


Ex1: Setup and Display











list_students.jsp:



Student Management System

Student deleted successfully

+ Add New Student

ID	Student Code	Full Name	Email	Major	Created At	Actions
5	SV005	David Wilson	david.w@email.com	Computer Science	2025-11-06 08:35:23.0	 Edit  Delete
4	SV004	Sarah Davis	sarah.d@email.com	Data Science	2025-11-06 08:35:23.0	 Edit  Delete
3	SV003	Michael Brown	michael.b@email.com	Software Engineering	2025-11-06 08:35:23.0	 Edit  Delete
2	SV002	Emily Johnson	emily.j@email.com	Information Technology	2025-11-06 08:35:23.0	 Edit  Delete
1	SV001	John Smith	john.smith@email.com	Computer Science	2025-11-06 08:35:23.0	 Edit  Delete

`Class.forName("com.mysql.cj.jdbc.Driver")` allows java to talk to MySQL.

`DriverManager.getConnection(...)` create a connection to student_management database.

"`SELECT * FROM students ORDER BY id DESC`" get all the student information from the database.

while (rs.next()) goes through each row and prints out student info inside <tr> table rows.

Each row includes an Edit and Delete links with the student's ID.

“Catch” shows a message if the JDBC driver or database fails.

“.close” frees the ResultSet, Statement, and Connection at the end.

Ex2: Create Operation


Add New Student

Student Code *

Full Name *

Email

Major

 Save Student



Student Management System

Student added successfully

+ Add New Student

ID	Student Code	Full Name	Email	Major	Created At	Actions
7	SV006	Adolf Hitler	PolandInvader@gmail.com	Art	2025-11-06 10:17:23.0	Edit Delete
5	SV005	David Wilson	david.w@email.com	Computer Science	2025-11-06 08:35:23.0	Edit Delete

add_student.jsp:

Create a “Add New Student” button.

If there’s an error in the URL (?error=...), it’ll show a message.

The form sends data via “POST” to process_add.jsp.

The form comprises:

Fields:

- student_code (required, 2 capital letters + number)
- full_name (required)
- email (optional, valid format)
- major (optional)

Buttons:

- Save Student => submits the form
- Cancel => goes back to the student list page

process_add.jsp:

Step 1: get the form's data:

Reads "student_code", "full_name", "email", and "major" from the form.

Step 2: validation:

If "student_code" or "full_name" is missing => display error under the text field.

Step 3: database connection:

Loads the MySQL driver and connects to the "student_management" database.

Step 4: insert data:

Uses a "PreparedStatement" to safely insert the student record:

"INSERT INTO students (student_code, full_name, email, major)

VALUES (?, ?, ?, ?)"

Step 5: redirect results:

- If successful => goes to list_students.jsp with a success message
- If failed => returns to add_students.jsp with an error message

Step 6: error handling:

Detects duplicate student codes and database or driver issue.

Step 7: close resources:

Closes database connections to prevent leaks.

Ex3:



Student Management System

Student updated successfully

[+ Add New Student](#)

ID	Student Code	Full Name	Email	Major	Created At	Actions
7	SV006	Leon Kennedy	RaccoonPD@gmail.com	Police	2025-11-06 10:17:23.0	Edit Delete

edit_student.jsp:

Step 1: gets “id” from the URL.

Step 2: checks if “id” is missing or invalid => redirects with error.

Step 3: converts “id” to integer

Step 4: connects to “student_management” database.

Step 5: prepares SQL: “SELECT * FROM students WHERE id = ?”.

Step 6: executes query and gets the student record.

Step 7: stores “student_code”, “full_name”, “email”, and “major”.

Step 8: if no record found => redirects with “Student not found”.

Step 9: shows error message if query fails.

Step 10: closes all database resources.

Process_edit.jsp:

Step 1: reads “id”, “full_name”, “email”, and “major” from the form.

Step 2: checks for missing data => redirects with error.

Step 3: converts “id” to integer.

Step 4: connects to “student_management” database.

Step 5: prepare SQL: “UPDATE students SET full_name=?, email=?, major=?
WHERE id=?”.

Step 6: fills the query with form values.

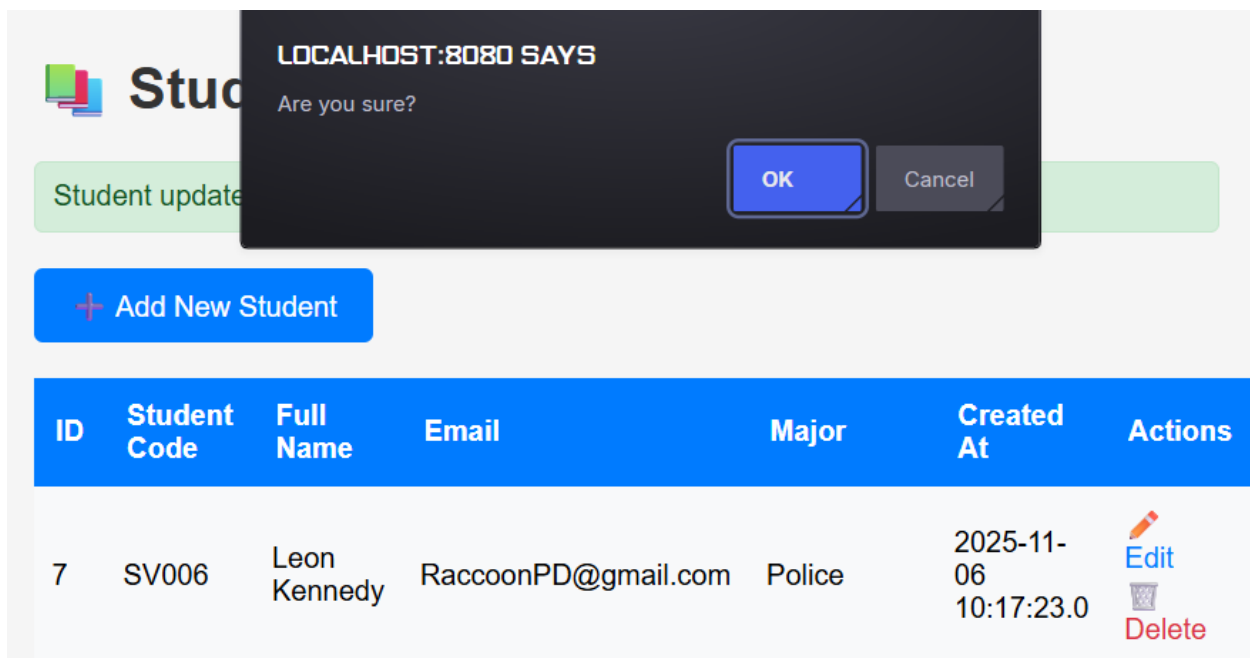
Step 7: executes update command.

Step 8: if successful => redirects to list with success message.

Step 9: if failed => redirects back with error.

Step 10: closes all database resources.

Ex4:



The screenshot shows a web application interface for student management. A dark grey confirmation dialog box is centered on the screen, displaying the text "LOCALHOST:8080 SAYS" and "Are you sure?". Below the text are two buttons: "OK" (blue) and "Cancel" (grey). The background interface includes a header with a logo and the word "Student", a green button labeled "Student update", a blue button labeled "+ Add New Student", and a table with student records.

ID	Student Code	Full Name	Email	Major	Created At	Actions
7	SV006	Leon Kennedy	RaccoonPD@gmail.com	Police	2025-11-06 10:17:23.0	Edit Delete

delete_student.jsp:

Step 1: gets the “id” parameter from the URL to know which student to delete.

Step 2: check if the “id” is valid; if not, redirects with an error.

Step 3: converts the “id” string to an integer.

Step 4: loads the MySQL JDBC driver and connects to the database.

Step 5: prepares a SQL “DELETE” statement using “PreparedStatement”.

Step 6: executes the delete query and checks if a row was affected.

Step 7: redirects with a success or “not found” message.

Step 8: catches SQL errors like foreign key constraints.

Step 9: closes the statement and database connection in “finally”.