CSE Department – Faculty of Engineering - MSA
Spring 2025
GSE122 GSE122i COM265 PROGRAMMING 2
Course Project
Course Instructor:Dr. Ahmed El Anany

Student Name	**Amr Hassan Abdel Moniem**	**Student ID**	**246005**
Student Name	**Ali Emad Ali**	**Student ID**	**221395**
Student Name	**Abdulrahman Mohamed Gaber**	**Student ID**	**238383**
Student Name	**Abdelrahman Waleed Ahmed**	**Student ID**	**232757**
Student Name	**Mustafa Mahmoud Elsheny**	**Student ID**	**243061**
TA_Name	**Eng. Dina Magdy**	**Grade:**	**/**

Project Title: Hospitals system management

Table of Contents

Project Overview

The **Hospital Management System** (HMS) is a software solution designed to manage various aspects of a hospital's operations. It facilitates the management of patient records, treatments, medicines, financial records, and more, all in one place. The system allows hospital staff to quickly and efficiently handle administrative tasks, which leads to smoother patient care and operations.

The HMS will be implemented using **Java** for the backend, with a **MySQL** database to store all the relevant data. A graphical user interface (GUI) will be provided for easier interaction with the system, using **Swing** components. The main goal of the HMS is to provide a system that helps the hospital's staff organize patient data efficiently, track medical treatments, manage financial records, and ultimately improve the quality of service offered to patients.

Objectives

1. **Database Management**:
    - Create and manage a MySQL database to store various hospital data, including **patient information**, **treatments**, **medications**, and **financial records**.
    - Implement CRUD (Create, Read, Update, Delete) operations on all the entities (patients, treatments, medicines, financial records).
2. **User-Friendly Interface**:
    - Develop a user-friendly graphical interface using Java Swing to interact with the system.
    - The interface will include forms to add new patients, search for patients, add treatments, and view financial records.
3. **Efficient Record Management**:
    - Provide the functionality to manage patient records effectively.
    - Support adding new patients, updating their details, and deleting records.
    - Support tracking and viewing medical treatments and medicines prescribed to each patient.
    - Manage financial records and payments for each patient.
4. **Data Integrity and Validation**:
    - Implement validation checks for user inputs to prevent invalid or inconsistent data.
    - Ensure the integrity of data with the use of primary and foreign keys in the database (e.g., patient ID linking patient records to treatments and medicines).
5. **Scalability and Maintainability**:
    - Design the system to be easily extensible, allowing new features or modifications to be added as the hospital's needs grow.
    - Ensure the code is modular and well-documented for ease of maintenance and future enhancements.
6. **Security**:
    - Implement basic security measures to ensure that sensitive patient information is stored and accessed securely.

Roles and Responsibilities

This section describes the roles of each team member.

Algorithm and External Libraries

The Hospital Management System is designed using a combination of core algorithms and external libraries to ensure smooth performance and user interaction with the system. The system leverages appropriate data structures and efficient data processing operations to cater to the needs of the hospital.

Algorithm Approach:

1. **Data Management (CRUD Operations):**
    - **Create Operations**: Patient records, treatments, medicines, and financial records are added through the user interface using algorithms that optimize data input and validate correctness (e.g., ensuring all fields are properly filled).
    - **Read Operations**: Efficient search algorithms are used to retrieve records, such as searching for patients by ID or name.
    - **Update Operations**: When updating patient records, treatments, or medicines, the system validates the entered data before saving it to ensure correctness.
    - **Delete Operations**: The system employs algorithms to ensure that records are correctly deleted, and the consequences of deletion (e.g., related records) are properly handled.
2. **Input Validation**: The system includes checks to ensure that user-entered data adheres to defined standards. For example, it checks that age is a positive number and that financial charges and dates are valid.
3. **Entity Relationships**: The system links patient records with treatments, medicines, and financial records using unique identifiers, which allows for quick and efficient retrieval of data for each patient.
4. **Storage and Display**: Information is stored in an organized manner using data structures like **ArrayLists** and **HashMaps** to store treatments, medicines, and financial details for each patient.

External Libraries:

1. **Swing (Graphical User Interface):**
    - The Swing library is used to create the graphical user interface (GUI) for the system. Swing provides flexible UI components such as buttons, text fields, labels, and panels to facilitate easy interaction between the user and the system.
2. **JDBC (Database Connection):**
    - The JDBC (Java Database Connectivity) library is used to connect the system with a database (like MySQL or SQLite) for storing and retrieving patient data, treatments, medicines, and financial records. This ensures data persistence and integrity.
3. **Java Collections Framework:**
    - The Java Collections Framework is utilized for managing data structures such as lists and maps, enabling efficient organization and manipulation of patient records, treatments, medicines, and financial transactions.

4. **JOptionPane (For Dialog Boxes):**
    - The `JOptionPane` class from the Swing library is used to display pop-up dialog boxes for user interactions, such as showing error messages, warnings, or confirmations. This helps enhance user experience by making the interface more interactive and responsive.
5. **SQL (Database Management):**
    - The `java.sql.*` package is utilized to establish database connections, execute SQL queries, and manage results. It supports interaction with databases such as MySQL, SQLite, or others, enabling the system to store, retrieve, and update patient-related data

GUI and Database Usage

The **Hospital Management System** utilizes a user-friendly Graphical User Interface (GUI) and integrates seamlessly with a database to manage patient records and associated information. Below is an explanation of how the GUI and database interact within the system.

GUI (Graphical User Interface) Usage:

1. **User Interaction:**
    - The system uses **Swing** components to create the GUI, providing an intuitive interface for hospital staff to interact with the system. It includes fields for entering patient details (ID, name, age, diagnosis) and buttons for various actions like adding new patients, viewing records, and adding treatments, medicines, or financial details.
    - The **JTextField** and **JTextArea** components are used for data input and display of patient information and treatment records. The **JPanel** layout managers organize the user interface into logical sections for easy access and clear visibility.
2. **Actions and Event Handling:**
    - The user can perform various actions like adding, searching, updating, and deleting patient records. These actions are triggered by button clicks, and event listeners are used to capture the user input and call corresponding methods.
    - When a user enters a patient's ID in the search field, the system uses this input to query the database and fetch relevant records, which are then displayed in the **JTextArea**.
3. **Real-Time Feedback:**
    - The GUI gives real-time feedback to users, such as error messages or confirmation dialogs, using **JOptionPane**. For example, if a user tries to add a patient with an already existing ID, a message will pop up informing them of the issue.
    - The output area is used to display messages related to the operations being performed, such as confirming the addition of a new patient or listing treatments for a specific patient.

Database Usage:

1. **Data Storage:**
    - The system interacts with a **DatabaseHandler** class that manages database operations using **JDBC** (Java Database Connectivity). This allows the system to persistently store and retrieve patient records, treatments, medicines, and financial records in a relational database.
    - Patient details like ID, name, age, and diagnosis are stored in the database. The database is designed to support other related entities such as treatments, medicines, and financial records, linking them to the appropriate patient using the patient's ID.
2. **CRUD Operations:**
    - The system supports **Create, Read, Update, and Delete (CRUD)** operations for managing patient and treatment data:
        - **Create**: New patient records are added to the database with the necessary details.
        - **Read**: The system retrieves existing records from the database using SQL queries, allowing users to search by patient ID or name.
        - **Update**: Patients' information or their treatment records are updated in the database when modified by the user.
        - **Delete**: If a patient's record needs to be removed, the system performs a delete operation in the database.
3. **Database Structure:**
    - The database is structured to have tables for each entity, such as **patients**, **treatments**, **medicines**, and **financial records**. Each table is linked by patient ID, ensuring that related records are associated with the correct patient.
    - For example, the **treatments** table stores treatment descriptions linked to the patient's ID, while the **financial records** table stores charge information associated with each patient.
4. **Querying and Data Retrieval:**
    - When searching for specific records (e.g., a patient's treatments or financial details), SQL queries are executed against the database to fetch relevant data based on the patient ID. The data is then displayed through the GUI in a readable format.
    - For example, when viewing the treatments for a patient, the system queries the database to get all treatments associated with that patient's ID and then displays them in the **JTextArea** of the GUI.
5. **Database Error Handling:**
    - The database operations are wrapped in try-catch blocks to handle potential errors, such as database connection failures or SQL exceptions. Error messages are shown in the GUI, providing clear feedback to the user on issues encountered during database interactions.

Code explaining

This section explains the functionality of the main Java classes used in the Hospital Management System project. Each class contributes to specific features and together forms the full application.

1. `Patient.java`

This is a basic model class representing a patient entity. It contains private attributes (`id`, `name`, `age`, `diagnosis`) and provides getters to access them. The `toString()` method formats patient data for display purposes.

2. `DatabaseHandler.java`

This class manages all interactions with the MySQL database. It uses JDBC (Java Database Connectivity) to:

- Connect to the database.
- Insert, retrieve, and delete patient records.
- Handle treatment, financial, and medicine records using SQL queries.
  Each operation is encapsulated in a method, and error handling is done using try-catch blocks and user feedback through `JOptionPane`.

3. HospitalGUI.java

This class defines the graphical user interface using Java Swing. It:

- Displays input fields for patient data.
- Has buttons for operations like Add, View, Search, Delete, Add Treatment/Medicine/Finance.
- Shows results in a text area.
  Event listeners are used to trigger operations from user actions (e.g., button clicks), and input validation ensures data correctness.

4. PatientRecordManager.java

This class simulates basic patient record operations, such as:

- Adding new records.
- Searching or editing records by name or number.
- Listing patients.
- Deleting records.
  It uses an `ArrayList` to temporarily store patient data. It also contains a nested class `PatientRecord` to represent detailed patient information like name, address, disease, and specialist room number.

5. FinancialRecordManager.java

This class provides methods to:

- Add a financial record for a patient (including charge, deposit, and balance).
- Display financial records using data fetched from `DatabaseHandler`.
  It contains a nested `FinancialRecord` class representing the financial structure of each patient.

6. TreatmentManager.java

This class helps manage treatments. It can:

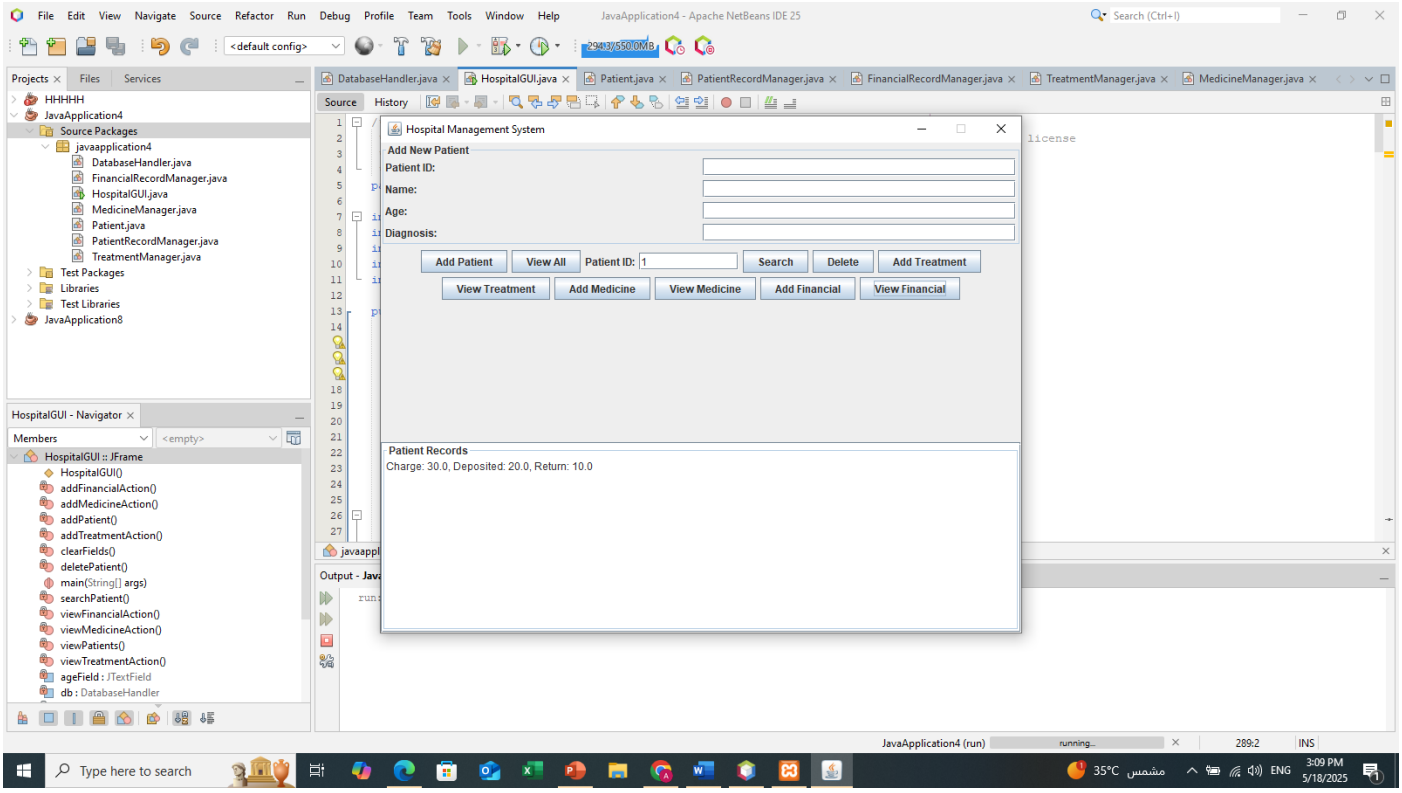
- Add a treatment to the database.
- View existing treatments.
- Update or delete a treatment.
  All these operations are linked to the patient ID and are processed through `DatabaseHandler`.

7. MedicineManager.java

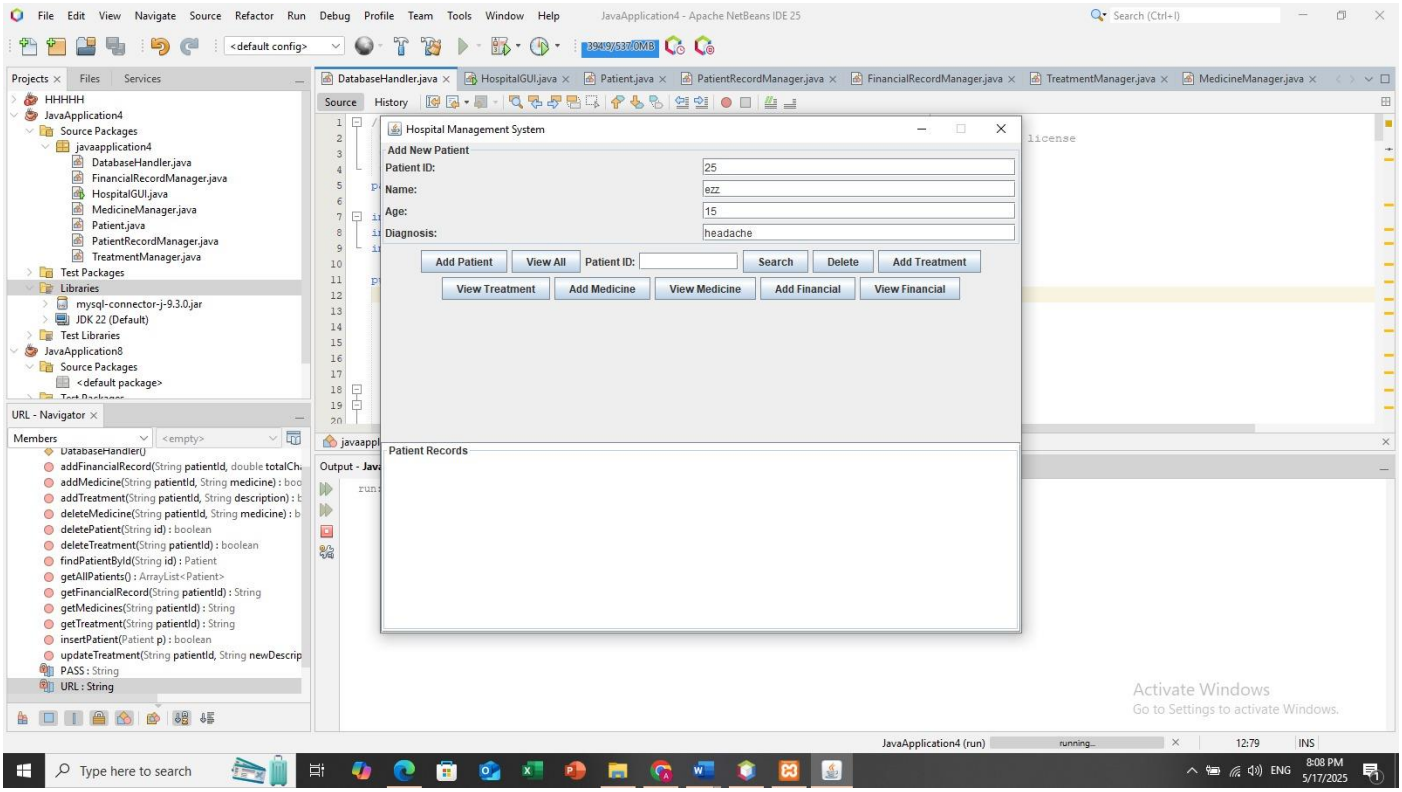
This class handles medicines prescribed to patients. It uses a `HashMap<Integer, ArrayList<String>>` to store medicine lists against patient record numbers and allows:

- Adding new medicines.
- Displaying all medicines for a patient.
- Deleting a specific medicine entry.

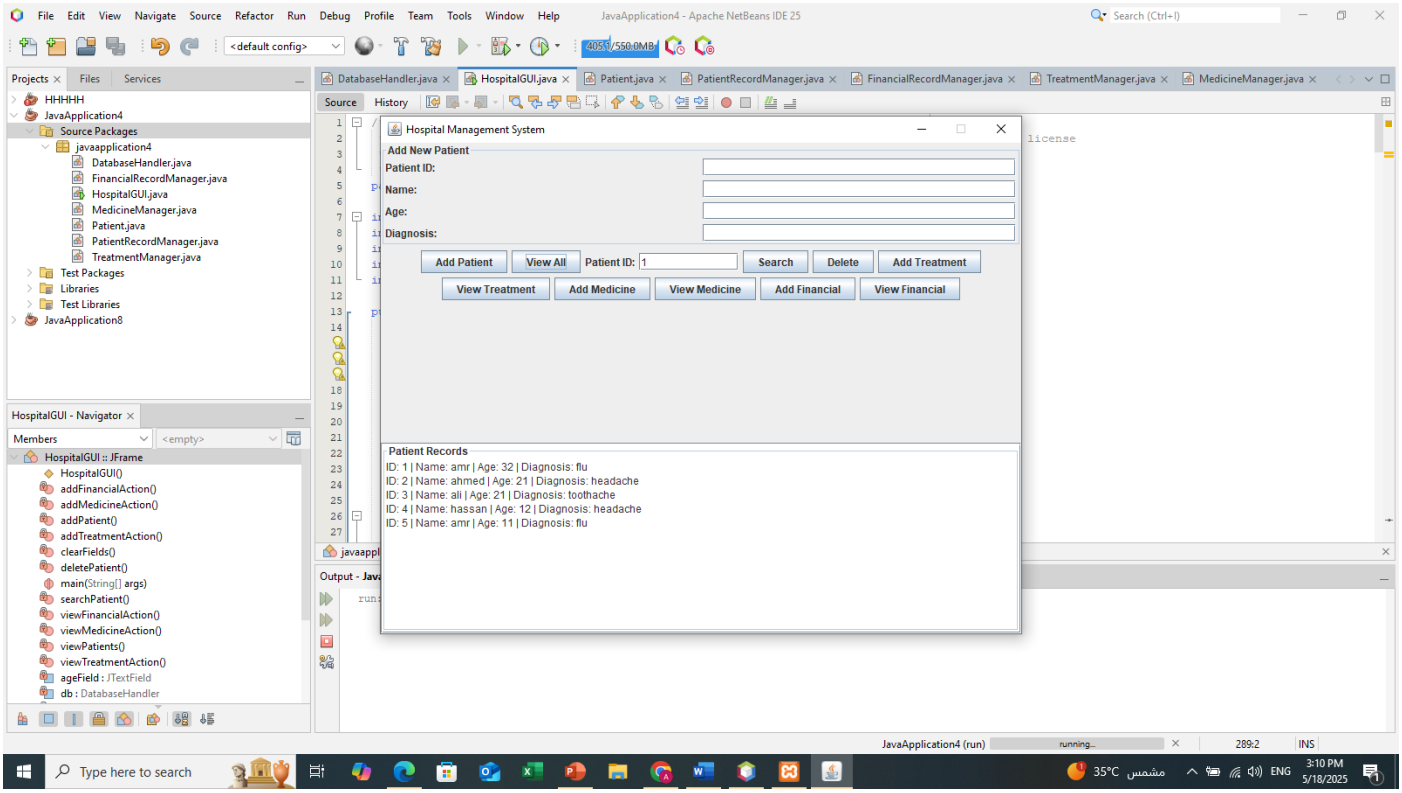
Output and results



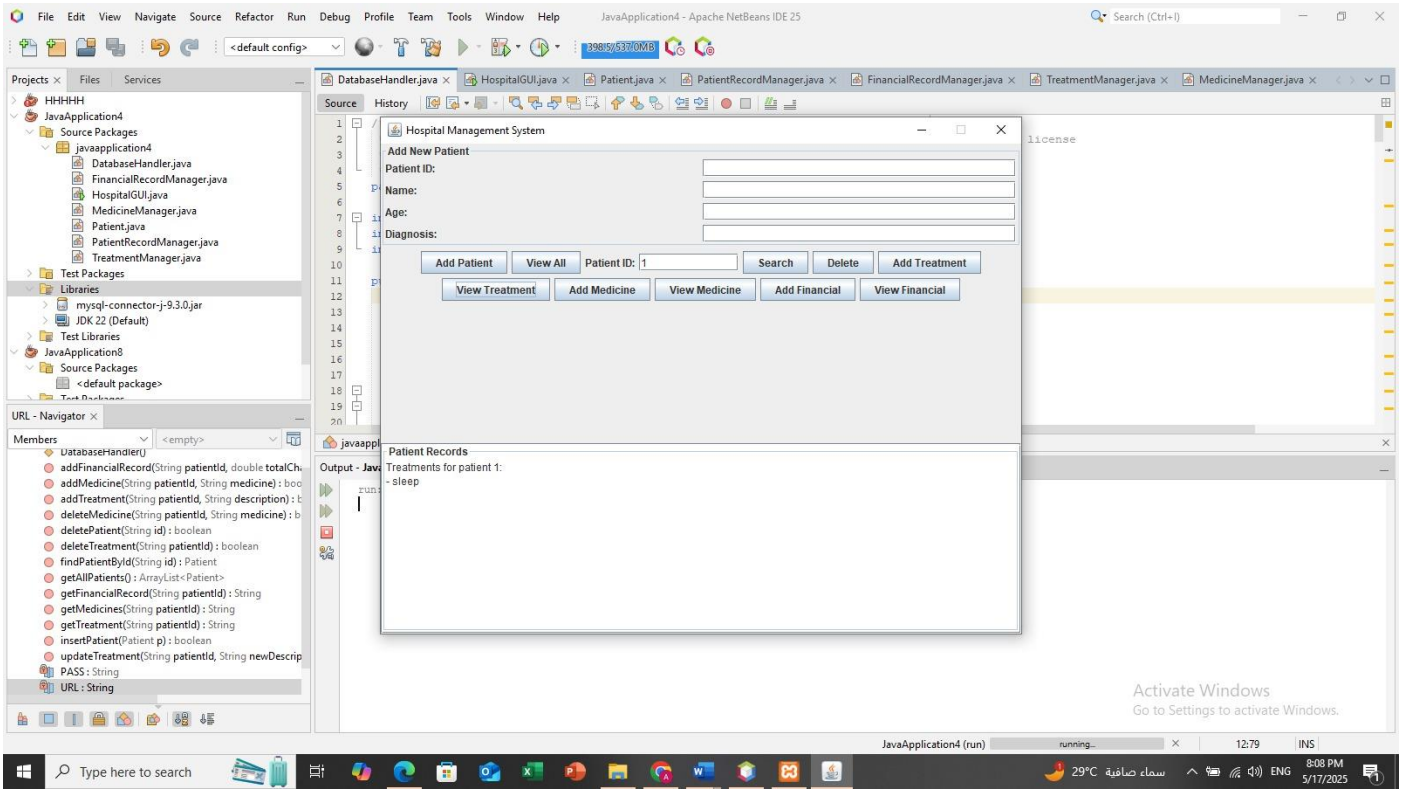
1. This screenshot shows the financial records section for Patient 1. It displays details such as the total charge for services, the amount the patient deposited, and the calculated return amount (if any). This helps hospital staff quickly track the patient's payments and balances.



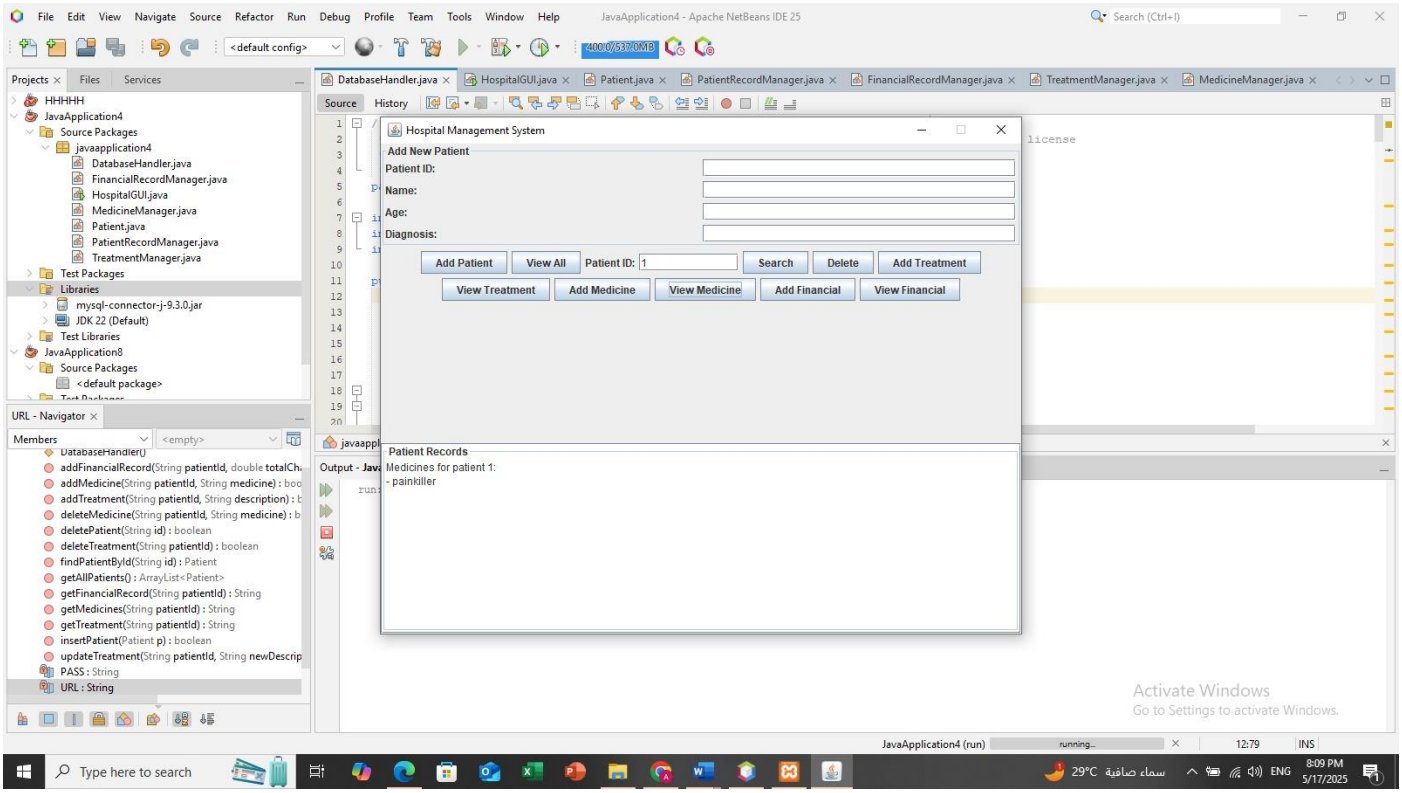
2. This screenshot shows the "Add Patient" form in the Hospital Management System. It includes input fields for entering the patient's ID, name, age, and diagnosis. After filling in the details, the user can click the "Add Patient" button to save the patient's information into the database.



3. This screenshot displays a list of all patients stored in the system. Each patient's details such as ID, name, age, and diagnosis are shown in the output area. This view is accessed by clicking the "View All" button, allowing hospital staff to quickly review patient records.



4. This screenshot shows the treatment records for patient with ID 1. It displays the list of treatments added for this patient. The treatments are entered by the hospital staff and can be viewed using the "View Treatment" button after selecting or searching for a specific patient ID.

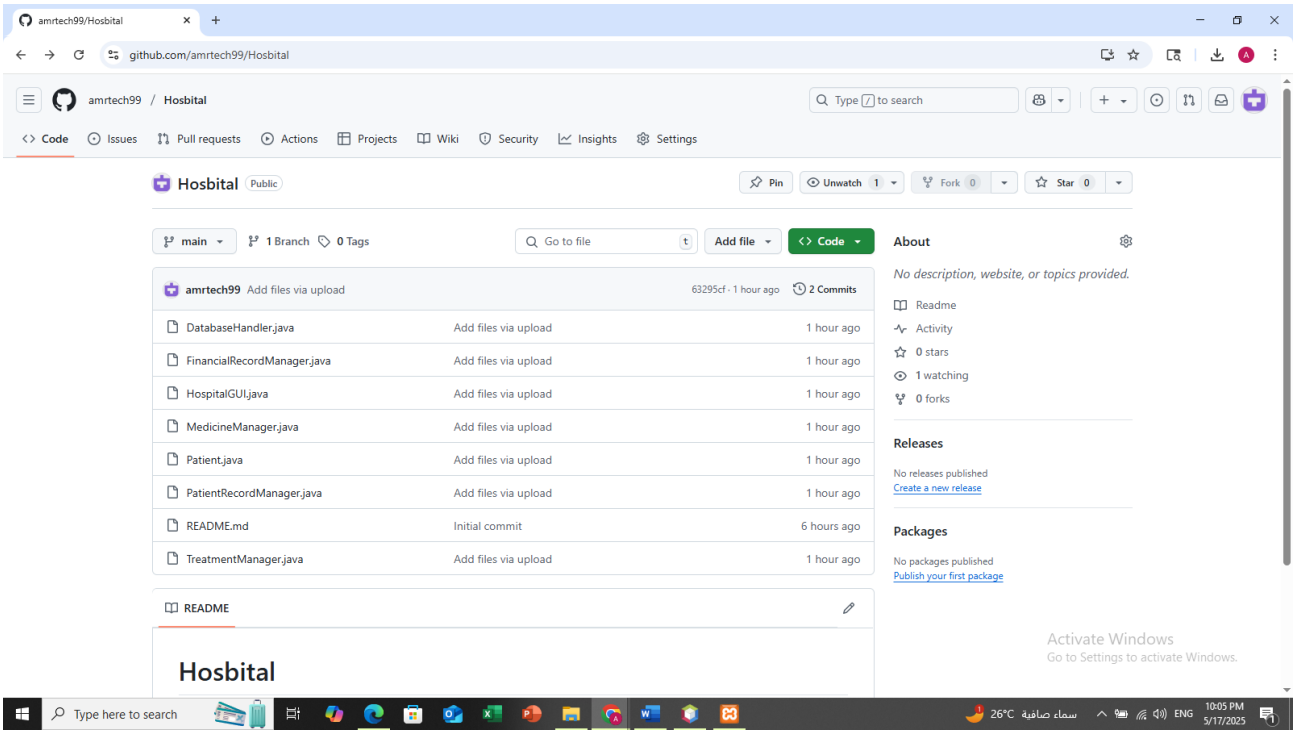


5. This screenshot displays the list of medicines assigned to patient with ID 1. After entering medicine names for the patient, the "View Medicine" button is used to retrieve and show them. It helps hospital staff keep track of medications given to each patient.

GitHub(optional)

Include link of github repo containing your project code and report and add screenshot of repo with commit logs

https://github.com/amrtech99/Hosbital.git



References

1. All lectures