESS: Patient has been added successfully
---------------------------------------------
Patient ID     : P003
Name           : Walid Jerjawi
Age            : 32
Contact Number : 709-765-0932
---------------------------------------------
```

Figure 2:Option1-Add a new patient

2: Add A New Doctor

- Allows adding a new doctor with ID, name, age, phone number, and specialization.
- Doctors can prescribe medications and manage patients.

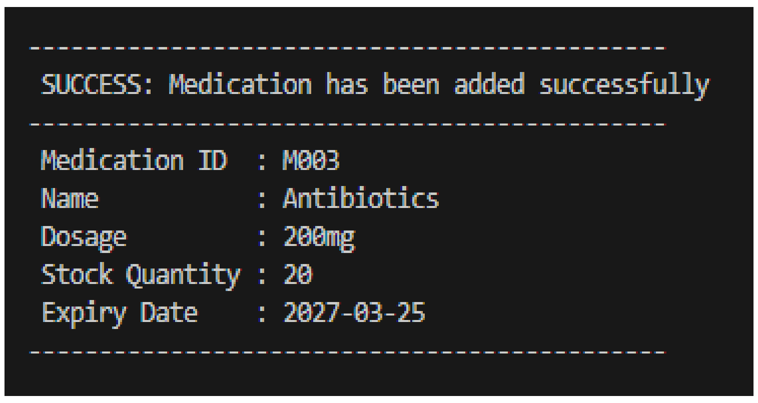


```
--------------------------------------------
 SUCCESS: Doctor has been added successfully
--------------------------------------------
 Doctor ID        : D003
 Name             : John Peter
 Age              : 54
 Contact Number   : 708-043-7894
 Specialization   : Dermatologist
--------------------------------------------
```

Figure 3: Option 2-Add a new doctor

3: Add A New Medication to The Pharmacy

- Adds a new medication to the system with details like:
  - Medication ID
  - Name
  - Dosage (e.g., 500mg)
    - Stock Quantity
    - Expiry Date (Auto-generated or manually entered)
- This ensures an updated list of available medications.

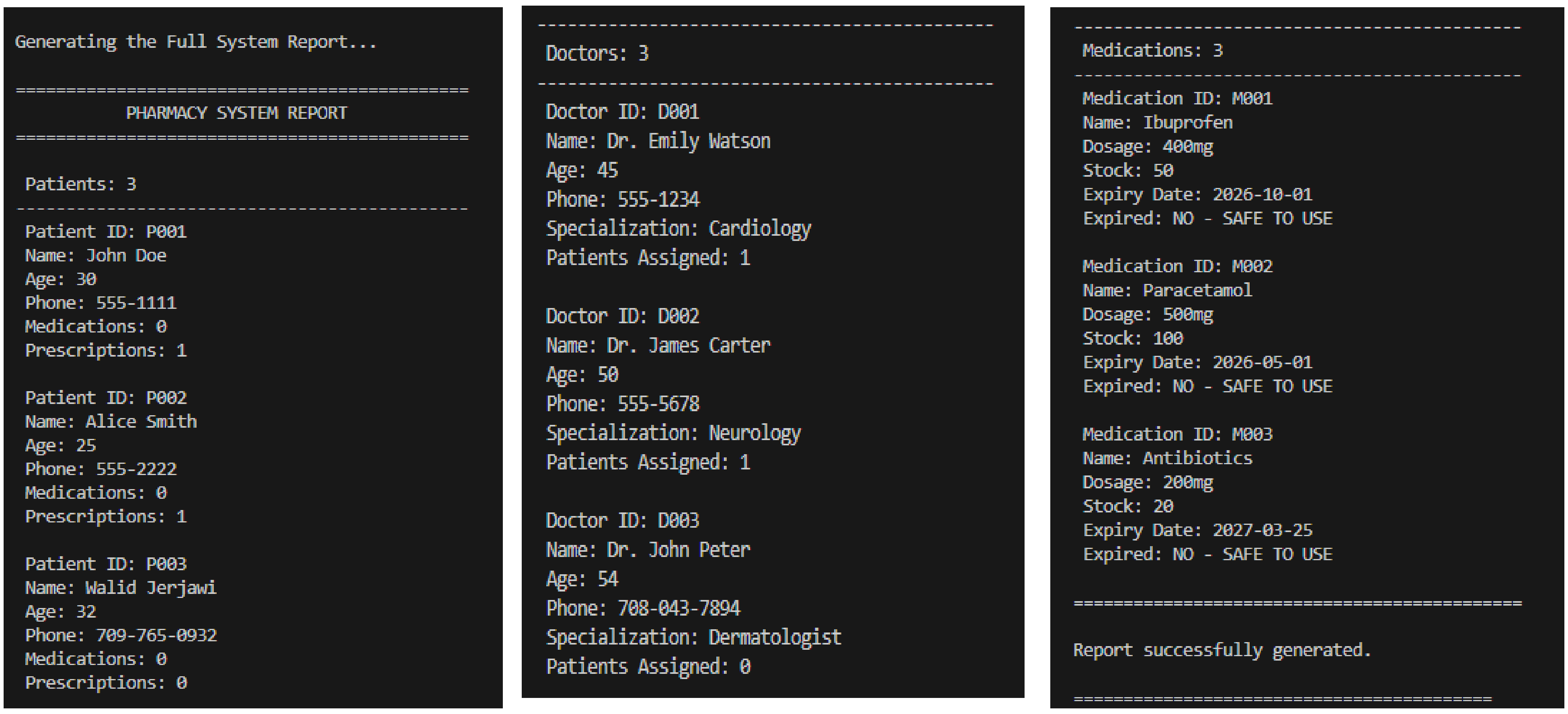


```
--------------------------------------------
 SUCCESS: Medication has been added successfully
--------------------------------------------
 Medication ID  : M003
 Name           : Antibiotics
 Dosage         : 200mg
 Stock Quantity : 20
 Expiry Date    : 2027-03-25
--------------------------------------------
```

Figure 4: Option 3- Add a new medication

4: Print System Report

- Generates a detailed report of:
  - All Patients (Names, IDs, Age, Contact Info, Medications, Prescriptions)
  - All Doctors (Names, IDs, Specializations, Contact Info)
  - All Medications (Names, IDs, Dosage, Stock, Expiry Date, Expired Status)



```
Generating the Full System Report...

============================================
          PHARMACY SYSTEM REPORT
============================================

 Patients: 3
--------------------------------------------
 Patient ID: P001
 Name: John Doe
 Age: 30
 Phone: 555-1111
 Medications: 0
 Prescriptions: 1

 Patient ID: P002
 Name: Alice Smith
 Age: 25
 Phone: 555-2222
 Medications: 0
 Prescriptions: 1

 Patient ID: P003
 Name: Walid Jerjawi
 Age: 32
 Phone: 709-765-0932
 Medications: 0
 Prescriptions: 0
```

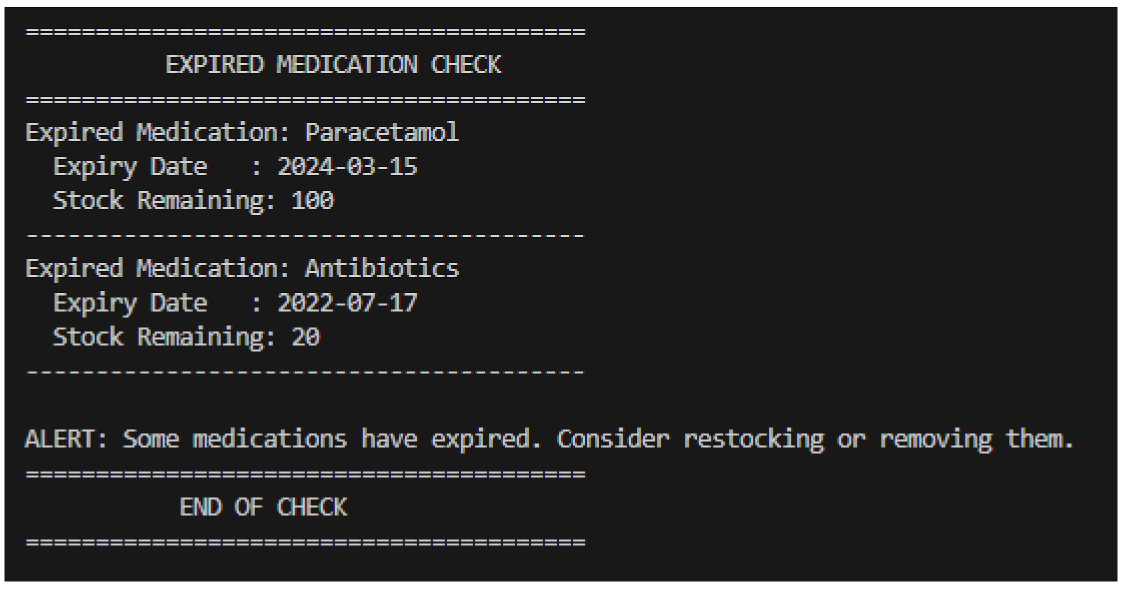
```
--------------------------------------------
 Doctors: 3
--------------------------------------------
 Doctor ID: D001
 Name: Dr. Emily Watson
 Age: 45
 Phone: 555-1234
 Specialization: Cardiology
 Patients Assigned: 1

 Doctor ID: D002
 Name: Dr. James Carter
 Age: 50
 Phone: 555-5678
 Specialization: Neurology
 Patients Assigned: 1

 Doctor ID: D003
 Name: Dr. John Peter
 Age: 54
 Phone: 708-043-7894
 Specialization: Dermatologist
 Patients Assigned: 0
```

```
--------------------------------------------
 Medications: 3
--------------------------------------------
 Medication ID: M001
 Name: Ibuprofen
 Dosage: 400mg
 Stock: 50
 Expiry Date: 2026-10-01
 Expired: NO - SAFE TO USE

 Medication ID: M002
 Name: Paracetamol
 Dosage: 500mg
 Stock: 100
 Expiry Date: 2026-05-01
 Expired: NO - SAFE TO USE

 Medication ID: M003
 Name: Antibiotics
 Dosage: 200mg
 Stock: 20
 Expiry Date: 2027-03-25
 Expired: NO - SAFE TO USE

============================================

Report successfully generated.

============================================
```

Figure 5:Option 4- Print system report for patients, doctors and medications

5: Check If Medications Are Expired

- Scans the list of medications and displays any expired ones.
- Notifies the user if restocking or removal is needed.

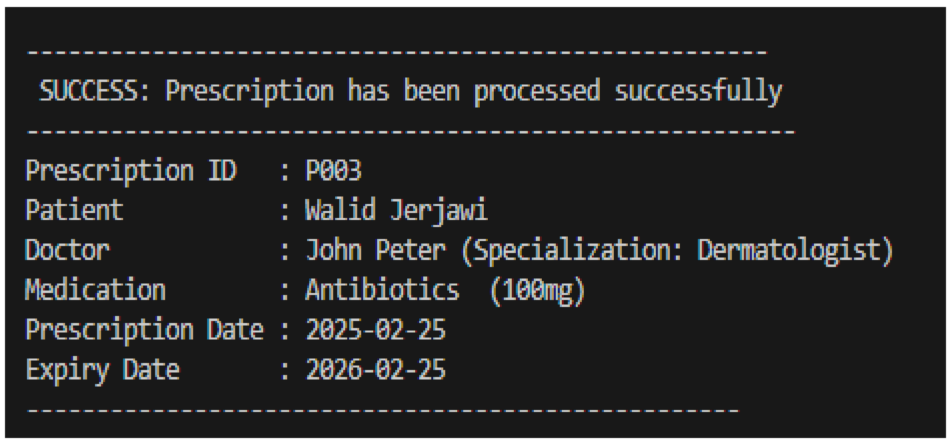


```
========================================
          EXPIRED MEDICATION CHECK
========================================
Expired Medication: Paracetamol
  Expiry Date    : 2024-03-15
  Stock Remaining: 100
----------------------------------------
Expired Medication: Antibiotics
  Expiry Date    : 2022-07-17
  Stock Remaining: 20
----------------------------------------

ALERT: Some medications have expired. Consider restocking or removing them.
========================================
           END OF CHECK
========================================
```

Figure 6: Option 5-Expired medication check

6: Process A New Prescription

- Allows a doctor to issue a **new prescription** for a patient.
- Requires details like:
  - Prescription ID
  - Doctor's Name
  - Patient's Name
  - Medication Name
  - Prescription Issue Date
- The prescription is automatically linked to the patient and checked for expiration.

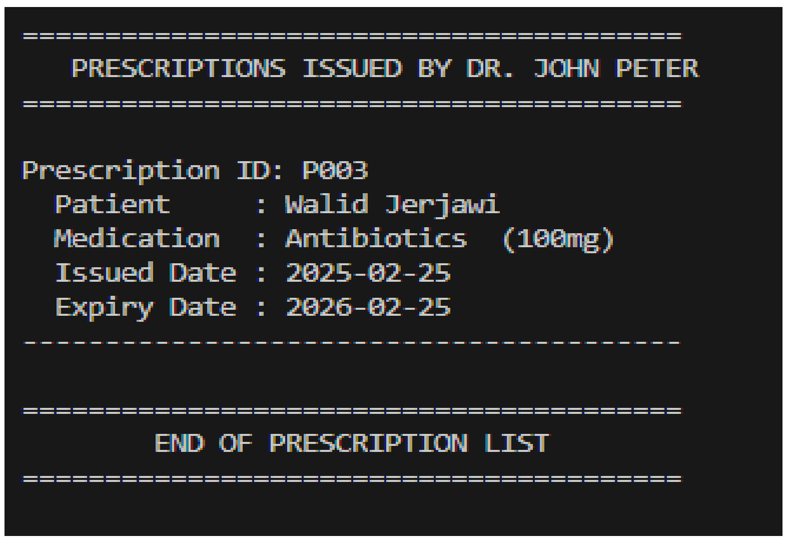


```
----------------------------------------------------
 SUCCESS: Prescription has been processed successfully
-----------------------------------------------------
Prescription ID   : P003
Patient           : Walid Jerjawi
Doctor            : John Peter (Specialization: Dermatologist)
Medication        : Antibiotics  (100mg)
Prescription Date : 2025-02-25
Expiry Date       : 2026-02-25
---------------------------------------------------
```

Figure 7:Option 6- Process a new prescription

7: Print All Prescriptions for A Specific Doctor

- Displays all prescriptions that a **specific doctor has issued**.
- Shows patient details and medication names.

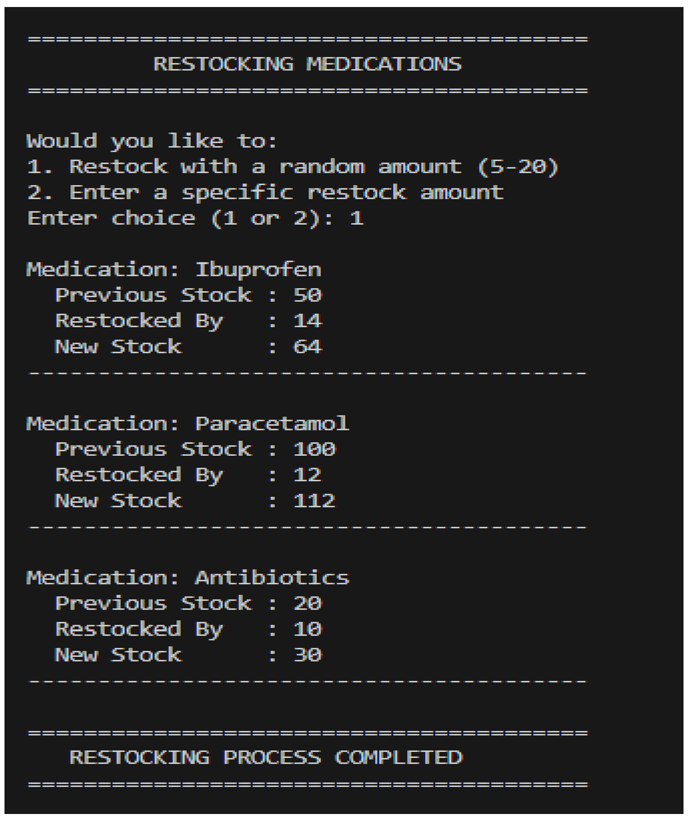


```
========================================
   PRESCRIPTIONS ISSUED BY DR. JOHN PETER
========================================

Prescription ID: P003
  Patient      : Walid Jerjawi
  Medication   : Antibiotics  (100mg)
  Issued Date  : 2025-02-25
  Expiry Date  : 2026-02-25
----------------------------------------

========================================
        END OF PRESCRIPTION LIST
========================================
```

Figure 8:Option 7-Print prescriptions for a specific doctor

8: Restock Medications in The Pharmacy

- Updates the stock quantity for **all** medications.
- Restocks automatically with a random amount between **5 and 20 units** per medication.

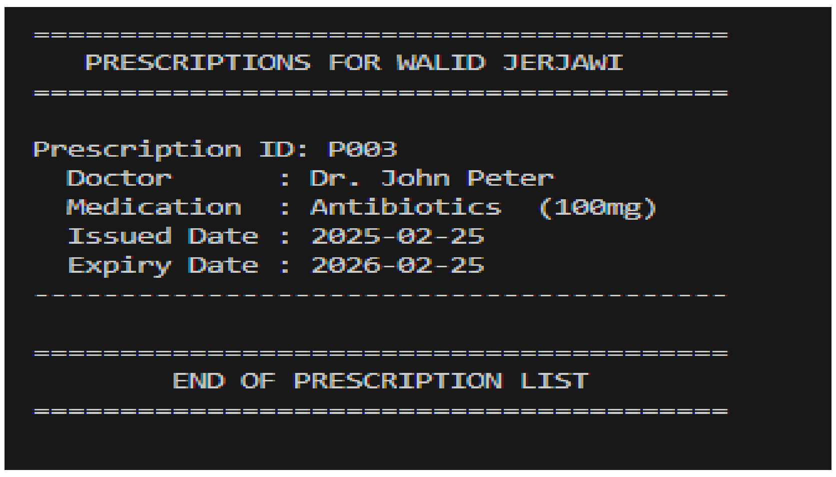


```
========================================
         RESTOCKING MEDICATIONS
========================================

Would you like to:
1. Restock with a random amount (5-20)
2. Enter a specific restock amount
Enter choice (1 or 2): 1

Medication: Ibuprofen
  Previous Stock : 50
  Restocked By   : 14
  New Stock      : 64
----------------------------------------

Medication: Paracetamol
  Previous Stock : 100
  Restocked By   : 12
  New Stock      : 112
----------------------------------------

Medication: Antibiotics
  Previous Stock : 20
  Restocked By   : 10
  New Stock      : 30
----------------------------------------

========================================
   RESTOCKING PROCESS COMPLETED
========================================
```

Figure 9:Option 8-Restock Medications

9: Print All Prescriptions for A Specific Patient

- Lists all prescriptions assigned to a specific patient.
- Includes details such as:
    - Prescription ID
    - Doctor's Name
    - Medication Name
    - Issue Date & Expiry Date

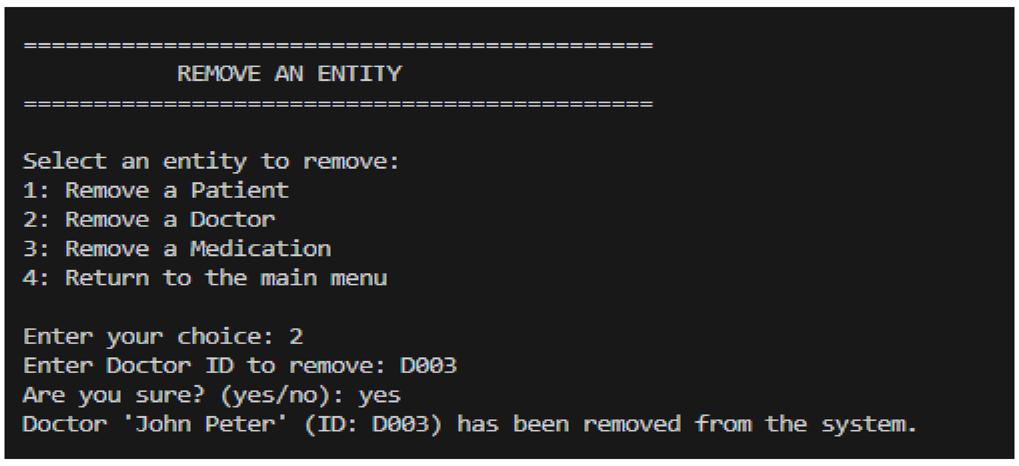


```
========================================
   PRESCRIPTIONS FOR WALID JERJAWI
========================================

Prescription ID: P003
  Doctor       : Dr. John Peter
  Medication   : Antibiotics  (100mg)
  Issued Date  : 2025-02-25
  Expiry Date  : 2026-02-25
----------------------------------------

========================================
       END OF PRESCRIPTION LIST
========================================
```

Figure 10:Option 9- Prescriptions for a specific patient

10: Remove A Patient, Doctor, Or Medication

- Removes a specific patient, doctor, or medication from the system using their ID.
- Displays confirmation when removal is successful.



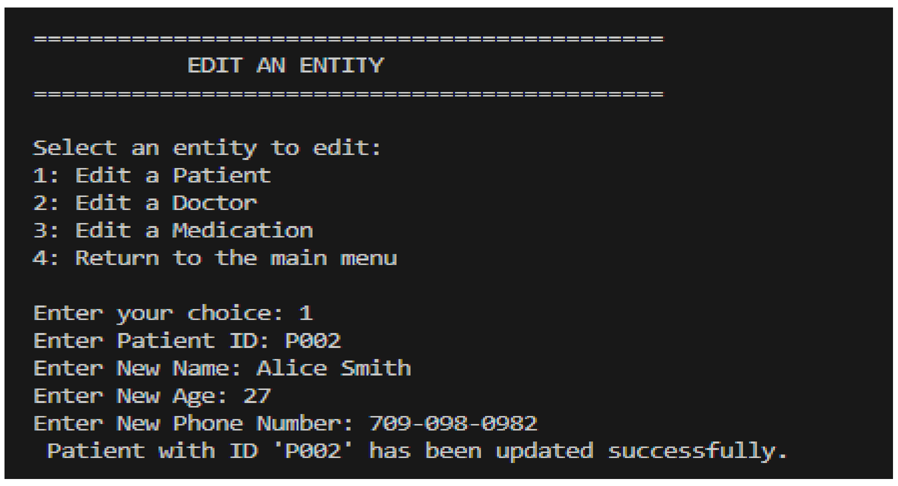
```
=============================================
           REMOVE AN ENTITY
=============================================

Select an entity to remove:
1: Remove a Patient
2: Remove a Doctor
3: Remove a Medication
4: Return to the main menu

Enter your choice: 2
Enter Doctor ID to remove: D003
Are you sure? (yes/no): yes
Doctor 'John Peter' (ID: D003) has been removed from the system.
```

Figure 11:Option10-remove an entity

11: Edit Details (Patient, Doctor, Medication)

- Allows the user to modify existing records, including:
    - Patient Details (Name, Age, Phone)

- Doctor Details (Name, Age, Phone, Specialization)
- Medication Details (Name, Dosage, Stock)

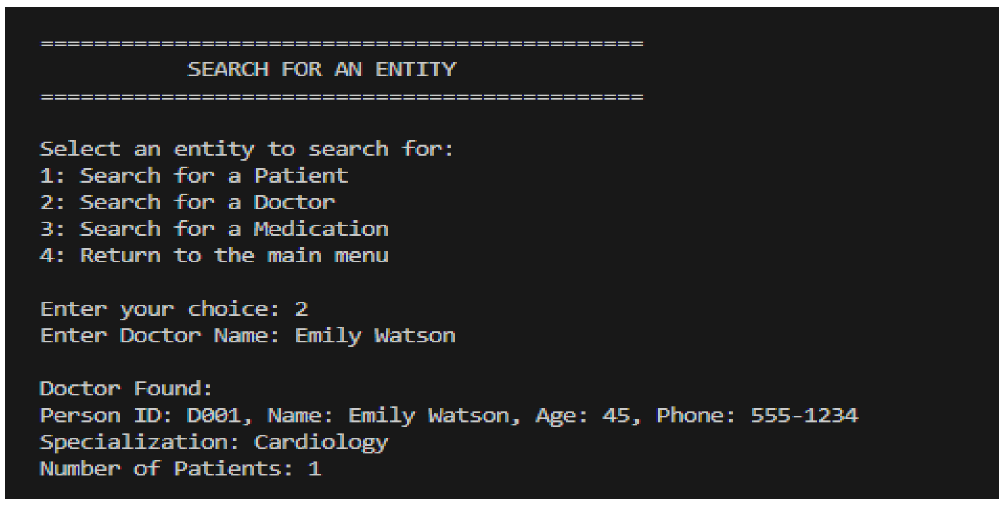


```
============================================
            EDIT AN ENTITY
============================================

Select an entity to edit:
1: Edit a Patient
2: Edit a Doctor
3: Edit a Medication
4: Return to the main menu

Enter your choice: 1
Enter Patient ID: P002
Enter New Name: Alice Smith
Enter New Age: 27
Enter New Phone Number: 709-098-0982
 Patient with ID 'P002' has been updated successfully.
```

Figure 12:Option11-Edit an entity

12: Search For a Patient, Doctor, Or Medication

- Finds and retrieves details of a patient, doctor, or medication based on the name entered.



```
============================================
           SEARCH FOR AN ENTITY
============================================

Select an entity to search for:
1: Search for a Patient
2: Search for a Doctor
3: Search for a Medication
4: Return to the main menu

Enter your choice: 2
Enter Doctor Name: Emily Watson

Doctor Found:
Person ID: D001, Name: Emily Watson, Age: 45, Phone: 555-1234
Specialization: Cardiology
Number of Patients: 1
```

Figure 13:Option 12-Search for an entity

13: Exit

- Closes the application and ends the session.

4. Class Diagram with Associations (UML)

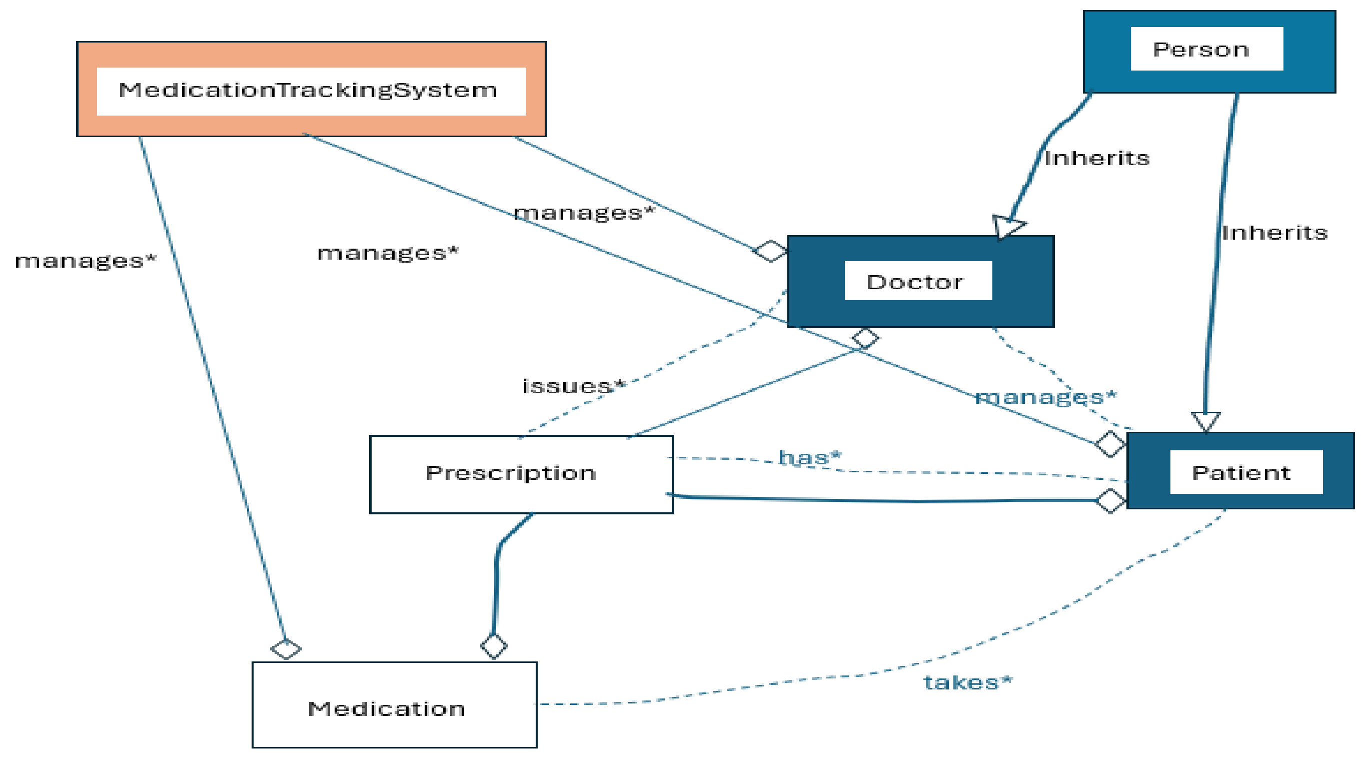


Figure 14:UML Class Diagram

1. **Inheritance**
   - Represented by a **triangle**.
   - Patient and Doctor **inherit** from Person.
2. **Aggregation**
   - Represented by a **diamond**.
   - Prescription **contains** Medication (Aggregation).
   - MedicationTrackingSystem **manages** Patient, Doctor, and Medication.

3. **Associations (Dashed Lines for References)**

- Doctor **manages** multiple Patients.
- Patient **has** multiple Prescriptions.
- Doctor **issues** multiple Prescriptions.
- Patient **takes** multiple Medications.
- The * symbol is used whenever a class has a List<>, array, or multiple instances of another class. (1:M or M:N relationships)

```
Person (Super Class)
--------------------------------------------
- ID: String
- Name: String
- age: int
- phoneNum: String
--------------------------------------------
+ Person(ID: String, Name: String,
   age: int, phoneNum: String)
+ getID(): String
+ getName(): String
+ getAge(): int
+ getPhoneNum: String
+ setID(String): void
+ setName(String): void
+ setAge(int): void
+ setPhoneNum(String): void
+toString: String
```

```
MedicalTrackingSystem
--------------------------------------------
- patients: List<Patient>
- doctors: List<Doctor>
- medications: List<Medication>
--------------------------------------------
+ MedicalTrackingSystem()
+ addPatient(patient:Patient):void
+ addDoctor(doctor:Doctor):void
+ addMedication(medication:Medication):void
+ removePatient(patientID: String):void
+ removeDoctor(doctorID: String):void
+ removeMedication(medicationID: String)
+ editPatient(patientID: String, newname: String,
   newAge: int, newPhone: String): void
+ editDoctor(doctorID: String, newname: String, newAge:int,
   newPhone: String, newSpecialization: String): void
+ editMedication(medicationID: String, newname: String,
  newDose: String, newStock: int): void
+ searchPatient(name:String): Patient
+ searchDoctort(name:String):Doctor
+ searchMedication(name:String): Medication
+ acceptPrescription(prescID: String, doctorName; String,
  patientName: String, medName: String, prescDate:
  LocalDate): void
+ assignPatientToDoctor(patientName: String, doctorName:
   String): void
+ generateReport(): void
+ checkForExpiredMedications(): void
+ printPrescriptionsByDoctor(doctorName: String): void
+ restockAllMedications(): void
+ printPrescriptionsByPatient(patientName: String)
```

```
Patient (SubClass)
--------------------------------------------
- medications: List<Medication>
- prescriptions: List<Prescription>
--------------------------------------------
+ Patient(ID: String, Name: String, age: int,
   phoneNum: String)
+ getMedications(): List<Medication>
+ getPrescriptions(): List<Prescription>
+ addMedication(Medication): void
+ addPrescription(Prescription): void
+ removeMedication(Medication): void
+ removePrescription(Prescription): void
+toString: String
```

```
Doctor (SubClass)
--------------------------------------------
- specialization: String
- patients: List<Patient>
--------------------------------------------
+ Doctor(ID: String, Name: String, age: int,
  phoneNum: String, specialization: String)
+ getSpecialization(): String
+ getPatients(): List<Patient>
+ addPatient(Patient): void
+ removePatient(Patient): void
+toString: String
```

```
Prescription
--------------------------------------------
- prescID: String
- doctor: Doctor
- patient: Patient
- medication: Medication
- prescDate: LocalDate
- prescExpiry: LocalDate
--------------------------------------------
+ Prescription (prescID: String, doctor: Doctor,
  patient: Patient, medication: Medication,
  prescDate: LocalDate)
+ getPrescID():String
+ getDoctor(): Doctor
+ getPatient(): Patient
+ getMedication(): Medication
+ getPrescDate(): LocalDate
+ getPrescExpiry(): LocalDate
+ setPrescID(prescID: String): void
+ setPatient( patient: Patient): void
+ setDoctor(doctor: Doctor): void
+ setMedication(medication: Medication): void
+ setPrescDate(prescDate: LocalDate): void
+ setPrescExpiry(prescExpiry: LocalDate):void
+ isExpired(): boolean
+ toString: String
```

```
Medication
--------------------------------------------
- medID: String
- medName: String
- does: String
- quantityStock: int
-expiryDate: LocalDate
--------------------------------------------
+ Medication(medID: String, medName: String,
  dose: String, quantityStock: int, expiryDate:
  LocalDate )
+ getMedID(): String
+ getMedName(): String
+ getDose(): String
+ getQuantityStock: int
+ getExpiryDate(): LocalDate
+ setMedID(medID: String): void
+ setMedName(medName: String):void
+ setDose(dose: String):void
+ setQuantityStock(quantityStock: int):void
+ setExpiryDate(expiryDate: LocalDate):void
+ isExpired(): boolean
+ restock(amount: int):void
+ toString: String
```

5. Javadoc Documentation

The Pharmacy Management System has been fully documented using Javadoc, following best practices for clarity, maintainability, and ease of use. Javadoc comments have been applied consistently across the system to ensure each class, attribute, constructor, and method is well-explained.

Javadoc Implementation

Throughout the project, **Javadoc comments** have been added to:

- **All classes** to describe their purpose and role in the system.
- **All attributes** to explain their significance.
- **All constructors** to specify required parameters for object initialization.
- **All methods** to detail their behavior, input parameters, and return values.

Key Javadoc Elements Used:

@param – Describes method parameters, including expected data types.
@return – Specifies the return type and expected value of a method.
@Override – Used in methods that override superclass behavior (e.g., toString()).

6. Source Code Directory Structure

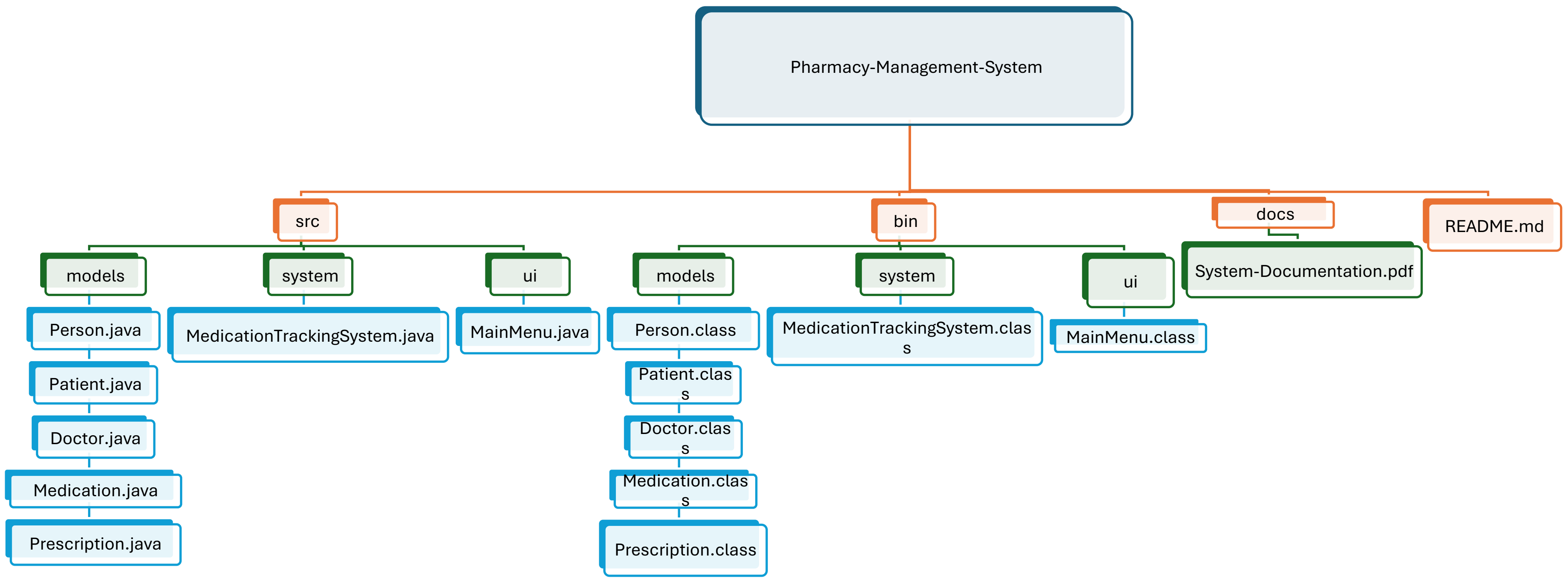


Figure 15:Source code Directory structure

7- Development Standards

Ensuring consistency and maintainability in our Java application, we follow these best practices:

7.1- Code Formatting Standards

A. Class Naming (PascalCase)

- Follows Java naming conventions where class names start with an uppercase letter, and each word is capitalized.
- **Example**:

```
public class MedicationTrackingSystem { }
public class Prescription { }
```

B. Variable & Method Naming (camelCase)

- Variables and methods start with a lowercase letter, but each new word is capitalized.
- **Example**:

```
private String patientName;
public void addNewMedication() { }
```

7.2- Java Best Practices Followed

A. Encapsulation (Using private fields with public getters & setters)

- **Example**

```
private String medID;
public String getMedID() {
    return medID;
}
```

B. Code Readability

- Indentation: 4 spaces per level.
- Braces {} always on the same line for methods and loops.
- **Example:**

```
if (medications.isEmpty()) {

    System.out.println("No medications available.");


}
```

C. Use of this Keyword

- Ensures no confusion between class attributes and method parameters.
- **Example:**

```
public void setPatientName(String patientName) {
    this.patientName = patientName;
}
```

7.3- Object-Oriented Principles Applied

A. Encapsulation:

All attributes are private, accessed via getters & setters.

B. Inheritance (extends keyword used for subclasses)

- **Example**:

```
public class Patient extends Person { }
public class Doctor extends Person { }
```

C. Polymorphism (Method Overriding used)

- **Example** (Overriding toString() in multiple classes):

```
@Override
public String toString() {
    return "Doctor: " + getName();
}
```

D. Aggregation & Composition (Relationships between classes)

- **Example** (Doctor has List<Patient> assigned):

```
private List<Patient> patients = new ArrayList<>();
```

7.4. Error Handling & Validation

- **Null Checking in Methods**

```
if (doctor == null) {
    System.out.println("Doctor not found.");
    return;
}
```

- **Validation Checks**

```
if (patients.contains(newPatient)) {
    System.out.println("Patient already exists.");
}
```

7.5- Code Documentation & Comments

- Javadoc for all public methods & classes
- Single-line (//) and multi-line (/* */) comments for clarity

```
/**
 * Searches for a doctor by name.
 * @param name The name of the doctor.
 * @return The doctor object if found, otherwise null.
 */
public Doctor searchDoctor(String name) { ... }
```

8. Theoretical Database Design (Entity Relationship Diagram - ERD)

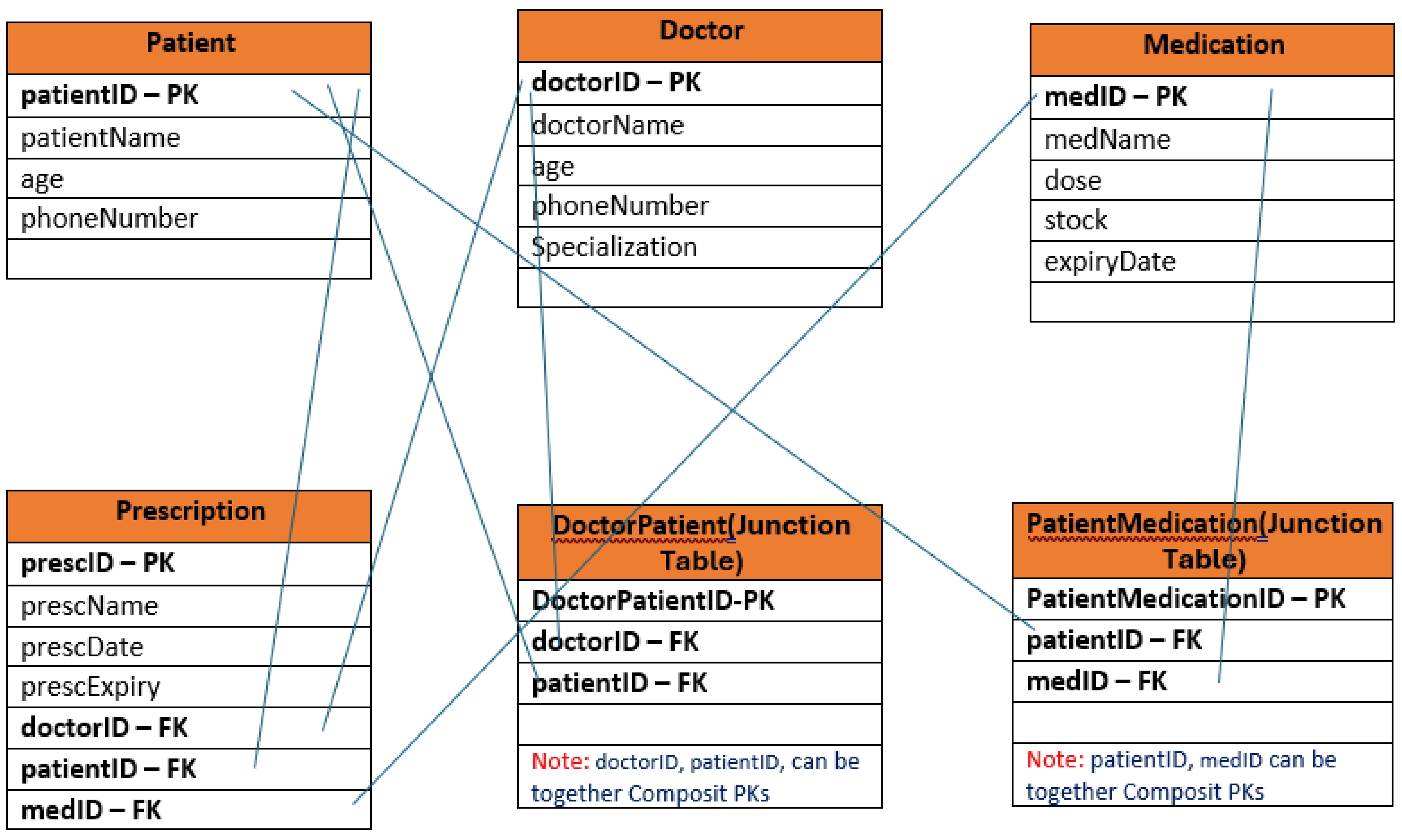


Figure 16:Entity Relationship Diagram

MedicationTrackingSystem (Management System)

- This class manages all entities but does not represent a database table.
- It interacts with all the above tables to perform CRUD operations.

Entity Relationship Diagram (ERD) Description

- **Doctor to Prescription**: **One-to-Many (1:M)** → A doctor can issue multiple prescriptions.
- **Patient to Prescription**: **One-to-Many (1:M)** → A patient can have multiple prescriptions.
- **Medication to Prescription**: **One-to-Many (1:M)** → A medication can be prescribed multiple times.
- **Doctor to Patient**: **Many-to-Many (M:N)** → A doctor can treat multiple patients, and a patient can be treated by multiple doctors.
- **Patient to Medication**: **Many-to-Many (M:N)** → A patient can take multiple medications, and a medication can be prescribed to multiple patients.

9. How to Get the Source Code from GitHub

To download or contribute to the Pharmacy Management System, follow these steps:

- **Ensure Git is Installed** (Windows: Download from git-scm.com, Mac/Linux: Install via terminal).
- **Clone the Repository Using Git** (Open terminal, navigate to a folder, clone the repository, and enter the project folder).

  git clone **https://github.com/WalidJer/Pharmacy-Management-System**

- **Download as a ZIP (Alternative Method)** (Visit GitHub, click "Code" > "Download ZIP," and extract the files).
- **Importing into an IDE**
  - **VS Code**: Open VS Code, go to **File > Open Folder**, and select the project folder.

        - **IntelliJ IDEA**: Open IntelliJ, go to **File > Open**, and select the project folder.
        - **Eclipse**: Open Eclipse, go to **File > Open Projects from File System**, and select the project folder.

- **Contributing to the Project (Optional)** (Create a branch, make changes, commit, push, and open a Pull Request on GitHub).