# Student Management System (JDBC-Based)

## 📌 Project Overview

This is a Student Management System built using Core Java and JDBC to perform CRUD (Create, Read, Update, Delete) operations on student data stored in a MySQL database.

## 🚀 Technologies Used

- Java (OOP concepts, Collections, Arrays, String, StringBuffer, StringBuilder)

- JDBC (Java Database Connectivity)

- MySQL (Database for storing student records)

## 🔹 Features

✅ Add a new student

✅ View all students

✅ Update student details

✅ Delete a student

✅ JDBC connection handling

## 🛠️ Setup Instructions

### 1️⃣ Database Configuration

Create a MySQL database and a table using the following SQL commands:

CREATE DATABASE student\_db;  
USE student\_db;  
  
CREATE TABLE students (  
 id INT AUTO\_INCREMENT PRIMARY KEY,  
 name VARCHAR(100) NOT NULL,  
 age INT NOT NULL,  
 grade VARCHAR(10) NOT NULL  
);

### 2️⃣ Update Database Connection in Java

Modify the `DBConnection.java` file with your MySQL credentials:

private static final String URL = "jdbc:mysql://localhost:3306/student\_db";  
private static final String USER = "your\_username";  
private static final String PASSWORD = "your\_password";

### 3️⃣ Run the Project

Compile and run `Main.java` to start the application.

javac Main.java  
java Main

## 📌 Code Explanation

### 🔹 1. JDBC Connection (DBConnection.java)

Handles the connection between Java and MySQL.

public class DBConnection {  
 private static final String URL = "jdbc:mysql://localhost:3306/student\_db";  
 private static final String USER = "root";  
 private static final String PASSWORD = "password";  
  
 public static Connection getConnection() throws SQLException {  
 return DriverManager.getConnection(URL, USER, PASSWORD);  
 }  
}

### 🔹 2. Student Class (Student.java)

Defines the Student object model.

public class Student {  
 private int id;  
 private String name;  
 private int age;  
 private String grade;  
  
 // Getters & Setters  
}

### 🔹 3. CRUD Operations (StudentDAO.java)

Implements functions to interact with the database.

public class StudentDAO {  
 public void addStudent(Student student) {  
 String query = "INSERT INTO students (name, age, grade) VALUES (?, ?, ?)";  
 try (Connection con = DBConnection.getConnection();  
 PreparedStatement ps = con.prepareStatement(query)) {  
 ps.setString(1, student.getName());  
 ps.setInt(2, student.getAge());  
 ps.setString(3, student.getGrade());  
 ps.executeUpdate();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}

### 🔹 4. Main Class (Main.java)

Handles user interaction via console-based menu.

public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 StudentDAO studentDAO = new StudentDAO();  
  
 System.out.println("Welcome to Student Management System");  
 while (true) {  
 System.out.println("1. Add Student\n2. View Students\n3. Update Student\n4. Delete Student\n5. Exit");  
 int choice = scanner.nextInt();  
 switch (choice) {  
 case 1:   
 // Logic to add student  
 break;  
 case 2:   
 // Logic to view students  
 break;  
 case 3:  
 // Logic to update student  
 break;  
 case 4:  
 // Logic to delete student  
 break;  
 case 5:  
 System.exit(0);  
 }  
 }  
 }  
}

## 📂 Folder Structure

StudentManagementSystem/  
│── src/  
│ ├── DBConnection.java  
│ ├── Student.java  
│ ├── StudentDAO.java  
│ ├── Main.java  
│── README.md

## 📌 How to Contribute

- Fork this repository.  
- Make your changes.  
- Submit a pull request!

## 📌 Author

Tejas Mohan Puri  
📧 tejaspuri789@gmail.com  
📍 Pune, India