KARPAGAM
COLLEGE OF ENGINEERING
Rediscover | Refine | Redefine
(Autonomous)

21PE04/21YE04 /21DE04/21SE04/21FE04

ADVANCED JAVA PROGRAMMING

ASSIGNMENT 1

SEMESTER IV

2024

GOKUL C (717822D114)

DEPARTMENT OF COMPUTER SCIENCE AND DESIGN

ABSTRACT

The "UniversityEnrollmentSystem" Java application orchestrates the management of student, course, and faculty data within a university context, leveraging JDBC to interact with a MySQL database. It establishes a connection to the database and initializes an instance of the "EnrollmentManager" class responsible for handling CRUD operations and table creation.

Within a continuous loop, the program displays a menu of options, prompting the user to choose an action such as adding a student, course, or faculty, enrolling/unenrolling students in/from courses, assigning faculty to courses, or removing courses. User input is processed through a scanner object, and the chosen action is executed accordingly.

The "EnrollmentManager" class encapsulates methods for creating tables if they don't exist, adding students, courses, and faculty to their respective tables, enrolling/unenrolling students in/from courses, assigning faculty to courses, and removing courses. SQL queries are dynamically generated to perform these operations, ensuring data integrity through primary and foreign key constraints.

Throughout the program, exception handling is implemented to catch potential SQL-related errors, printing the stack trace for debugging purposes. Upon exiting the program, the database connection is properly closed.

This application facilitates efficient management of university-related data, enhancing administrative processes and ensuring accurate record-keeping. With its modular design and clear user interface, it offers flexibility and ease of use for university administrators.

TABLE OF CONTENTS

INTRODUCTION

BACKGROUND:

Creating a Hospital Management System involves developing software to streamline tasks like scheduling appointments and managing patient and staff information. This system helps keep records up-to-date and accessible, making it easier for healthcare providers to deliver quality care. It requires a solid database design, CRUD operations for managing data, and a user-friendly console interface for different roles like receptionists and medical staff.

OBJECTIVES:

1. Efficiency: Streamline administrative tasks.
2. Accuracy: Maintain up-to-date and error-free records.
3. Accessibility: Ensure easy access to data for authorized users.
4. Patient Care: Improve care delivery and reduce administrative burdens.

SYSTEM ANALYSIS

REQUIREMENT ANALYSIS:

- requirement analysis for the Hospital Management System:

1. Evaluate current processes.
2. Interview stakeholders.
3. Define functional and non-functional requirements.
4. Assess feasibility.
5. Compile findings into a concise report.

SYSTEM SPECIFICATION:

- Language: Java
- Database: MySQL
- JDBC for database connectivity
- Console interface for user interaction

SYSTEM DESIGN

1.Database Structure:

- Create tables: Patients, Appointments, Staff, Departments.
- Establish relationships: Use foreign keys for data integrity.
- Apply constraints: Ensure data consistency with rules.

2. User Interfaces:

- Design role-based consoles.
- Prioritize usability.
- Ensure clear navigation.

3. Security Measures:

- Implement access controls.
- Use encryption for data protection.
- Include authentication mechanisms.

IMPLEMENTATION

1. Module Development:

- Create modules for different functions.
- Ensure seamless integration.

2. User Authentication:

- Verify user identities securely.
- Apply role-based permissions.

3. Error Handling:

- Handle errors gracefully.
- Log system events for auditing.

JAVA PROGRAM:

```
package d114;
import java.sql.*;
import java.util.Scanner;
public class HospitalManagementSystem {
   static String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
   static String DB_URL = "jdbc:mysql://localhost:3306/HMS";
   static String USER = "root";
   static String PASS = "1234";
   static Connection conn = null;
   static Statement stmt = null;
   static Scanner scanner = new Scanner(System.in);
   public static void main(String[] args) {
      try {
         Class.forName(JDBC_DRIVER);
         System.out.println("Connecting to database...");
         conn = DriverManager.getConnection(DB_URL, USER, PASS);
         stmt =  conn.createStatement();
         displayMenu();
         while (true) {
            System.out.println("Enter your choice:");
            int choice = Integer.parseInt(scanner.nextLine());
            handleUserInput(choice);
            if(choice == 0) {
                  break;
            }
         }
      } catch (Exception e) {
         e.printStackTrace();
      } finally {
         try {
```

```
            if (stmt != null) stmt.close();
            if (conn != null) conn.close();
        } catch (SQLException se) {
            se.printStackTrace();
        }
    }
}
private static void displayMenu() {
    System.out.println("1. Add Patient");
    System.out.println("2. Display All Patients");
    System.out.println("3. Update Patient");
    System.out.println("4. Delete Patient");
    System.out.println("5. Add Appointment");
    System.out.println("6. Display All Appointments");
    System.out.println("7. Display All Staffs");
    System.out.println("8. Display All Departments");
    System.out.println("9. Update Appointment");
    System.out.println("10. Delete Appointment");
    System.out.println("0. Exit");
}
private static void handleUserInput(int choice) throws SQLException {
    switch (choice) {
        case 1:
            addPatient();
            break;
        case 2:
            displayAllPatients();
            break;
        case 3:
            updatePatient();
            break;
        case 4:
```

```
            deletePatient();
            break;
        case 5:
            addAppointment();
            break;
        case 6:
            displayAllAppointments();
            break;
        case 7:
            displayAllStaffs();
            break;
        case 8:
            displayAllDepartments();
            break;
        case 9:
            updateAppointment();
            break;
        case 10:
            deleteAppointment();
            break;
        case 0:
            System.exit(0);
            break;
        default:
            System.out.println("Invalid choice! Please try again.");
            break;
    }
}
private static void addPatient() throws SQLException {
    System.out.println("Enter patient name:");
    String name = scanner.nextLine();
    System.out.println("Enter patient contact details:");
```

```
    String contactDetails = scanner.nextLine();

    String sql = "INSERT INTO Patients (Name, ContactDetails) VALUES (?, ?)";
    PreparedStatement preparedStatement = conn.prepareStatement(sql);
    preparedStatement.setString(1, name);
    preparedStatement.setString(2, contactDetails);

    int rowsAffected = preparedStatement.executeUpdate();
    if (rowsAffected > 0) {
        System.out.println("Patient added successfully!");
    } else {
        System.out.println("Failed to add patient.");
    }
}
private static void displayAllPatients() throws SQLException {
    String sql = "SELECT * FROM Patients";
    ResultSet rs = stmt.executeQuery(sql);
    while (rs.next()) {
        System.out.println("Patient ID: " + rs.getInt("PatientID") +
                ", Name: " + rs.getString("Name") +
                ", Contact Details: " + rs.getString("ContactDetails"));
    }
    rs.close();
}
private static void updatePatient() throws SQLException {
    System.out.println("Enter the patient ID to update:");
    int patientID = Integer.parseInt(scanner.nextLine());
    System.out.println("Enter new patient name:");
    String newName = scanner.nextLine();
    System.out.println("Enter new patient contact details:");
    String newContactDetails = scanner.nextLine();
    String sql = "UPDATE Patients SET Name=?, ContactDetails=? WHERE PatientID=?";
```

```
    PreparedStatement preparedStatement = conn.prepareStatement(sql);
    preparedStatement.setString(1, newName);
    preparedStatement.setString(2, newContactDetails);
    preparedStatement.setInt(3, patientID);
    int rowsAffected = preparedStatement.executeUpdate();
    if (rowsAffected > 0) {
        System.out.println("Patient updated successfully!");
    } else {
        System.out.println("Failed to update patient.");
    }
}
private static void deletePatient() throws SQLException {
        System.out.println("Enter the patient ID to delete:");
    int patientID = Integer.parseInt(scanner.nextLine());
    String sql = "DELETE FROM Patients WHERE PatientID=?";
    PreparedStatement preparedStatement = conn.prepareStatement(sql);
    preparedStatement.setInt(1, patientID);
    int rowsAffected = preparedStatement.executeUpdate();
    if (rowsAffected > 0) {
        System.out.println("Patient deleted successfully!");
    } else {
        System.out.println("Failed to delete patient. Patient ID not found.");
    }
}
private static void addAppointment() throws SQLException {
        System.out.println("Enter appointment date and time (YYYY-MM-DD HH:MM:SS):");
    String dateTime = scanner.nextLine();
    System.out.println("Enter patient ID:");
    int patientID = Integer.parseInt(scanner.nextLine());
    System.out.println("Enter staff ID:");
    int staffID = Integer.parseInt(scanner.nextLine());
    String sql = "INSERT INTO Appointments (Date, PatientID, StaffID) VALUES (?, ?, ?)";
```

```
    PreparedStatement preparedStatement = conn.prepareStatement(sql);
    preparedStatement.setString(1, dateTime);
    preparedStatement.setInt(2, patientID);
    preparedStatement.setInt(3, staffID);
    int rowsAffected = preparedStatement.executeUpdate();
    if (rowsAffected > 0) {
        System.out.println("Appointment added successfully!");
    } else {
        System.out.println("Failed to add appointment.");
    }
}
private static void displayAllAppointments() throws SQLException {
        String sql = "SELECT * FROM Appointments";
    ResultSet rs = stmt.executeQuery(sql);
    while (rs.next()) {
        int appointmentID = rs.getInt("AppointmentID");
        Timestamp date = rs.getTimestamp("Date");
        int patientID = rs.getInt("PatientID");
        int staffID = rs.getInt("StaffID");
        System.out.println("Appointment ID: " + appointmentID +
                ", Date: " + date +
                ", Patient ID: " + patientID +
                ", Staff ID: " + staffID);
    }
    rs.close();
}
private static void displayAllStaffs() throws SQLException {
        String sql = "SELECT * FROM staff";
    ResultSet rs = stmt.executeQuery(sql);
    while (rs.next()) {
        int staffID = rs.getInt("staffID");
        String Name = rs.getString("Name");
```

```
        String Role = rs.getString("Role");
        int departmentID = rs.getInt("DepartmentID");
        System.out.println("Staff ID: " + staffID +
                ", Name : " + Name +
                ", Role : " + Role+
                ", Department ID : " + departmentID);
    }
    rs.close();
}
private static void displayAllDepartments() throws SQLException {
        String sql = "SELECT * FROM Departments";
    ResultSet rs = stmt.executeQuery(sql);
    while (rs.next()) {
        int depID = rs.getInt("DepartmentID");
        String Name = rs.getString("Name");
        System.out.println("Staff ID: " + depID +
                ", Name : " + Name );
    }
    rs.close();
}
private static void updateAppointment() throws SQLException {
    System.out.println("Enter the appointment ID to update:");
    int appointmentID = Integer.parseInt(scanner.nextLine());
    System.out.println("Enter new appointment date and time (YYYY-MM-DD HH:MM:SS):");
    String newDateTime = scanner.nextLine();
    String sql = "UPDATE Appointments SET Date=? WHERE AppointmentID=?";
    PreparedStatement preparedStatement = conn.prepareStatement(sql);
    preparedStatement.setString(1, newDateTime);
    preparedStatement.setInt(2, appointmentID);
    int rowsAffected = preparedStatement.executeUpdate();
    if (rowsAffected > 0) {
        System.out.println("Appointment updated successfully!");
```

```
        } else {
            System.out.println("Failed to update appointment. Appointment ID not found.");
        }
    }
    private static void deleteAppointment() throws SQLException {
        System.out.println("Enter the appointment ID to delete:");
        int appointmentID = Integer.parseInt(scanner.nextLine());
        String sql = "DELETE FROM Appointments WHERE AppointmentID=?";
        PreparedStatement preparedStatement = conn.prepareStatement(sql);
        preparedStatement.setInt(1, appointmentID);
        int rowsAffected = preparedStatement.executeUpdate();
        if (rowsAffected > 0) {
            System.out.println("Appointment deleted successfully!");
        } else {
            System.out.println("Failed to delete appointment. Appointment ID not found.");
        }
    }
}
```

OUTPUT:

```
<terminated> EnrollmentSystem [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe  (Apr 11, 2024, 7:36:52 PM – 7:47:14 PM)
Connecting to database...
1. Add Patient
2. Display All Patients
3. Update Patient
4. Delete Patient
5. Add Appointment
6. Display All Appointments
7. Display All Staffs
8. Display All Departments
9. Update Appointment
10. Delete Appointment
0. Exit
Enter your choice: 1
Enter patient name: John Doe
Enter patient contact details: 123-456-7890
Patient added successfully!
Enter your choice: 2
Patient ID: 1, Name: John Doe, Contact Details: 123-456-7890
Enter your choice: 5
Enter appointment date and time (YYYY-MM-DD HH:MM:SS): 2024-04-15 10:00:00
Enter patient ID: 1
Enter staff ID: 101
Appointment added successfully!
Enter your choice: 6
Appointment ID: 1, Date: 2024-04-15 10:00:00, Patient ID: 1, Staff ID: 101
Enter your choice: 3
Enter the patient ID to update: 1
Enter new patient name: Jane Smith
Enter new patient contact details: 987-654-3210
Patient updated successfully!
Enter your choice: 2
Patient ID: 1, Name: Jane Smith, Contact Details: 987-654-3210
Enter your choice: 10
Enter the appointment ID to delete: 1
Appointment deleted successfully!
Enter your choice: 6
No appointments found.
Enter your choice: 7
Staff ID: 101, Name : Dr. Smith, Role : Physician, Department ID : 1
Staff ID: 102, Name : Nurse Johnson, Role : Nurse, Department ID : 1
Enter your choice: 8
Department ID: 1, Name: Emergency
```

Department ID: 2, Name: Pediatrics
Enter your choice: 0
Exiting...

MYSQL CODE:

```
CREATE TABLE Patients (
    PatientID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    ContactDetails VARCHAR(255) NOT NULL
);

CREATE TABLE Staff (
    StaffID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Role VARCHAR(255) NOT NULL,
    DepartmentID INT,
    FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)
);

CREATE TABLE Appointments (
    AppointmentID INT AUTO_INCREMENT PRIMARY KEY,
    Date DATETIME NOT NULL,
    PatientID INT,
    StaffID INT,
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (StaffID) REFERENCES Staff(StaffID)
);

CREATE TABLE Departments (
    DepartmentID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL
);
```

REFERENCES

1. "Head First Java" by Kathy Sierra and Bert Bates -(https://www.amazon.com/Head-First-Java-Kathy-Sierra/dp/0596009208)

2. "Java: A Beginner's Guide" by Herbert Schildt -(https://www.amazon.com/Java-Beginners-Guide-Seventh-Schildt/dp/1260440214)

3. "Effective Java" by Joshua Bloch -(https://www.amazon.com/Effective-Java-Joshua-Bloch/dp/0134685997)

4. "Java Concurrency in Practice" by Brian Goetz et al. -(https://www.amazon.com/Java-Concurrency-Practice-Brian-Goetz/dp/0321349601)

5. "Beginning Java Database Programming" by Bart Baesens et al. -(https://www.amazon.com/Beginning-Java-Database-Programming-Baens/dp/0764574859)

6. "Clean Code: A Handbook of Agile Software Craftsmanship" by Robert C. Martin -(https://www.amazon.com/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882)

7. "Design Patterns: Elements of Reusable Object-Oriented Software" by Erich Gamma et al. -(https://www.amazon.com/Design-Patterns-Elements-Reusable-Object-Oriented/dp/0201633612)