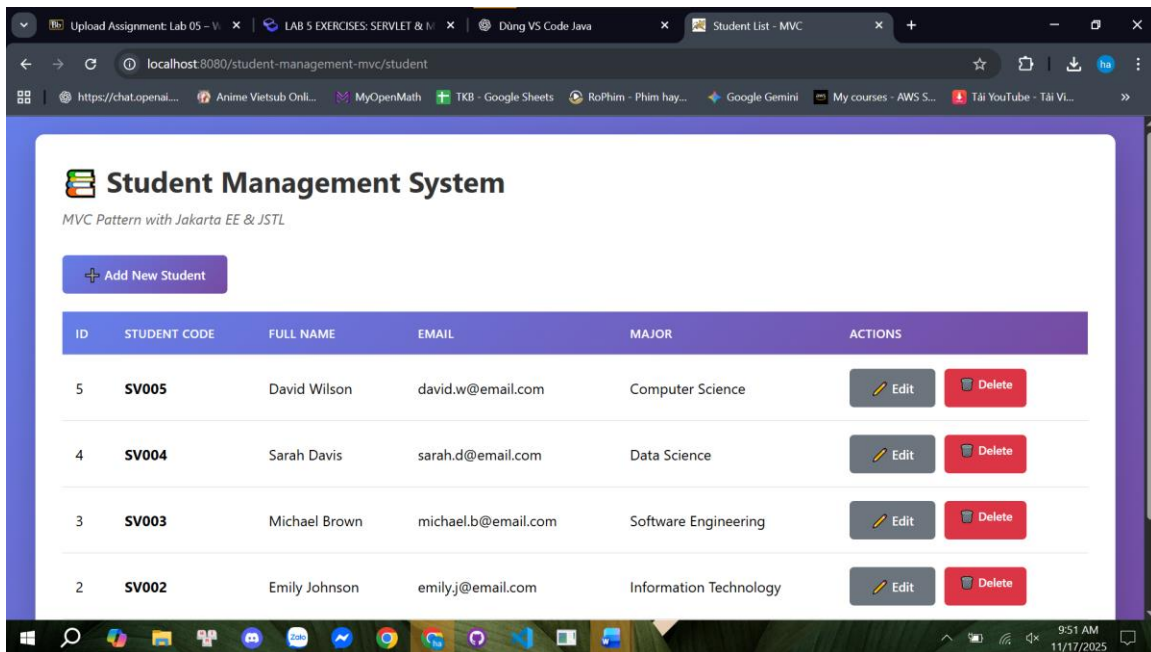


## Lab 05 – Nguyễn Hà Thanh – ITITI21144

### 1. LIST – Navigate to /student

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



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

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

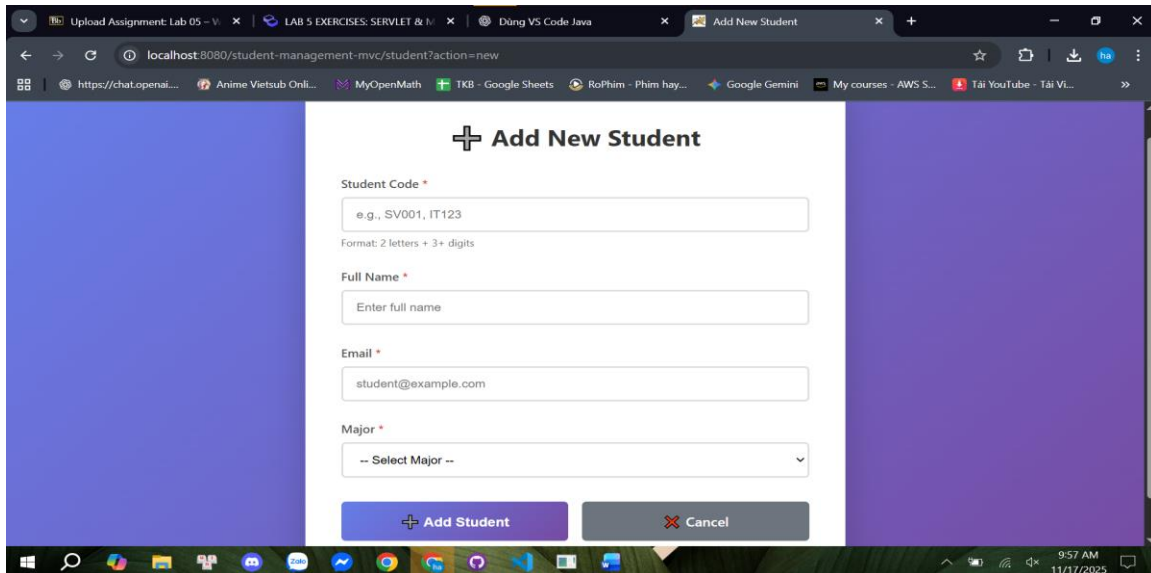
```
“request.setAttribute("students", students);”
```

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 screenshot shows a web browser window with the URL `localhost:8080/student-management-mvc/student?action=new`. The page has a purple header and a white form area. The form is titled "Add New Student" with a plus icon. It contains four input fields: "Student Code" (with a hint "e.g., SV001, IT123" and a format note "Format: 2 letters + 3+ digits"), "Full Name" (with a hint "Enter full name"), "Email" (with a hint "student@example.com"), and "Major" (a dropdown menu with "-- Select Major --"). At the bottom of the form are two buttons: "Add Student" (purple) and "Cancel" (grey).

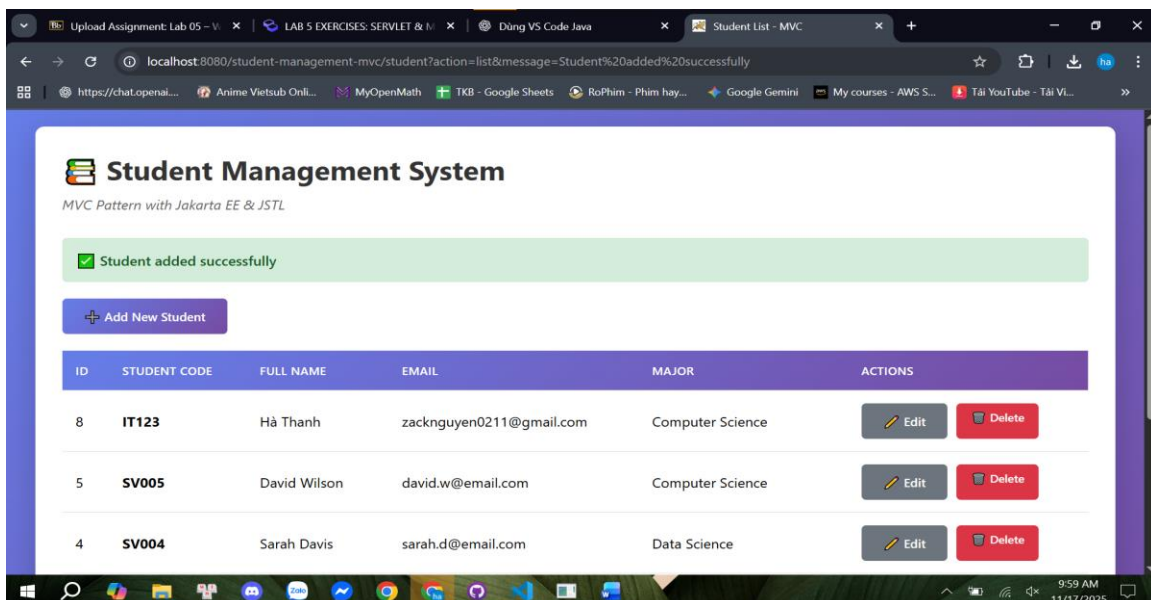
The controller forwards the request to `student-form.jsp`, showing an empty form.

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.

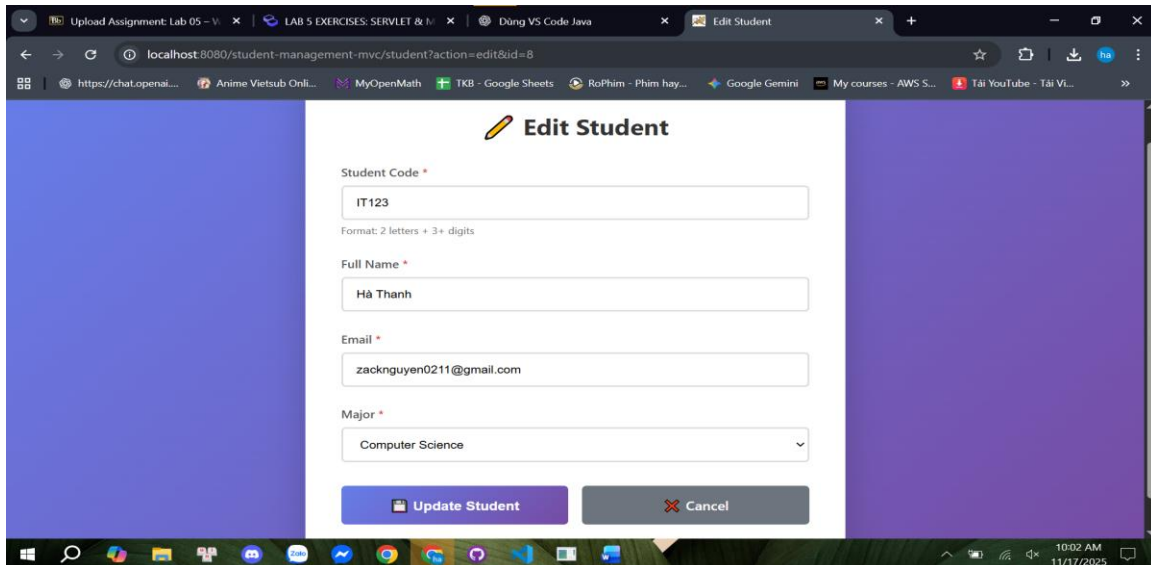


The screenshot shows a web browser window with the URL `localhost:8080/student-management-mvc/student?action=list&message=Student%20added%20successfully`. The page has a purple header and a white content area. At the top, there is a green success message: "Student added successfully". Below the message is a button "Add New Student" (purple). The main part of the page is a table with the following columns: ID, STUDENT CODE, FULL NAME, EMAIL, MAJOR, and ACTIONS. The table contains three rows of student data.

ID	STUDENT CODE	FULL NAME	EMAIL	MAJOR	ACTIONS
8	IT123	Hà Thanh	zacknguyen0211@gmail.com	Computer Science	<a href="#">Edit</a> <a href="#">Delete</a>
5	SV005	David Wilson	david.w@email.com	Computer Science	<a href="#">Edit</a> <a href="#">Delete</a>
4	SV004	Sarah Davis	sarah.d@email.com	Data Science	<a href="#">Edit</a> <a href="#">Delete</a>

### 3. EDIT – Edit Existing Student

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



The screenshot shows a web browser window with the URL `localhost:8080/student-management-mvc/student?action=edit&id=8`. The page title is "Edit Student". The form contains the following fields:

- Student Code \* (Text input): IT123
- Full Name \* (Text input): Hà Thanh
- Email \* (Text input): zacknguyen0211@gmail.com
- Major \* (Dropdown menu): Computer Science

At the bottom of the form are two buttons: "Update Student" and "Cancel".

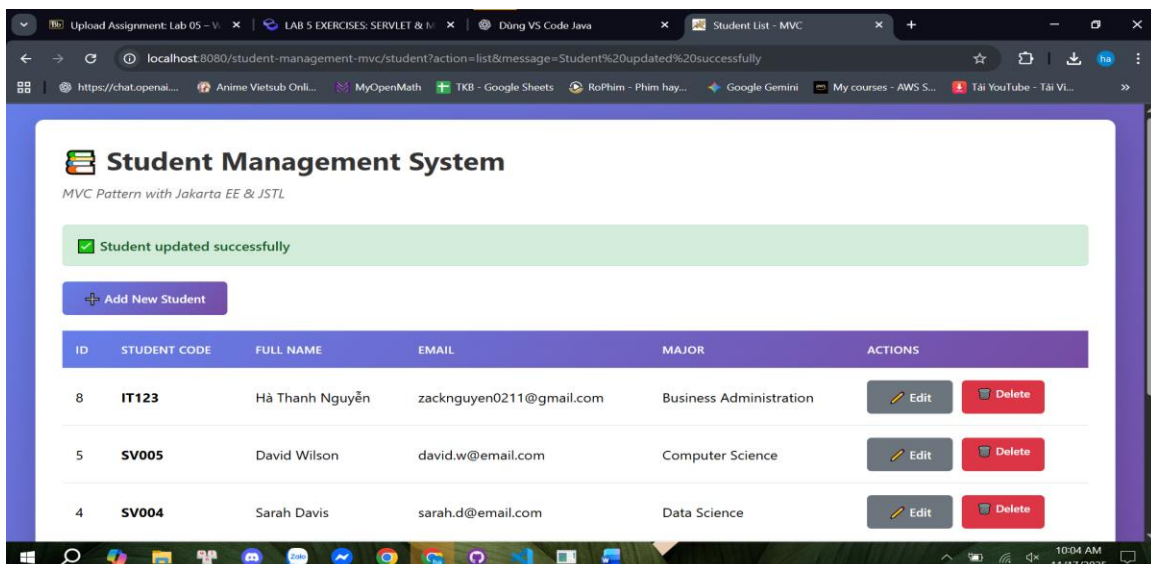
The controller loads the student: `"Student existingStudent = studentDAO.getStudentById(id);"`  
`request.setAttribute("student", existingStudent);`

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.



The screenshot shows a web browser window with the URL `localhost:8080/student-management-mvc/student?action=list&message=Student%20updated%20successfully`. The page title is "Student List - MVC". The page displays a message: "Student updated successfully". Below the message is a button "Add New Student". The main content is a table with the following columns: ID, Student Code, Full Name, Email, Major, and Actions.

ID	STUDENT CODE	FULL NAME	EMAIL	MAJOR	ACTIONS
8	IT123	Hà Thanh Nguyễn	zacknguyen0211@gmail.com	Business Administration	<a href="#">Edit</a> <a href="#">Delete</a>
5	SV005	David Wilson	david.w@email.com	Computer Science	<a href="#">Edit</a> <a href="#">Delete</a>
4	SV004	Sarah Davis	sarah.d@email.com	Data Science	<a href="#">Edit</a> <a href="#">Delete</a>

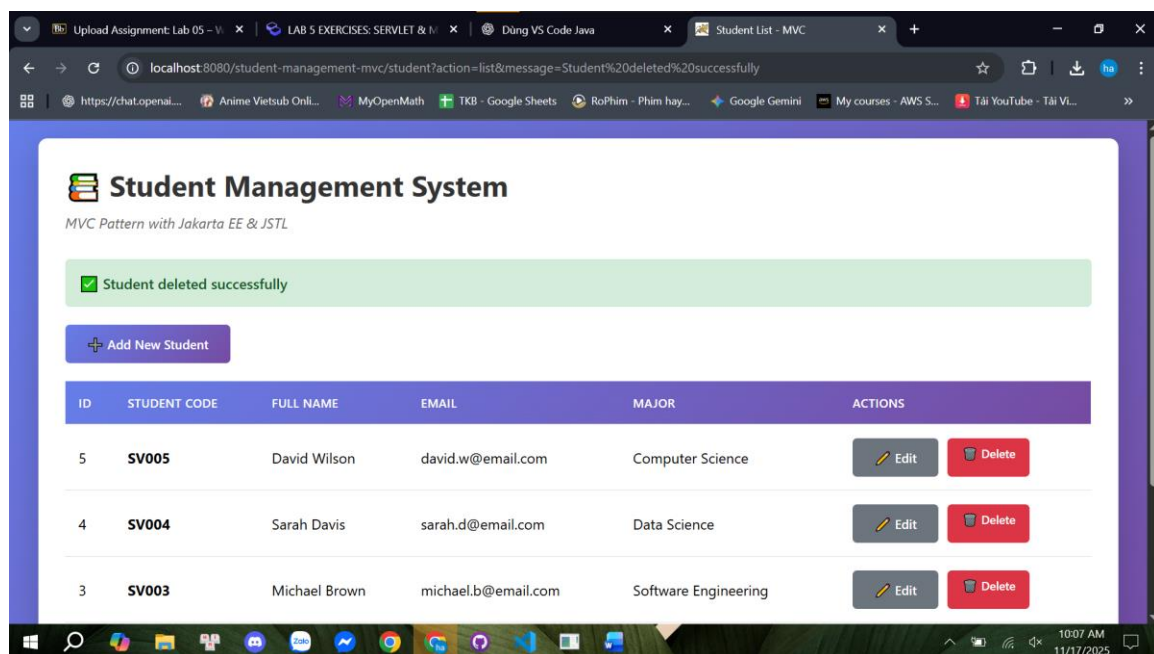
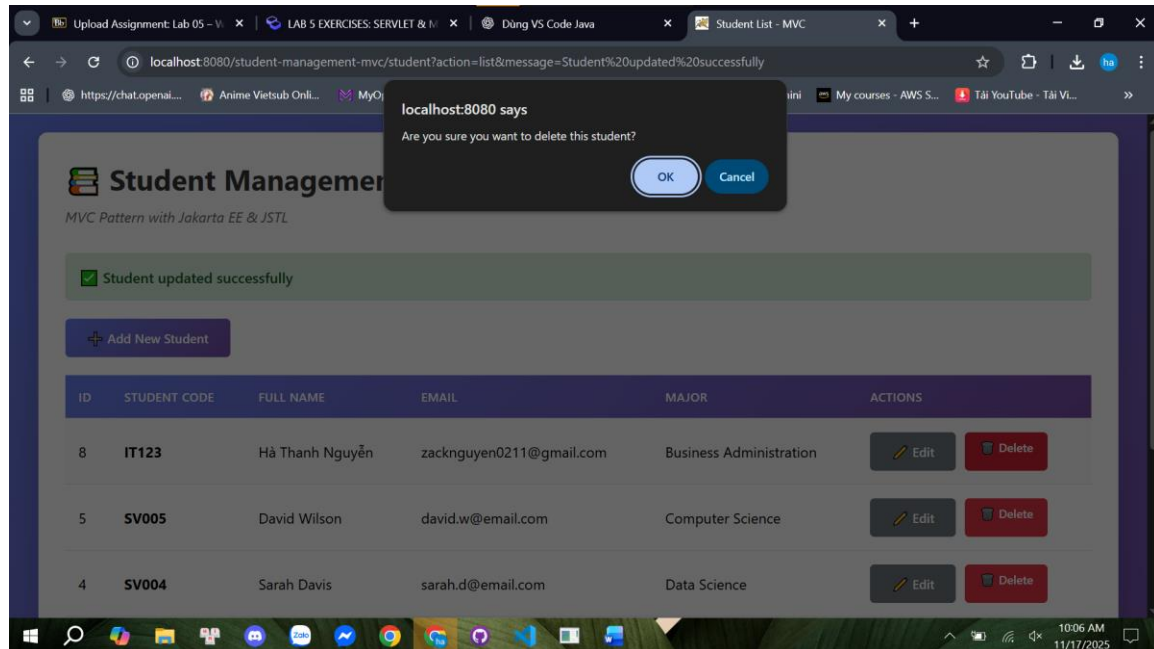
#### 4. DELETE – Remove Student

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.



## 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.

