Student Entity:

```
package com.example.studentmanagement;

import jakarta.persistence.*;
import jakarta.validation.constraints.*;
import lombok.*;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Student {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long studentId;

    @NotBlank(message = "Name is required")
    private String name;

    @Min(value = 3, message = "Age must be at least 3")
    @Max(value = 100, message = "Age must be less than 100")
    private Integer age;

    @NotBlank(message = "Class is required")
    private String studentClass;

    @Email(message = "Email should be valid")
    private String email;

    private String address;
}
```

StudentRepository Interface:

```
package com.example.studentmanagement;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;

public interface StudentRepository extends JpaRepository<Student, Long> {
    Page<Student> findByNameContainingIgnoreCaseOrStudentClassContainingIgnoreCase(String
name, String studentClass, Pageable pageable);
}
```

StudentService Class:

```
package com.example.studentmanagement;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;

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

    public Page<Student> listAll(String keyword, Pageable pageable) {
        if (keyword != null && !keyword.isEmpty()) {
                                                                                return
repo.findByNameContainingIgnoreCaseOrStudentClassContainingIgnoreCase(keyword,    keyword,
pageable);
        }
        return repo.findAll(pageable);
    }

    public Student save(Student student) {
        return repo.save(student);
    }

    public Student get(Long id) {
        return repo.findById(id).orElse(null);
    }

    public void delete(Long id) {
        repo.deleteById(id);
    }
}
```

StudentController Class:

```
package com.example.studentmanagement;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import jakarta.validation.Valid;

@Controller
public class StudentController {
    @Autowired
    private StudentService service;

    @GetMapping("/")
    public String viewHomePage(Model model,
        @RequestParam(defaultValue = "0") int page,
        @RequestParam(defaultValue = "") String keyword) {
        Page<Student> studentsPage = service.listAll(keyword, PageRequest.of(page, 5));
        model.addAttribute("studentsPage", studentsPage);
        model.addAttribute("currentPage", page);
        model.addAttribute("keyword", keyword);
        return "index";
    }

    @GetMapping("/new")
    public String showNewStudentForm(Model model) {
        model.addAttribute("student", new Student());
        return "new_student";
    }

    @PostMapping("/save")
      public String saveStudent(@Valid @ModelAttribute("student") Student student, BindingResult
bindingResult) {
        if (bindingResult.hasErrors()) {
            return "new_student";
        }
        service.save(student);
        return "redirect:/";
    }
```

```
    @GetMapping("/edit/{id}")
    public String showEditForm(@PathVariable("id") Long id, Model model) {
        Student student = service.get(id);
        if (student == null) {
            return "redirect:/";
        }
        model.addAttribute("student", student);
        return "edit_student";
    }

    @PostMapping("/update/{id}")
      public String updateStudent(@PathVariable("id") Long id, @Valid @ModelAttribute("student")
Student student, BindingResult bindingResult) {
        if (bindingResult.hasErrors()) {
            return "edit_student";
        }
        student.setStudentId(id);
        service.save(student);
        return "redirect:/";
    }

    @GetMapping("/delete/{id}")
    public String deleteStudent(@PathVariable("id") Long id) {
        service.delete(id);
        return "redirect:/";
    }
}
```

Thymeleaf Template - index.html:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
   <title>Student List</title>
</head>
<body>
<h1>Student List</h1>
<form method="get" action="/" th:action="@{/}">
            <input type="text" name="keyword" placeholder="Search by name or class" th:value="${keyword}">
   <button type="submit">Search</button>
</form>
<a href="/new">Add New Student</a>
<table border="1">
   <tr>

<th>ID</th><th>Name</th><th>Age</th><th>Class</th><th>Email</th><th>Address</th><th>Actions</th>
   </tr>
   <tr th:each="student : ${studentsPage.content}">
      <td th:text="${student.studentId}"></td>
      <td th:text="${student.name}"></td>
      <td th:text="${student.age}"></td>
      <td th:text="${student.studentClass}"></td>
      <td th:text="${student.email}"></td>
      <td th:text="${student.address}"></td>
      <td>
         <a th:href="@{'/edit/' + ${student.studentId}}">Edit</a> |
                  <a th:href="@{'/delete/' + ${student.studentId}}" onclick="return confirm('Are you sure?')">Delete</a>
      </td>
   </tr>
</table>
<div>
                  <span>Page: <span th:text="${currentPage + 1}"></span> / <span th:text="${studentsPage.totalPages}"></span></span>
   <div th:if="${studentsPage.hasPrevious()}">
      <a th:href="@{/(page=${currentPage - 1}, keyword=${keyword})}">Previous</a>
   </div>
   <div th:if="${studentsPage.hasNext()}">
      <a th:href="@{/(page=${currentPage + 1}, keyword=${keyword})}">Next</a>
   </div>
</div>
```

</body>
</html>

Thymeleaf Template - new_student.html:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Add New Student</title>
</head>
<body>
<h1>Add New Student</h1>
<form th:action="@{/save}" th:object="${student}" method="post">
    <label>Name: </label><input type="text" th:field="*{name}"/>
    <div th:if="${#fields.hasErrors('name')}" th:errors="*{name}"></div>

    <label>Age: </label><input type="number" th:field="*{age}"/>
    <div th:if="${#fields.hasErrors('age')}" th:errors="*{age}"></div>

    <label>Class: </label><input type="text" th:field="*{studentClass}"/>
    <div th:if="${#fields.hasErrors('studentClass')}" th:errors="*{studentClass}"></div>

    <label>Email: </label><input type="email" th:field="*{email}"/>
    <div th:if="${#fields.hasErrors('email')}" th:errors="*{email}"></div>

    <label>Address: </label><input type="text" th:field="*{address}"/>

    <button type="submit">Save</button>
</form>
<a href="/">Back</a>
</body>
</html>
```

Thymeleaf Template - edit_student.html:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Edit Student</title>
</head>
<body>
<h1>Edit Student</h1>
<form th:action="@{'/update/' + ${student.studentId}}" th:object="${student}" method="post">
    <label>Name: </label><input type="text" th:field="*{name}"/>
    <div th:if="${#fields.hasErrors('name')}" th:errors="*{name}"></div>

    <label>Age: </label><input type="number" th:field="*{age}"/>
    <div th:if="${#fields.hasErrors('age')}" th:errors="*{age}"></div>

    <label>Class: </label><input type="text" th:field="*{studentClass}"/>
    <div th:if="${#fields.hasErrors('studentClass')}" th:errors="*{studentClass}"></div>

    <label>Email: </label><input type="email" th:field="*{email}"/>
    <div th:if="${#fields.hasErrors('email')}" th:errors="*{email}"></div>

    <label>Address: </label><input type="text" th:field="*{address}"/>

    <button type="submit">Update</button>
</form>
<a href="/">Back</a>
</body>
</html>
```

Explanation:

This application helps manage student records for a school or college. You can add new students, update their info, delete students, and see a list of all students.
It saves the data in a database automatically. You can also search students by their name or class and see the results page by page. The web pages use a simple design to show forms and lists,
and the app makes sure you enter correct information like valid emails and ages. If there are errors, it shows friendly messages so you can fix them.