

Lab 05 – Nguyễn Hà Thanh – ITITIU21144

1. LIST – Navigate to /student

When the user navigates to /student without providing an action parameter, the controller defaults the action to 'list':

ID	STUDENT CODE	FULL NAME	EMAIL	MAJOR	ACTIONS
5	SV005	David Wilson	david.w@email.com	Computer Science	<button>Edit</button> <button>Delete</button>
4	SV004	Sarah Davis	sarah.d@email.com	Data Science	<button>Edit</button> <button>Delete</button>
3	SV003	Michael Brown	michael.b@email.com	Software Engineering	<button>Edit</button> <button>Delete</button>
2	SV002	Emily Johnson	emily.j@email.com	Information Technology	<button>Edit</button> <button>Delete</button>

The controller then calls `listStudents()`, which retrieves all students from the database:

```
List<Student> students = studentDAO.getAllStudents();
```

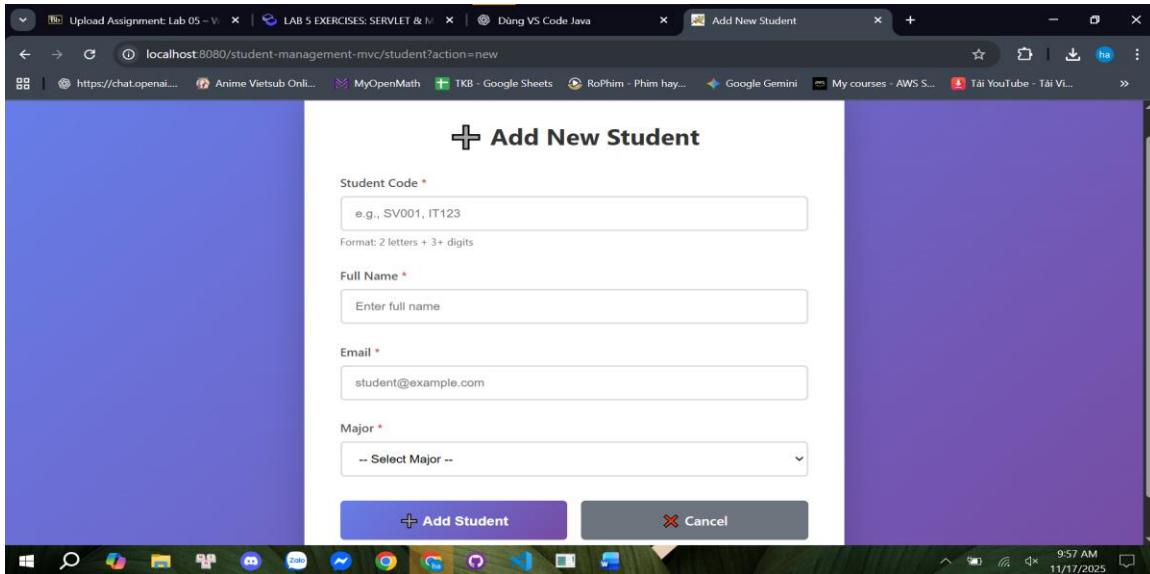
```
"request.setAttribute("students", students);"
```

Finally, it forwards the request to `student-list.jsp`:

`student-list.jsp` loops through the list and displays each student in a table. This shows all existing students retrieved from the database.

2. ADD – Add New Student

When the user clicks “Add New Student”, the browser goes to: “/student?action=new”



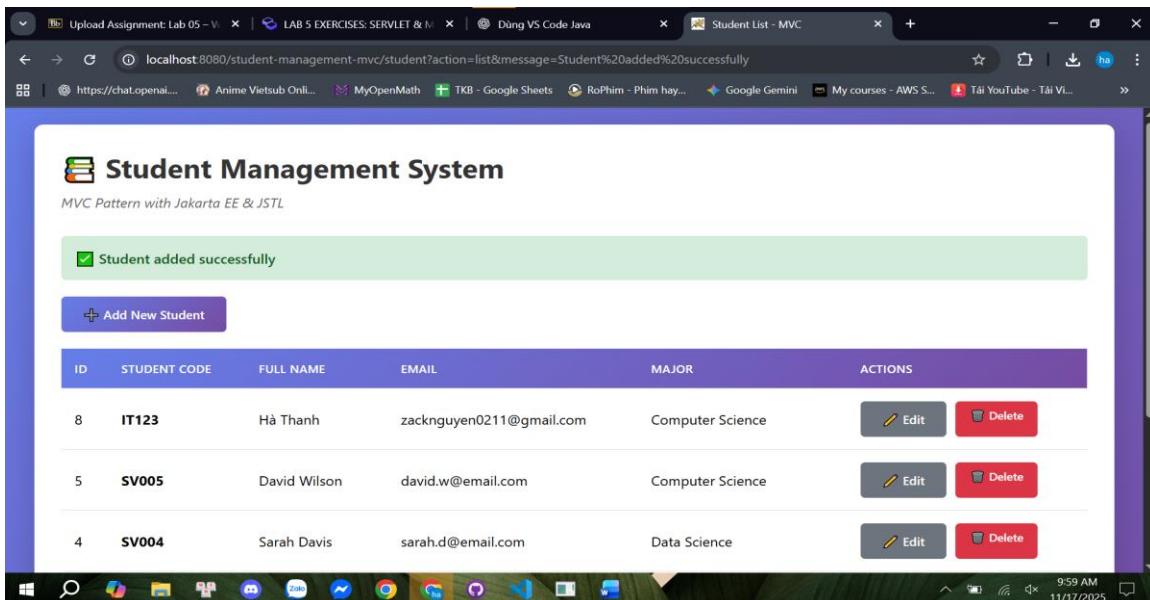
The controller forwards the request to student-form.jsp, showing an empty form.

When the user submits the form, a POST request is sent to: “/student?action=insert”

The controller extracts form data: “String studentCode =
request.getParameter("studentCode");”

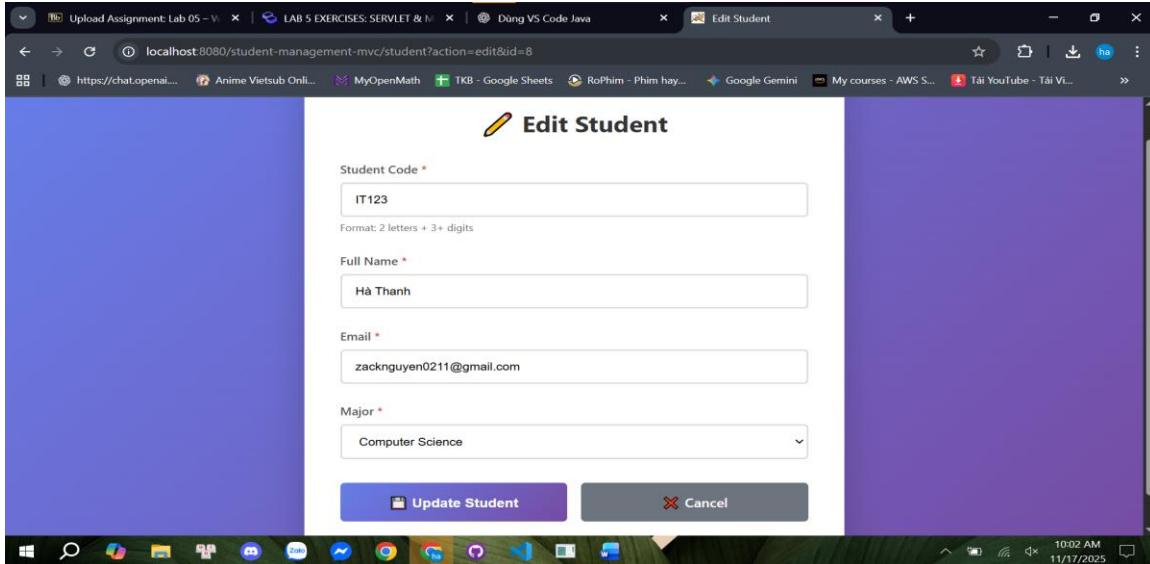
Creates a new Student object and calls DAO: “studentDAO.addStudent(newStudent);”

A SQL INSERT is executed, and the controller redirects back to the list with a success message.



3. EDIT – Edit Existing Student

When clicking Edit, the browser navigates to: "/student?action=edit&id=<studentId>"



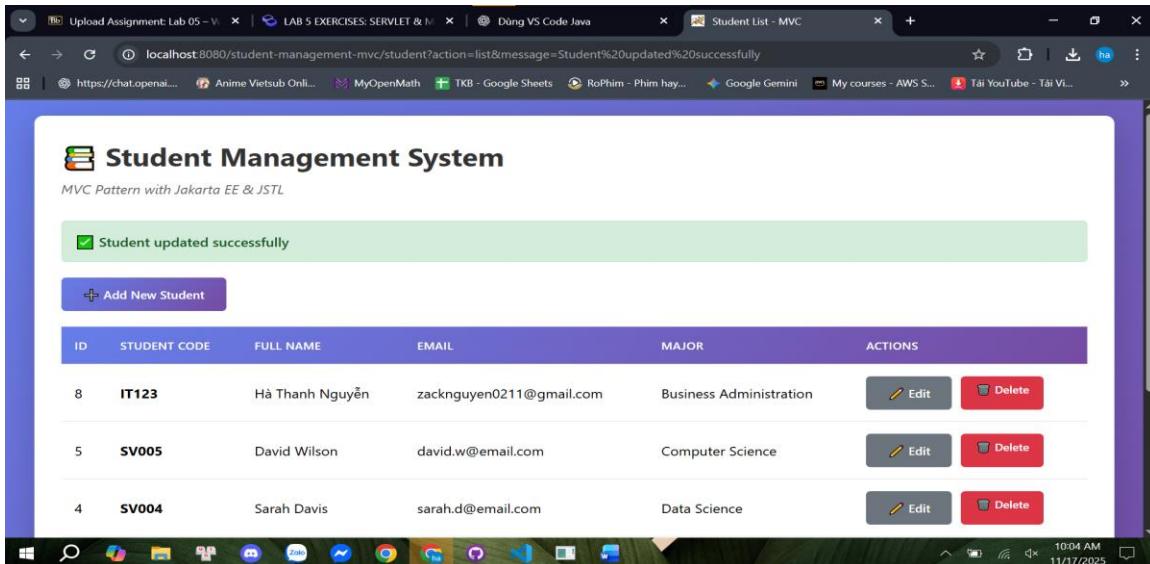
The controller loads the student: "Student existingStudent = studentDAO.getStudentById(id);"
request.setAttribute("student", existingStudent);

Then forwards to student-form.jsp, where fields are pre-filled.

After submitting, a POST is sent to: "/student?action=update"

The controller reads updated fields, calls: "studentDAO.updateStudent(student);"

This executes an UPDATE SQL statement and redirects back to the list with an update message.



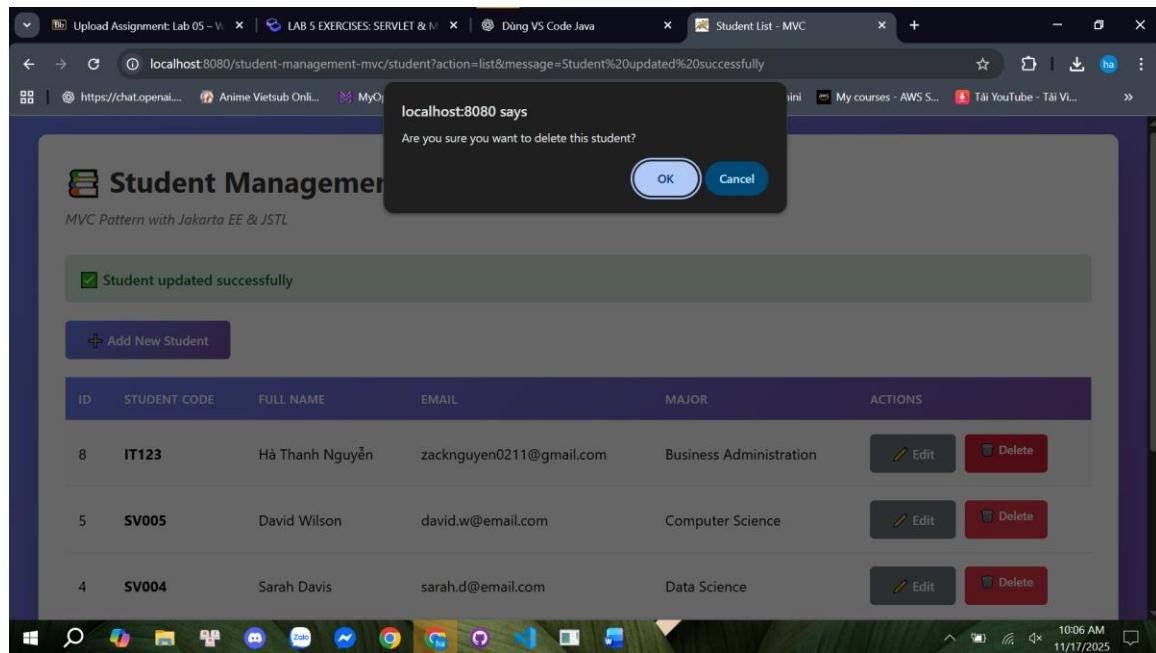
4. DELETE – Remove Student

Clicking Delete sends: "/student?action=delete&id=<studentId>"

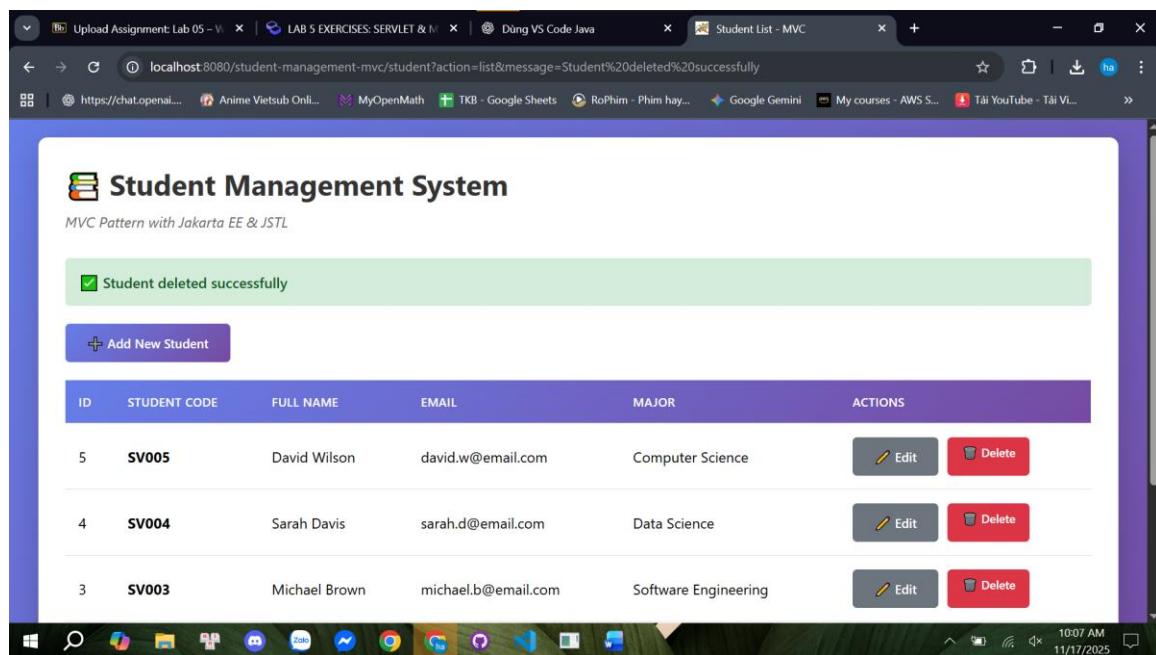
The controller calls: 'studentDAO.deleteStudent(id);'

The DAO executes a SQL DELETE statement, and the controller redirects back with a deletion message.

If all students are deleted, the list will be empty.



A screenshot of a web browser window displaying the "Student Management" application. The title bar shows the URL "localhost:8080/student-management-mvc/student?action=list&message=Student%20updated%20successfully". A modal dialog box is centered on the screen with the text "localhost:8080 says" and "Are you sure you want to delete this student?". Below the dialog, a green success message box displays "Student updated successfully". The main content area shows a table of student data with three rows. The first row has ID 8, Student Code IT123, Full Name Hà Thanh Nguyễn, Email zacknguyen0211@gmail.com, Major Business Administration, and actions "Edit" and "Delete". The second row has ID 5, Student Code SV005, Full Name David Wilson, Email david.w@email.com, Major Computer Science, and actions "Edit" and "Delete". The third row has ID 4, Student Code SV004, Full Name Sarah Davis, Email sarah.d@email.com, Major Data Science, and actions "Edit" and "Delete". The browser's taskbar at the bottom shows various open tabs and system icons.



A screenshot of the same web browser window after a student has been deleted. The title bar shows the URL "localhost:8080/student-management-mvc/student?action=list&message=Student%20deleted%20successfully". The modal dialog box from the previous screenshot is still present. The main content area now shows a table with only two rows of student data. The first row has ID 5, Student Code SV005, Full Name David Wilson, Email david.w@email.com, Major Computer Science, and actions "Edit" and "Delete". The second row has ID 4, Student Code SV004, Full Name Sarah Davis, Email sarah.d@email.com, Major Data Science, and actions "Edit" and "Delete". The browser's taskbar at the bottom shows various open tabs and system icons.

5. EMPTY STATE – No Students Found

When all students have been deleted, StudentDAO.getAllStudents() returns an empty list.

The JSP checks:

“<c:if test="\${empty students}>

No students found

</c:if>"

The UI displays a friendly message indicating that the system contains no students.

