

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 | <a href="#"> Edit</a><br><a href="#"> Delete</a> |
| 4  | SV004        | Sarah Davis   | sarah.d@email.com    | Data Science           | 2025-11-06 08:35:23.0 | <a href="#"> Edit</a><br><a href="#"> Delete</a> |
| 3  | SV003        | Michael Brown | michael.b@email.com  | Software Engineering   | 2025-11-06 08:35:23.0 | <a href="#"> Edit</a><br><a href="#"> Delete</a> |
| 2  | SV002        | Emily Johnson | emily.j@email.com    | Information Technology | 2025-11-06 08:35:23.0 | <a href="#"> Edit</a><br><a href="#"> Delete</a> |
| 1  | SV001        | John Smith    | john.smith@email.com | Computer Science       | 2025-11-06 08:35:23.0 | <a href="#"> Edit</a><br><a href="#"> Delete</a> |

`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 \***  
SV006

**Full Name \***  
Adolf Hitler

**Email**  
PolandInvader@gmail.com

**Major**  
Art

 Save Student     Cancel



# 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 | <a href="#"> Edit</a><br><a href="#"> Delete</a> |
| 5  | SV005        | David Wilson | david.w@email.com       | Computer Science | 2025-11-06 08:35:23.0 | <a href="#"> Edit</a><br><a href="#"> Delete</a> |

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<br>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 for managing students. At the top, there is a navigation bar with a logo and some menu items. Below it, a modal dialog box is open, asking "Are you sure?". The main content area displays a table of student data. The table has columns for ID, Student Code, Full Name, Email, Major, Created At, and Actions. One row is visible, showing ID 7, Student Code SV006, Full Name Leon Kennedy, Email RaccoonPD@gmail.com, Major Police, and Created At 2025-11-06 10:17:23.0. The Actions column contains links for Edit and Delete.

| 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 | <a href="#">Edit</a> <a href="#">Delete</a> |

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”.