

International University

School of Computer Science and Engineering

Web Application Development

Laboratory

IT093IU

Lab #4

Submitted by

Nguyễn Hồng Ngọc Hân - ITCSIU22229

EXERCISE 5: SEARCH FUNCTIONALITY

- 5.1: Create Search Form.**
- 5.2: Implement Search Logic.**

The screenshot shows a web browser window titled "Student List" with the URL "localhost:8080/StudentManagement/list_students.jsp". The page is titled "Student Management System" and features a search bar with placeholder text "Search by name or code...". Below the search bar are two buttons: "+ Add New Student" and "Export CSV". The main content is a table with the following data:

ID	Student Code	Full Name	Email	Major	Created At	Actions
6	SV006	John Doe	john.d@email.com	Computer Science	2025-11-08 13:52:26.0	Edit Delete
5	SV005	David Wilson	david.w@email.com	Computer Science	2025-11-08 13:02:17.0	Edit Delete
4	SV004	Sarah Davis	sarah.d@email.com	Data Science	2025-11-08 13:02:17.0	Edit Delete
3	SV003	Michael Brown	michael.b@email.com	Software Engineering	2025-11-08 13:02:17.0	Edit Delete
2	SV002	Emily Johnson	emily.j@email.com	Information Technology	2025-11-08 13:02:17.0	Edit Delete

Pagination controls at the bottom show pages 1, 2, and Next.

Explain:

This page lists all students and includes a search bar that lets users find records by name or student code. The form uses the GET method so the keyword appears in the URL. When submitted, the server reads the parameter and, if not empty, runs a query with the LIKE operator:

SELECT * FROM students WHERE full_name LIKE ? OR student_code LIKE ?
The query uses wildcards (%keyword%) with a PreparedStatement to prevent SQL injection. If no keyword is provided, it defaults to showing all records ordered by ID. Pagination with LIMIT and OFFSET ensures only 10 results per page, and the results appear in a responsive HTML table with edit, delete, and navigation links.

- **Test Cases 1:**

The screenshot shows a web browser window titled "Student List" with the URL "localhost:8080/StudentManagement/list_students.jsp?keyword=John". The page displays a table of student records. A search bar at the top contains the value "John". The table has columns: ID, Student Code, Full Name, Email, Major, Created At, and Actions. There are three rows of data:

ID	Student Code	Full Name	Email	Major	Created At	Actions
6	SV006	John Doe	john.d@email.com	Computer Science	2025-11-08 13:52:26.0	Edit Delete
2	SV002	Emily Johnson	emily.j@email.com	Information Technology	2025-11-08 13:02:17.0	Edit Delete
1	SV001	John Smith	john.smith@email.com	Computer Science	2025-11-08 13:02:17.0	Edit Delete

A small blue box highlights the number "1" at the bottom center of the table.

Explain:

In this scenario, the user types “John” into the search box and clicks Search. The form sends a GET request to `list_students.jsp?keyword=John`. On the server side, the code detects that the keyword is not null and executes the prepared SQL statement with `LIKE '%John%'` for both the `full_name` and `student_code` columns. The database returns all records where the name contains “John,” such as John Doe, Johnny Nguyen, or Peter Johnson. These records are then rendered in the HTML table, while pagination continues to function normally if multiple matches exist.

- **Test Cases 2:**

The screenshot shows a web browser window titled "Student List" with the URL "localhost:8080/StudentManagement/list_students.jsp?keyword=SV001". The page displays a table of student records. A search bar at the top contains the value "SV001". The table has columns: ID, Student Code, Full Name, Email, Major, Created At, and Actions. One record is listed: ID 1, Student Code SV001, Full Name John Smith, Email john.smith@email.com, Major Computer Science, Created At 2025-11-08 13:02:17.0. The Actions column for this row contains "Edit" and "Delete" links.

ID	Student Code	Full Name	Email	Major	Created At	Actions
1	SV001	John Smith	john.smith@email.com	Computer Science	2025-11-08 13:02:17.0	Edit Delete

Explain:

Here, the user types “SV001” and presses Search. The same process occurs: the request URL becomes `list_students.jsp?keyword=SV001`. The SQL query runs with wildcards `%SV001%`, comparing against both `student_code` and `full_name`. Since `student_code` is unique, the query finds exactly one match, such as SV001 – Nguyen Van A. The result set contains that single record, which is displayed clearly in the table.

- **Test Cases 3:**

The screenshot shows a web browser window titled "Student List". The address bar displays "localhost:8080/StudentManagement/list_students.jsp?keyword=science". The main content area is titled "Student Management System" and features a search bar with the value "science", a "Search" button, and a "Clear" button. Below the search bar are two buttons: "+ Add New Student" and "Export CSV". A table is displayed with the following columns: ID, Student Code, Full Name, Email, Major, Created At, and Actions. The table currently has no data rows.

Explain:

In this case, the user enters “science” in the search box to look for students whose major includes “Computer Science” or “Data Science.” The request is sent as `list_students.jsp?keyword=science`. The server again runs the prepared SQL query with `LIKE '%science%'`. Since the keyword matches part of the major or full_name field (depending on how data is entered), all students enrolled in majors like “Computer Science,” “Information Science,” or “Data Science” are retrieved.

- **Test Cases 4:**

The screenshot shows a web browser window titled "Student List" with the URL "localhost:8080/StudentManagement/list_students.jsp?keyword=". The page displays a table of student records with columns: ID, Student Code, Full Name, Email, Major, Created At, and Actions. There are 6 rows of data. At the bottom, there are navigation links for page 1, 2, and Next.

ID	Student Code	Full Name	Email	Major	Created At	Actions
6	SV006	John Doe	john.d@email.com	Computer Science	2025-11-08 13:52:26.0	Edit Delete
5	SV005	David Wilson	david.w@email.com	Computer Science	2025-11-08 13:02:17.0	Edit Delete
4	SV004	Sarah Davis	sarah.d@email.com	Data Science	2025-11-08 13:02:17.0	Edit Delete
3	SV003	Michael Brown	michael.b@email.com	Software Engineering	2025-11-08 13:02:17.0	Edit Delete
2	SV002	Emily Johnson	emily.j@email.com	Information Technology	2025-11-08 13:02:17.0	Edit Delete

Explain:

When the search field is left blank and the user clicks Search, the browser still sends a request to `list_students.jsp` but with an empty keyword value. The server checks: if `(keyword != null && !keyword.trim().isEmpty())` Since this condition is false, the code executes the default SQL: `SELECT * FROM students ORDER BY id DESC`. This query retrieves all students from the database, sorted from newest to oldest. Pagination remains active, displaying 10 records per page, and users can navigate using the page links at the bottom.

EXERCISE 6: VALIDATION ENHANCEMENT

6.1: Email Validation

Explain:

- **Test Cases 1:**

The screenshots illustrate the process of adding a new student record. In the first screenshot, the 'Add New Student' form is displayed with the following data:

Field	Value
Student Code *	SV007
Full Name *	John Gem
Email	john@email.com
Major	Computer Science

In the second screenshot, the 'Student Management System' dashboard shows the newly added student in the list:

ID	Student Code	Full Name	Email	Major	Created At	Actions
10	SV007	John Gem	john@email.com	Computer Science	2025-11-13 22:13:05.0	Edit Delete
6	SV006	John Doe	john.d@email.com	Computer Science	2025-11-08 13:52:26.0	Edit Delete
5	SV005	David Wilson	david.w@email.com	Computer Science	2025-11-08 13:02:17.0	Edit Delete
4	SV004	Sarah Davis	sarah.d@email.com	Data Science	2025-11-08 13:02:17.0	Edit Delete
3	SV003	Michael Brown	michael.b@email.com	Software Engineering	2025-11-08 13:02:17.0	Edit Delete

Explain:

When the user enters this email, it matches the regex pattern because it has valid characters before and after the "@" symbol. The system accepts the input, continues the insertion, and displays the message "*Student added successfully.*"

- **Test Cases 2:**

ID	Student Code	Full Name	Email	Major	Created At	Actions
11	SV008	John Doe02	john.doe@company.co.uk	Computer Science	2025-11-13 22:14:15.0	Edit Delete
10	SV007	John Gem	john@email.com	Computer Science	2025-11-13 22:13:05.0	Edit Delete
6	SV006	John Doe	john.d@email.com	Computer Science	2025-11-08 13:52:26.0	Edit Delete
5	SV005	David Wilson	david.w@email.com	Computer Science	2025-11-08 13:02:17.0	Edit Delete
4	SV004	Sarah Davis	sarah.d@email.com	Data Science	2025-11-08 13:02:17.0	Edit Delete

Explain:

This case includes multiple dots in the domain name but still fits the regex structure. The validation passes, and the student record is successfully inserted into the database with no errors.

- Test Cases 3:

+ Add New Student

Student Code *

SV009

Full Name *

John

Email

john@email

Major

Computer Science

Save Student Cancel

Student Management System

✓ Student added successfully

Search by name or code... Search Clear

+ Add New Student Export CSV

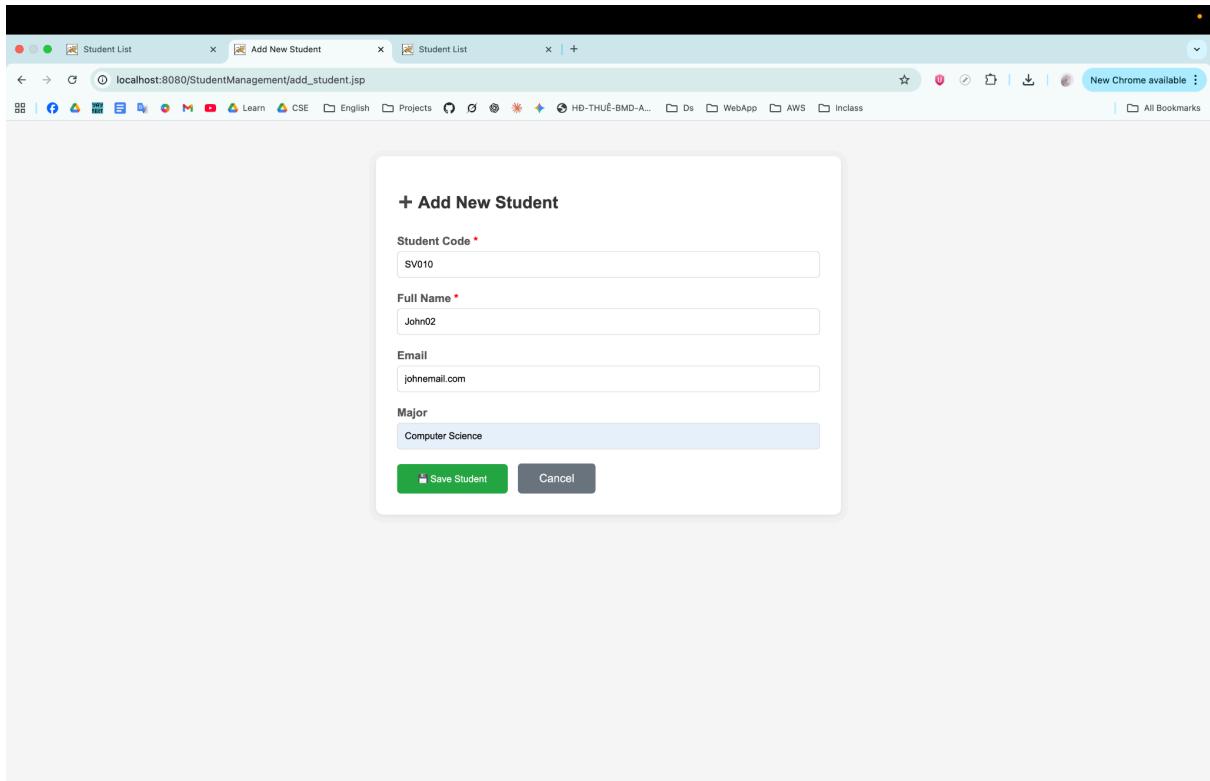
ID	Student Code	Full Name	Email	Major	Created At	Actions
14	SV009	John	john@email	Computer Science	2025-11-13 22:20:04.0	Edit Delete
11	SV008	John Doe02	john.doe@company.co.uk	Computer Science	2025-11-13 22:14:15.0	Edit Delete
10	SV007	John Gem	john@email.com	Computer Science	2025-11-13 22:13:05.0	Edit Delete
6	SV006	John Doe	john.d@email.com	Computer Science	2025-11-08 13:52:26.0	Edit Delete
5	SV005	David Wilson	david.w@email.com	Computer Science	2025-11-08 13:02:17.0	Edit Delete

1 2 Next

Explain:

Here, the input is missing the top-level domain (e.g., .com), so it sticks to match the regex “[A-Za-z0-9+_.-]+@[.]+\$”.

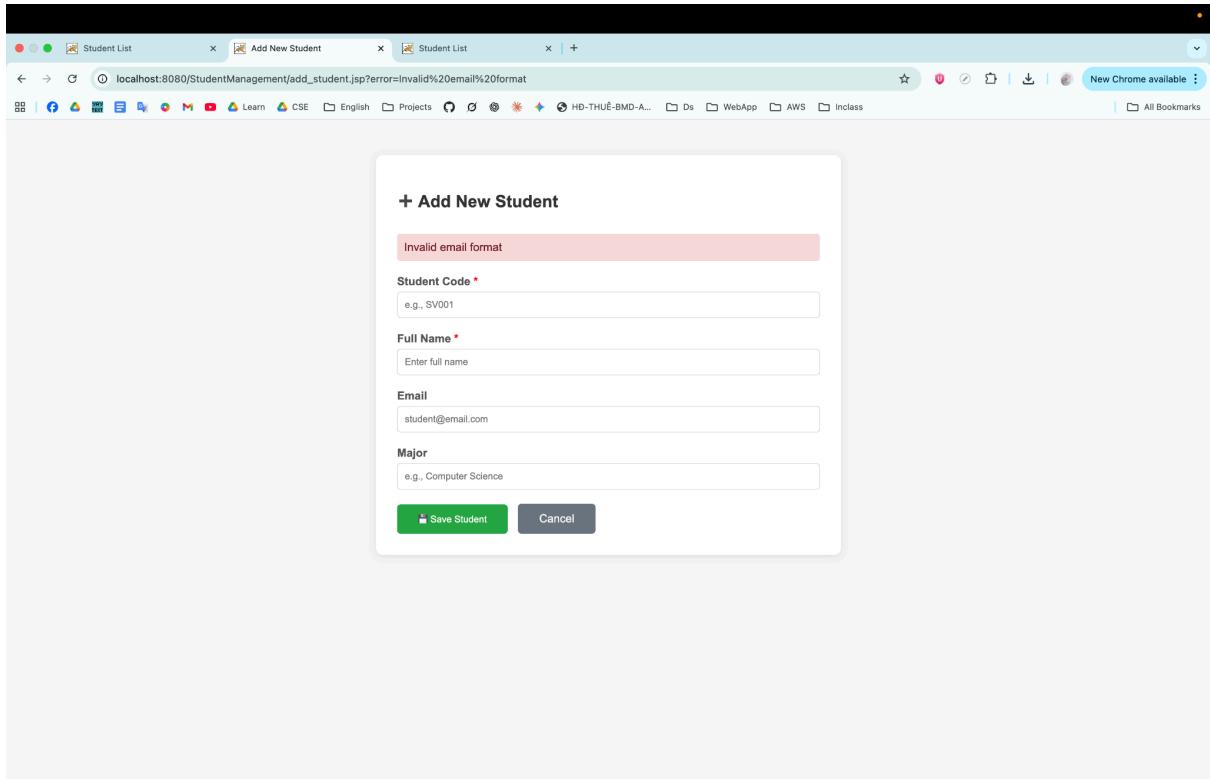
- **Test Cases 4:**



The screenshot shows a web browser window with three tabs: "Student List", "Add New Student", and "Student List". The "Add New Student" tab is active, displaying a form titled "+ Add New Student". The form fields are as follows:

- Student Code ***: SV010
- Full Name ***: John02
- Email**: johnemail.com
- Major**: Computer Science

At the bottom are two buttons: a green "Save Student" button and a grey "Cancel" button.



The screenshot shows the same browser window after an attempt to submit the form with invalid email input. The "Email" field now contains "student@email.com" and has a red border, indicating an error. A red error message "Invalid email format" is displayed above the "Email" field. The other form fields remain the same as in the first screenshot.

Explain:

Since the "@" symbol is missing, this input clearly violates the email pattern. The validation fails, and the user is redirected with the same error, "*Invalid email format.*" The database remains unchanged.

- **Test Cases 5:**

The top screenshot shows the 'Add New Student' form. The fields are filled as follows: Student Code (SV010), Full Name (John03), Email (student@email.com), and Major (Computer Science). The 'Save Student' button is highlighted. The bottom screenshot shows the 'Student List' page with a success message: '✓ Student added successfully'. The table lists six students, including the newly added one (ID 16, SV010, John03, student@email.com, Computer Science, 2025-11-13 22:22:36.0).

ID	Student Code	Full Name	Email	Major	Created At	Actions
16	SV010	John03		Computer Science	2025-11-13 22:22:36.0	Edit Delete
14	SV009	John	john@email	Computer Science	2025-11-13 22:22:04.0	Edit Delete
11	SV008	John Doe02	john.doe@company.co.uk	Computer Science	2025-11-13 22:14:15.0	Edit Delete
10	SV007	John Gem	john@email.com	Computer Science	2025-11-13 22:13:05.0	Edit Delete
6	SV006	John Doe	john.d@email.com	Computer Science	2025-11-08 13:52:26.0	Edit Delete

Explain:

When the email field is left blank, the code checks the condition:

```
if (email != null && !email.trim().isEmpty()) { ... }
```

Because the field is empty, this block is skipped, meaning the system treats the email as optional. The student record is inserted successfully without an email value.

6.2: Student Code Pattern Validation

- **Test cases valid:**

IT005
 IT003
 IT002
 IT001
 SV010
 SV009
 SV008
 SV007
 SV006
 SV005

Explain:

When the user clicks Add, fills in the form with a valid code such as SV001, completes the other fields, and clicks Save, the system checks the pattern using `.matches("[A-Z]{2}[0-9]{3,}")`. Since the code follows the rule (two capital letters and at least three digits), validation passes. The system proceeds to insert the data into the database using the prepared SQL statement, and the user is redirected to `list_students.jsp` with a success message — “Student added successfully.”

- **Test cases invalid:**

The figure consists of two side-by-side screenshots of a web browser displaying a student management application. Both screenshots show the same 'Add New Student' form.

Left Screenshot (Valid Input):

- Form Fields:**
 - Student Code: sv001
 - Full Name: John
 - Email: student@email.com
 - Major: Computer Science
- Buttons:** Save Student (green button) and Cancel.

Right Screenshot (Invalid Input):

- Form Fields:**
 - Student Code: sv001 (highlighted in red with an error message: "Invalid student code format (Example: SV001)")
 - Full Name: John
 - Email: student@email.com
 - Major: Computer Science
- Buttons:** Save Student (disabled/greyed out) and Cancel.

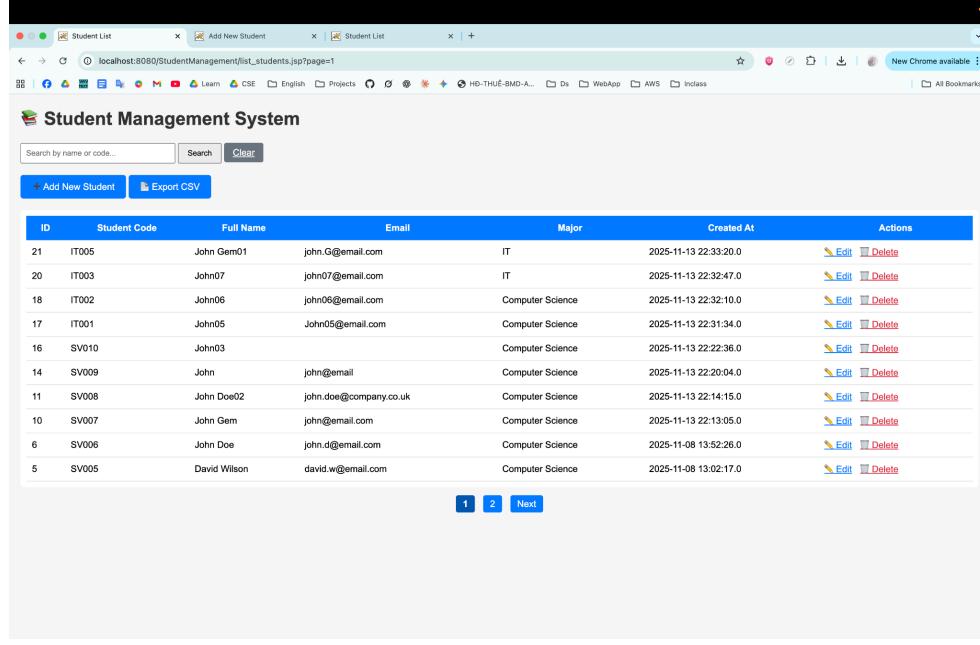
Explain:

If the user fills the form with lowercase or too-short codes (like sv001, S001, or SV12) and clicks Save, the validation check fails because the input doesn't match the required uppercase and digit pattern. The code executes the redirect command: `response.sendRedirect("add_student.jsp?error=Invalid student code format (Example: SV001)");`

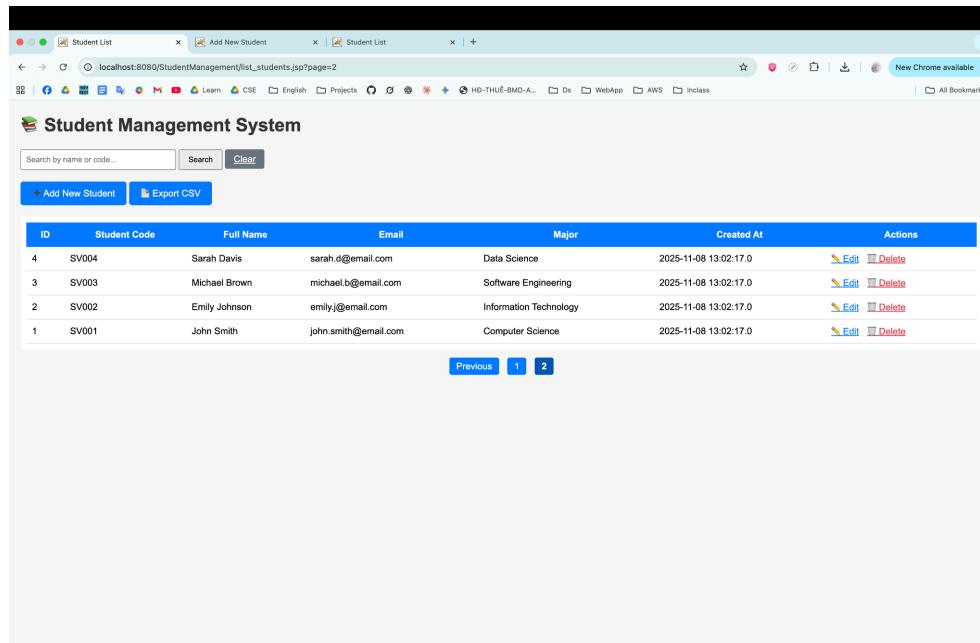
The page reloads with a red error message, and no new record is inserted into the database.

EXERCISE 7: USER EXPERIENCE IMPROVEMENTS

7.1: Pagination



ID	Student Code	Full Name	Email	Major	Created At	Actions
21	IT005	John Gem01	john.G@email.com	IT	2025-11-13 22:33:20.0	Edit Delete
20	IT003	John07	john07@email.com	IT	2025-11-13 22:32:47.0	Edit Delete
18	IT002	John06	john06@email.com	Computer Science	2025-11-13 22:32:10.0	Edit Delete
17	IT001	John05	John05@email.com	Computer Science	2025-11-13 22:31:34.0	Edit Delete
16	SV010	John03		Computer Science	2025-11-13 22:22:36.0	Edit Delete
14	SV009	John	john@email	Computer Science	2025-11-13 22:20:04.0	Edit Delete
11	SV008	John Doe02	john.doe@company.co.uk	Computer Science	2025-11-13 22:14:15.0	Edit Delete
10	SV007	John Gem	john@email.com	Computer Science	2025-11-13 22:13:05.0	Edit Delete
6	SV006	John Doe	john.d@email.com	Computer Science	2025-11-08 13:52:26.0	Edit Delete
5	SV005	David Wilson	david.w@email.com	Computer Science	2025-11-08 13:02:17.0	Edit Delete



ID	Student Code	Full Name	Email	Major	Created At	Actions
4	SV004	Sarah Davis	sarah.d@email.com	Data Science	2025-11-08 13:02:17.0	Edit Delete
3	SV003	Michael Brown	michael.b@email.com	Software Engineering	2025-11-08 13:02:17.0	Edit Delete
2	SV002	Emily Johnson	emily.j@email.com	Information Technology	2025-11-08 13:02:17.0	Edit Delete
1	SV001	John Smith	john.smith@email.com	Computer Science	2025-11-08 13:02:17.0	Edit Delete

Explain:

After the user clicks Add New Student, fills in the form, and presses Save, the new student is inserted into the database. When redirected to list_students.jsp, the page now shows only 10 records per page. The pagination logic retrieves the page parameter from the URL using

```
String pageParam = request.getParameter("page");
```

and calculates the offset with

```
int offset = (currentPage - 1) * recordsPerPage;
```

The SQL query includes LIMIT ? OFFSET ? to fetch only the students for that page. The code also counts total records with a separate query (SELECT COUNT(*) FROM students) to calculate total pages, then displays numbered page links with Previous and Next buttons at the bottom.

When the user clicks on a page number (e.g., “2”), the browser sends a new GET request like list_students.jsp?page=2, and the system shows the next 10 students accordingly.

7.2: Improved UI/UX

a) Success/Error Message Styling

- Add distinct colors (green for success, red for error)
- Add icons (✓ for success, ✗ for error)
- Auto-hide after 3 seconds (JavaScript)

The image contains two screenshots of a web application titled "Student Management System".

Screenshot 1 (Top): A green success message box with a checkmark icon and the text "✓ Student added successfully".

Screenshot 2 (Bottom): A red error message box with an X icon and the text "✗ Student code already exists".

Below the messages:

- Left Screenshot:** Shows a table of student data with columns: ID, Student Code, Full Name, Email, Major, Created At, Actions. One row has a red background and an X icon in the Actions column.
- Right Screenshot:** Shows a table of student data with columns: ID, Student Code, Full Name, Email, Major, Created At, Actions. All rows have a light blue background and a checkmark icon in the Actions column.

Explain:

When the user clicks Add New Student, fills in the form, and presses Save, the system now responds with smoother visual feedback. If the data is valid, the page redirects to list_students.jsp and shows a green success box with a ✓ icon and the message “Student added successfully.” If the data is invalid (for example, missing required fields or incorrect format), the system instead displays a red error box with a ✗ icon and an appropriate error message such as “Invalid email format.” In both cases, these messages automatically disappear after 3 seconds using JavaScript for a clean interface.

b) Loading States (2 points)

- Disable submit button after clicking to prevent double submission

- Show "Processing..." text

The screenshot shows a web browser window titled "Add New Student" with the URL "localhost:8080/StudentManagement/add_student.jsp?student_code=SV007&full_name=John&email=dichfb181%40gmail.com&major=Computer+Science". The main content is a form titled "+ Add New Student" with four input fields: "Student Code" (value: SV001), "Full Name" (value: John), "Email" (value: dichfb181@gmail.com), and "Major" (value: Computer Science). Below the form are two buttons: a green "Processing..." button and a grey "Cancel" button.

Explain:

When the user submits any form, JavaScript runs the submitForm() function. The Save button becomes disabled and its text changes to “Processing...”. This prevents double-clicks or repeated inserts while waiting for the response. Once redirected, the button returns to normal.

c) Responsive Table (3 points)

- Table scrollable on small screens
- Better mobile layout

The screenshot shows a web application titled "Student Management System". The title bar includes a logo of three books, the text "Student List", and a search icon. The address bar shows "localhost..." with a star icon and a message "New Chrome available". Below the address bar are various browser icons and a "All Bookmarks" link.

Student Management System

Search by name or code...

ID	Student Code	Full Name	Email	Major	Created At	Actions
21	IT005	John Gem1	john.G@email.com	IT	2025-11-13 22:33:20.0	Edit Delete
20	IT003	John07	john07@email.com	IT	2025-11-13 22:32:47.0	Edit Delete
18	IT002	John06	john06@email.com	Computer Science	2025-11-13 22:32:10.0	Edit Delete
17	IT001	John05	John05@email.com	Computer Science	2025-11-13 22:31:34.0	Edit Delete
16	SV010	John03		Computer Science	2025-11-13 22:22:36.0	Edit Delete

Explain:

When viewing on smaller screens, the .table-responsive CSS makes the table horizontally scrollable, while font size and padding adjust for visibility.

BONUS EXERCISES

BONUS 1: Export to CSV

The screenshot shows the Student Management System interface. At the top, there is a logo of three books and the text "Student Management System". Below the logo are search and clear buttons. There are two blue buttons: "+ Add New Student" and "Export CSV". The main area displays a table of student records with columns: ID, Student Code, Full Name, Email, Major, and Created At. Each row has edit and delete links in the Actions column. The table includes records from ID 5 to 21. Below the table are navigation buttons for page 1, 2, and Next. In the background, a browser window titled "Student List" is open at the URL "localhost:8080/StudentManagement/list_students.jsp". The browser's address bar also shows "localhost:8080/StudentManagement/export_csv.jsp". The browser's sidebar shows a file named "students.csv" with a size of 1,205 B and a timestamp of 586 B - 1 hour ago.

ID	Student Code	Full Name	Email	Major	Created At	Actions
21	IT005	John Gem1	john.G@email.com	IT	2025-11-13 22:33:20.0	Edit Delete
20	IT003	John07	john07@email.com	IT	2025-11-13 22:32:47.0	Edit Delete
18	IT002	John06	john06@email.com	Computer Science	2025-11-13 22:32:10.0	Edit Delete
17	IT001	John05	John05@email.com	Computer Science	2025-11-13 22:31:34.0	Edit Delete
16	SV010	John03		Computer Science	2025-11-13 22:22:36.0	Edit Delete
14	SV009	John	john@email	Computer Science	2025-11-13 22:20:04.0	Edit Delete
11	SV008	John Doe02	john.doe@company.co.uk	Computer Science	2025-11-13 22:14:15.0	Edit Delete
10	SV007	John Gem	john@email.com	Computer Science	2025-11-13 22:13:05.0	Edit Delete
6	SV006	John Doe	john.d@email.com	Computer Science	2025-11-08 13:52:26.0	Edit Delete
5	SV005	David Wilson	david.w@email.com	Computer Science	2025-11-08 13:02:17.0	Edit Delete

Explain:

When the user clicks Add New Student, fills in the form, and saves, the student record is successfully stored in the database. On the list_students.jsp page, the user can then click the Export CSV button. This action sends a request to export_csv.jsp, which dynamically generates a CSV file from the database and automatically downloads it to the user's computer as students.csv.

The JSP sets the response headers using
response.setContentType("text/csv");
response.setHeader("Content-Disposition", "attachment; filename=\"students.csv\"");
so the browser recognizes it as a downloadable CSV file. The code then connects to the database using JDBC, retrieves all student records (or filtered results if a

keyword search is active), and prints each record as a CSV line using `out.println()`. The output includes columns such as ID, Student Code, Full Name, Email, Major, and Created At.