

VIETNAM NATIONAL UNIVERSITY OF HO CHI MINH CITY  
THE INTERNATIONAL UNIVERSITY  
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING



**Web Application Development - IT093IU**

Homework Lab 5: SERVLET & MVC PATTERN

Nguyễn Vũ Thành Tính - ITCSIU23039

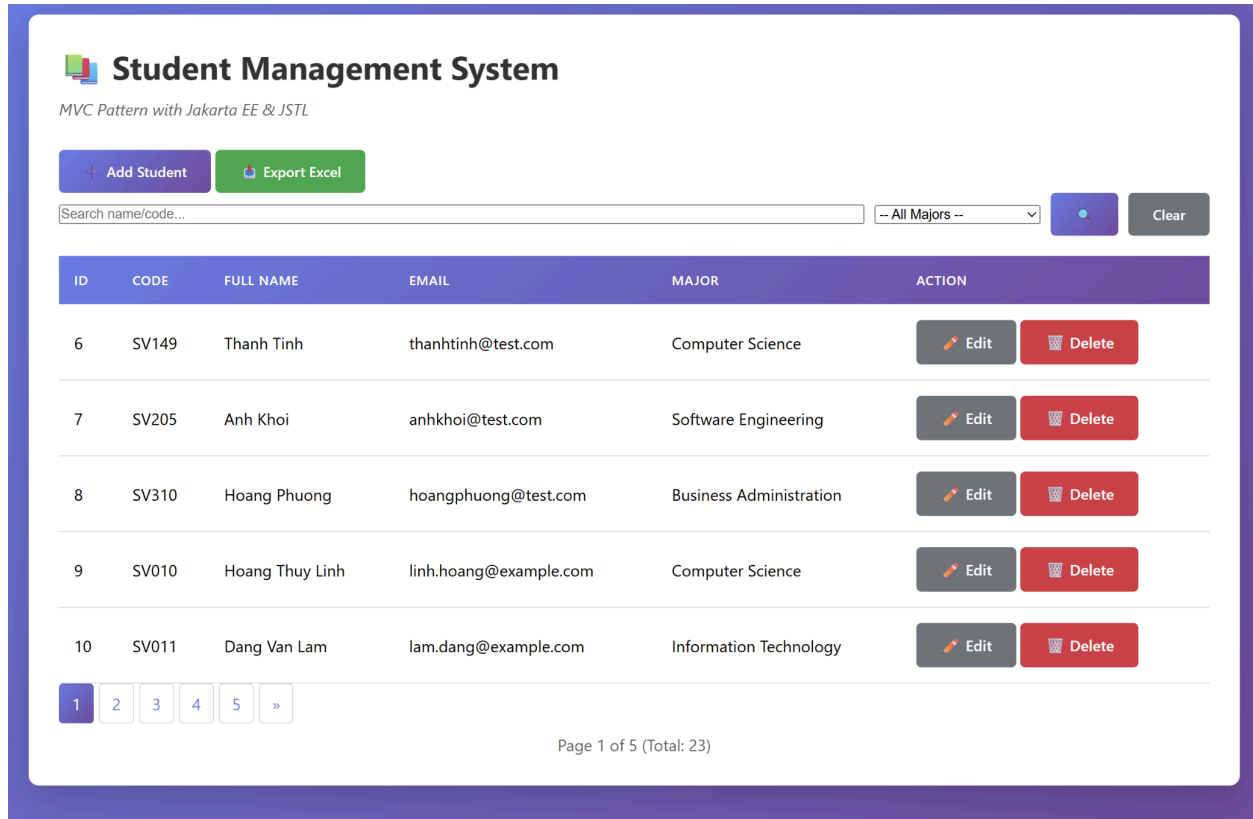
**Lab Instructor:** N.T.Nghia

<b>Github Link.....</b>	<b>3</b>
<b>1. Home page with student list.....</b>	<b>3</b>
<b>3. Edit student form (pre-filled).....</b>	<b>6</b>
<b>4. Search results.....</b>	<b>7</b>
<b>5. Validation errors.....</b>	<b>8</b>
<b>6. Sorted list.....</b>	<b>9</b>
<b>7. Filtered list.....</b>	<b>10</b>

## Github Link

[https://github.com/ngvuthtinh/Web\\_Lab/tree/main/Lab\\_5/Homework](https://github.com/ngvuthtinh/Web_Lab/tree/main/Lab_5/Homework)

### 1. Home page with student list



**Student Management System**  
MVC Pattern with Jakarta EE & JSTL

[Add Student](#) [Export Excel](#)

Search name/code... -- All Majors -- [Filter](#) [Clear](#)

ID	CODE	FULL NAME	EMAIL	MAJOR	ACTION
6	SV149	Thanh Tinh	thanhtinh@test.com	Computer Science	<a href="#">Edit</a> <a href="#">Delete</a>
7	SV205	Anh Khoi	anhkhoi@test.com	Software Engineering	<a href="#">Edit</a> <a href="#">Delete</a>
8	SV310	Hoang Phuong	hoangphuong@test.com	Business Administration	<a href="#">Edit</a> <a href="#">Delete</a>
9	SV010	Hoang Thuy Linh	linh.hoang@example.com	Computer Science	<a href="#">Edit</a> <a href="#">Delete</a>
10	SV011	Dang Van Lam	lam.dang@example.com	Information Technology	<a href="#">Edit</a> <a href="#">Delete</a>

1 2 3 4 5 »

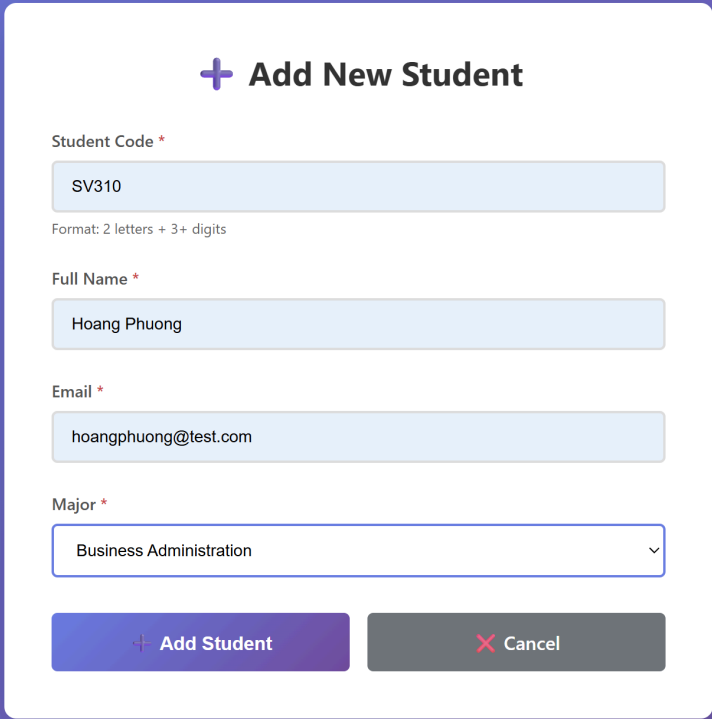
Page 1 of 5 (Total: 23)

#### Explanation:

- Request:** The user navigates to `/student` or clicks a pagination link. Tomcat maps the request to `StudentController`.
- Controller (doGet):** The `doGet` method routes the request to the `handleList` method (the central handler for viewing data).
- Parameter Extraction:** The controller extracts parameters: `page` (for pagination), `keyword` (search), `major` (filter), and `sortBy` (sorting).
- Pagination Calculation:** It sets `recordsPerPage` (e.g., 5) and calculates the SQL offset:  $\text{offset} = (\text{page} - 1) * \text{recordsPerPage}$ .

5. **DAO Call:** The controller calls `studentDAO.getStudentsCombined(...)`. This "Master Method" builds a dynamic SQL query including `WHERE` clauses for filtering and `LIMIT/OFFSET` for pagination.
6. **Set Attributes:** The controller sets crucial attributes for the view: the `students` list, `totalPages`, `currentPage`, and preserves all filter parameters (e.g., `currentMajor`) to keep the UI state.
7. **Forward to JSP:** The request is forwarded to `/views/student-list.jsp`.
8. **Render View:** The JSP uses `<c:forEach>` to render the student table. It also generates dynamic pagination links (Previous/Next) that retain the current search and filter state.

## 2. Add student form



The image shows a web form titled "+ Add New Student" inside a purple-bordered container. The form has four input fields: "Student Code \*" with the value "SV310" and a hint "Format: 2 letters + 3+ digits"; "Full Name \*" with the value "Hoang Phuong"; "Email \*" with the value "hoangphuong@test.com"; and "Major \*" which is a dropdown menu currently showing "Business Administration". At the bottom are two buttons: a purple "+ Add Student" button and a grey "X Cancel" button.

### + Add New Student

Student Code \*  
SV310  
Format: 2 letters + 3+ digits

Full Name \*  
Hoang Phuong

Email \*  
hoangphuong@test.com

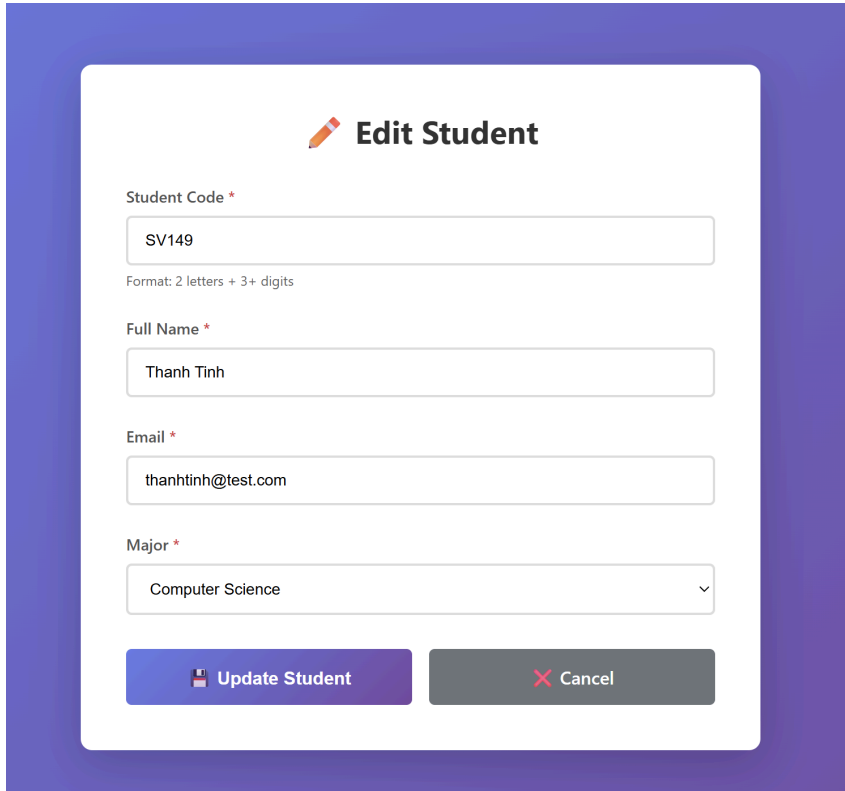
Major \*  
Business Administration


+ Add Student    X Cancel

## Explanation:

1. **Request (GET):** User clicks "Add New Student". URL becomes `/student?action=new`.
2. **Controller (doGet):** The `StudentController` captures `action=new` and calls the `showNewForm` method.
3. **Forward:** The request is forwarded to `/views/student-form.jsp`. The page renders an empty form with the title "Add New Student".
4. **Submission (POST):** User fills data and submits. The form sends a POST request to `/student` with hidden field `action=insert`.
5. **Controller (doPost):** The controller switches on `action=insert` and calls `insertStudent`.
6. **Validation:** The `validateStudent` method checks input rules (Regex, required fields). If invalid, it forwards back to the form with error messages.
7. **DAO Call:** If valid, `studentDAO.addStudent(newStudent)` is called to execute the SQL `INSERT` command.
8. **Redirect:** Upon success, the response redirects to the list page using `response.sendRedirect("student?action=list")`.

### 3. Edit student form (pre-filled)





 **Edit Student**

Student Code \*  
  
Format: 2 letters + 3+ digits

Full Name \*

Email \*

Major \*

 **Update Student**  **Cancel**

#### Explanation:

1. **Request (GET):** User clicks "Edit" on a student row. URL becomes `/student?action=edit&id=5`.
2. **Controller:** Captures `action=edit` and gets the `id` parameter from the request.
3. **DAO Call:** Calls `studentDAO.getStudentById(id)`. The DAO executes `SELECT * FROM students WHERE id = ?` and returns a specific `Student` object.
4. **Set Attribute:** The found student object is attached to the request:  
`request.setAttribute("student", existingStudent)`.
5. **Render View:** Forwards to `student-form.jsp`. The JSP checks if `${student}` exists and pre-fills all input fields (e.g., `value="${student.fullName}"`) for editing.

6. **Update (POST):** When submitted, `action=update` triggers the `updateStudent` method in Controller, which calls DAO to run the SQL `UPDATE` command.

## 4. Search results

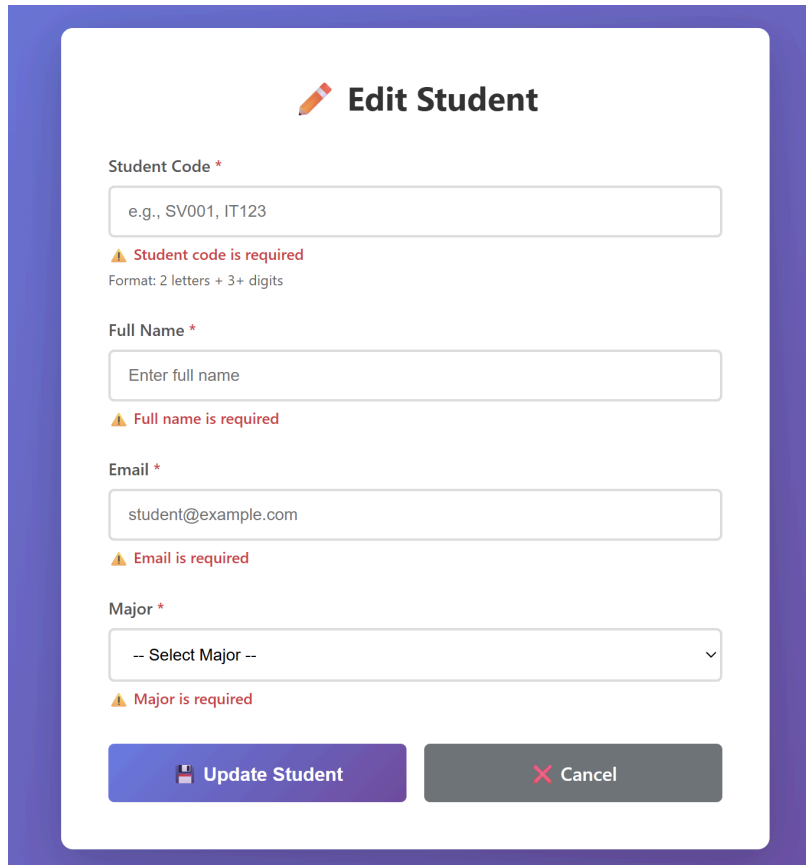
The screenshot shows a web application titled "Student Management System" with the subtitle "MVC Pattern with Jakarta EE & JSTL". It features two buttons: "Add Student" and "Export Excel". Below these is a search bar containing the text "Thanh Tinh", a dropdown menu set to "-- All Majors --", a search icon, and a "Clear" button. A table displays the search results with columns: ID, CODE, FULL NAME, EMAIL, MAJOR, and ACTION. The table contains one row for a student with ID 6, CODE SV149, FULL NAME Thanh Tinh, EMAIL thanh tinh@test.com, and MAJOR Computer Science. The ACTION column for this row has two buttons: "Edit" and "Delete".

ID	CODE	FULL NAME	EMAIL	MAJOR	ACTION
6	SV149	Thanh Tinh	thanh tinh@test.com	Computer Science	<button>Edit</button> <button>Delete</button>

### Explanation:

1. **Request:** User types a keyword and clicks Search. URL becomes `/student?action=list&keyword=John`.
2. **Controller:** The `handleList` method retrieves the `keyword` parameter.
3. **DAO Call:** Calls `studentDAO.getStudentsCombined(...)`. The DAO dynamically builds a SQL query using `LIKE: WHERE full_name LIKE ? OR student_code LIKE ?`.
4. **Wildcards:** The keyword is wrapped with `%` (e.g., `%John%`) to allow partial matching.
5. **Set Attribute:** The filtered list is set as `students`. Crucially, the `keyword` is also set as an attribute to keep it inside the search box after reloading.
6. **Render View:** The JSP displays only the students matching the criteria.

## 5. Validation errors



The screenshot shows a web form titled "Edit Student" with a pencil icon. It contains four input fields, each with a red error message below it:

- Student Code \***: The input field contains "e.g., SV001, IT123". The error message is "Student code is required" with a warning icon. Below it, the format "Format: 2 letters + 3+ digits" is displayed.
- Full Name \***: The input field contains "Enter full name". The error message is "Full name is required" with a warning icon.
- Email \***: The input field contains "student@example.com". The error message is "Email is required" with a warning icon.
- Major \***: The input field is a dropdown menu showing "-- Select Major --". The error message is "Major is required" with a warning icon.


At the bottom of the form are two buttons: a purple "Update Student" button with a save icon and a grey "Cancel" button with a red X icon.

### Explanation:

1. **Trigger:** Occurs inside `insertStudent` or `updateStudent` methods before calling the DAO.
2. **Validation Logic:** The `validateStudent()` method checks inputs. For example, it uses `matches("[A-Z]{2}[0-9]{3,}")` to validate the Student Code format.
3. **Error Handling:** If a rule is violated, an error message is added to the request (e.g., `request.setAttribute("errorCode", "Invalid format")`).
4. **Flow Interruption:** The controller stops processing and immediately **forwards** back to `student-form.jsp` (instead of redirecting).
5. **Feedback:** The JSP uses `<c:if test="${not empty errorCode}">` to display the red error message next to the specific field.



## 6. Sorted list

 **Student Management System**

MVC Pattern with Jakarta EE & JSTL

Add Student

Export Excel

-- All Majors --

Clear

ID	CODE	FULL NAME	EMAIL	MAJOR	ACTION
28	SV029	Bui Hoang Viet Anh	anh.bui@example.com	Business Administration	<div>Edit</div> <div>Delete</div>
27	SV028	Nguyen Thanh Chung	chung.nguyen@example.com	Software Engineering	<div>Edit</div> <div>Delete</div>
26	SV027	Ho Tan Tai	tai.ho@example.com	Information Technology	<div>Edit</div> <div>Delete</div>
25	SV026	Nguyen Tien Linh	linh.nguyen@example.com	Computer Science	<div>Edit</div> <div>Delete</div>
24	SV025	Pham Duc Huy	huy.pham@example.com	Business Administration	<div>Edit</div> <div>Delete</div>

1

2

3

4

5


»

Page 1 of 5 (Total: 23)

### Explanation:

- Request:** User clicks a column header (e.g., "Full Name"). URL becomes `/student?...&sortBy=full_name&order=asc`.
- Controller:** The `handleList` method captures `sortBy` and `order` parameters.
- Security Check:** The DAO's `validateSortBy` method checks the column name against a whitelist to prevent SQL Injection.
- DAO Call:** The SQL query appends the sorting clause: `ORDER BY full_name ASC`.
- Render View:** The JSP renders the sorted list. It also toggles the link for the next click (if current is `asc`, the link generates `desc`).

## 7. Filtered list



### Student Management System

MVC Pattern with Jakarta EE & JSTL

Add Student

Export Excel

Information Technology

Clear

ID	CODE	FULL NAME	EMAIL	MAJOR	ACTION
26	SV027	Ho Tan Tai	tai.ho@example.com	Information Technology	<div>Edit</div> <div>Delete</div>
22	SV023	Nguyen Phong Hong Duy	duy.nguyen@example.com	Information Technology	<div>Edit</div> <div>Delete</div>
18	SV019	Luong Xuan Truong	truong.luong@example.com	Information Technology	<div>Edit</div> <div>Delete</div>
14	SV015	Que Ngoc Hai	hai.que@example.com	Information Technology	<div>Edit</div> <div>Delete</div>
10	SV011	Dang Van Lam	lam.dang@example.com	Information Technology	<div>Edit</div> <div>Delete</div>

### Explanation:

- Request:** User selects "Computer Science" from the dropdown. URL becomes `/student?...&major=Computer Science`.
- Controller:** The `handleList` method captures the `major` parameter.
- DAO Call:** The DAO adds a `WHERE` clause to the SQL: `AND major = ?`.
- Combined Logic:** Since we use the "Master Method" in DAO, this filter works simultaneously with Search and Sort parameters.
- Render View:** The dropdown in JSP uses a conditional check `(${currentMajor == 'CS' ? 'selected' : ''})` to show the active filter.