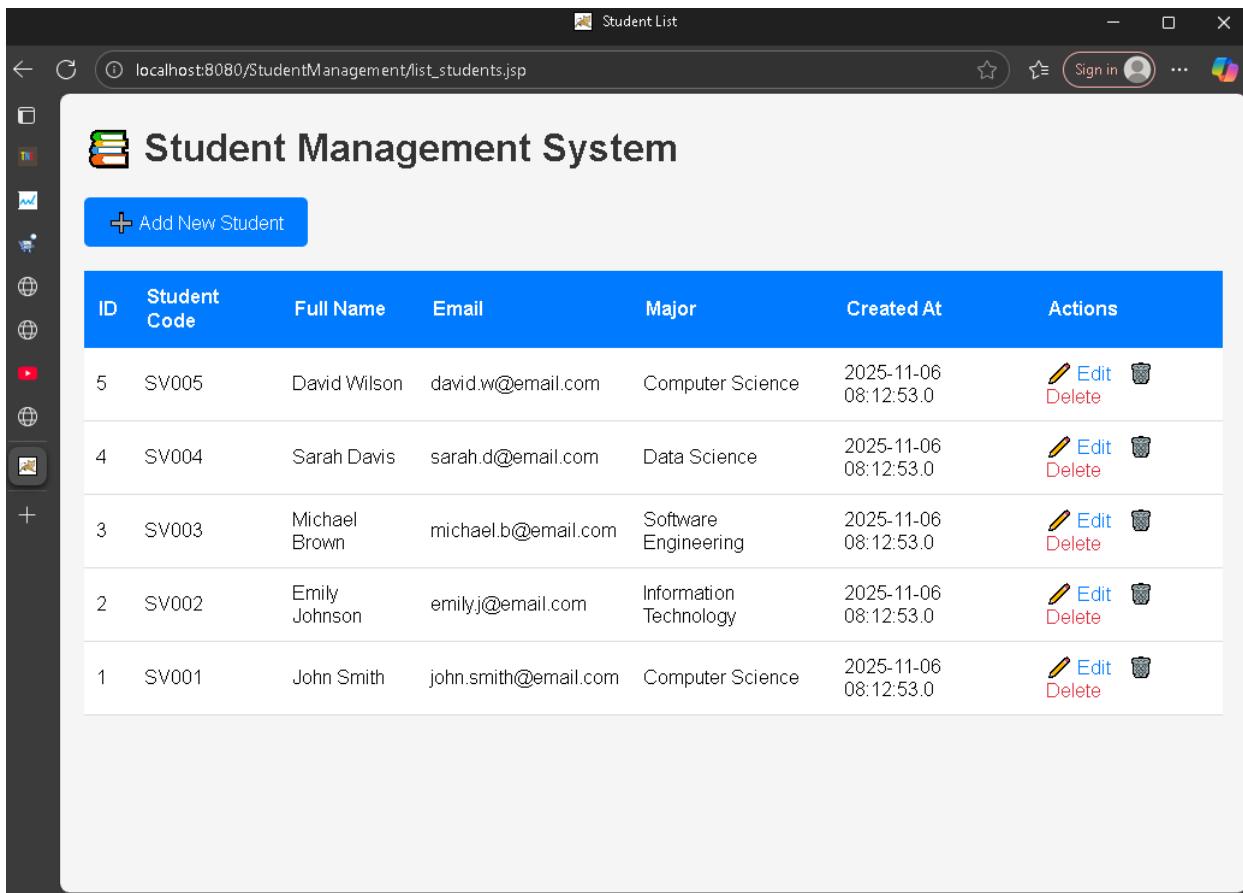


1. Display Student List



The screenshot shows a web browser window titled "Student List" with the URL "localhost:8080/StudentManagement/list_students.jsp". The page features a sidebar with various icons and a main content area titled "Student Management System". A blue button labeled "+ Add New Student" is visible. Below it is a table with columns: ID, Student Code, Full Name, Email, Major, Created At, and Actions. The table contains five rows of student data:

ID	Student Code	Full Name	Email	Major	Created At	Actions
5	SV005	David Wilson	david.w@email.com	Computer Science	2025-11-06 08:12:53.0	Edit Delete
4	SV004	Sarah Davis	sarah.d@email.com	Data Science	2025-11-06 08:12:53.0	Edit Delete
3	SV003	Michael Brown	michael.b@email.com	Software Engineering	2025-11-06 08:12:53.0	Edit Delete
2	SV002	Emily Johnson	emily.j@email.com	Information Technology	2025-11-06 08:12:53.0	Edit Delete
1	SV001	John Smith	john.smith@email.com	Computer Science	2025-11-06 08:12:53.0	Edit Delete

When the user accesses the `list_students.jsp` page, Tomcat compiles the JSP file into a Java servlet and executes it on the server. The page begins by importing the necessary `java.sql` classes and setting UTF-8 encoding to support Unicode text. It then defines the user interface using HTML and CSS, which provides a clean layout for displaying the student list and action buttons. Next, the JSP script connects to the MySQL database `student_management` using the JDBC driver (`com.mysql.cj.jdbc.Driver`) and the `DriverManager.getConnection()` method. A SQL query (`SELECT * FROM students ORDER BY id DESC`) is executed to retrieve all student records. The result set is processed in a while loop, and each row is dynamically printed into the HTML table using JSP expression tags (`<%= ... %>`).

The page also supports user feedback through messages—if redirected from another page with a message or error parameter, it displays a green success box or a red error box at the top. Finally, the code ensures that all database resources (`ResultSet`, `Statement`, and `Connection`) are closed properly in a `finally` block to avoid memory leaks.

As a result, `list_students.jsp` serves as the main view for displaying student data dynamically from the database, with interactive links to **Add**, **Edit**, and **Delete** student records.

2.1 Create Add Student Form

When the user opens the Add New Student page, Tomcat renders the JSP as an HTML form for inputting student information. The form uses UTF-8 encoding and styled CSS for a clean interface. It contains required fields for *Student Code* and *Full Name*, optional fields for *Email* and *Major*, and uses an HTML validation pattern ([A-Z]{2}[0-9]{3,}) to ensure proper student code format.

If the page is redirected with an error parameter (e.g., from process_add.jsp), a red error box appears at the top to notify the user.

When the form is submitted, data is sent via POST to process_add.jsp, which handles validation and database insertion.

The page also includes a Cancel button linking back to list_students.jsp.

As a result, add_student.jsp provides a user-friendly interface for entering new student data, performing basic client-side validation before submission, and displaying errors if necessary.

The screenshot shows a web browser window with the URL 'localhost:8080/StudentManagement/add_student.jsp'. The main content is a form titled '+ Add New Student'. It has four input fields: 'Student Code *' (with placeholder 'e.g., SV001'), 'Full Name *' (placeholder 'Enter full name'), 'Email' (placeholder 'student@email.com'), and 'Major' (placeholder 'e.g., Computer Science'). Below the form are two buttons: a green 'Save Student' button with a disk icon and a grey 'Cancel' button.

2.2 Process Add Student

When the user clicks the Add New Student, process_add.jsp is executed on the server.

The page retrieves input values from the form (student_code, full_name, email, major) and

checks for missing required fields.

If required fields are missing, it redirects back to add_student.jsp with an error message.

Otherwise, it connects to the student_management MySQL database using JDBC and executes an **INSERT** statement to add the new record.

If the operation succeeds, the user is redirected to list_students.jsp with a success message.

If the student code already exists or another SQL error occurs, it redirects with an appropriate error.

All database resources are closed properly at the end.

Valid data:

Add New Student

Student Code *: SV006

Full Name *: John Doe

Email: JohnDoe@Studentemail.com

Major: IT

Save Student Cancel

Student Management System

Student added successfully

ID	Student Code	Full Name	Email	Major	Created At	Actions
6	SV006	John Doe	JohnDoe@Studentemail.com	Information Technology	2025-11-06 10:24:29.0	Edit Delete
5	SV005	David Wilson	david.w@email.com	Computer Science	2025-11-06 08:12:53.0	Edit Delete
4	SV004	Sarah Davis	sarah.d@email.com	Data Science	2025-11-06 08:12:53.0	Edit Delete
3	SV003	Michael Brown	michael.b@email.com	Software Engineering	2025-11-06 08:12:53.0	Edit Delete
2	SV002	Emily Johnson	emily.j@email.com	Information Technology	2025-11-06 08:12:53.0	Edit Delete
1	SV001	John Smith	john.smith@email.com	Computer Science	2025-11-06 08:12:53.0	Edit Delete

Redirects to list_students.jsp. The student appears at the top of the table. A green success message box at the top displays: "Student added successfully".

Empty Required Field:

Redirects back to add_student.jsp. A error box appears showing: “**Please fill out this field**”. No data is added.

The image displays two screenshots of a web application interface. Both screenshots show a modal dialog titled "+ Add New Student".

Top Screenshot (Initial State):

- Student Code ***: An input field containing "e.g., SV001".
- Full Name ***: An input field containing "John".
- Email**: An input field containing "John@Studentemail.com".
- Major**: An input field containing "Information Technology".
- Buttons**: "Save Student" (green button) and "Cancel".

Bottom Screenshot (After Submission):

- Student Code ***: An input field containing "e.g., SV001", highlighted with a red border.
- Full Name ***: An input field containing "John". To the right of this field is a red-bordered error message box containing the text "Please fill out this field."
- Email**: An input field containing "John@Studentemail.com".
- Major**: An input field containing "Information Technology".
- Buttons**: "Save Student" (green button) and "Cancel".

Duplicate Code

Add New Student

Student Code *

Full Name *

Email

Major

Save StudentCancel

Add New Student

Student code already exists

Student Code *

Full Name *

Email

Major

Save StudentCancel

Redirects to add_student.jsp. A red error box appears with the text: “Student code already exists”. The database remains unchanged.

3.1: Create Edit Form

When the user accesses `edit_student.jsp`, Tomcat compiles and runs it to load the student data for editing. The page reads the student ID from the request parameter (`?id=`). If the ID is missing or invalid, it redirects back to `list_students.jsp` with an error message.

Next, the script connects to the `student_management` MySQL database using JDBC, retrieves the student's record with a `SELECT * FROM students WHERE id = ?` query, and fills the form fields with existing values. The student code field is read-only, while the name, email, and major fields can be edited.

If an error or invalid ID occurs, a red error box is displayed at the top of the page. The form then submits the updated information via POST to `process_edit.jsp` for processing and saving changes.

Overall, `edit_student.jsp` provides an interface for updating existing student details, ensuring proper validation and safe database access.

The screenshot shows a web browser window with the URL `localhost:8080/StudentManagement/edit_student.jsp?id=6`. The page title is "Edit Student Information". The form contains the following fields:

- Student Code**: A read-only input field containing "SV006". Below it, a note says "Cannot be changed".
- Full Name ***: An input field containing "John Doe".
- Email**: An input field containing "JohnDoe@Studentemail.com".
- Major**: An input field containing "Information Technology".

At the bottom of the form are two buttons: a yellow "Update" button with a disk icon and a grey "Cancel" button.

When the Edit Student button is clicked, the browser navigates to `edit_student.jsp?id=....`. The page loads the selected student's information from the database and displays it in a pre-filled form.

At the top, you can see the student's details such as Student Code, Full Name, Email, and Major. The Student Code field is marked as *read-only*, meaning it cannot be modified. The other fields are editable, allowing the user to update the student's information easily.

The layout is clean and centered, with labeled input boxes and two buttons —  Update (to save changes) and Cancel (to return to the list). This confirms that the form correctly retrieves and displays the student's existing data for editing.

3.2: Process Update

When the user submits the Edit Student form, this JSP page processes the update request. It retrieves the input values (id, full_name, email, major) from the form and validates them — ensuring that the ID and name are not empty.

The code then connects to the student_management MySQL database using the JDBC driver and executes an SQL UPDATE query to modify the selected student's record. If the update is successful, the user is redirected to list_students.jsp with a success message. Otherwise, an error message is shown on the edit page.

All database resources are properly closed in the finally block to prevent memory leaks. This page ensures that only valid updates are applied and provides clear feedback to the user after submission.

Case

Change name from "John Smith" to "John Doe"

Edit Student Information

Student Code
SV001
Cannot be changed

Full Name *
John Doe

Email
john.smith@email.com

Major
Computer Science

Update **Cancel**

Student Management System

Student updated successfully

[+ Add New Student](#)

ID	Student Code	Full Name	Email	Major	Created At	Actions
6	SV006	John Doe	JohnDoe@Studentemail.com	Information Technology	2025-11-06 10:24:29.0	Edit Delete
5	SV005	David Wilson	david.w@email.com	Computer Science	2025-11-06 08:12:53.0	Edit Delete
4	SV004	Sarah Davis	sarah.d@email.com	Data Science	2025-11-06 08:12:53.0	Edit Delete
3	SV003	Michael Brown	michael.b@email.com	Software Engineering	2025-11-06 08:12:53.0	Edit Delete
2	SV002	Emily Johnson	emily.j@email.com	Information Technology	2025-11-06 08:12:53.0	Edit Delete
1	SV001	John Doe	john.smith@email.com	Computer Science	2025-11-06 08:12:53.0	Edit Delete

When changing the name from “John Smith” to “John Doe” and clicking Update, the system

redirects to the student list page.

The success message “Student updated successfully” appears in a green box, and the list now shows John Doe instead of the old name.

Access edit page with id=999

The screenshot shows a web browser window for the "Student Management System". The URL in the address bar is `localhost:8080/StudentManagement/list_students.jsp?error=Student%20not%20found`. The main content area displays a red box containing the text "Student not found". Below this, there is a blue button labeled "+ Add New Student". A table lists six student records:

ID	Student Code	Full Name	Email	Major	Created At	Actions
6	SV006	John Doe	JohnDoe@Studentemail.com	Information Technology	2025-11-06 10:24:29.0	Edit Delete
5	SV005	David Wilson	david.w@email.com	Computer Science	2025-11-06 08:12:53.0	Edit Delete
4	SV004	Sarah Davis	sarah.d@email.com	Data Science	2025-11-06 08:12:53.0	Edit Delete
3	SV003	Michael Brown	michael.b@email.com	Software Engineering	2025-11-06 08:12:53.0	Edit Delete
2	SV002	Emily Johnson	emily.j@email.com	Information Technology	2025-11-06 08:12:53.0	Edit Delete
1	SV001	John Doe	john.smith@email.com	Computer Science	2025-11-06 08:12:53.0	Edit Delete

When trying to access the edit page with a non-existent ID (e.g., id=999), the system redirects back to the list page.

An error message “Student not found” appears in a red box at the top, and no data is displayed.

Submit with blank name

Edit Student Information

Student Code

SV001

Cannot be changed

Full Name *

I

Email

john.smith@email.com

Major

Computer Science

 Update

Cancel

Edit Student Information

Student Code

SV001

Cannot be changed

Full Name *

Email

john.smith@email.com

 Please fill out this field.

Major

Computer Science

 Update

Cancel

When submitting the edit form with the **Full Name** field left blank, the system prevents the update.

The page displays an error message “**Please fill out this field**”, indicating that the name field must be filled before saving.

4.1: Implement Delete

When the user clicks “Delete”, the page retrieves the student’s id from the request and checks if it’s valid.

Then, it connects to the student_management database and executes:

```
DELETE FROM students WHERE id = ?
```

If the student exists, it redirects to list_students.jsp with a success message.

If no rows are affected, it shows “Student not found”.

The code also handles errors:

- Foreign key constraint → shows “Cannot delete: has related records”
- SQL or connection error → shows “Database error”

Finally, all database resources are closed properly to prevent leaks.

4.2: DELETE OPERATION

Enhance list_students.jsp to include a Delete feature for managing student records.

The table now allows users to easily delete students with confirmation and visual feedback through success or error messages.

Click “Delete” on *Test Student* → Confirm in dialog.

The student was successfully deleted and no longer appears in the table. A green success notification is shown at the top of the page.

Student List

localhost:8080 says
Are you sure?

Add New Student

ID	Student Code	Full Name	Email	Major	Created At	Actions
6	SV006	John Doe	JohnDoe@Studentemail.com	Information Technology	2025-11-06 10:24:29.0	
5	SV005	David Wilson	david.w@email.com	Computer Science	2025-11-06 08:12:53.0	
4	SV004	Sarah Davis	sarah.d@email.com	Data Science	2025-11-06 08:12:53.0	
3	SV003	Michael Brown	michael.b@email.com	Software Engineering	2025-11-06 08:12:53.0	
2	SV002	Emily Johnson	emily.j@email.com	Information Technology	2025-11-06 08:12:53.0	
1	SV001	John Doe	john.smith@email.com	Computer Science	2025-11-06 08:12:53.0	

localhost:8080/StudentManagement/delete_student.jsp?id=5

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 08:12:53.0	
4	SV004	Sarah Davis	sarah.d@email.com	Data Science	2025-11-06 08:12:53.0	
3	SV003	Michael Brown	michael.b@email.com	Software Engineering	2025-11-06 08:12:53.0	
2	SV002	Emily Johnson	emily.j@email.com	Information Technology	2025-11-06 08:12:53.0	
1	SV001	John Doe	john.smith@email.com	Computer Science	2025-11-06 08:12:53.0	

Click “Delete” on *Test Student* → Click **Cancel** in the confirmation dialog.
The dialog closed, no deletion occurred, and the student record stayed in the table

The screenshot shows a web-based Student Management System interface. At the top, there is a header bar with a logo, the URL "localhost:8080/StudentManagement/list_students.jsp", and various browser icons. Below the header is a title "Student Management System" with a book icon. A blue button labeled "+ Add New Student" is visible. The main content is a table with the following data:

ID	Student Code	Full Name	Email	Major	Created At	Actions
5	SV005	David Wilson	david.w@email.com	Computer Science	2025-11-06 08:12:53.0	
4	SV004	Sarah Davis	sarah.d@email.com	Data Science	2025-11-06 08:12:53.0	
3	SV003	Michael Brown	michael.b@email.com	Software Engineering	2025-11-06 08:12:53.0	
2	SV002	Emily Johnson	emily.j@email.com	Information Technology	2025-11-06 08:12:53.0	
1	SV001	John Doe	john.smith@email.com	Computer Science	2025-11-06 08:12:53.0	

