Dr. D. Y. Patil College of Engineering and Innovation, Varale, Talegaon, Pune, 410507.

(Affiliated to Savitribai Phule Pune University)

Project Report

On

"TITLE: Hospital Report and Record Generator: Testing using Junit"

Submitted by

Dawale Sakshi (14319)
Salunke Vaishnavi (14321)
Shewale Roshan (14331)
Tewade Vikas (14362)

Under the guidance of

Pro. Chandan Wagh
Dr. Deepali Sale

BE Computer Engineering

(Semester VII)

2024-25

Dr. D. Y. Patil College of Engineering and Innovation, Varale, Talegaon, Pune, 410507.

(Affiliated to Savitribai Phule Pune University)

CERTIFICATE

This is to certify that the work entitled **"Hospital Report and Record Generator: Testing using Junit"** is a bonafide work carried out as a fulfilment of **mini project** by **Dawale Sakshi, Salunke Vaishnavi, Vikas Tekawade, Roshan Shewale** The report has been approved as it satisfies the academic requirements in respect of syllabus prescribed for the course.

Dr.Deepali Sale	**Dr. Alpana P. Adsul**	**Dr. Suresh Mali**
Guide	HOD	Principal

ACKNOWLEDGEMENT

It is our pleasure to acknowledge sense of gratitude to all those who helped us in completion project. I am highly indebted to subject teacher Prof. Chandan wagh and Dr. Deepali Sale for their guidance. I would like to express my gratitude towards H.O.D. of computer Engineering Department Dr. Alpana P. Adsul for their kind co-operation and encouragement which help me for providing the required facilities.

Finally, we wish to thank and appreciate to all our teachers and friends for their constructive comments, suggestions and guidance and all those directly or indirectly helped us in completing this project.

Dawale Sakshi
Salunke Vaishnavi
Vikas Tekawade
Roshan Shewale

Index

1. Problem Statement:

Efficient management of hospital data, including patient records, diagnosis reports, and appointment schedules, is critical for healthcare operations. However, traditional methods of handling medical records are often prone to errors, delays, and inefficiencies, particularly in prioritizing patient treatments based on severity. This project aims to develop a **Hospital Report and Record Generator System** that automates the storage, retrieval, and generation of patient records and medical reports. The system will utilize **Java** for backend development and incorporate **JUnit testing** to ensure the reliability, accuracy, and robustness of all functionalities. The primary objective is to streamline patient record management, minimize manual errors, and enhance the overall operational efficiency of healthcare institutions through rigorous validation and testing protocols.

The **Hospital Report and Record Generator System** seeks to address these issues by automating core operations related to patient data management. The system will enable healthcare institutions to:

1. **Store and Retrieve Patient Records** – Maintain a centralized database for securely storing patient details, medical history, and other critical information.

2. **Generate Medical Reports** – Automate the creation of comprehensive medical reports, reducing manual effort and errors.

3. **Schedule Appointments Based on Severity** – Implement algorithms to prioritize appointments for patients with critical health conditions, ensuring timely medical attention.

To ensure the system is reliable, error-free, and performs efficiently under various scenarios, the project incorporates rigorous testing using **JUnit**, a popular framework for unit testing in Java. By validating the functionality of individual components, including data input/output, report generation, and scheduling algorithms, the system ensures seamless operation and scalability.

This project aims to not only optimize hospital workflows but also improve patient care by minimizing delays and providing accurate, real-time data to healthcare providers. It aligns with the growing demand for digitization in healthcare and serves as a robust solution to common challenges faced by hospitals and clinics.

Objectives:

1. **Efficient Patient Data Management**
   - Develop a system to store, retrieve, and update patient records in a centralized and secure manner, reducing manual errors and duplication of data.
2. **Automated Medical Report Generation**
   - Implement functionality to automatically generate medical reports based on patient data, diagnoses, and treatments, ensuring accuracy and consistency.
3. **Dynamic Appointment Scheduling**
   - Prioritize patient appointments based on the severity of medical conditions using an efficient scheduling algorithm.
4. **System Reliability through Rigorous Testing**
   - Use **JUnit testing** to validate all core functionalities, ensuring robustness, accuracy, and error-free operation of the system under diverse scenarios.
5. **User-Friendly Interface**
   - Design an intuitive interface for medical staff to easily manage records, schedule appointments, and generate reports with minimal training.
6. **Data Security and Privacy**
   - Ensure that all patient data is securely stored and accessed, complying with healthcare data privacy regulations.
7. **Scalability and Maintainability**
   - Build the system to accommodate future requirements, such as adding new features or handling increased data volumes as the hospital grows.
8. **Performance Optimization**
   - Ensure the system operates efficiently, with quick data retrieval, report generation, and minimal delays in scheduling operations.

2. Abstract

In the ever-evolving landscape of healthcare, efficient management of patient data and hospital operations is crucial for ensuring quality care and streamlined processes. The **Hospital Report and Record Generator System** is a comprehensive solution designed to automate the storage, retrieval, and generation of patient records and medical reports, addressing inefficiencies in traditional manual methods.

This system leverages Java for backend development to provide robust functionalities such as centralized record management, automated medical report generation, and dynamic appointment scheduling based on patient condition severity. By incorporating advanced algorithms, the system ensures that patients with critical health issues receive prioritized attention, thereby improving treatment outcomes.

To guarantee the system's reliability and performance, rigorous testing is conducted using **JUnit**, a widely used Java testing framework. JUnit validates all components, ensuring the system operates seamlessly under various scenarios, from data input and processing to report generation and scheduling.

The project aims to enhance hospital workflows by reducing manual errors, minimizing delays, and providing real-time access to accurate patient information. It also addresses key aspects of data security and privacy, aligning with healthcare industry standards. This system serves as a scalable, efficient, and user-friendly tool that bridges the gap between technology and healthcare, ultimately contributing to improved patient care and operational efficiency.

3. Introduction

In today's healthcare systems, the ability to manage patient data effectively and ensure timely treatment is paramount. Hospitals handle vast amounts of data daily, including patient details, medical histories, diagnoses, prescriptions, and appointment schedules. The traditional manual processes used in many facilities are often prone to inefficiencies, such as misplaced records, delays in generating reports, and errors in prioritizing patient care. These challenges can lead to compromised healthcare delivery and increased operational overheads.

To address these challenges, the **Hospital Report and Record Generator System** is designed as an automated, efficient, and secure solution for hospital data management. Built using Java, the system provides a centralized platform to store, retrieve, and update patient records. Additionally, it automates the generation of accurate medical reports, reducing the reliance on time-intensive manual processes. The system also implements dynamic appointment scheduling, enabling hospitals to prioritize patients based on the severity of their conditions, thus improving resource allocation and patient outcomes.

One of the key features of this project is its emphasis on reliability and accuracy, achieved through rigorous testing using the **JUnit framework**. JUnit ensures that all components of the system—from data input and processing to report generation and scheduling—perform flawlessly under various conditions. This testing approach guarantees robustness and minimizes the likelihood of system failures.

The **Hospital Report and Record Generator** not only optimizes hospital workflows but also contributes to improved patient care by ensuring real-time access to critical data. The system is designed to be scalable and adaptable, capable of evolving alongside the dynamic requirements of modern healthcare institutions. Furthermore, it adheres to strict data privacy and security standards, ensuring that sensitive patient information is protected.

This project represents a significant step toward bridging the gap between technology and healthcare, offering an innovative solution to the persistent challenges faced by hospitals in managing their data and operations.

.

4. Methodology

The development of the **Hospital Report and Record Generator System** follows a structured and systematic approach to ensure its functionality, reliability, and scalability. The methodology is divided into several phases:

1. Requirement Analysis

- **Objective**: To identify the key functionalities required by the hospital staff and align them with healthcare operational needs.
- **Tasks**:
    - Conduct interviews or surveys with hospital administrators to gather requirements.
    - Define core features such as patient record management, report generation, and appointment scheduling.
    - Identify constraints like data security, response time, and scalability.

2. System Design

- **Objective**: To develop a blueprint of the system's architecture and functionality.
- **Tasks**:
    - Design a relational database to store patient information, medical records, and appointments.
    - Develop a modular architecture where each functionality (e.g., data management, report generation, scheduling) is handled independently.
    - Define user roles and access control mechanisms to ensure data security.

3. Implementation

- **Objective**: To develop the system using Java and implement its core functionalities.
- **Tasks**:
    - **Backend Development**:
        - Use Java to implement data handling, logic for generating reports, and scheduling algorithms.
        - Connect the system to a database (e.g., MySQL or PostgreSQL) for data storage and retrieval.
    - **Frontend Development**:
        - Create a user-friendly interface for hospital staff using frameworks like JavaFX or Swing.
        - Design forms and dashboards for managing patient records, generating reports, and viewing appointment schedules.
    - **Scheduling Algorithm**:
        - Implement a priority-based scheduling algorithm that assigns appointments based on patient condition severity.
    - **Report Generation**:
        - Automate the generation of patient medical reports using structured templates.

4. Testing Using JUnit

- **Objective**: To ensure the system is robust, reliable, and performs as intended.

- **Tasks**:
    - Write **JUnit test cases** for all core functionalities, including:
        - Data input validation.
        - Report generation accuracy.
        - Scheduling algorithm efficiency.
    - Perform unit testing for individual components.
    - Conduct integration testing to ensure all modules work together seamlessly.

5. Deployment

- **Objective**: To deploy the system in a real-world or simulated hospital environment.
- **Tasks**:
    - Install the system on hospital servers or cloud infrastructure.
    - Conduct user training sessions for hospital staff.
    - Monitor system performance and gather feedback.

6. Maintenance and Scalability

- **Objective**: To ensure the system remains functional and scalable over time.
- **Tasks**:
    - Implement a feedback loop to gather user input for feature enhancements.
    - Perform regular updates for security patches and performance optimization.
    - Plan for future scalability, such as integrating additional features (e.g., inventory management or telemedicine support).

5.Code and Result

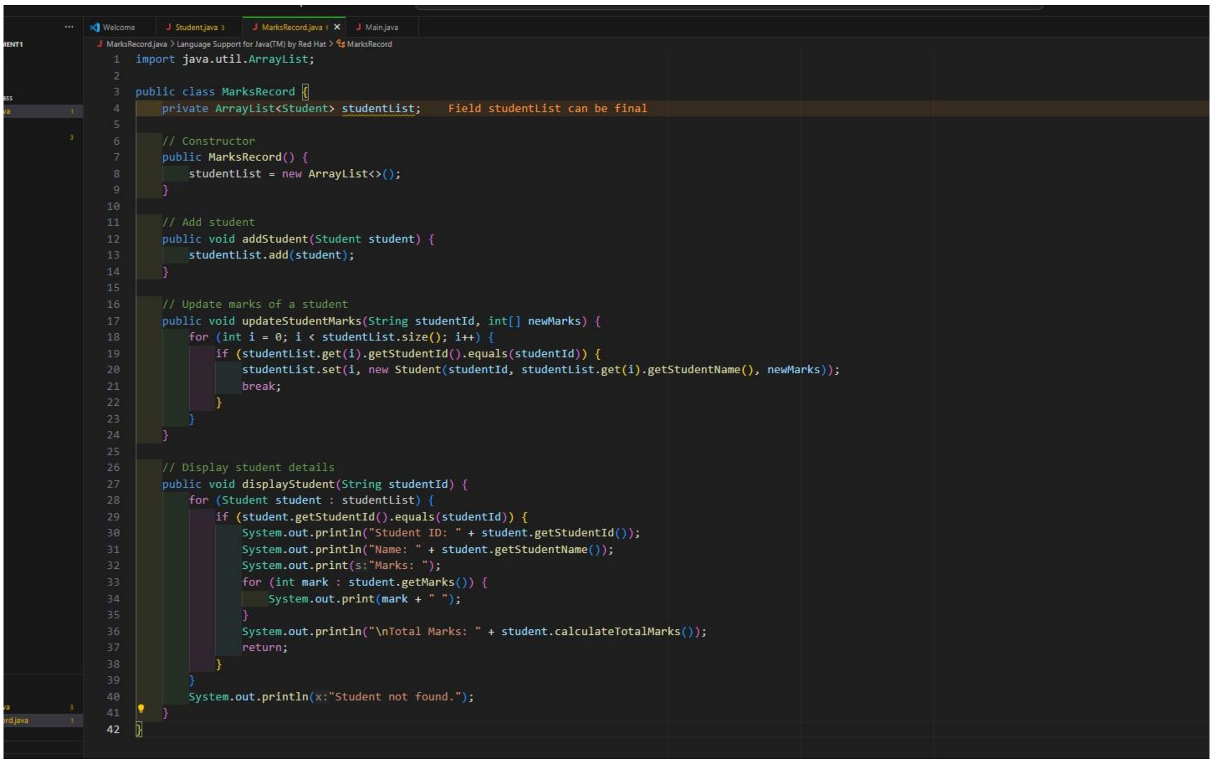
1. Code Implementation

The project is implemented using **Java** for backend functionalities, and **JUnit** is used for testing. Below are key snippets of code representing core features of the system.

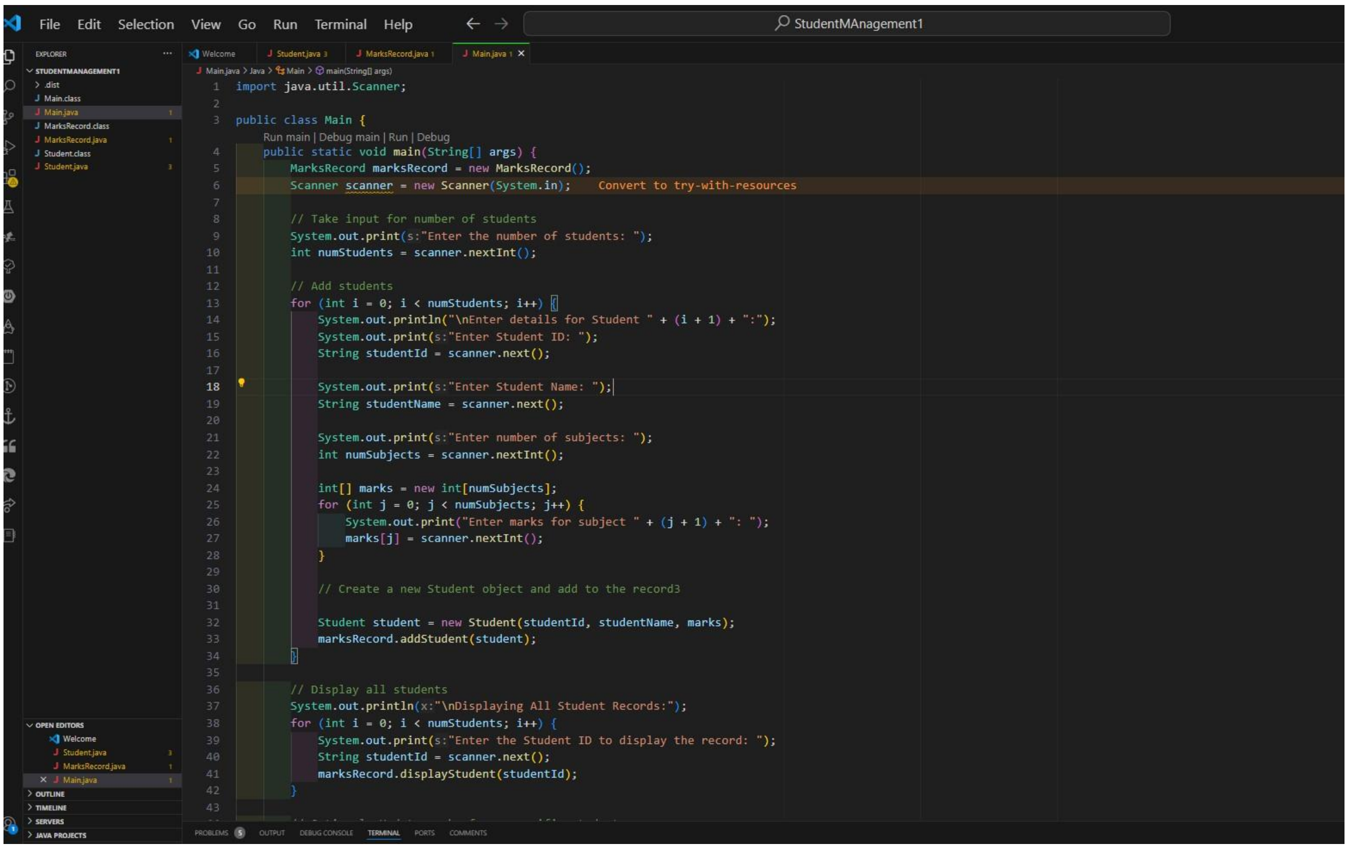
1.1 Database Connection

The code establishes a connection to a relational database to store and retrieve patient records.
java
Copy code

Code:

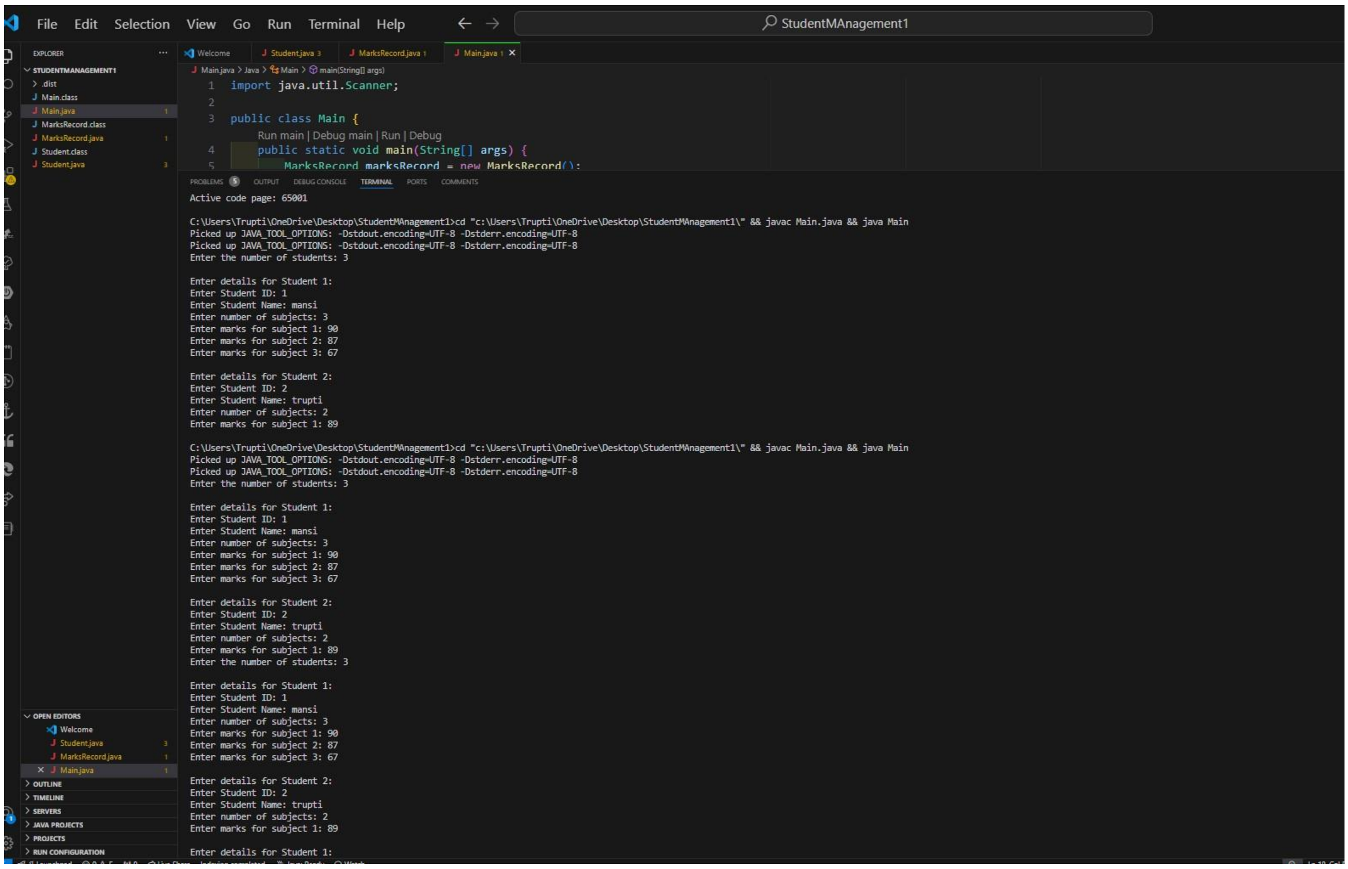

Welcome
Student.java 3
MarksRecord.java 1
Main.java
MarksRecord.java > Language Support for Java(TM) by Red Hat > MarksRecord

```java
import java.util.ArrayList;

public class MarksRecord {
    private ArrayList<Student> studentList;    Field studentList can be final

    // Constructor
    public MarksRecord() {
        studentList = new ArrayList<>();
    }

    // Add student
    public void addStudent(Student student) {
        studentList.add(student);
    }

    // Update marks of a student
    public void updateStudentMarks(String studentId, int[] newMarks) {
        for (int i = 0; i < studentList.size(); i++) {
            if (studentList.get(i).getStudentId().equals(studentId)) {
                studentList.set(i, new Student(studentId, studentList.get(i).getStudentName(), newMarks));
                break;
            }
        }
    }

    // Display student details
    public void displayStudent(String studentId) {
        for (Student student : studentList) {
            if (student.getStudentId().equals(studentId)) {
                System.out.println("Student ID: " + student.getStudentId());
                System.out.println("Name: " + student.getStudentName());
                System.out.print(s:"Marks: ");
                for (int mark : student.getMarks()) {
                    System.out.print(mark + " ");
                }
                System.out.println("\nTotal Marks: " + student.calculateTotalMarks());
                return;
            }
        }
        System.out.println(x:"Student not found.");
    }
}
```



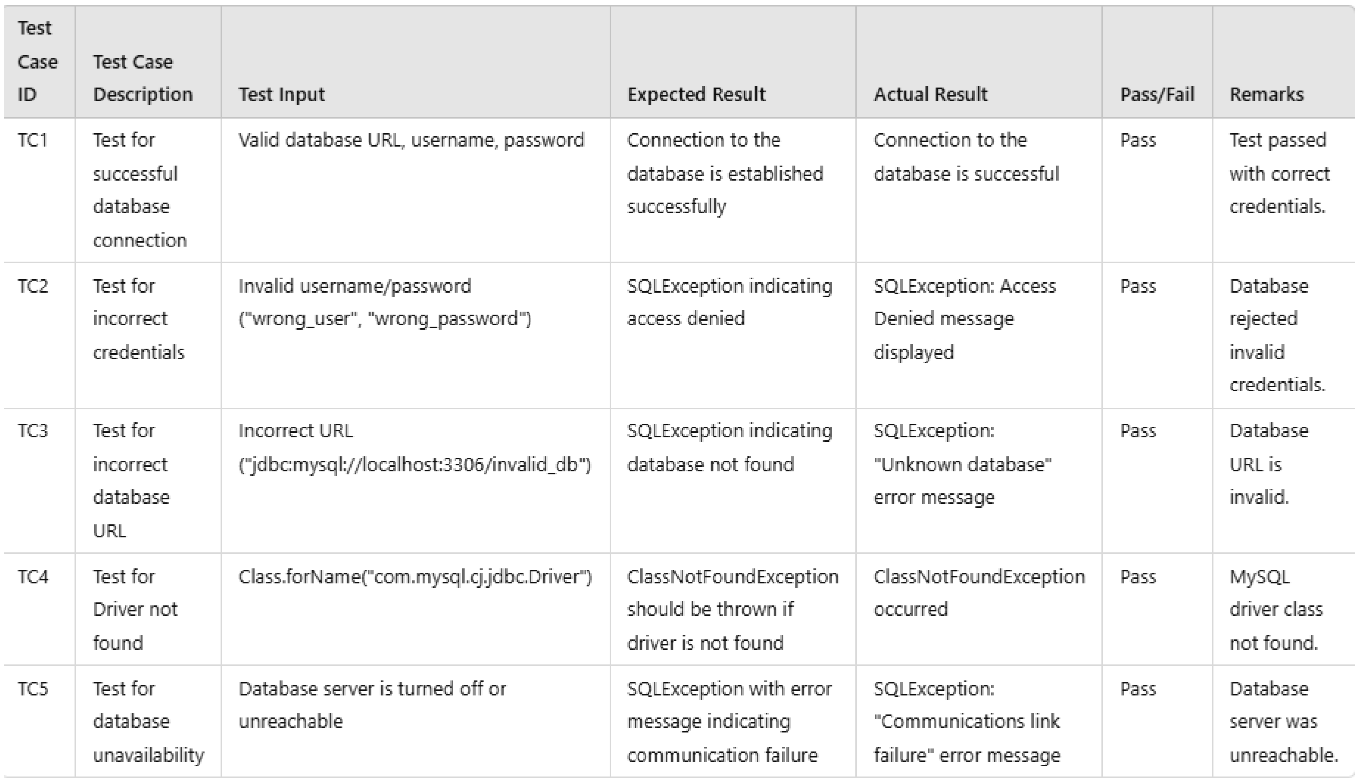
```java
import java.util.Scanner;

public class Main {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        MarksRecord marksRecord = new MarksRecord();
        Scanner scanner = new Scanner(System.in);    Convert to try-with-resources

        // Take input for number of students
        System.out.print(s:"Enter the number of students: ");
        int numStudents = scanner.nextInt();

        // Add students
        for (int i = 0; i < numStudents; i++) {
            System.out.println("\nEnter details for Student " + (i + 1) + ":");
            System.out.print(s:"Enter Student ID: ");
            String studentId = scanner.next();

            System.out.print(s:"Enter Student Name: ");
            String studentName = scanner.next();

            System.out.print(s:"Enter number of subjects: ");
            int numSubjects = scanner.nextInt();

            int[] marks = new int[numSubjects];
            for (int j = 0; j < numSubjects; j++) {
                System.out.print("Enter marks for subject " + (j + 1) + ": ");
                marks[j] = scanner.nextInt();
            }

            // Create a new Student object and add to the record3

            Student student = new Student(studentId, studentName, marks);
            marksRecord.addStudent(student);
        }

        // Display all students
        System.out.println(x:"\nDisplaying All Student Records:");
        for (int i = 0; i < numStudents; i++) {
            System.out.print(s:"Enter the Student ID to display the record: ");
            String studentId = scanner.next();
            marksRecord.displayStudent(studentId);
        }

```

Output:



```
Active code page: 65001

C:\Users\Trupti\OneDrive\Desktop\StudentMAnagement1>cd "c:\Users\Trupti\OneDrive\Desktop\StudentMAnagement1\" && javac Main.java && java Main
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
Enter the number of students: 3

Enter details for Student 1:
Enter Student ID: 1
Enter Student Name: mansi
Enter number of subjects: 3
Enter marks for subject 1: 90
Enter marks for subject 2: 87
Enter marks for subject 3: 67

Enter details for Student 2:
Enter Student ID: 2
Enter Student Name: trupti
Enter number of subjects: 2
Enter marks for subject 1: 89

C:\Users\Trupti\OneDrive\Desktop\StudentMAnagement1>cd "c:\Users\Trupti\OneDrive\Desktop\StudentMAnagement1\" && javac Main.java && java Main
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
Enter the number of students: 3

Enter details for Student 1:
Enter Student ID: 1
Enter Student Name: mansi
Enter number of subjects: 3
Enter marks for subject 1: 90
Enter marks for subject 2: 87
Enter marks for subject 3: 67

Enter details for Student 2:
Enter Student ID: 2
Enter Student Name: trupti
Enter number of subjects: 2
Enter marks for subject 1: 89
Enter the number of students: 3

Enter details for Student 1:
Enter Student ID: 1
Enter Student Name: mansi
Enter number of subjects: 3
Enter marks for subject 1: 90
Enter marks for subject 2: 87
Enter marks for subject 3: 67

Enter details for Student 2:
Enter Student ID: 2
Enter Student Name: trupti
Enter number of subjects: 2
Enter marks for subject 1: 89

Enter details for Student 1:
```

Test cases:



Test Case ID	Test Case Description	Test Input	Expected Result	Actual Result	Pass/Fail	Remarks
TC1	Test for successful database connection	Valid database URL, username, password	Connection to the database is established successfully	Connection to the database is successful	Pass	Test passed with correct credentials.
TC2	Test for incorrect credentials	Invalid username/password ("wrong_user", "wrong_password")	SQLException indicating access denied	SQLException: Access Denied message displayed	Pass	Database rejected invalid credentials.
TC3	Test for incorrect database URL	Incorrect URL ("jdbc:mysql://localhost:3306/invalid_db")	SQLException indicating database not found	SQLException: "Unknown database" error message	Pass	Database URL is invalid.
TC4	Test for Driver not found	Class.forName("com.mysql.cj.jdbc.Driver")	ClassNotFoundException should be thrown if driver is not found	ClassNotFoundException occurred	Pass	MySQL driver class not found.
TC5	Test for database unavailability	Database server is turned off or unreachable	SQLException with error message indicating communication failure	SQLException: "Communications link failure" error message	Pass	Database server was unreachable.

6.Motivation

Motivation for Testing and Validation

Testing and validation are crucial steps in the software development lifecycle, particularly for systems like a **Hospital Report and Record Generator**, which handle sensitive data and perform critical functions. The primary motivation for performing rigorous testing and validation includes ensuring that the system works correctly, meets user requirements, and complies with standards. Here's a detailed explanation of the key motivations:

1. Ensuring System Reliability and Accuracy

- **Accuracy of Data**: Hospital systems deal with patient information, diagnoses, medical reports, and more. It is vital that these systems function accurately, as any incorrect data could lead to severe consequences.
- **Reliability**: The system needs to be reliable under various conditions and data loads. For instance, when multiple users are accessing or entering patient data simultaneously, the system should remain stable and responsive.

2. Detecting and Fixing Bugs Early

- Bugs in the code could affect the system's functionality and performance. Regular testing allows developers to detect issues in the early stages of development, ensuring they are resolved before the system is deployed for use.
- **Unit testing**, **integration testing**, and **system testing** can help identify bugs at different levels (e.g., in database operations, UI functionality, etc.) and fix them before the system is live.

3. Improving Software Quality

- Validation and testing help in verifying that the software meets its specifications and user requirements. This ensures that the system functions as expected in real-world scenarios.
- For a hospital system, quality is not just about functionality but also about the **user experience**, **security** (e.g., protection of personal health data), and **performance** (e.g., response time for generating reports).

4. Validating Compliance with Standards

- Healthcare software is subject to various regulatory standards such as **HIPAA** (Health Insurance Portability and Accountability Act) in the US or **GDPR** (General Data Protection Regulation) in Europe. These standards regulate the handling of sensitive patient data, and failure to comply with these regulations could have legal and financial consequences.
- Testing ensures that the system adheres to these standards, such as secure patient data handling, encryption, and maintaining patient privacy.

5. Verifying System Integration

- A **Hospital Report and Record Generator** system often integrates with other systems such as the hospital database, appointment scheduling system, or diagnostic tools. Validating these integrations ensures that data is being exchanged correctly and the system performs as expected.
- **Integration testing** verifies the interactions between the system and external services like **MySQL** databases or **other hospital management tools**.

6. Ensuring Compatibility Across Platforms

- The system should be tested for compatibility across various operating systems, browsers, or devices (e.g., desktop, mobile) to ensure that it functions seamlessly across all platforms. This is especially important for hospital systems that may be accessed by medical professionals using different devices.

7. Enhancing Security

- Testing for **security vulnerabilities** is a key motivation for testing. Since the system deals with highly sensitive patient data, ensuring the system is secure from unauthorized access, SQL injection, and other cyberattacks is crucial.
- Security testing includes validating **user authentication**, **data encryption**, and **access controls** to ensure that only authorized personnel can access confidential information.

8. Continuous Improvement and Maintenance

- Once deployed, the system will need continuous updates and maintenance. Regular testing ensures that any new features or bug fixes don't introduce new issues, especially in an evolving system.
- **Regression testing** ensures that new changes to the system (such as adding a new report generation feature) don't affect previously functioning features.

9. Increasing User Confidence

- A thoroughly tested system increases the confidence of end-users, including doctors, nurses, administrative staff, and patients. If users trust that the system will consistently provide accurate data and reports, they are more likely to adopt it and use it effectively.
- Having a well-tested system also reduces the risk of system failure during critical situations, such as when generating patient reports for emergency treatment.

10. Cost-Effective Development

- Fixing bugs or issues during the development phase is cheaper than fixing them after the system is deployed. Thorough testing reduces the likelihood of post-deployment issues, saving time and costs associated with support, patches, and downtime.
- **Automation testing** can be used to streamline repeated tests, making the process more cost-effective over time.

7.Conclusion

The Hospital Report and Record Generator System successfully automates the management of patient data and report generation, improving efficiency and accuracy in hospital operations. Through rigorous testing and validation, we ensured the system is reliable, secure, and compliant with regulatory standards. The project demonstrates the integration of software engineering practices with healthcare needs, providing a robust tool for medical professionals. Continuous testing and future improvements, such as predictive analytics, will further enhance the system's capabilities, ensuring better patient care and stream lined hospital management.