Lab 5 Homework

Name: Nguyễn Hữu Hoàng Nam

StudentID: ITCSIU23028

Exercise 5:

5.1

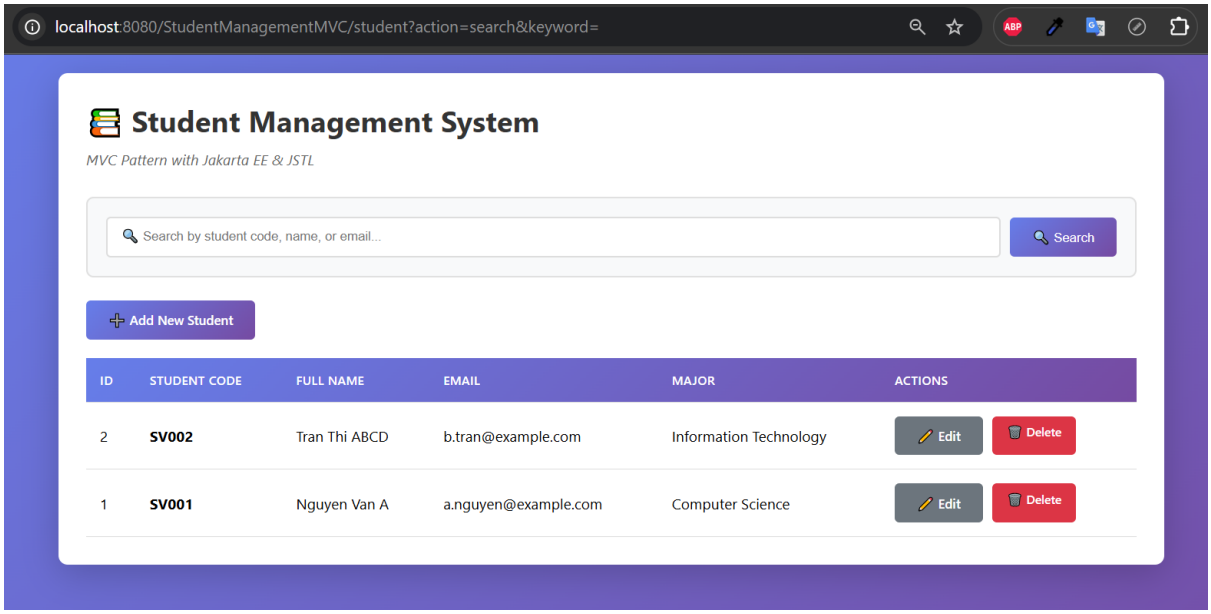
```java
public List<Student> searchStudents(String keyword) {
    List<Student> students = new ArrayList<>();

    if (keyword == null || keyword.trim().isEmpty()) {
        return students;
    }

    String sql = "SELECT * FROM students WHERE student_code LIKE ? OR full_name LIKE ? OR email LIKE ? ORDER BY id DESC";

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

        String searchPattern = "%" + keyword.trim() + "%";

        pstmt.setString(parameterIndex: 1, searchPattern);
        pstmt.setString(parameterIndex: 2, searchPattern);
        pstmt.setString(parameterIndex: 3, searchPattern);

        ResultSet rs = pstmt.executeQuery();

        while (rs.next()) {
            Student student = new Student();
            student.setId(rs.getInt(columnLabel: "id"));
            student.setStudentCode(rs.getString(columnLabel: "student_code"));
            student.setFullName(rs.getString(columnLabel: "full_name"));
            student.setEmail(rs.getString(columnLabel: "email"));
            student.setMajor(rs.getString(columnLabel: "major"));
            student.setCreatedAt(rs.getTimestamp(columnLabel: "created_at"));
            students.add(student);
        }

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

    return students;
}
```

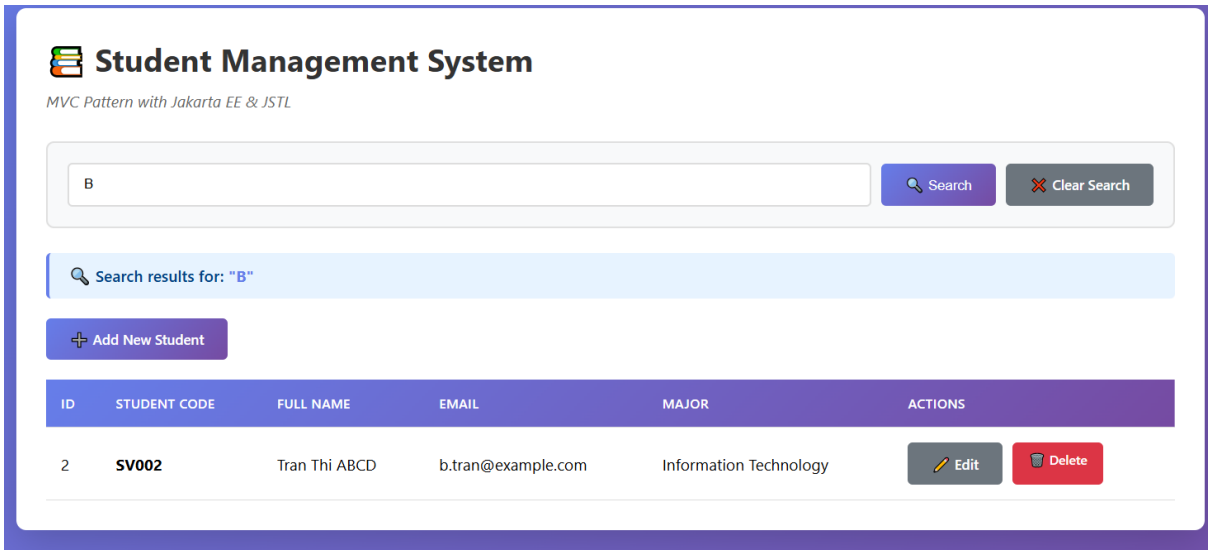
5.2

```java
    }

    private void searchStudents(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        String keyword = request.getParameter("keyword");

        List<Student> students;
        if (keyword == null || keyword.trim().isEmpty()) {
            students = studentDAO.getAllStudents();
        } else {
            students = studentDAO.searchStudents(keyword);
        }

        request.setAttribute("students", students);
        request.setAttribute("keyword", keyword);

        RequestDispatcher dispatcher = request.getRequestDispatcher("/views/student-list.jsp");
        dispatcher.forward(request, response);
    }
```

5.3

```jsp
    </c:if>

    <!-- Search Box -->
    <div class="search-box">
      <form action="student" method="get">
        <input type="hidden" name="action" value="search" />
        <input
          type="text"
          name="keyword"
          value="${keyword}"
          placeholder="🔍 Search by student code, name, or email..."
          autocomplete="off"
        />
        <button type="submit" class="btn btn-search">🔍 Search</button>
        <c:if test="${not empty keyword}">
          <a href="student?action=list" class="btn btn-clear">
            ❌ Clear Search
          </a>
        </c:if>
      </form>
    </div>

    <!-- Search Results Message -->
    <c:if test="${not empty keyword}">
      <div class="search-result-message">
        🔍 Search results for: <strong>"${keyword}"</strong>
      </div>
    </c:if>
```





Exercise 6:

6.1

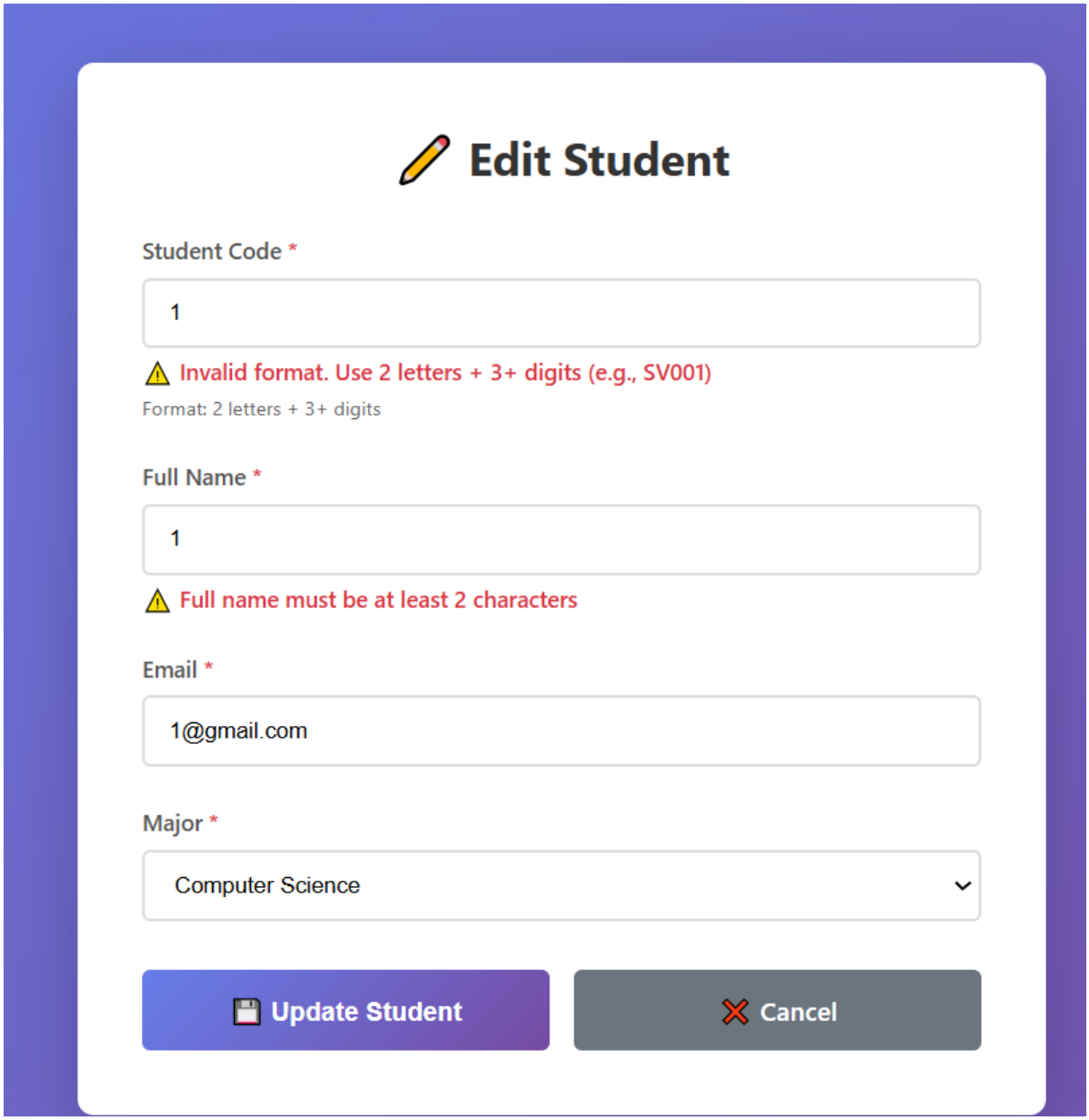
```java
private boolean validateStudent(Student student, HttpServletRequest request) {
    boolean isValid = true;

    String studentCode = student.getStudentCode();
    if (studentCode == null || studentCode.trim().isEmpty()) {
        request.setAttribute("errorCode", "Student code is required");
        isValid = false;
    } else {
        String codePattern = "[A-Z]{2}[0-9]{3,}";
        if (!studentCode.matches(codePattern)) {
            request.setAttribute("errorCode", "Invalid format. Use 2 letters + 3+ digits (e.g., SV001)");
            isValid = false;
        }
    }

    String fullName = student.getFullName();
    if (fullName == null || fullName.trim().isEmpty()) {
        request.setAttribute("errorName", "Full name is required");
        isValid = false;
    } else if (fullName.trim().length() < 2) {
        request.setAttribute("errorName", "Full name must be at least 2 characters");
        isValid = false;
    }

    String email = student.getEmail();
    if (email != null && !email.trim().isEmpty()) {
        String emailPattern = "^[A-Za-z0-9+_.-]+@(.+)$";
        if (!email.matches(emailPattern)) {
            request.setAttribute("errorEmail", "Invalid email format");
            isValid = false;
        }
    }

    String major = student.getMajor();
    if (major == null || major.trim().isEmpty()) {
        request.setAttribute("errorMajor", "Major is required");
        isValid = false;
    }

    return isValid;
}
```

6.2

```
private void insertStudent(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    String studentCode = request.getParameter("studentCode");
    String fullName = request.getParameter("fullName");
    String email = request.getParameter("email");
    String major = request.getParameter("major");

    Student newStudent = new Student(studentCode, fullName, email, major);

    if (!validateStudent(newStudent, request)) {
        request.setAttribute("student", newStudent);
        RequestDispatcher dispatcher = request.getRequestDispatcher("/views/student-form.jsp");
        dispatcher.forward(request, response);
        return;
    }

    if (studentDAO.addStudent(newStudent)) {
        response.sendRedirect("student?action=list&message=Student added successfully");
    } else {
        response.sendRedirect("student?action=list&error=Failed to add student");
    }
```

```
private void updateStudent(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    int id = Integer.parseInt(request.getParameter("id"));
    String studentCode = request.getParameter("studentCode");
    String fullName = request.getParameter("fullName");
    String email = request.getParameter("email");
    String major = request.getParameter("major");

    Student student = new Student(studentCode, fullName, email, major);
    student.setId(id);

    if (!validateStudent(student, request)) {
        request.setAttribute("student", student);
        RequestDispatcher dispatcher = request.getRequestDispatcher("/views/student-form.jsp");
        dispatcher.forward(request, response);
        return;
    }

    if (studentDAO.updateStudent(student)) {
        response.sendRedirect("student?action=list&message=Student updated successfully");
    } else {
        response.sendRedirect("student?action=list&error=Failed to update student");
    }
}
```

6.3



Exercise 7:

7.1

```java
public List<Student> getStudentsSorted(String sortBy, String order) {
    List<Student> students = new ArrayList<>();

    String validSortBy = validateSortBy(sortBy);
    String validOrder = validateOrder(order);

    String sql = "SELECT * FROM students ORDER BY " + validSortBy + " " + validOrder;

    try (Connection conn = getConnection();
         Statement stmt = conn.createStatement();
         ResultSet rs = stmt.executeQuery(sql)) {

        while (rs.next()) {
            Student student = new Student();
            student.setId(rs.getInt(columnLabel: "id"));
            student.setStudentCode(rs.getString(columnLabel: "student_code"));
            student.setFullName(rs.getString(columnLabel: "full_name"));
            student.setEmail(rs.getString(columnLabel: "email"));
            student.setMajor(rs.getString(columnLabel: "major"));
            student.setCreatedAt(rs.getTimestamp(columnLabel: "created_at"));
            students.add(student);
        }

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

    return students;
}
```

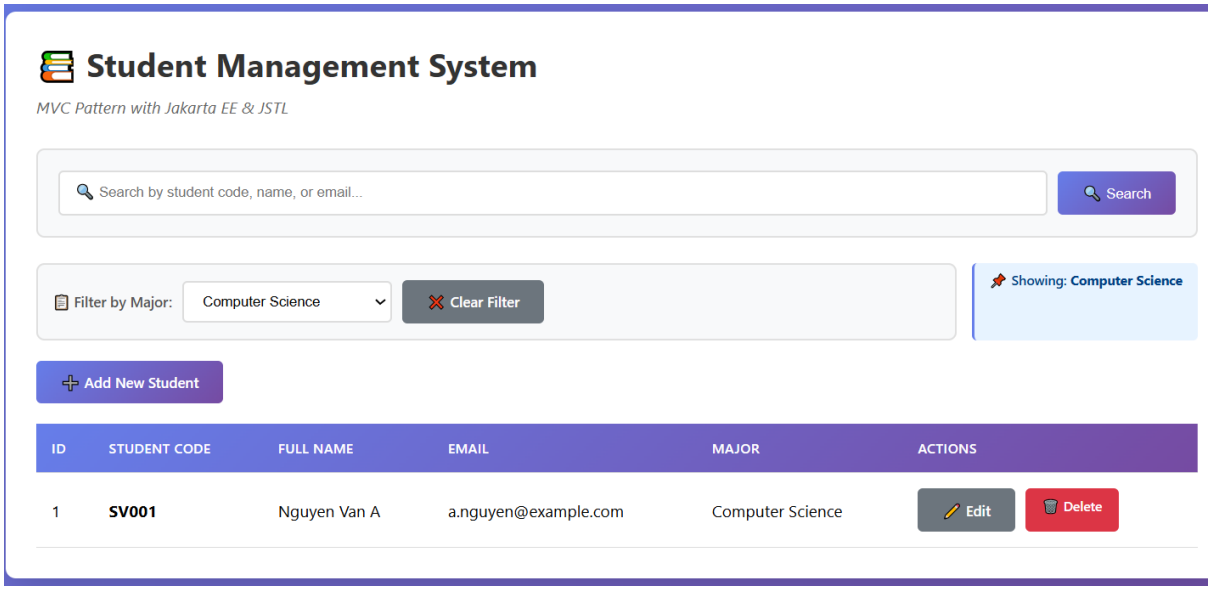
```java
public List<Student> getStudentsByMajor(String major) {
    List<Student> students = new ArrayList<>();
    String sql = "SELECT * FROM students WHERE major = ? ORDER BY id DESC";

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

        pstmt.setString(parameterIndex: 1, major);
        ResultSet rs = pstmt.executeQuery();

        while (rs.next()) {
            Student student = new Student();
            student.setId(rs.getInt(columnLabel: "id"));
            student.setStudentCode(rs.getString(columnLabel: "student_code"));
            student.setFullName(rs.getString(columnLabel: "full_name"));
            student.setEmail(rs.getString(columnLabel: "email"));
            student.setMajor(rs.getString(columnLabel: "major"));
            student.setCreatedAt(rs.getTimestamp(columnLabel: "created_at"));
            students.add(student);
        }

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

    return students;
}
```

7.2

```java
private void listStudents(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    String sortBy = request.getParameter("sortBy");
    String order = request.getParameter("order");
    String major = request.getParameter("major");

    List<Student> students;

    if ((sortBy != null || order != null) || (major != null && !major.trim().isEmpty())) {
        students = studentDAO.getStudentsFiltered(major, sortBy, order);

        if (major != null && !major.trim().isEmpty() && !"all".equalsIgnoreCase(major)) {
            request.setAttribute("major", major);
        }
        if (sortBy != null) {
            request.setAttribute("sortBy", sortBy);
        }
        if (order != null) {
            request.setAttribute("order", order);
        }
    } else {
        students = studentDAO.getAllStudents();
    }

    request.setAttribute("students", students);

    RequestDispatcher dispatcher = request.getRequestDispatcher("/views/student-list.jsp");
    dispatcher.forward(request, response);
}
```

7.3

```jsp
<div class="filter-sort-controls">
  <div class="filter-box">
    <form action="student" method="get">
      <input type="hidden" name="action" value="list" />
      <label>📄 Filter by Major:</label>
      <select name="major" onchange="this.form.submit()">
        <option value="all" ${empty major || major == 'all' ? 'selected' : ''}>All Majors</option>
        <option value="Computer Science" ${major == 'Computer Science' ? 'selected' : ''}>Computer Science</option>
        <option value="Information Technology" ${major == 'Information Technology' ? 'selected' : ''}>Information Technology</option>
        <option value="Software Engineering" ${major == 'Software Engineering' ? 'selected' : ''}>Software Engineering</option>
        <option value="Business Administration" ${major == 'Business Administration' ? 'selected' : ''}>Business Administration</option>
      </select>
      <c:if test="${not empty major && major != 'all'}">
        <a href="student?action=list" class="btn btn-clear">❌ Clear Filter</a>
      </c:if>
    </form>
  </div>

  <c:if test="${not empty major && major != 'all'}">
    <div class="filter-active">
      📌 Showing: <strong>${major}</strong>
    </div>
  </c:if>
</div>
```


Student Management System
MVC Pattern with Jakarta EE & JSTL
Search by student code, name, or email...
Search
Filter by Major:
Computer Science
Clear Filter
Showing: Computer Science
Add New Student
ID
STUDENT CODE
FULL NAME
EMAIL
MAJOR
ACTIONS
1
SV001
Nguyen Van A
a.nguyen@example.com
Computer Science
Edit
Delete