Student Info System Dev

You need to develop an information system with a backend (MySQL DB and server-side) and a front-end (Client-side; web, Java console app). You can pick your own domain (a topic) and a platform (Spring
Boot, etc.) to develop an application with webservices (both server and client). However, all four CRUD
operations need to be implemented at least for one entity. Besides, students need to maintain a GitHub
repository with regular submissions and meaningful comments on their development (All the commits need
to be error free as it is possible to ask you to pull from any point and run).
system:
• A student information system that mange (Add, edit, view, delete) four attributes such as IndexNo, Name, DoB, and GPA of students.

Give me the complete code for this project with front end.

Below is the complete code for a student information system. The backend uses **Spring Boot** with a MySQL database, and the front end uses **HTML/CSS/JavaScript**. It implements CRUD operations for the attributes: IndexNo, Name, DoB, and GPA.

Backend: Spring Boot with MySQL

1. Create the Spring Boot project

Use **Spring Initializr** or your preferred IDE to generate a Spring Boot project with the following dependencies:

- Spring Web
- Spring Data JPA

- MySQL Driver
- Spring Boot DevTools

---

2. application.properties

```
properties


  spring.datasource.url=jdbc:mysql://localhost:3306/student_db
  spring.datasource.username=root
  spring.datasource.password=yourpassword
  spring.jpa.hibernate.ddl-auto=update
  spring.jpa.show-sql=true
```

---

3. Entity Class

```
java


  package com.example.studentinfo.model;

  import jakarta.persistence.*;
  import java.time.LocalDate;

  @Entity
  public class Student {
      @Id
      @GeneratedValue(strategy = GenerationType.IDENTITY)
      private Long id;
      private String indexNo;
      private String name;
      private LocalDate dob;
      private Double gpa;

      // Getters and Setters
  }
```

---

4. Repository

```
java


  package com.example.studentinfo.repository;

  import com.example.studentinfo.model.Student;
  import org.springframework.data.jpa.repository.JpaRepository;

  public interface StudentRepository extends JpaRepository<Student, Long> {
  }
```

---

5. Service

```java
package com.example.studentinfo.service;

import com.example.studentinfo.model.Student;
import com.example.studentinfo.repository.StudentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class StudentService {
    @Autowired
    private StudentRepository studentRepository;

    public List<Student> getAllStudents() {
        return studentRepository.findAll();
    }

    public Student getStudentById(Long id) {
        return studentRepository.findById(id).orElse(null);
    }

    public Student addStudent(Student student) {
        return studentRepository.save(student);
    }

    public Student updateStudent(Long id, Student studentDetails) {
        Student student = getStudentById(id);
        if (student != null) {
            student.setIndexNo(studentDetails.getIndexNo());
            student.setName(studentDetails.getName());
            student.setDob(studentDetails.getDob());
            student.setGpa(studentDetails.getGpa());
            return studentRepository.save(student);
        }
        return null;
    }

    public void deleteStudent(Long id) {
        studentRepository.deleteById(id);
    }
}
```

---

6. Controller

```java
package com.example.studentinfo.controller;

import com.example.studentinfo.model.Student;
import com.example.studentinfo.service.StudentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/students")
public class StudentController {
    @Autowired
    private StudentService studentService;
```

```
    @GetMapping
    public List<Student> getAllStudents() {
        return studentService.getAllStudents();
    }

    @GetMapping("/{id}")
    public Student getStudentById(@PathVariable Long id) {
        return studentService.getStudentById(id);
    }

    @PostMapping
    public Student addStudent(@RequestBody Student student) {
        return studentService.addStudent(student);
    }

    @PutMapping("/{id}")
    public Student updateStudent(@PathVariable Long id, @RequestBody Student student) {
        return studentService.updateStudent(id, student);
    }

    @DeleteMapping("/{id}")
    public void deleteStudent(@PathVariable Long id) {
        studentService.deleteStudent(id);
    }
}
```

Frontend: HTML/CSS/JavaScript

1. index.html

```
html


<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Information System</title>
    <style>
        body {
            font-family: Arial, sans-serif;
        }
        .form-container {
            margin: 20px;
        }
        table {
            width: 100%;
            border-collapse: collapse;
        }
        table, th, td {
            border: 1px solid black;
        }
        th, td {
            padding: 8px;
            text-align: left;
        }
    </style>
</head>
<body>
    <h1>Student Information System</h1>
    <div class="form-container">
        <form id="studentForm">
            <label>Index No: <input type="text" id="indexNo"></label><br>
```

```
            <label>Name: <input type="text" id="name"></label><br>
            <label>Date of Birth: <input type="date" id="dob"></label><br>
            <label>GPA: <input type="number" step="0.01" id="gpa"></label><br>
            <button type="button" onclick="addStudent()">Add Student</button>
        </form>
    </div>
    <div>
        <h2>Student List</h2>
        <table>
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Index No</th>
                    <th>Name</th>
                    <th>Date of Birth</th>
                    <th>GPA</th>
                    <th>Actions</th>
                </tr>
            </thead>
            <tbody id="studentTable"></tbody>
        </table>
    </div>

    <script>
        const API_URL = "http://localhost:8080/api/students";

        function fetchStudents() {
            fetch(API_URL)
                .then(response => response.json())
                .then(data => {
                    const table = document.getElementById('studentTable');
                    table.innerHTML = '';
                    data.forEach(student => {
                        table.innerHTML += `
                            <tr>
                                <td>${student.id}</td>
                                <td>${student.indexNo}</td>
                                <td>${student.name}</td>
                                <td>${student.dob}</td>
                                <td>${student.gpa}</td>
                                <td>
                                    <button onclick="deleteStudent(${student.id})">Delete</b
utton>
                                </td>
                            </tr>
                        `;
                    });
                });
        }

        function addStudent() {
            const student = {
                indexNo: document.getElementById('indexNo').value,
                name: document.getElementById('name').value,
                dob: document.getElementById('dob').value,
                gpa: document.getElementById('gpa').value,
            };

            fetch(API_URL, {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify(student)
            }).then(() => fetchStudents());
        }

        function deleteStudent(id) {
            fetch(`${API_URL}/${id}`, {
                method: 'DELETE',
            }).then(() => fetchStudents());
        }
```

```
            window.onload = fetchStudents;
        </script>
    </body>
    </html>
```

This setup implements all CRUD operations and ensures seamless interaction between the front end and the backend via RESTful web services. Ensure you have MySQL running and the database created before starting the Spring Boot application.