Nguyễn Ngọc Đình Trung

ITITIU20331

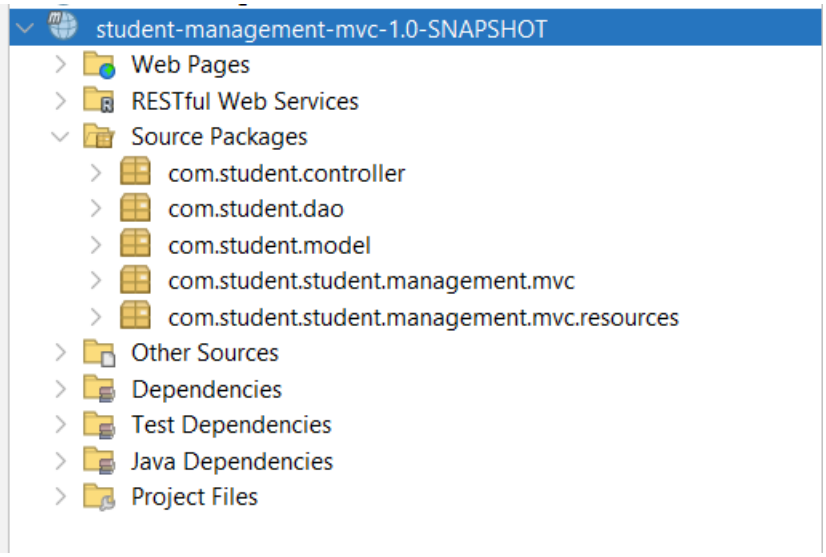
WEB APP REPORT

LAB 05

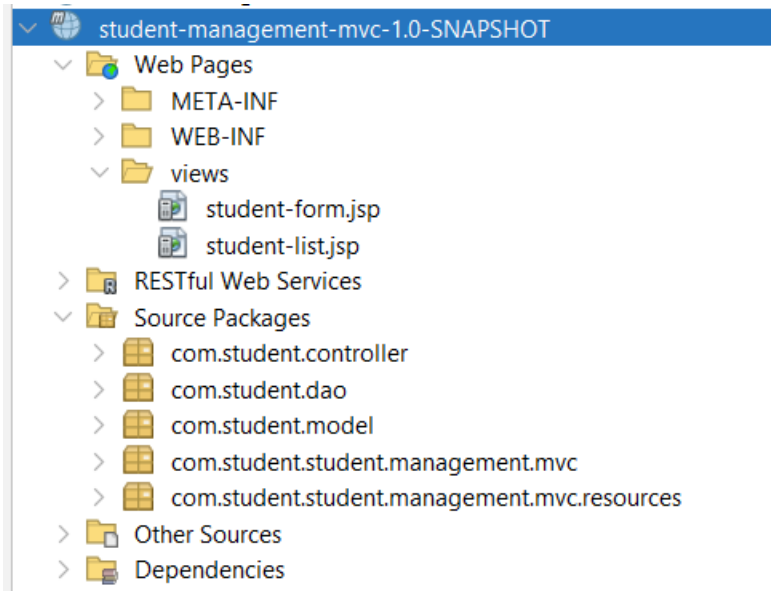
EX1:

TASK 1.1

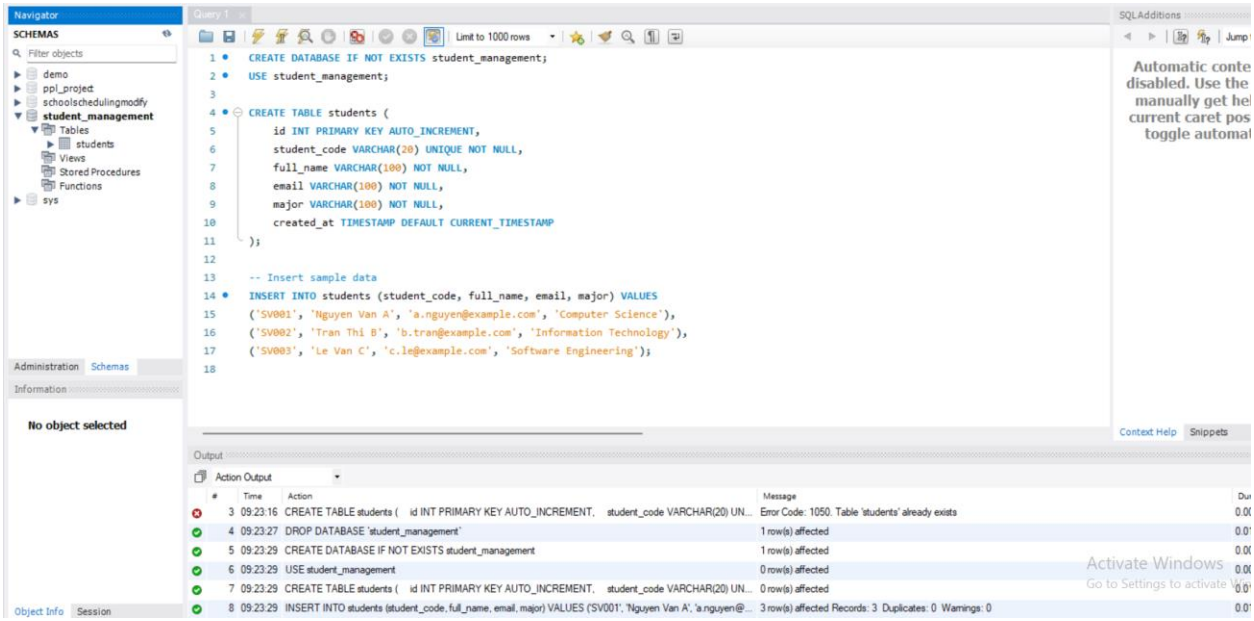
CREATE PROJECT AND CREATE JAVA PACKAGE



CREATE VIEWS:

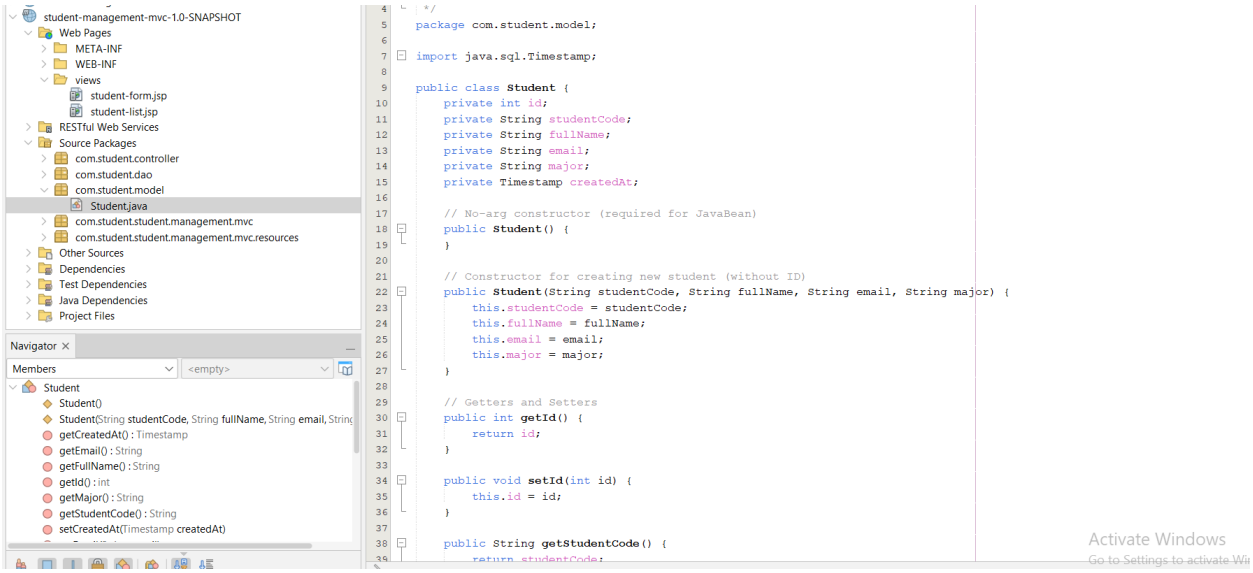


CONNECT MYSQL



TASK 1.2

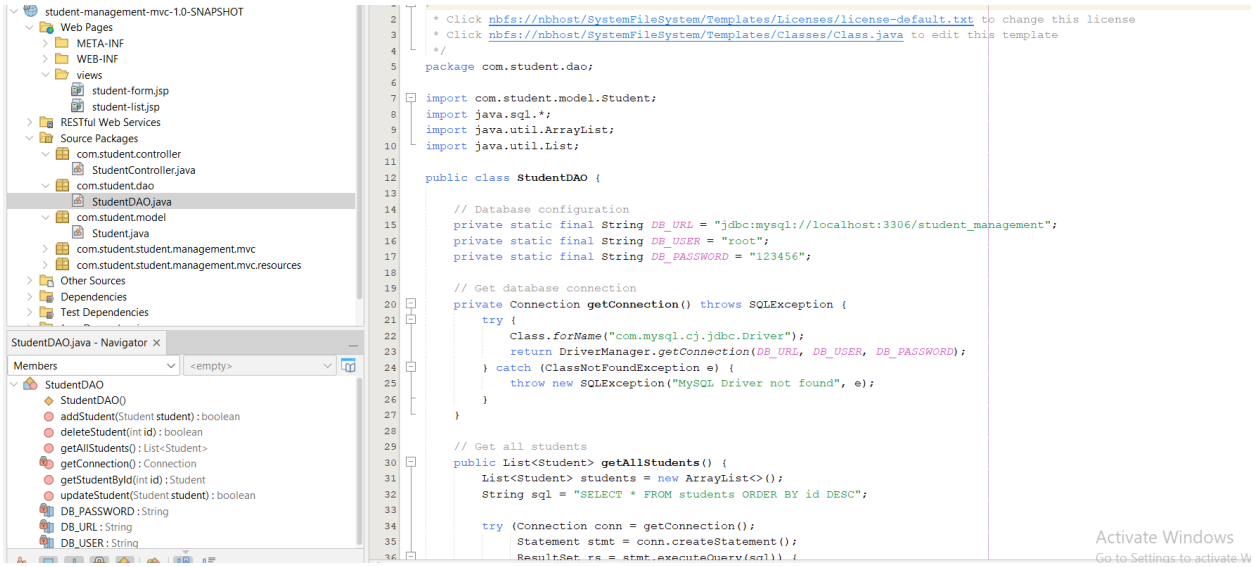
Create Student JavaBean



- CONSTRUCTORS: Many frameworks require a no-arg constructor so they can instantiate the object and then set properties using setters.
  This.studentCode = studentCode;
- This constructor helps you create an object quickly with its main properties.
- GETTER AND SETTER: This pattern is part of the JavaBeans convention: private fields + public getters/setters + no-arg constructor. It makes the object compatible with tools and frameworks that use reflection.
- @Override: means this method replaces the default

TASK 1.3

Create StudentDAO

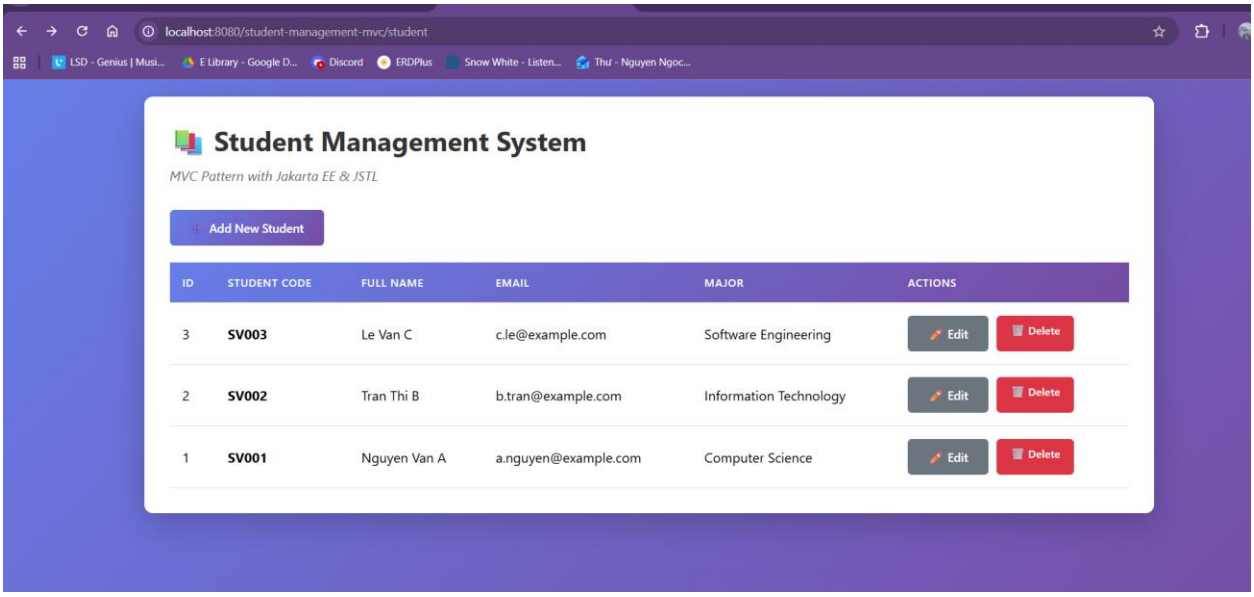


DAO.Java: include connect dtb (MySQL)

Database Configuration: The class defines constants for database connection details (DB_URL, DB_USER, DB_PASSWORD). These values are used to establish connections with the MySQL database.

- Class.forName(): to load driver
- DriverManager.getConnection(): to connect dtb (MySQl)

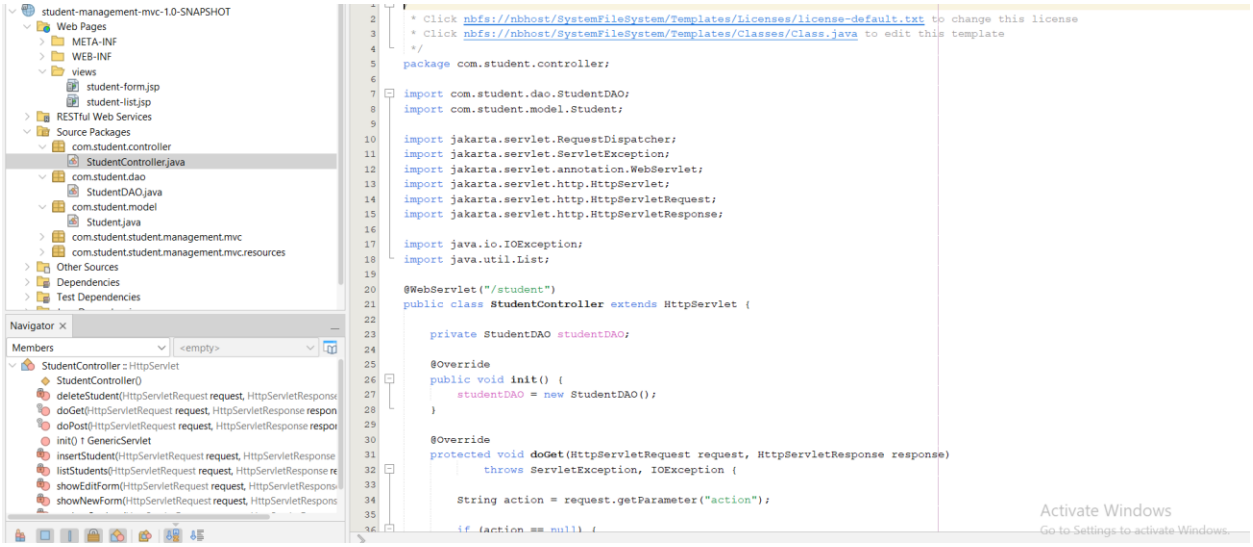
CRUD Operations: The class provides full CRUD functionality (Create, Read, Update, Delete) for the Student model.



EX2:

TASK 2.1

Create Basic Servlet



TASK 2.2

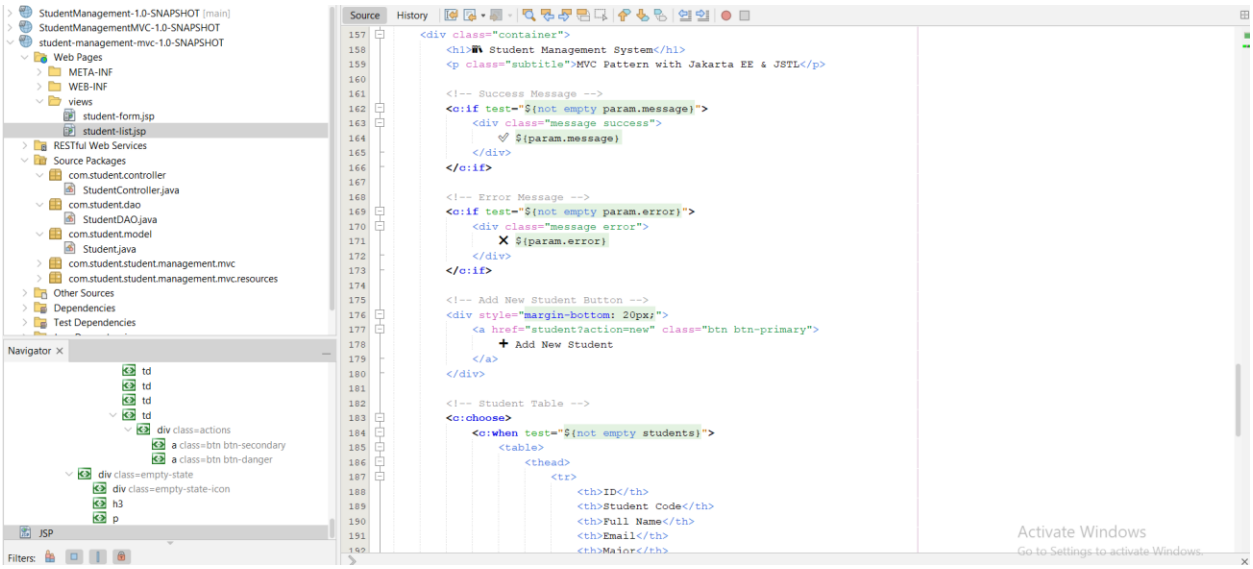
Add More CRUD Methods

- showNewForm() forwards to form: method correctly forwards the user to the JSP form used for creating a new student.
- showEditForm() loads student and forwards: method loads the specific student by ID and sends it to the form for editing.
- doPost() routes correctly: handles form submissions and routes by action:
  switch (action) {
  case "insert": insertStudent(...); break;
  case "update": updateStudent(...); break;
  }
- insertStudent() creates student: The method accepts form fields (studentCode, fullName, email, major), builds a new Student object, and sends it to studentDAO.addStudent().
- updateStudent() modifies student: retrieves updated form data, sets the ID, and forwards it to DAO for update
- deleteStudent() removes student: The method deletes a student based on ID

EX3:

TASK 3.1

Create Student List View



- Add JSTL taglib directive
  => This directive enables the use of JSTL Core tags in the JSP page.
- No scriptlets (no <% %>)
  => which are JSP scriptlets (embedding Java code directly into JSP).
- Use <c:if> for messages
  => conditionally displays HTML content. Use for showing success message when param.message exists or param.error exists
- Use <c:forEach> for student list
  => loops through the list of students provided by the controller.
- Use EL ${} for all data access
  => is used to print each field. Used to access: request parameters, request attributes and object properties
- Handle empty list case
  => works like a Java if–else statement.

TASK 3.2

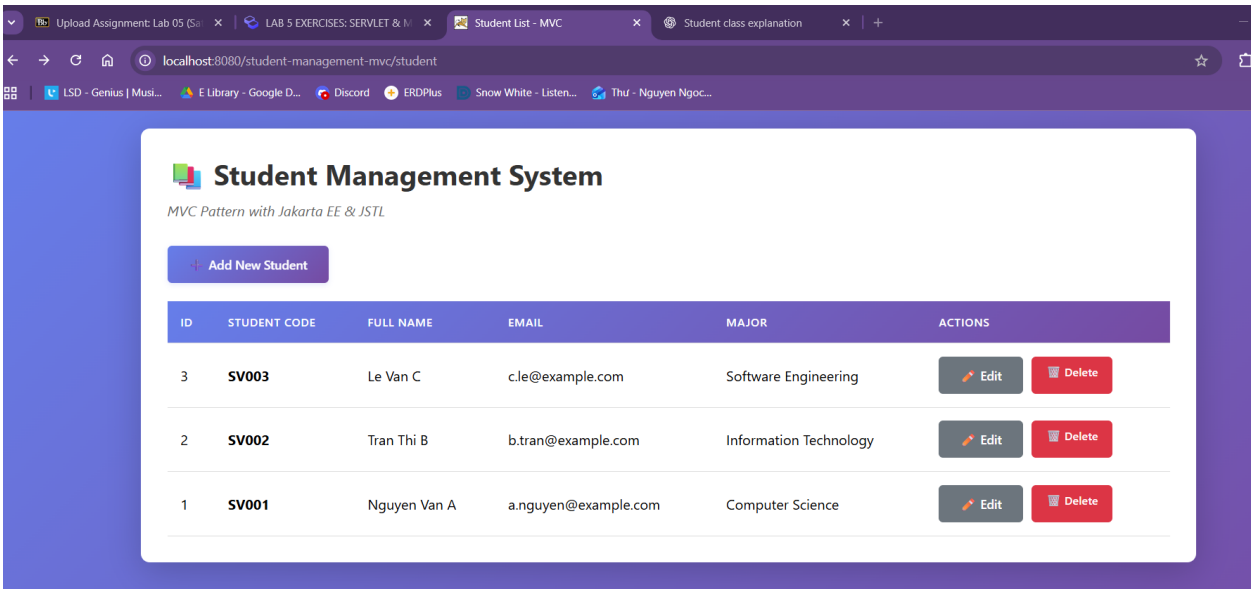
- Single form for both Add and Edit
  => Instead of having two separate JSP files (student-add.jsp and student-edit.jsp), the page dynamically adjusts based on whether a Student object is present in the request.
- Use <c:choose> for dynamic title
  => works like a switch or if-else
- Use <c:if> to show/hide fields conditionally
  => conditionally includes HTML elements

- Pre-fill values for edit mode
- => This improves user experience and reduces input errors
- No scriptlets
  => The form avoids any Java code (<% %>)

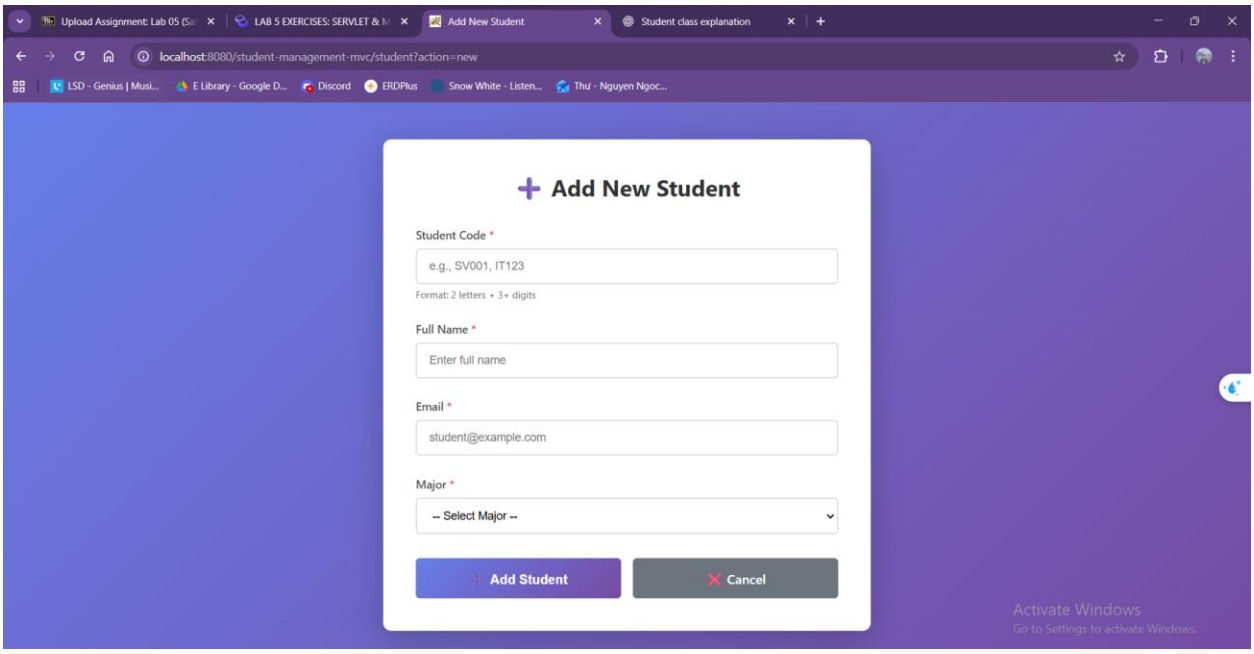
EX4:
TASK 4.1

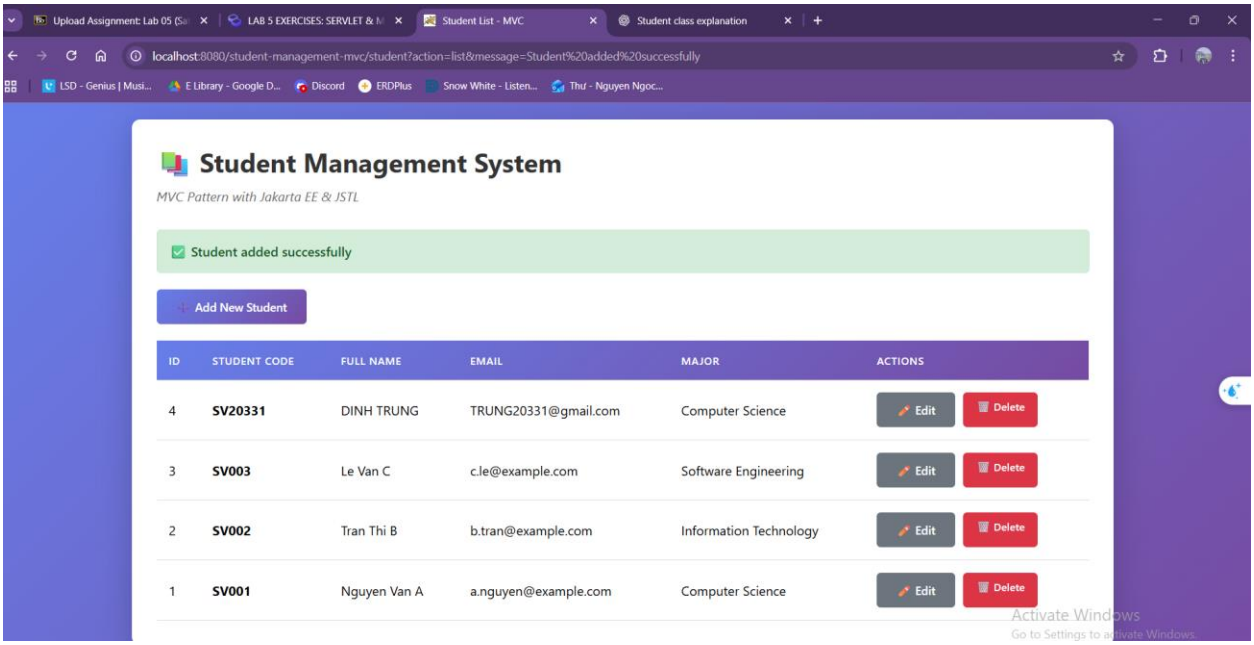
```java
// Add new student
public boolean addStudent(Student student) {
    String sql = "INSERT INTO students (student_code, full_name, email, major) VALUES (?, ?, ?, ?)";

    try (Connection conn = getConnection();
         PreparedStatement pstmt = conn.prepareStatement(sql)) {

        pstmt.setString(1, student.getStudentCode());
        pstmt.setString(2, student.getFullName());
        pstmt.setString(3, student.getEmail());
        pstmt.setString(4, student.getMajor());

        int rowsAffected = pstmt.executeUpdate();
        return rowsAffected > 0;

    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}

// Update student
public boolean updateStudent(Student student) {
    String sql = "UPDATE students SET student_code = ?, full_name = ?, email = ?, major = ? WHERE id = ?";

    try (Connection conn = getConnection();
         PreparedStatement pstmt = conn.prepareStatement(sql)) {

        pstmt.setString(1, student.getStudentCode());
        pstmt.setString(2, student.getFullName());
        pstmt.setString(3, student.getEmail());
        pstmt.setString(4, student.getMajor());
        pstmt.setInt(5, student.getId());

        int rowsAffected = pstmt.executeUpdate();
```


TASK 4.2
TEST:

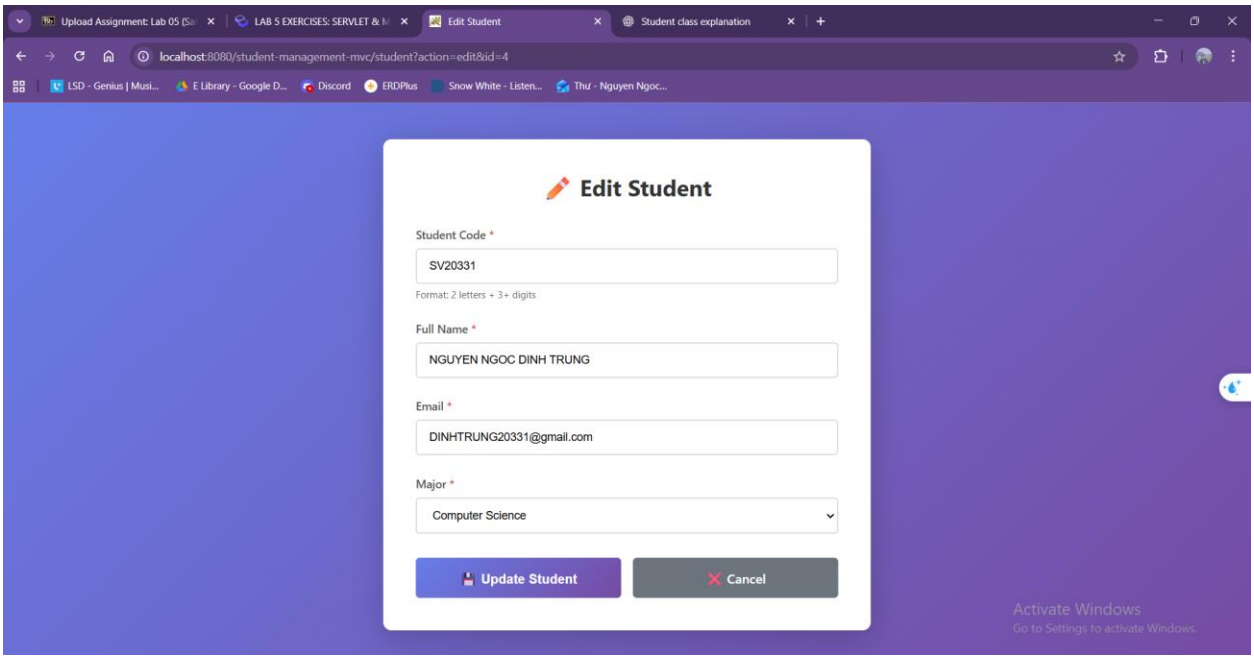


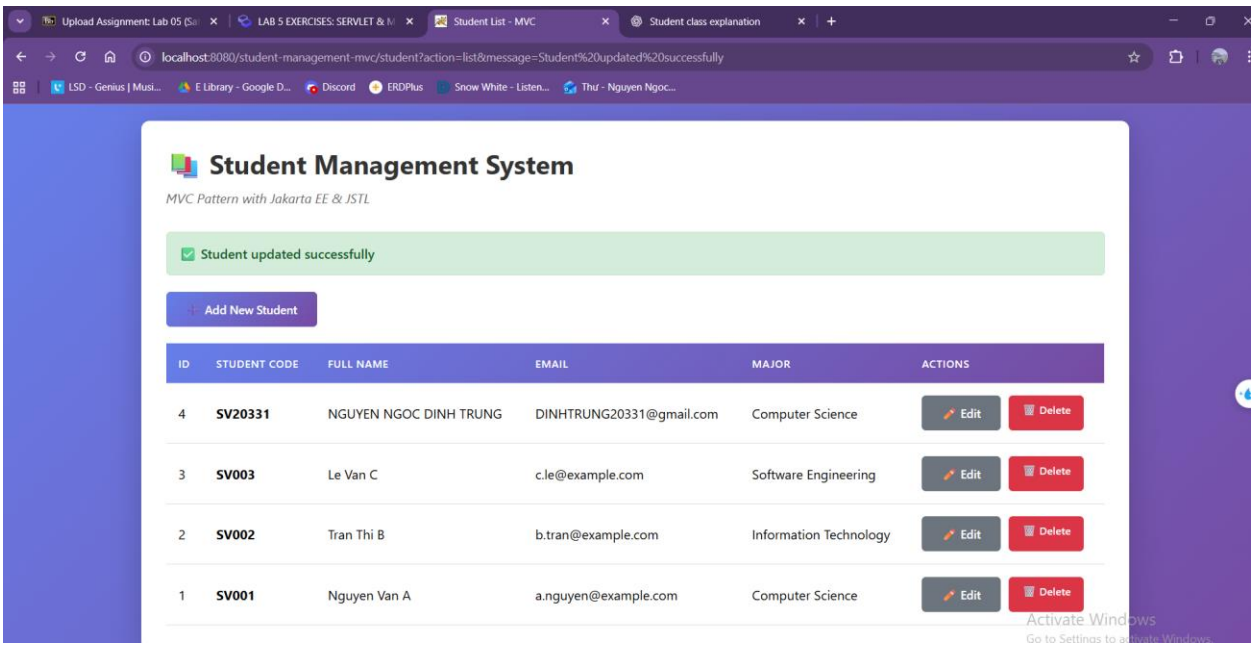
Add: Click "Add New Student"



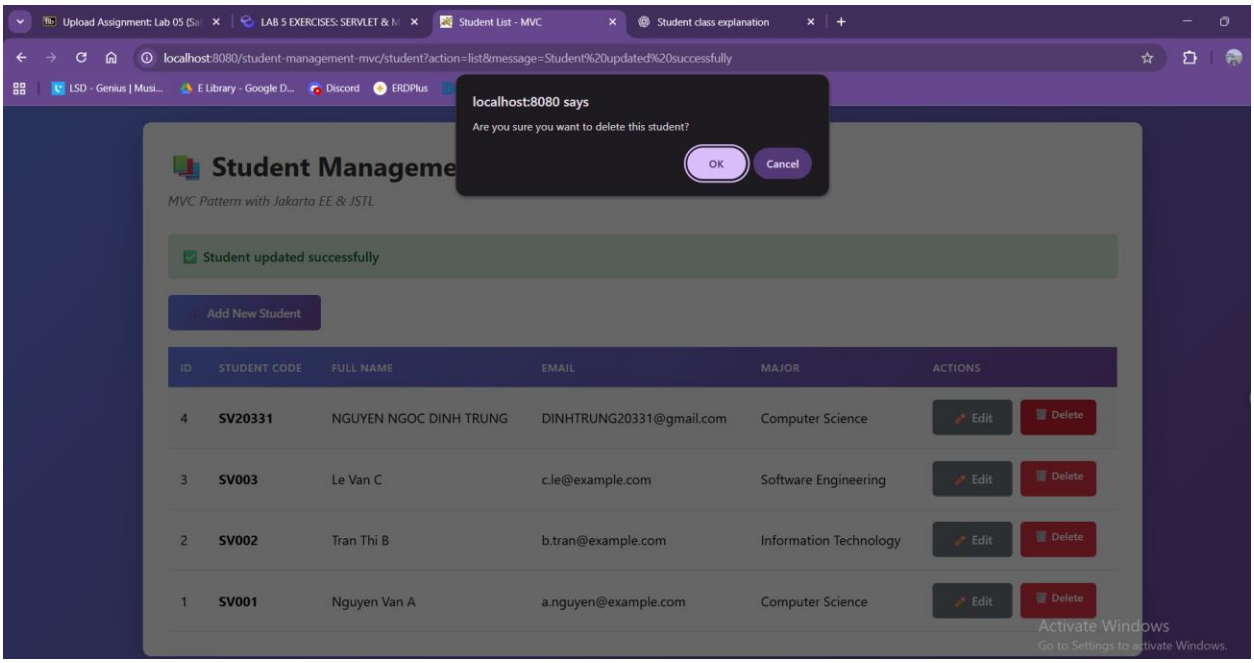


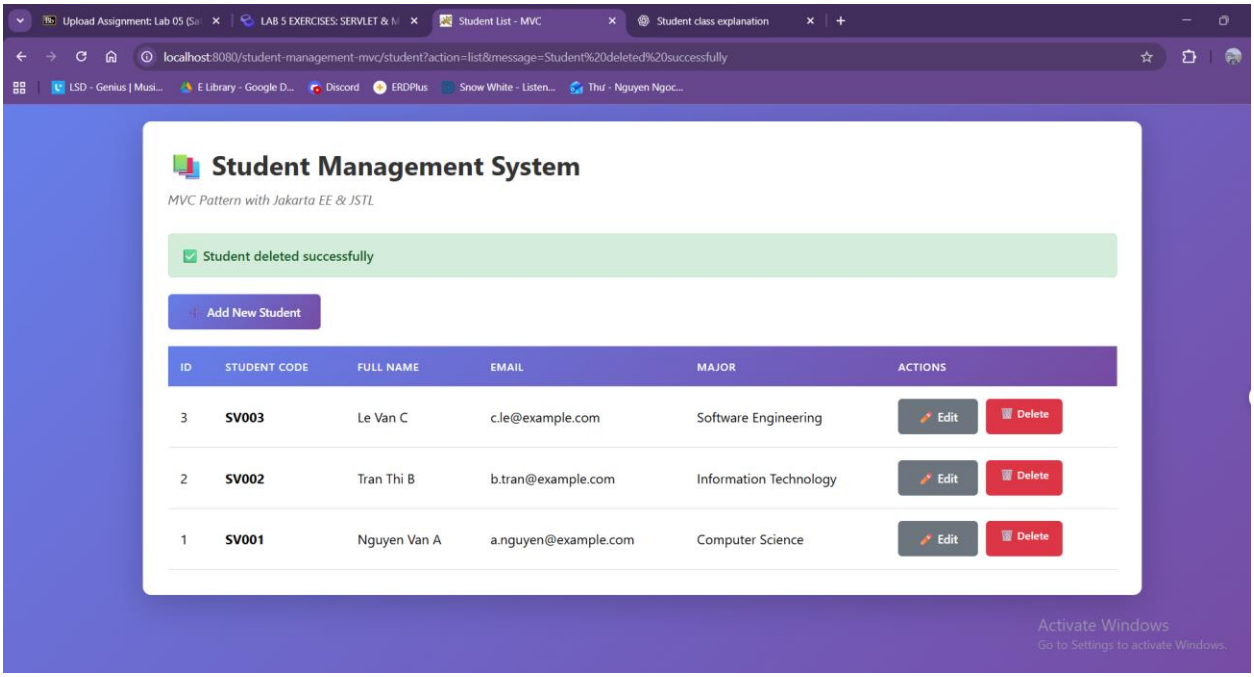
Edit: Click "Edit" on test student



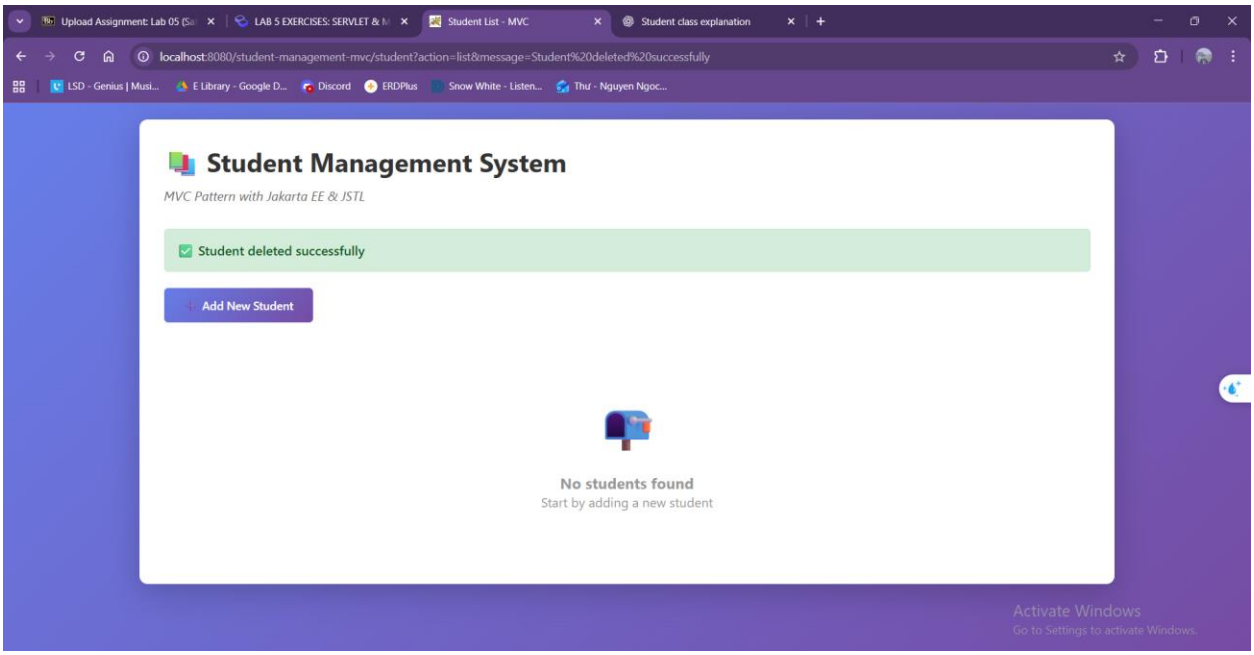


Delete: Click "Delete" on test student





Empty State: Delete all students


Upload Assignment: Lab 05 (Sa
LAB 5 EXERCISES: SERVLET & M
Student List - MVC
Student class explanation
localhost:8080/student-management-mvc/student?action=list&message=Student%20deleted%20successfully
LSD - Genius | Musi...
E Library - Google D...
Discord
ERDPlus
Snow White - Listen...
Thu - Nguyen Ngoc...
Student Management System
MVC Pattern with Jakarta EE & JSTL
Student deleted successfully
Add New Student
No students found
Start by adding a new student
Activate Windows
Go to Settings to activate Windows.