

## Web Application Development Lab 05

### SERVLET & MVC PATTERN

#### *Part B - HOMEWORK*

### I. SEARCH FUNCTIONALITY

#### 1. Search results

\*Screenshots taken after the completion of most of the exercises

The screenshot shows a web browser window titled "Student List - MVC". The URL is "localhost:8080/student-management-mvc/student?action=listCombined&keyword=john&major=&s sortBy=maior&order=desc". The page title is "Student Management System" with the subtitle "MVC Pattern with Jakarta EE & JSTL". There are two buttons: "Add New Student" and "Export to Excel". A search bar contains "john" and a dropdown menu shows "All Majors". Below the search bar are "Search" and "Clear Filters" buttons. A message "Search results for: john" is displayed. A table lists student data:

ID	CODE	NAME	EMAIL	MAJOR ▾	ACTIONS
26	JK001	Das John Doeminion	DasJohnDoes@email.com	Julustu Science	<button>Edit</button> <button>Delete</button>
22	JJ001	thisIsJohn	john@email.com	Johnning Science	<button>Edit</button> <button>Delete</button>
23	JJ002	thisIsJohnDoe	john@email.com	Johnning Science	<button>Edit</button> <button>Delete</button>
2	SV002	Emily Johnson	emily.j@email.com	Information Technology	<button>Edit</button> <button>Delete</button>
27	IT001	John IT Jr	johnit@email.com	Information Technology	<button>Edit</button> <button>Delete</button>

*Searching for 'john'*

The screenshot shows a web browser window titled "Student List - MVC". The URL is `localhost:8080/student-management-mvc/student?action=listCombined&sortBy=id&order=asc&keyword=&major=`. The page title is "Student Management System" with the subtitle "MVC Pattern with Jakarta EE & JSTL". Below the title are two buttons: "Add New Student" and "Export to Excel". There is a search bar with a placeholder "Search..." and a dropdown menu set to "All Majors". A "Search" button is next to the search bar. The main area is a table with columns: ID, CODE, NAME, EMAIL, MAJOR, and ACTIONS. The table contains six rows of student data:

ID	CODE	NAME	EMAIL	MAJOR	ACTIONS
1	SV001	John Smith	john.smith@email.com	Computer Science	<button>Edit</button> <button>Delete</button>
2	SV002	Emily Johnson	emily.j@email.com	Information Technology	<button>Edit</button> <button>Delete</button>
3	SV003	Michael Brown	michael.b@email.com	Software Engineering	<button>Edit</button> <button>Delete</button>
4	SV004	Sarah Davis	sarah.d@email.com	Data Science	<button>Edit</button> <button>Delete</button>
5	SV005	David Wilson	david.w@email.com	Computer Science	<button>Edit</button> <button>Delete</button>
16	SV006	Ivp	john@email	CompSci	<button>Edit</button> <button>Delete</button>

*Handles empty searches*

The screenshot shows a web browser window titled "Student List - MVC". The URL is `localhost:8080/student-management-mvc/student?action=listCombined&keyword=science&major=&sortBy=id&order=asc`. The page title is "Student Management System" with the subtitle "MVC Pattern with Jakarta EE & JSTL". Below the title are two buttons: "Add New Student" and "Export to Excel". There is a search bar with the value "science", a dropdown menu set to "All Majors", a "Search" button, and a "Clear Filters" button. Below the search bar, a message says "Search results for: science". In the center, there is a small icon of a person with a graduation cap. Below the icon, the text "No students found" is displayed, followed by the subtext "Start by adding a new student".

*Handles 'no students found'*

\*Issue: - 'Major' is used as a filter, as such, it cannot be used for searches.

## II. SERVER-SIDE VALIDATION

### 1. Display Validation Errors in Form

>Edit Student

Student Code \*

IT00

Format: 2 letters + 3+ digits  
Invalid format. Use 2 letters + 3+ digits (e.g., SV001)

Full Name \*

John D

Email \*

john@email

Major \*

Computer Science

Update Student Cancel

## 2. Errors appear in appropriate fields

>Edit Student

Student Code \*

IT00

Format: 2 letters + 3+ digits  
Invalid format. Use 2 letters + 3+ digits (e.g., SV001)

Full Name \*

K

Full name must be at least 2 characters

Email \*

john@em

Invalid email format

Major \*

Computer Science

Update Student Cancel

## 3. For editing

The screenshot shows a web browser window with the title "Edit Student". The URL in the address bar is "localhost:8080/student-management-mvc/student". The main content is a modal dialog titled "Edit Student".  
Fields and their values:

- Student Code \*: IT003 (read-only)
- Full Name \*: M (Error: Full name must be at least 2 characters)
- Email \*: macit@e (Error: Invalid email format)
- Major \*: Information Technology

Buttons at the bottom:

- Update Student
- Cancel

*Student code set as 'read-only'*

### III. SORTING & FILTERING

#### 1. Filtering

The screenshot shows a web browser window titled "Student List - MVC". The URL is "localhost:8080/student-management-mvc/student?action=listCombined&keyword=Computer+Science&sortBy=&order=". The page title is "Student Management System" with the subtitle "MVC Pattern with Jakarta EE & JSTL". It features two buttons: "Add New Student" and "Export to Excel". Below them are search fields for "Search..." and "MAJOR" (set to "Computer Science"), and buttons for "Search" and "Clear Filters". A table lists student data with columns: ID, CODE, NAME, EMAIL, MAJOR, and ACTIONS (Edit and Delete). Two rows are visible: one for John Smith (SV001) and one for David Wilson (SV005).

ID	CODE	NAME	EMAIL	MAJOR	ACTIONS
1	SV001	John Smith	john.smith@email.com	Computer Science	<button>Edit</button> <button>Delete</button>
5	SV005	David Wilson	david.w@email.com	Computer Science	<button>Edit</button> <button>Delete</button>

## 2. Sorting

The screenshot shows a web browser window titled "Student List - MVC". The URL is "localhost:8080/student-management-mvc/student?action=listCombined". The page title is "Student Management System" with the subtitle "MVC Pattern with Jakarta EE & JSTL". It features two buttons: "Add New Student" and "Export to Excel". Below them are search fields for "Search..." and "MAJOR" (set to "All Majors"), and a "Search" button. A table lists student data with columns: ID, CODE, NAME, EMAIL, MAJOR, and ACTIONS (Edit and Delete). Seven rows are visible, showing data for multiple students across different majors.

ID	CODE	NAME	EMAIL	MAJOR	ACTIONS
1	SV001	John Smith	john.smith@email.com	Computer Science	<button>Edit</button> <button>Delete</button>
2	SV002	Emily Johnson	emily.j@email.com	Information Technology	<button>Edit</button> <button>Delete</button>
3	SV003	Michael Brown	michael.b@email.com	Software Engineering	<button>Edit</button> <button>Delete</button>
4	SV004	Sarah Davis	sarah.d@email.com	Data Science	<button>Edit</button> <button>Delete</button>
5	SV005	David Wilson	david.w@email.com	Computer Science	<button>Edit</button> <button>Delete</button>
16	SV006	lvp	john@email	CompSci	<button>Edit</button> <button>Delete</button>
17	SV007	lvp	john@email	CompSci	<button>Edit</button> <button>Delete</button>

*Sort by Id in Ascending order*

The screenshot shows a web application titled "Student Management System" running on a local host. The interface includes a search bar with dropdown filters for "Search..." and "All Majors", and a "Search" button. Below the search bar is a table with columns: ID, CODE, NAME, EMAIL, MAJOR, and ACTIONS. The table contains six student records. Each record has "Edit" and "Delete" buttons in the ACTIONS column. The table header has a "Sort By" dropdown set to "ID".

ID	CODE	NAME	EMAIL	MAJOR	ACTIONS
34	IT003	Mac IT	macit@email.com	Information Technology	<button>Edit</button> <button>Delete</button>
32	IT002	John D	johnD@email.com	Information Technology	<button>Edit</button> <button>Delete</button>
27	IT001	John IT Jr	johnIT@email.com	Information Technology	<button>Edit</button> <button>Delete</button>
26	JK001	Das John Doeminion	DasJohnDoes@email.com	JuJustu Science	<button>Edit</button> <button>Delete</button>
23	JJ002	thisIsJohnDoe	john@email.com	Johnning Science	<button>Edit</button> <button>Delete</button>
22	JJ001	thisIsJohn	john@email.com	Johnning Science	<button>Edit</button> <button>Delete</button>

*Sort by Id in Descending order*

\*The current filter&sort is the same implementation as that of the bonus exercise: Search&Filter&Sort

Code submission:

StudentDAO.java

```
public List<Student> getStudentsCombined(String keyword, String major, String sortBy, String order) {
    List<Student> students = new ArrayList<>();
    StringBuilder sql = new StringBuilder("SELECT * FROM students WHERE 1=1");
    if (keyword != null && !keyword.trim().isEmpty()) {
        sql.append(" AND (student_code LIKE ? OR full_name LIKE ? OR email LIKE ?)");
    }
}
```

```
if (major != null && !major.trim().isEmpty()) {  
    sql.append(" AND major = ?");  
}  
  
sql.append(" ORDER BY ").append(validateSortBy(sortBy)).append(")  
").append(validateOrder(order));  
  
try (Connection conn = getConnection();  
    PreparedStatement pstmt = conn.prepareStatement(sql.toString())) {  
  
    int index = 1;  
    if (keyword != null && !keyword.trim().isEmpty()) {  
        String pattern = "%" + keyword.trim() + "%";  
        pstmt.setString(index++, pattern);  
        pstmt.setString(index++, pattern);  
        pstmt.setString(index++, pattern);  
    }  
  
    if (major != null && !major.trim().isEmpty()) {  
        pstmt.setString(index++, major);  
    }  
  
    ResultSet rs = pstmt.executeQuery();  
    while (rs.next()) {  
        Student student = new Student();  
    }  
}
```

```
        student.setId(rs.getInt("id"));

        student.setStudentCode(rs.getString("student_code"));

        student.setFullName(rs.getString("full_name"));

        student.setEmail(rs.getString("email"));

        student.setMajor(rs.getString("major"));

        student.setCreatedAt(rs.getTimestamp("created_at"));

        students.add(student);

    }

}

} catch (SQLException e) {

    e.printStackTrace();

}

return students;

}
```

### StudentController.java

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

String keyword = request.getParameter("keyword");
String major = request.getParameter("major");
String sortBy = request.getParameter("sortBy");
```

```
String order = request.getParameter("order");

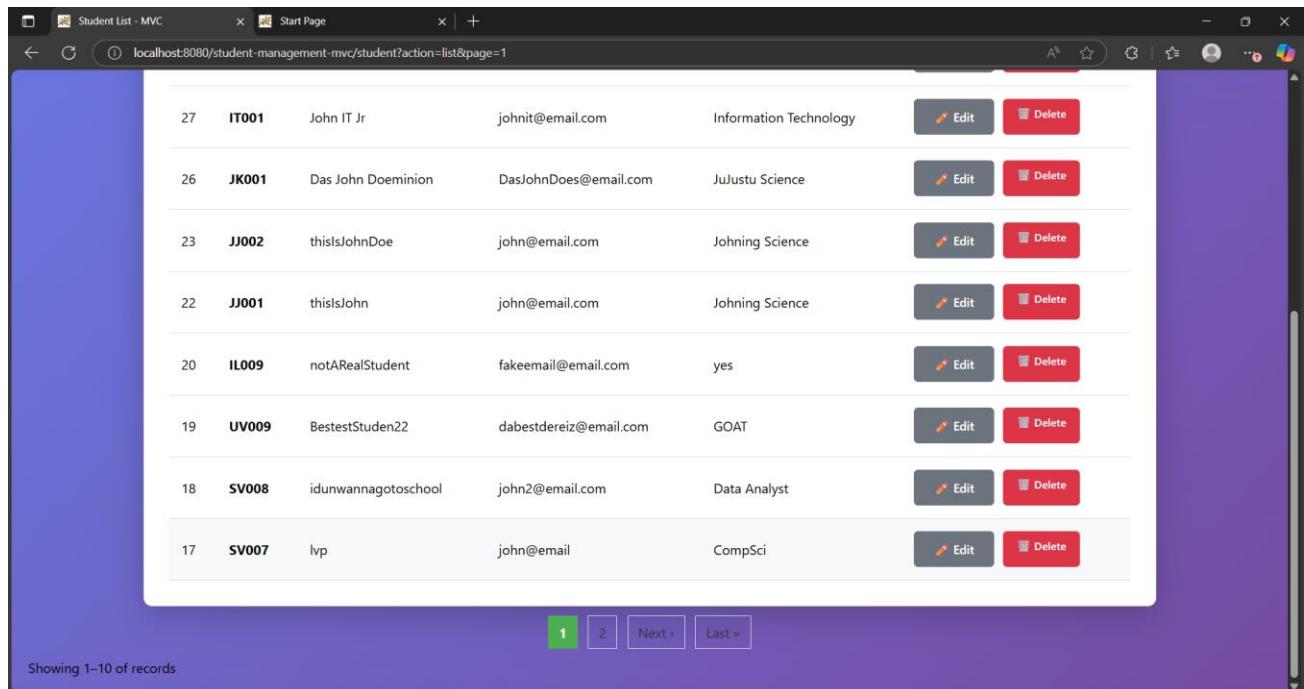
List<Student> students = studentDAO.getStudentsCombined(keyword,
major, sortBy, order);

request.setAttribute("students", students);
request.setAttribute("keyword", keyword != null ? keyword : "");
request.setAttribute("selectedMajor", major != null ? major : "");
request.setAttribute("sortBy", sortBy != null ? sortBy : "id");
request.setAttribute("order", order != null ? order : "asc");

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

}
```

## IV. PAGINATION

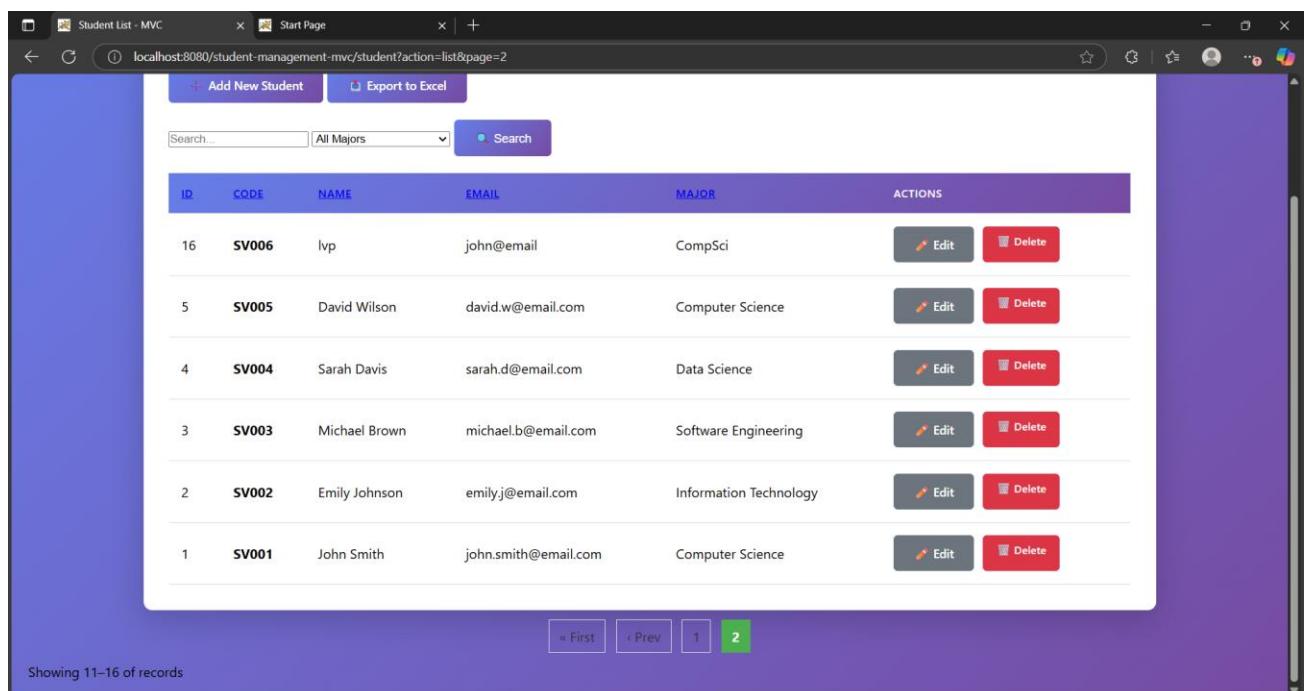


The screenshot shows a web browser window titled "Student List - MVC". The URL is "localhost:8080/student-management-mvc/student?action=list&page=1". The page displays a table of student records with columns: ID, CODE, NAME, EMAIL, MAJOR, and ACTIONS. The data includes:

ID	CODE	NAME	EMAIL	MAJOR	ACTIONS
27	<b>IT001</b>	John IT Jr	johnit@email.com	Information Technology	<button>Edit</button> <button>Delete</button>
26	<b>JK001</b>	Das John Doeminion	DasJohnDoes@email.com	JuJustu Science	<button>Edit</button> <button>Delete</button>
23	<b>JJ002</b>	thisIsJohnDoe	john@email.com	Johning Science	<button>Edit</button> <button>Delete</button>
22	<b>JJ001</b>	thisIsJohn	john@email.com	Johning Science	<button>Edit</button> <button>Delete</button>
20	<b>IL009</b>	notARealStudent	fakeemail@email.com	yes	<button>Edit</button> <button>Delete</button>
19	<b>UV009</b>	BestestStuden22	dabestdereiz@email.com	GOAT	<button>Edit</button> <button>Delete</button>
18	<b>SV008</b>	idunwannagotoschool	john2@email.com	Data Analyst	<button>Edit</button> <button>Delete</button>
17	<b>SV007</b>	lvp	john@email	CompSci	<button>Edit</button> <button>Delete</button>

At the bottom, there is a navigation bar with buttons for "1" (highlighted in green), "2", "Next >", and "Last »". Below the table, a message says "Showing 1–10 of records".

*First page*



The screenshot shows a web browser window titled "Student List - MVC". The URL is "localhost:8080/student-management-mvc/student?action=list&page=2". The page displays a table of student records with columns: ID, CODE, NAME, EMAIL, MAJOR, and ACTIONS. The data includes:

ID	CODE	NAME	EMAIL	MAJOR	ACTIONS
16	<b>SV006</b>	lvp	john@email	CompSci	<button>Edit</button> <button>Delete</button>
5	<b>SV005</b>	David Wilson	david.w@email.com	Computer Science	<button>Edit</button> <button>Delete</button>
4	<b>SV004</b>	Sarah Davis	sarah.d@email.com	Data Science	<button>Edit</button> <button>Delete</button>
3	<b>SV003</b>	Michael Brown	michael.b@email.com	Software Engineering	<button>Edit</button> <button>Delete</button>
2	<b>SV002</b>	Emily Johnson	emily.j@email.com	Information Technology	<button>Edit</button> <button>Delete</button>
1	<b>SV001</b>	John Smith	john.smith@email.com	Computer Science	<button>Edit</button> <button>Delete</button>

At the bottom, there is a navigation bar with buttons for "< First", "< Prev", "1", "2" (highlighted in green), and "Next >". Below the table, a message says "Showing 11–16 of records".

*Second page*

## Code Implementation:

```
<c:if test="${totalPages > 1}">
    <div class="pagination">
        <c:if test="${currentPage > 1}">
            <a href="student?action=list&page=1"><< First</a>
        </c:if>

        <c:if test="${currentPage > 1}">
            <a href="student?action=list&page=${currentPage - 1}">< Prev</a>
        </c:if>

        <c:set var="startPage" value="${currentPage - 2}" />
        <c:set var="endPage" value="${currentPage + 2}" />
        <c:if test="${startPage < 1}"><c:set var="startPage" value="1" /></c:if>
        <c:if test="${endPage > totalPages}"><c:set var="endPage" value="${totalPages}" /></c:if>

        <c:forEach begin="${startPage}" end="${endPage}" var="i">
            <c:choose>
                <c:when test="${i == currentPage}">
                    <strong>${i}</strong>
                </c:when>
                <c:otherwise>
                    <a href="student?action=list&page=${i}">${i}</a>
                </c:otherwise>
            </c:choose>
        
```

```
</c:forEach>

<c:if test="${currentPage < totalPages}">
    <a href="student?action=list&page=${currentPage + 1}">Next </a>
</c:if>

<c:if test="${currentPage < totalPages}">
    <a href="student?action=list&page=${totalPages}">Last </a>
</c:if>

</div>

<c:set var="recordsPerPage" value="10" />
<c:set var="startRecord" value="${(currentPage - 1) * recordsPerPage +
1}" />
<c:set var="endRecord" value="${startRecord + students.size() - 1}" />
<p>Showing ${startRecord}–${endRecord} of ${totalRecords}
records</p>
</c:if>
```

\*The pagination logic was taken from Lab04

## V. BONUS EXERCISES

### 1. Export to Excel

The screenshot shows a web application titled "Student Management System" running on a local host. The interface includes a search bar, a dropdown menu for majors, and a "Search" button. Below is a table of student data with columns: ID, CODE, NAME, EMAIL, MAJOR, and ACTIONS (Edit and Delete buttons). At the top right of the table is a blue button labeled "Export to Excel".

ID	CODE	NAME	EMAIL	MAJOR	ACTIONS
34	IT003	Mac IT	macit@email.com	Information Technology	<button>Edit</button> <button>Delete</button>
32	IT002	John D	johnd@email.com	Information Technology	<button>Edit</button> <button>Delete</button>
27	IT001	John IT Jr	johnit@email.com	Information Technology	<button>Edit</button> <button>Delete</button>
26	JK001	Das John Doeminion	DasJohnDoes@email.com	JuJustu Science	<button>Edit</button> <button>Delete</button>
23	JJ002	thisIsJohnDoe	john@email.com	Johnning Science	<button>Edit</button> <button>Delete</button>
22	JJ001	thisIsJohn	john@email.com	Johnning Science	<button>Edit</button> <button>Delete</button>

'Export to Excel' button

The screenshot shows an Excel spreadsheet titled "students.xlsx" with data corresponding to the table above. The columns are labeled A through Q, and the rows are numbered 1 through 26. The data includes student IDs, names, emails, majors, and creation dates.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	ID	Student Code	Full Name	Email	Major	Created At										
2	34	IT003	Mac IT	macit@email.com	Information Technology	2025-11-21 20:16:16.0										
3	32	IT002	John D	johnd@email.com	Information Technology	2025-11-21 20:13:44.0										
4	27	IT001	John IT Jr	johnit@email.com	Information Technology	2025-11-21 18:41:43.0										
5	26	JK001	Das John Doeminion	DasJohnDoes@email.com	JuJustu Science	2025-11-14 16:03:46.0										
6	23	JJ002	thisIsJohnDoe	john@email.com	Johnning Science	2025-11-14 15:44:14.0										
7	22	JJ001	thisIsJohn	john@email.com	Johnning Science	2025-11-14 15:42:35.0										
8	20	IL009	notRealStudent	fakemail@email.com	yes	2025-11-13 16:07:00.0										
9	19	UV009	BestestStuden22	dabestdereiz@email.com	GOAT	2025-11-13 16:06:24.0										
10	18	SV008	idumwannagotschool	john2@email.com	Data Analyst	2025-11-13 16:05:46.0										
11	17	SV007	lvp	john@email	CompSci	2025-11-13 15:25:17.0										
12	16	SV006	lvp	john@email	CompSci	2025-11-13 15:23:51.0										
13	5	SV005	David Wilson	david.w@email.com	Computer Science	2025-11-08 09:23:13.0										
14	4	SV004	Sarah Davis	sarah.d@email.com	Data Science	2025-11-08 09:23:13.0										
15	3	SV003	Michael Brown	michael.b@email.com	Software Engineering	2025-11-08 09:23:13.0										
16	2	SV002	Emily Johnson	emily.j@email.com	Information Technology	2025-11-08 09:23:13.0										
17	1	SV001	John Smith	john.smith@email.com	Computer Science	2025-11-08 09:23:13.0										
18																
19																
20																
21																
22																
23																
24																
25																
26																

Excel view

**Dependency:**

```
Pom.xml

<dependency>

    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>5.2.3</version>
</dependency>
```

**Implementation:**

```
package com.student.controller;

import com.student.dao.StudentDAO;
import com.student.model.Student;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import org.apache.poi.ss.usermodel.*;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import java.io.IOException;
import java.util.List;
```

```
@WebServlet("/export")
public class excelExport extends HttpServlet {

    private StudentDAO studentDAO;

    @Override
    public void init() {
        studentDAO = new StudentDAO();
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {

        // students
        List<Student> students = studentDAO.getAllStudents();

        try (
            Workbook workbook = new XSSFWorkbook() {
                Sheet sheet = workbook.createSheet("Students");

                Row header = sheet.createRow(0);
                String[] columns = {"ID", "Student Code", "Full Name", "Email",
"Major", "Created At"};
                for (int i = 0; i < columns.length; i++) {
```

```
Cell cell = header.createCell(i);
cell.setCellValue(columns[i]);
}

int rowNum = 1;
for (Student s : students) {
    Row row = sheet.createRow(rowNum++);
    row.createCell(0).setCellValue(s.getId());
    row.createCell(1).setCellValue(s.getStudentCode());
    row.createCell(2).setCellValue(s.getFullName());
    row.createCell(3).setCellValue(s.getEmail());
    row.createCell(4).setCellValue(s.getMajor());
    row.createCell(5).setCellValue(s.getCreatedAt().toString());
}

for (int i = 0; i < columns.length; i++) {
    sheet.autoSizeColumn(i);
}

response.setContentType("application/vnd.ms-excel");
response.setHeader("Content-Disposition", "attachment;
filename=students.xlsx");

workbook.write(response.getOutputStream());
}
```

{}

## 2. Search&Filter&Sort

The screenshot shows a web browser window titled "Student List - MVC". The URL is "localhost:8080/student-management-mvc/student?action=listCombined&keyword=john&major=&sortBy=&order=". The page title is "Student Management System" with the subtitle "MVC Pattern with Jakarta EE & JSTL". At the top, there are two buttons: "Add New Student" and "Export to Excel". Below them is a search bar with the input "john", a dropdown menu set to "All Majors", a "Search" button, and a "Clear Filters" button. The main content area displays a table with student data. The table has columns: ID, CODE, NAME, EMAIL, MAJOR, and ACTIONS. The data rows are:

ID	CODE	NAME	EMAIL	MAJOR	ACTIONS
1	<b>SV001</b>	John Smith	john.smith@email.com	Computer Science	<button>Edit</button> <button>Delete</button>
2	<b>SV002</b>	Emily Johnson	emily.j@email.com	Information Technology	<button>Edit</button> <button>Delete</button>
16	<b>SV006</b>	lvp	john@email	CompSci	<button>Edit</button> <button>Delete</button>
17	<b>SV007</b>	lvp	john@email	CompSci	<button>Edit</button> <button>Delete</button>
18	<b>SV008</b>	idunwannagotoschool	john2@email.com	Data Analyst	<button>Edit</button> <button>Delete</button>

*Search results for 'John'*

The screenshot shows a web browser window titled "Student List - MVC". The URL is "localhost:8080/student-management-mvc/student?action=listCombined&keyword=john&major=Information+Technology&sortBy=&order=". The page title is "Student Management System" with the subtitle "MVC Pattern with Jakarta EE & JSTL". At the top, there are two buttons: "Add New Student" and "Export to Excel". Below them is a search bar with the input "john", a dropdown menu set to "Information Technology", a "Search" button, and a "Clear Filters" button. The main content area displays a table with student data. The table has columns: ID, CODE, NAME, EMAIL, MAJOR, and ACTIONS. The data rows are:

ID	CODE	NAME	EMAIL	MAJOR	ACTIONS
2	<b>SV002</b>	Emily Johnson	emily.j@email.com	Information Technology	<button>Edit</button> <button>Delete</button>
27	<b>IT001</b>	John IT Jr	johnit@email.com	Information Technology	<button>Edit</button> <button>Delete</button>
32	<b>IT002</b>	John D	johnd@email.com	Information Technology	<button>Edit</button> <button>Delete</button>

*Search results for 'john' & filtered by 'Information Technology'*

The screenshot shows a web application titled "Student Management System" running on localhost:8080. The page displays a table of student records. A search bar at the top has "john" in the input field and "Information Technology" selected in the dropdown. Below the search bar is a button labeled "Search". The table has columns: ID, CODE, NAME, EMAIL, MAJOR, and ACTIONS. The data in the table is:

ID	CODE	NAME	EMAIL	MAJOR	ACTIONS
32	IT002	John D	johnd@email.com	Information Technology	<button>Edit</button> <button>Delete</button>
27	IT001	John IT Jr	johnit@email.com	Information Technology	<button>Edit</button> <button>Delete</button>
2	SV002	Emily Johnson	emilyj@email.com	Information Technology	<button>Edit</button> <button>Delete</button>

*Search results for 'john' & filtered by 'Information Technology' & in descending order'*

\*A similar problem was solved in Lab04, which I took as reference

### Implementation

#### StudentController.java

```
private void listCombined(HttpServletRequest request, HttpServletResponse response)
```

```
throws ServletException, IOException {
```

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

```
    String major = request.getParameter("major");
```

```
    String sortBy = request.getParameter("sortBy");
```

```
    String order = request.getParameter("order");
```

```
List<Student> students = studentDAO.getStudentsCombined(keyword,  
major, sortBy, order);
```

```
request.setAttribute("students", students);  
request.setAttribute("keyword", keyword != null ? keyword : "");  
request.setAttribute("selectedMajor", major != null ? major : "");  
request.setAttribute("sortBy", sortBy != null ? sortBy : "id");  
request.setAttribute("order", order != null ? order : "asc");
```

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

### StudentDao.java

```
public List<Student> getStudentsCombined(String keyword, String major,  
String sortBy, String order) {  
  
    List<Student> students = new ArrayList<>();  
  
    StringBuilder sql = new StringBuilder("SELECT * FROM students WHERE  
1=1");  
  
    if (keyword != null && !keyword.trim().isEmpty()) {  
        sql.append(" AND (student_code LIKE ? OR full_name LIKE ? OR email  
LIKE ?)");  
    }  
  
    if (major != null && !major.trim().isEmpty()) {  
        sql.append(" AND major = ?");  
    }  
  
    try {  
        Connection conn = DatabaseUtil.getConnection();  
        PreparedStatement pstmt = conn.prepareStatement(sql.toString());  
        pstmt.setString(1, "%" + keyword + "%");  
        pstmt.setString(2, "%" + keyword + "%");  
        pstmt.setString(3, "%" + keyword + "%");  
        if (major != null) {  
            pstmt.setString(4, major);  
        }  
        ResultSet rs = pstmt.executeQuery();  
        while (rs.next()) {  
            Student student = new Student();  
            student.setId(rs.getInt("id"));  
            student.setStudentCode(rs.getString("student_code"));  
            student.setFullName(rs.getString("full_name"));  
            student.setEmail(rs.getString("email"));  
            student.setMajor(rs.getString("major"));  
            student.setSortBy(rs.getString("sort_by"));  
            student.setOrder(rs.getString("order"));  
            students.add(student);  
        }  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
}
```

```
sql.append(" AND major = ?");  
}  
  
sql.append(" ORDER BY ").append(validateSortBy(sortBy)).append(  
").append(validateOrder(order));  
  
try (Connection conn = getConnection();  
PreparedStatement pstmt = conn.prepareStatement(sql.toString())) {  
  
    int index = 1;  
    if (keyword != null && !keyword.trim().isEmpty()) {  
        String pattern = "%" + keyword.trim() + "%";  
        pstmt.setString(index++, pattern);  
        pstmt.setString(index++, pattern);  
        pstmt.setString(index++, pattern);  
    }  
  
    if (major != null && !major.trim().isEmpty()) {  
        pstmt.setString(index++, major);  
    }  
  
    ResultSet rs = pstmt.executeQuery();  
    while (rs.next()) {  
        Student student = new Student();  
        student.setId(rs.getInt("id"));  
        student.setStudentCode(rs.getString("student_code"));  
    }  
}
```

```
        student.setFullName(rs.getString("full_name"));

        student.setEmail(rs.getString("email"));

        student.setMajor(rs.getString("major"));

        student.setCreatedAt(rs.getTimestamp("created_at"));

        students.add(student);

    }

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

return students;
}
```

## References:

1. <https://poi.apache.org/apidocs/dev/org/apache/poi/xssf/usermodel/XSSFWorkbook.html>
2. <https://stackoverflow.com/questions/5985318/apache-poi-xssf-reading-in-excel-files>
3. <https://viblo.asia/p/huong-dan-doc-va-ghi-file-excel-trong-java-su-dung-thu-vien-apache-poi-RQqKLENpZ7z>
4. <https://www.baeldung.com/java-dao-pattern>
5. <https://jakarta.ee/specifications/tags/>
6. <https://docs.oracle.com/javaee/7/tutorial/servlets.htm>