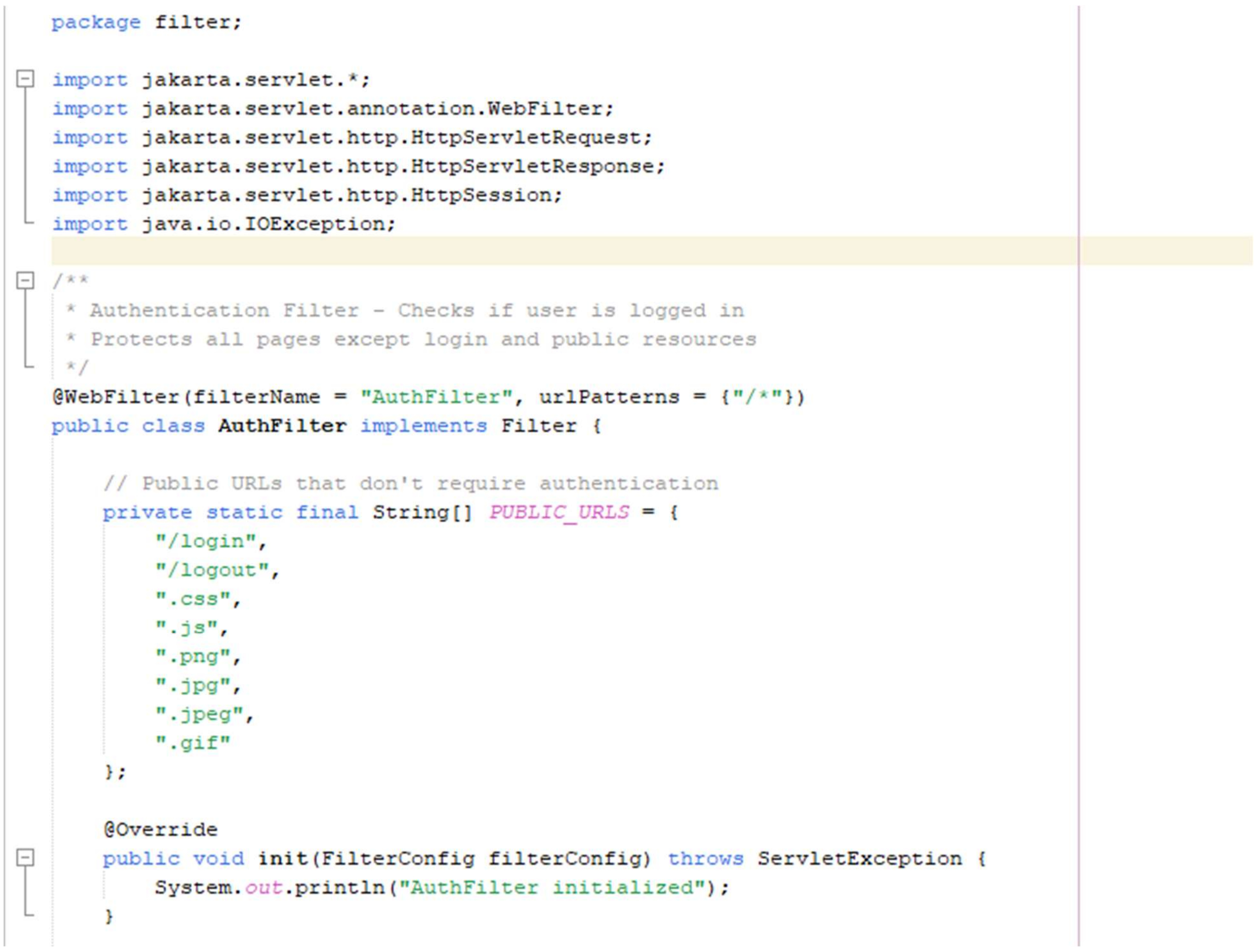
Lab 6 – Homework report

Trần Quốc Nam – ITITIU21250

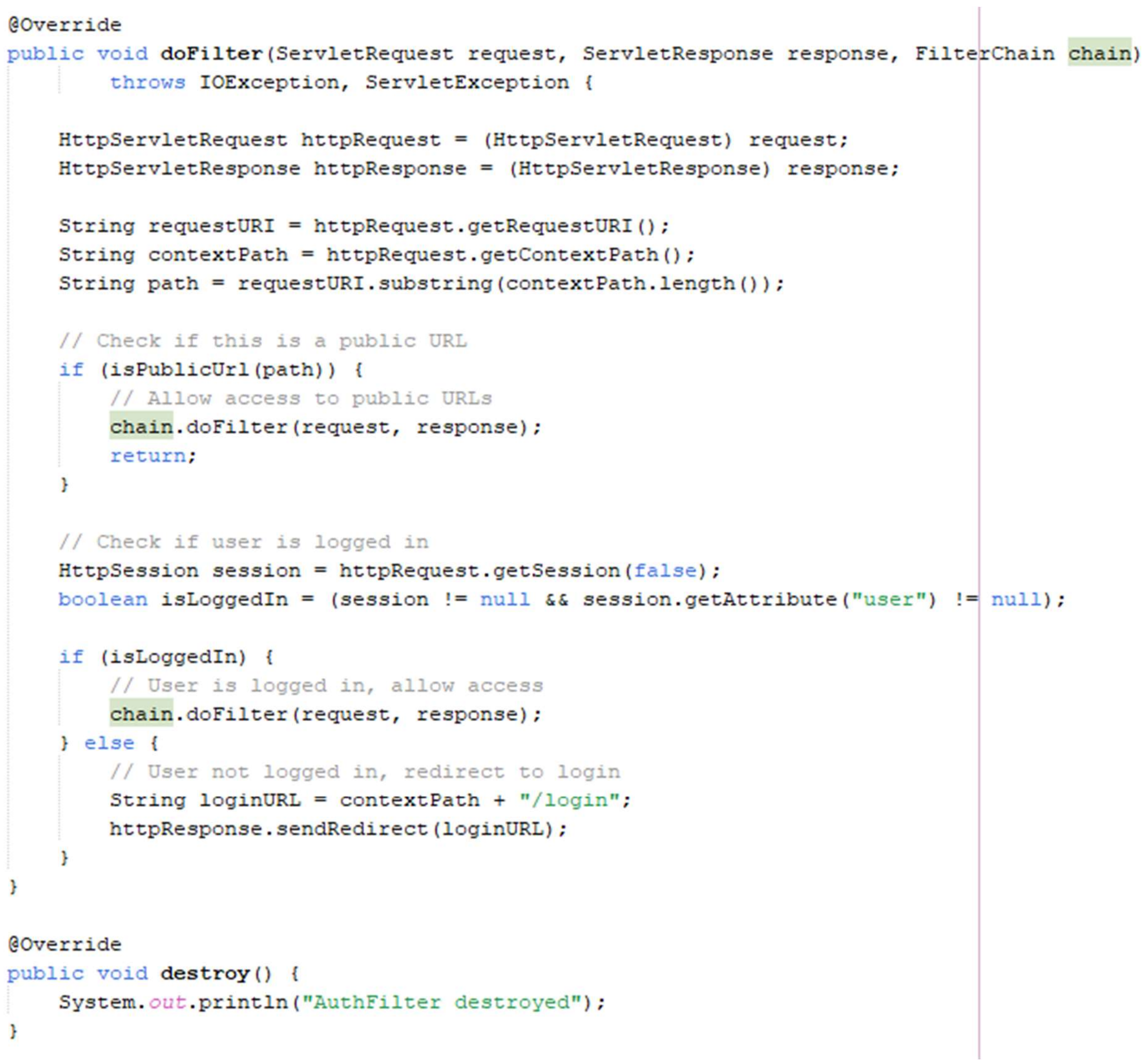
Table of content

5. Authentication filter

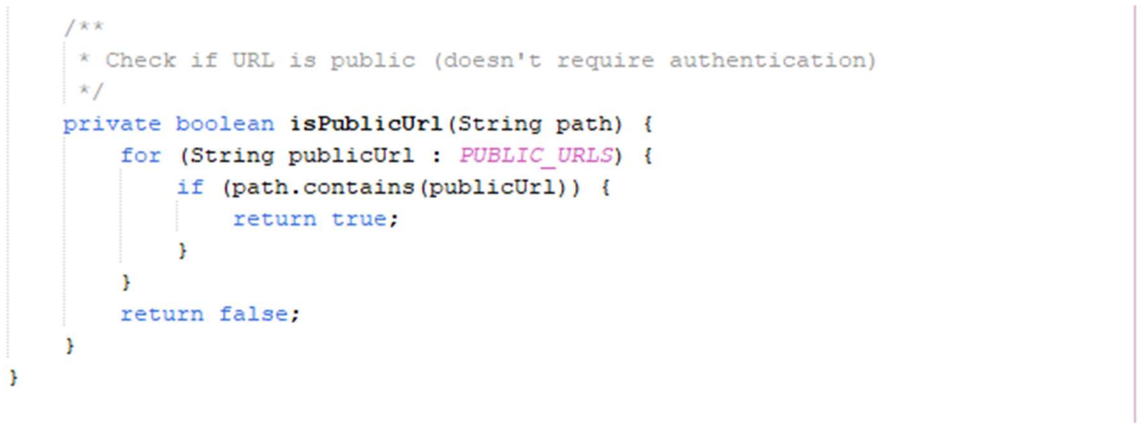
Code:



```java
package filter;

import jakarta.servlet.*;
import jakarta.servlet.annotation.WebFilter;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import java.io.IOException;

/**
 * Authentication Filter - Checks if user is logged in
 * Protects all pages except login and public resources
 */
@WebFilter(filterName = "AuthFilter", urlPatterns = {"/*"})
public class AuthFilter implements Filter {

    // Public URLs that don't require authentication
    private static final String[] PUBLIC_URLS = {
        "/login",
        "/logout",
        ".css",
        ".js",
        ".png",
        ".jpg",
        ".jpeg",
        ".gif"
    };

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        System.out.println("AuthFilter initialized");
    }
```



```java
    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
            throws IOException, ServletException {

        HttpServletRequest httpRequest = (HttpServletRequest) request;
        HttpServletResponse httpResponse = (HttpServletResponse) response;

        String requestURI = httpRequest.getRequestURI();
        String contextPath = httpRequest.getContextPath();
        String path = requestURI.substring(contextPath.length());

        // Check if this is a public URL
        if (isPublicUrl(path)) {
            // Allow access to public URLs
            chain.doFilter(request, response);
            return;
        }

        // Check if user is logged in
        HttpSession session = httpRequest.getSession(false);
        boolean isLoggedIn = (session != null && session.getAttribute("user") != null);

        if (isLoggedIn) {
            // User is logged in, allow access
            chain.doFilter(request, response);
        } else {
            // User not logged in, redirect to login
            String loginURL = contextPath + "/login";
            httpResponse.sendRedirect(loginURL);
        }
    }

    @Override
    public void destroy() {
        System.out.println("AuthFilter destroyed");
    }
```

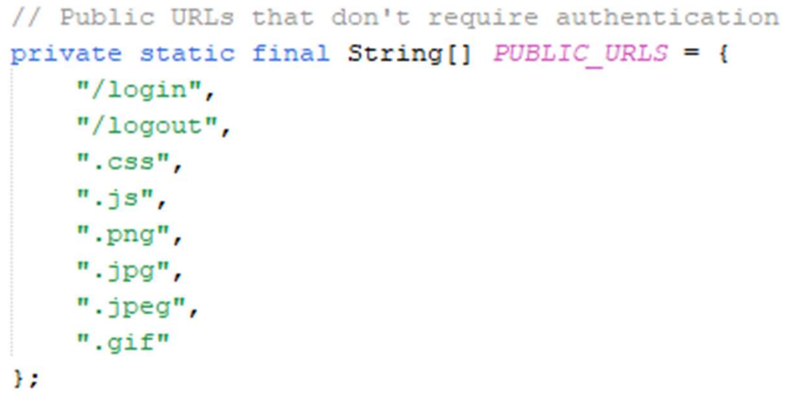


```java
    /**
     * Check if URL is public (doesn't require authentication)
     */
    private boolean isPublicUrl(String path) {
        for (String publicUrl : PUBLIC_URLS) {
            if (path.contains(publicUrl)) {
                return true;
            }
        }
        return false;
    }
}
```

Explanation:

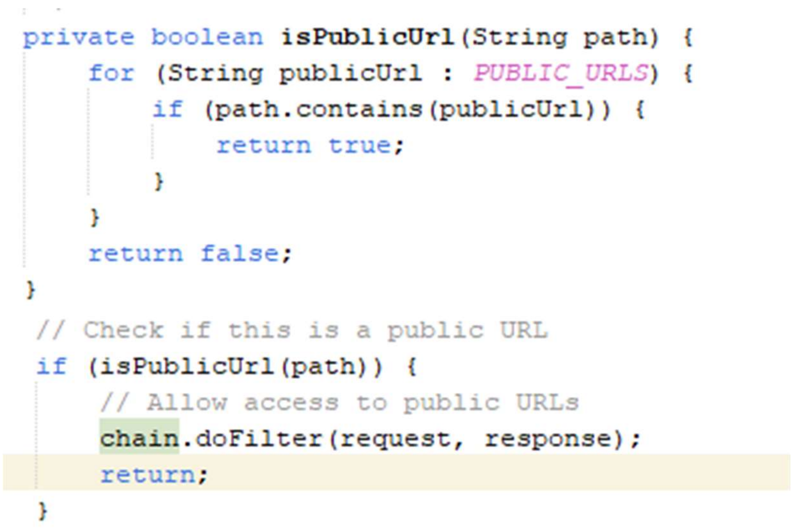
Before user can interact with any functionality of the website, authentication must be taken to define user’s role in the website.

At first, we will define public URLs which user still can access and interact without authentication steps.



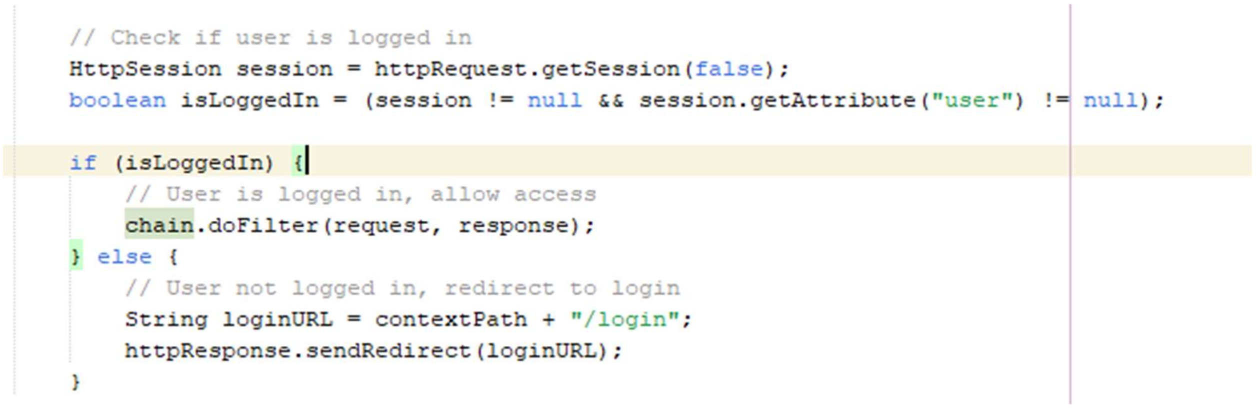
```
// Public URLs that don't require authentication
private static final String[] PUBLIC_URLS = {
    "/login",
    "/logout",
    ".css",
    ".js",
    ".png",
    ".jpg",
    ".jpeg",
    ".gif"
};
```

In dofilter function, we will create a isPublicURL(path) with return type Boolean to check action performed is whether it is public or not.
If isPublicURl return True -> will continue
Else, doFilter() function will terminated



```
private boolean isPublicUrl(String path) {
    for (String publicUrl : PUBLIC_URLS) {
        if (path.contains(publicUrl)) {
            return true;
        }
    }
    return false;
}
 // Check if this is a public URL
 if (isPublicUrl(path)) {
     // Allow access to public URLs
     chain.doFilter(request, response);
     return;
 }
```

once we define if the URL path is not public path, we will call HTTPs session to check the login status of the user. If user performs action in private path while not logged in, user will be redirect to login page, else , the action performed by user will be executed normally.

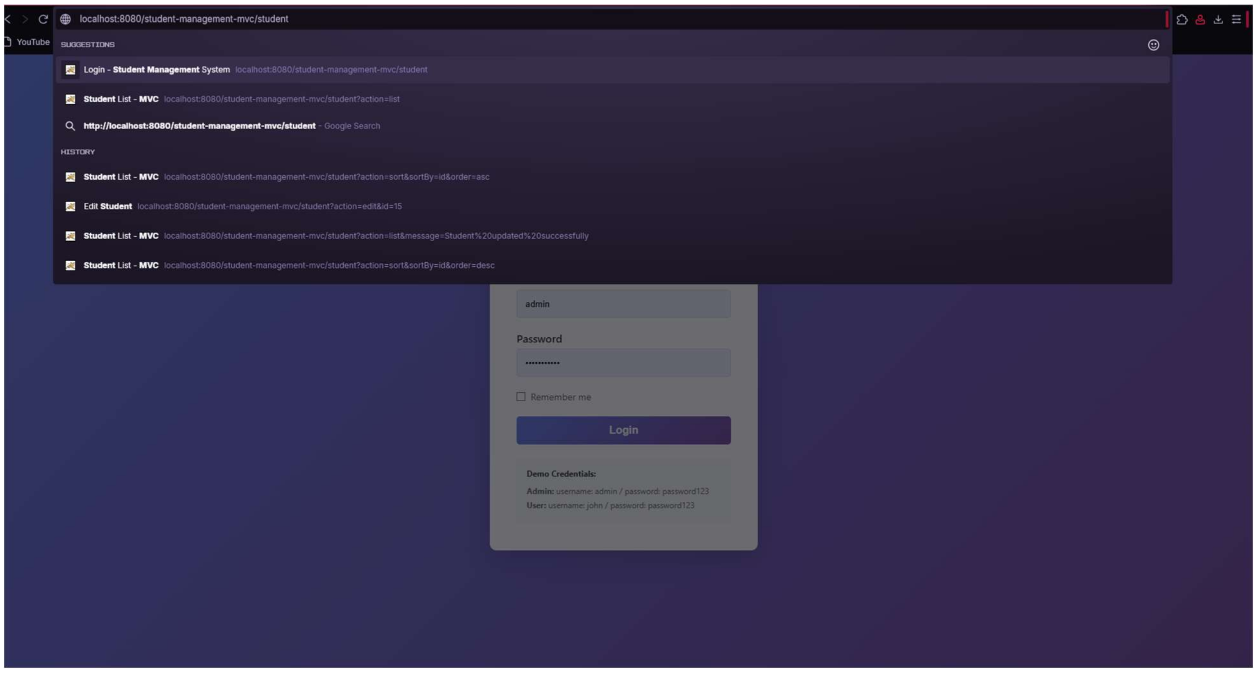


```java
// Check if user is logged in
HttpSession session = httpRequest.getSession(false);
boolean isLoggedIn = (session != null && session.getAttribute("user") != null);

if (isLoggedIn) {
    // User is logged in, allow access
    chain.doFilter(request, response);
} else {
    // User not logged in, redirect to login
    String loginURL = contextPath + "/login";
    httpResponse.sendRedirect(loginURL);
}
```

<u>Conclusion</u>:

Case user not logged in :

User performed action URL -> isPublicURL() return true -> dofilter() get browser session -> user is not logged in -> redirect user to login page.



localhost:8080/student-management-mvc/student

SUGGESTIONS

Login - **Student Management** System localhost:8080/student-management-mvc/student

Student List - **MVC** localhost:8080/student-management-mvc/student?action=list

http://localhost:8080/student-management-mvc/student - Google Search

HISTORY

Student List - **MVC** localhost:8080/student-management-mvc/student?action=sort&sortBy=id&order=asc

Edit **Student** localhost:8080/student-management-mvc/student?action=edit&id=15

Student List - **MVC** localhost:8080/student-management-mvc/student?action=list&message=Student%20updated%20successfully

Student List - **MVC** localhost:8080/student-management-mvc/student?action=sort&sortBy=id&order=desc

admin

Password

•••••••••••

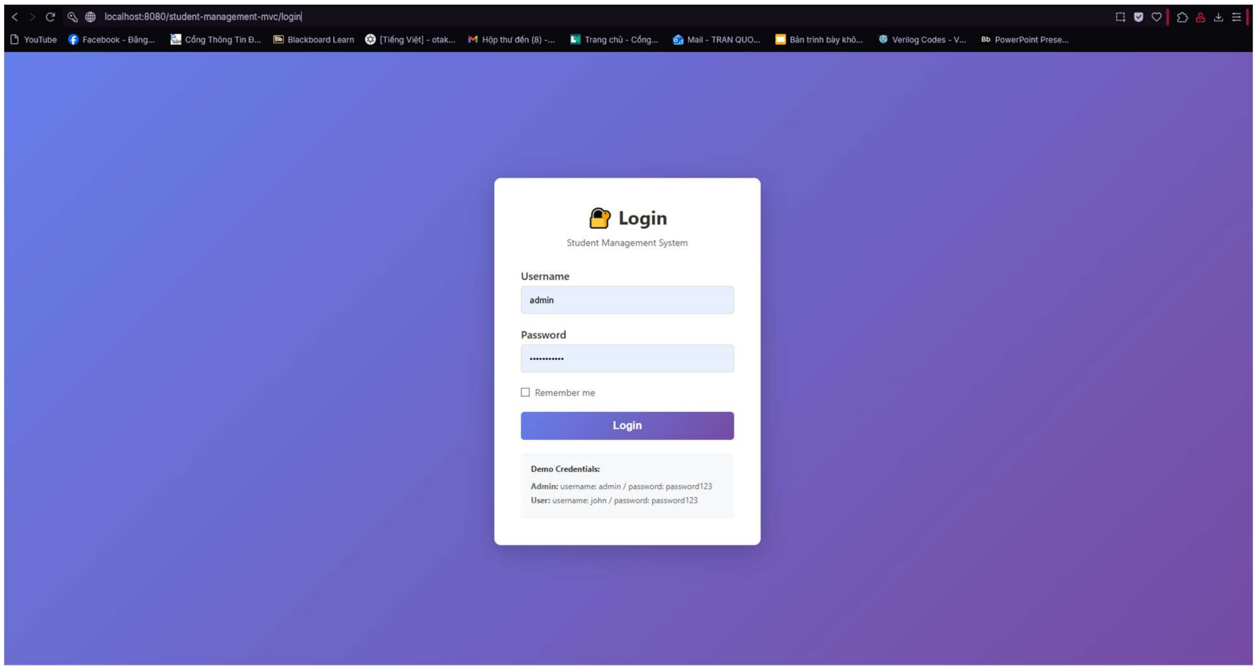
☐ Remember me

Login

Demo Credentials:

Admin: username: admin / password: password123

User: username: john / password: password123


localhost:8080/student-management-mvc/login

Login

Student Management System

Username: admin

Password: ••••••••••••

☐ Remember me

Login

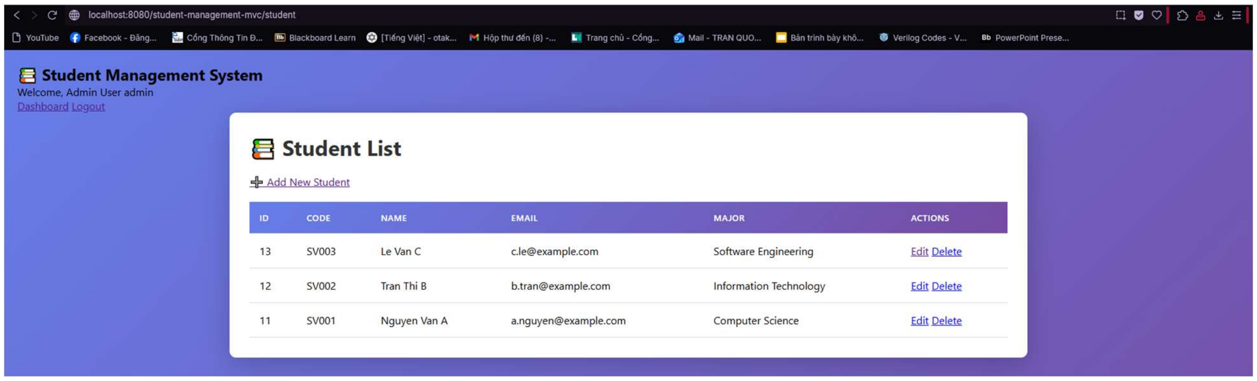
Demo Credentials:

Admin: username: admin / password: password123

User: username: john / password: password123

Case user is logged in :

User performed action in URL -> isPublicURL() return true -> dofilter() get browser session - > user is logged in -> execute action.


localhost:8080/student-management-mvc/student

Student Management System

Welcome, Admin User admin

Dashboard Logout

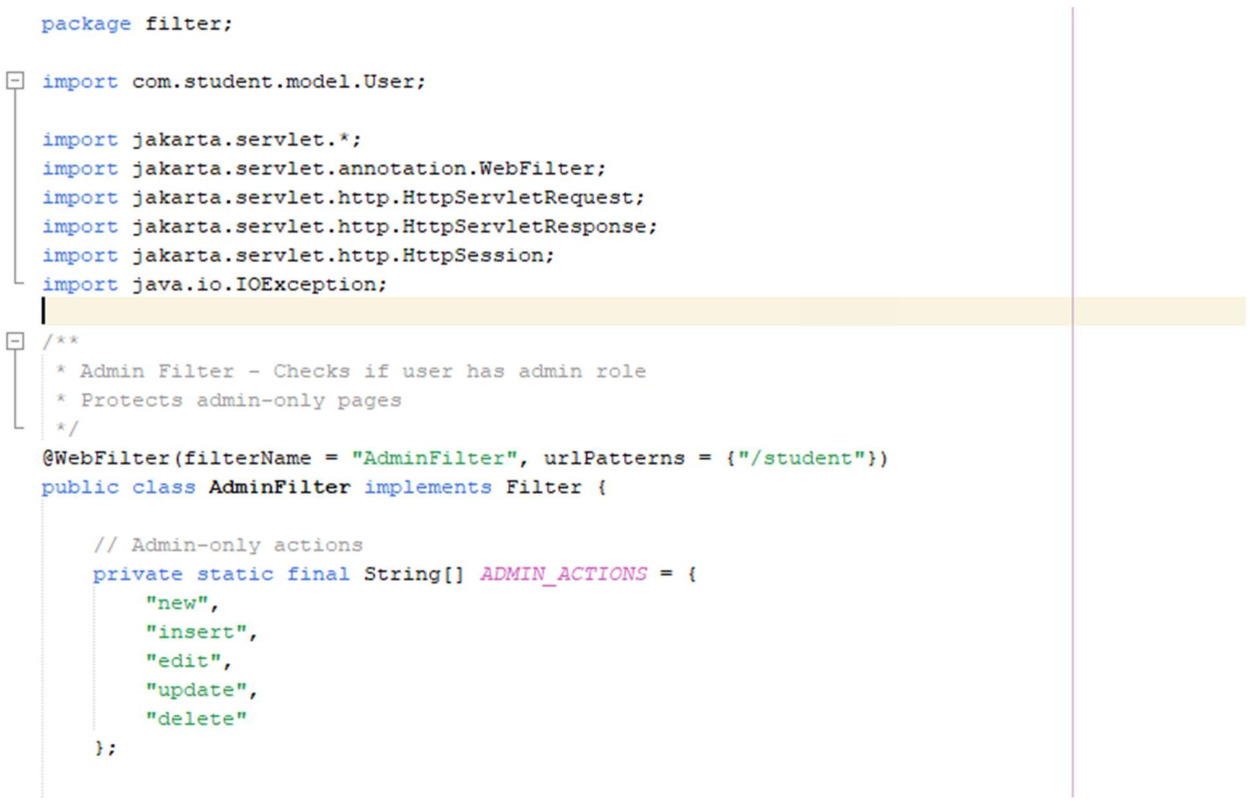
Student List

Add New Student

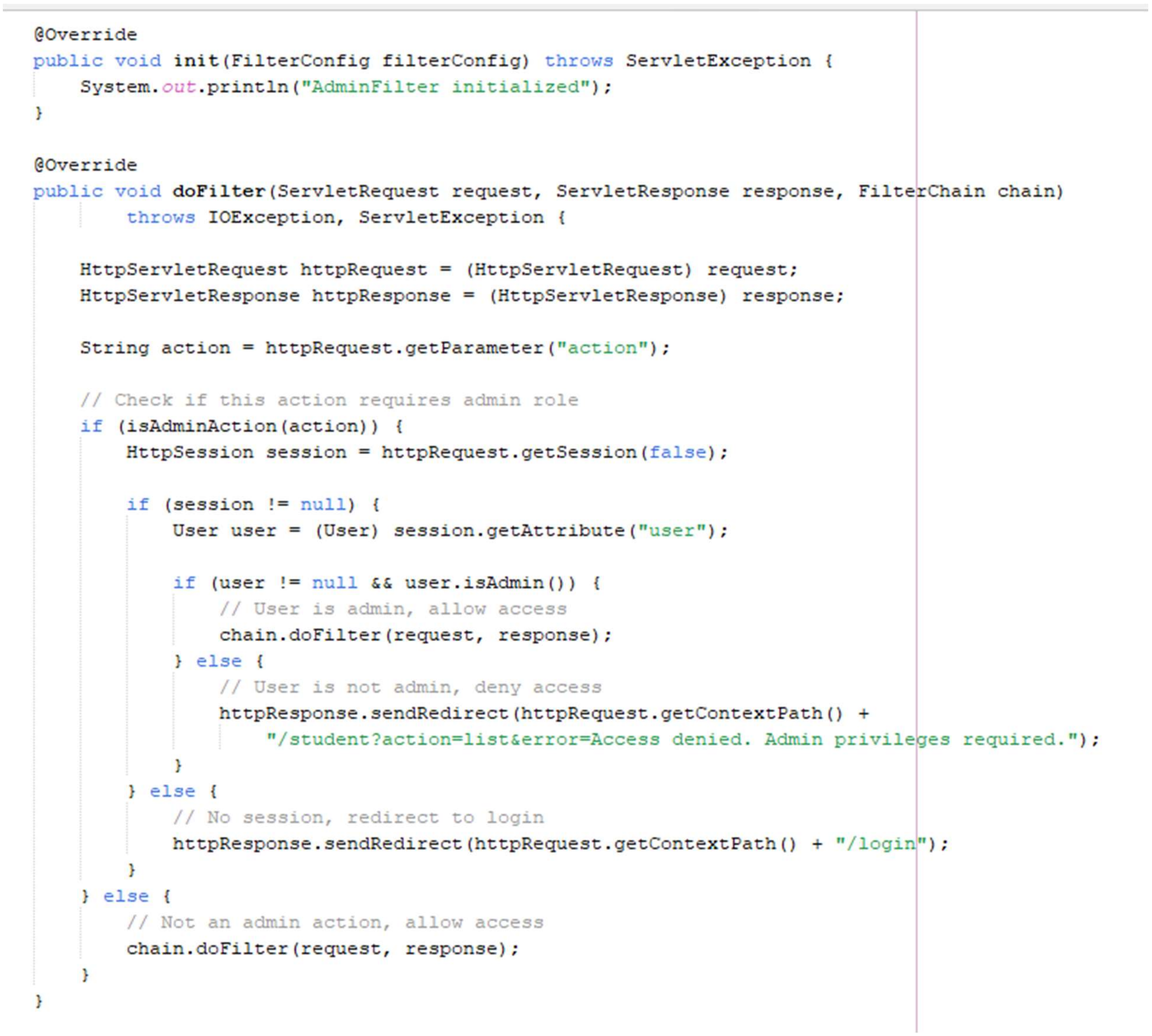
ID	CODE	NAME	EMAIL	MAJOR	ACTIONS
13	SV003	Le Van C	c.le@example.com	Software Engineering	Edit Delete
12	SV002	Tran Thi B	b.tran@example.com	Information Technology	Edit Delete
11	SV001	Nguyen Van A	a.nguyen@example.com	Computer Science	Edit Delete

6. Admin authorization

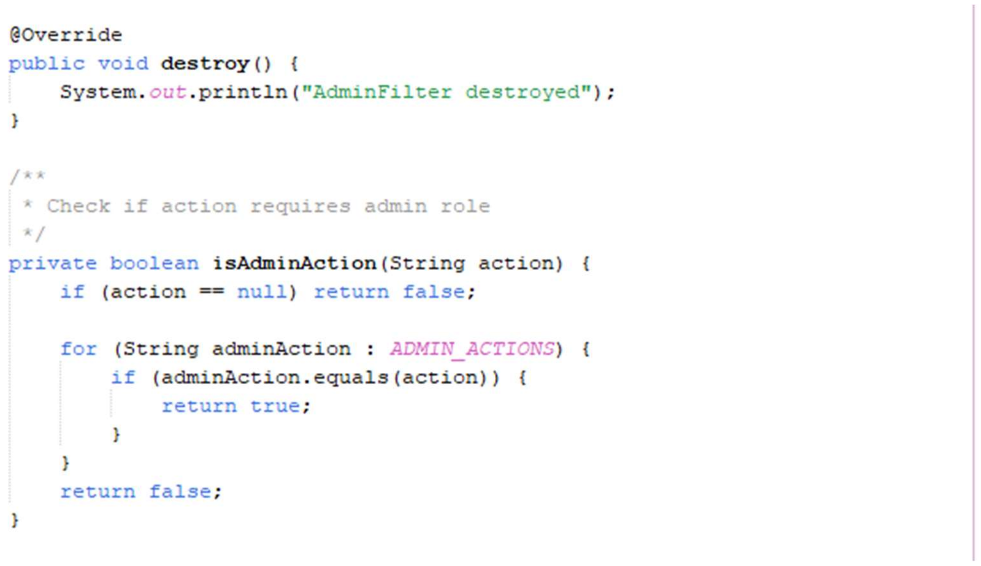
Code:



```java
package filter;

import com.student.model.User;

import jakarta.servlet.*;
import jakarta.servlet.annotation.WebFilter;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import java.io.IOException;

/**
 * Admin Filter - Checks if user has admin role
 * Protects admin-only pages
 */
@WebFilter(filterName = "AdminFilter", urlPatterns = {"/student"})
public class AdminFilter implements Filter {

    // Admin-only actions
    private static final String[] ADMIN_ACTIONS = {
        "new",
        "insert",
        "edit",
        "update",
        "delete"
    };

```



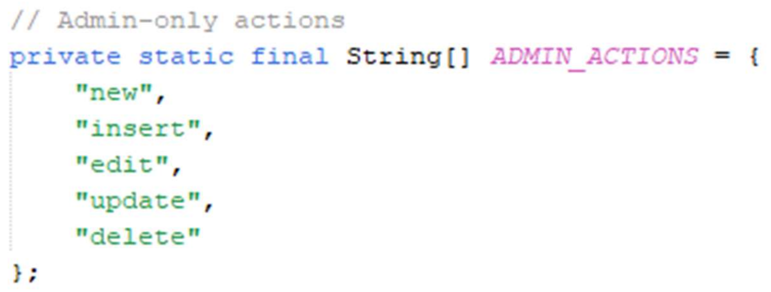
```java
@Override
public void init(FilterConfig filterConfig) throws ServletException {
    System.out.println("AdminFilter initialized");
}

@Override
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {

    HttpServletRequest httpRequest = (HttpServletRequest) request;
    HttpServletResponse httpResponse = (HttpServletResponse) response;

    String action = httpRequest.getParameter("action");

    // Check if this action requires admin role
    if (isAdminAction(action)) {
        HttpSession session = httpRequest.getSession(false);

        if (session != null) {
            User user = (User) session.getAttribute("user");

            if (user != null && user.isAdmin()) {
                // User is admin, allow access
                chain.doFilter(request, response);
            } else {
                // User is not admin, deny access
                httpResponse.sendRedirect(httpRequest.getContextPath() +
                    "/student?action=list&error=Access denied. Admin privileges required.");
            }
        } else {
            // No session, redirect to login
            httpResponse.sendRedirect(httpRequest.getContextPath() + "/login");
        }
    } else {
        // Not an admin action, allow access
        chain.doFilter(request, response);
    }
}
```



```java
@Override
public void destroy() {
    System.out.println("AdminFilter destroyed");
}

/**
 * Check if action requires admin role
 */
private boolean isAdminAction(String action) {
    if (action == null) return false;

    for (String adminAction : ADMIN_ACTIONS) {
        if (adminAction.equals(action)) {
            return true;
        }
    }
    return false;
}
```

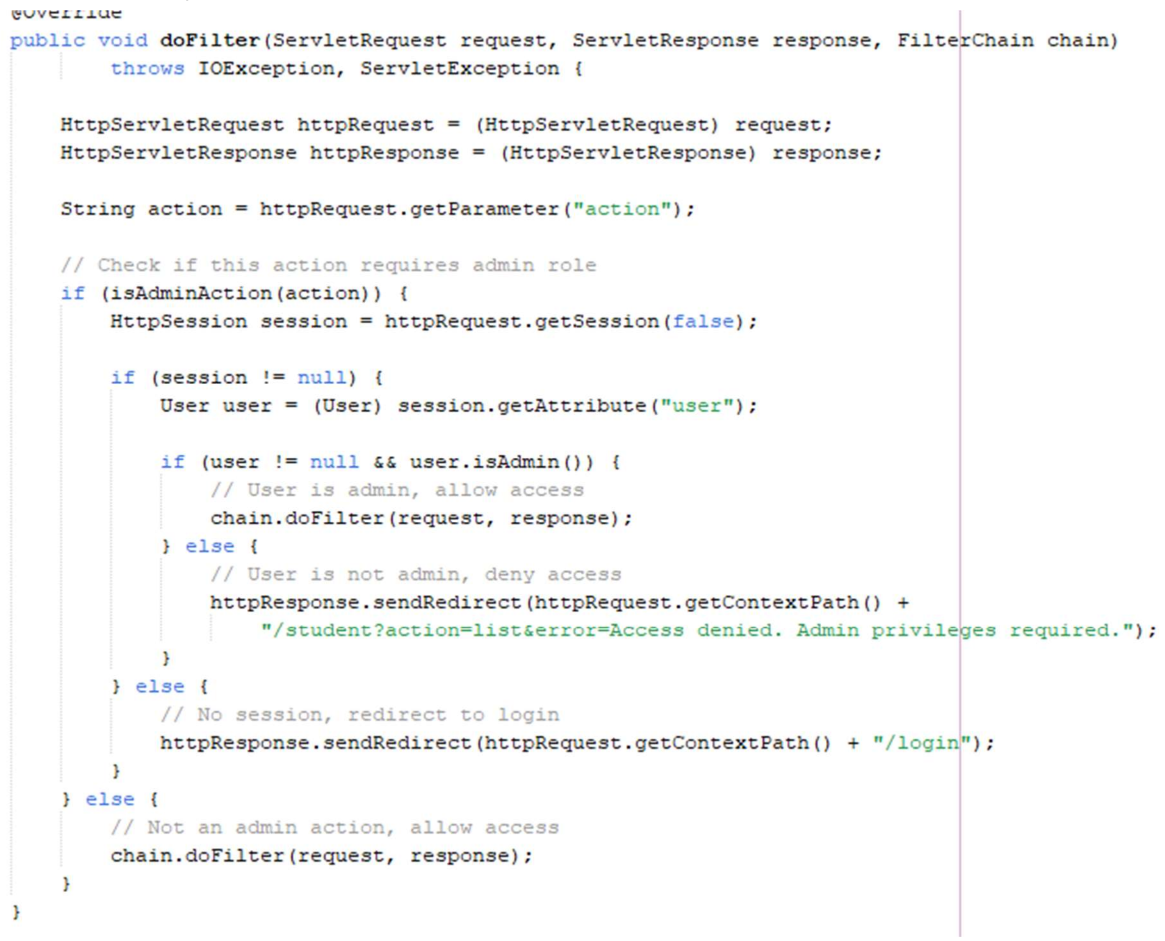
<u>Explanation:</u>

This admin filter is executed once the user logged in to the page with a role. Firstly, we define actions that are Admin exclusive



```java
// Admin-only actions
private static final String[] ADMIN_ACTIONS = {
    "new",
    "insert",
    "edit",
    "update",
    "delete"
};
```

In dofilter() for admin authorization will check whether the performed action is belonged to admin or not. If action is belonged to admin, will get the browser session to check if user session is null then check user role. Once if statements are satisfied perform the admin exclusive actions.



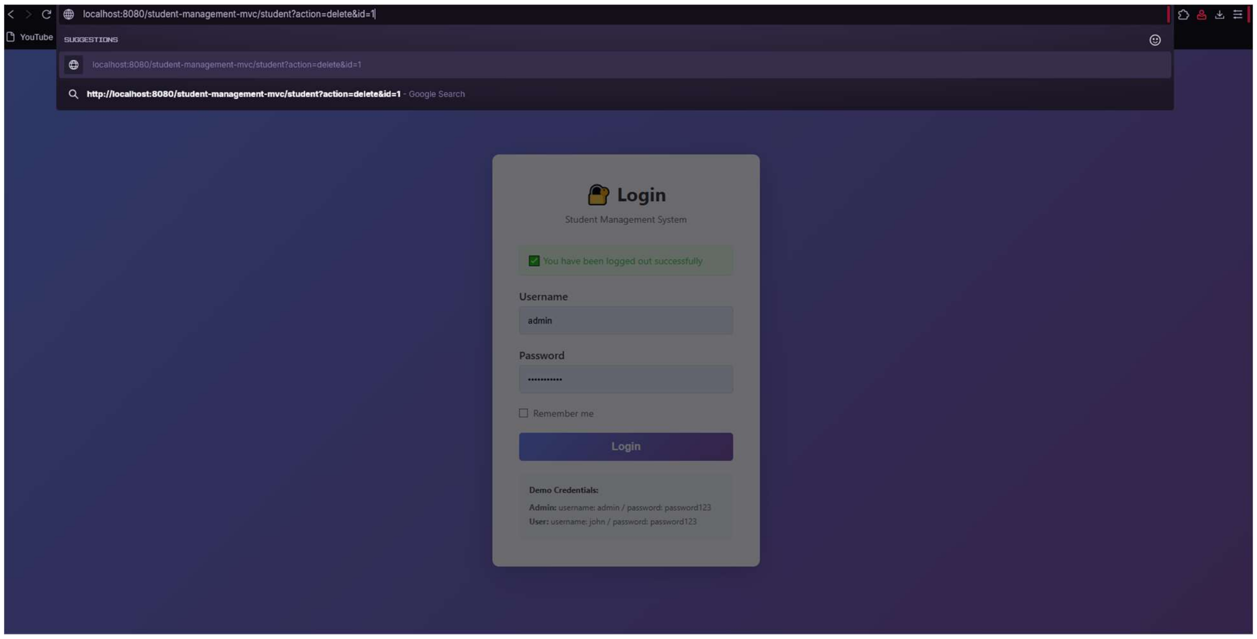
```java
@Override
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {

    HttpServletRequest httpRequest = (HttpServletRequest) request;
    HttpServletResponse httpResponse = (HttpServletResponse) response;

    String action = httpRequest.getParameter("action");

    // Check if this action requires admin role
    if (isAdminAction(action)) {
        HttpSession session = httpRequest.getSession(false);

        if (session != null) {
            User user = (User) session.getAttribute("user");

            if (user != null && user.isAdmin()) {
                // User is admin, allow access
                chain.doFilter(request, response);
            } else {
                // User is not admin, deny access
                httpResponse.sendRedirect(httpRequest.getContextPath() +
                    "/student?action=list&error=Access denied. Admin privileges required.");
            }
        } else {
            // No session, redirect to login
            httpResponse.sendRedirect(httpRequest.getContextPath() + "/login");
        }
    } else {
        // Not an admin action, allow access
        chain.doFilter(request, response);
    }
}
```

Else redirect to login page if session equals null or deny access if user is not admin

Conclusion:

Case user is not logged in :

User tries to perform admin exclusive actions -> isAdminAction() return true ->dofilter() get session -> user not logged in -> redirect to login page



localhost:8080/student-management-mvc/student?action=delete&id=1

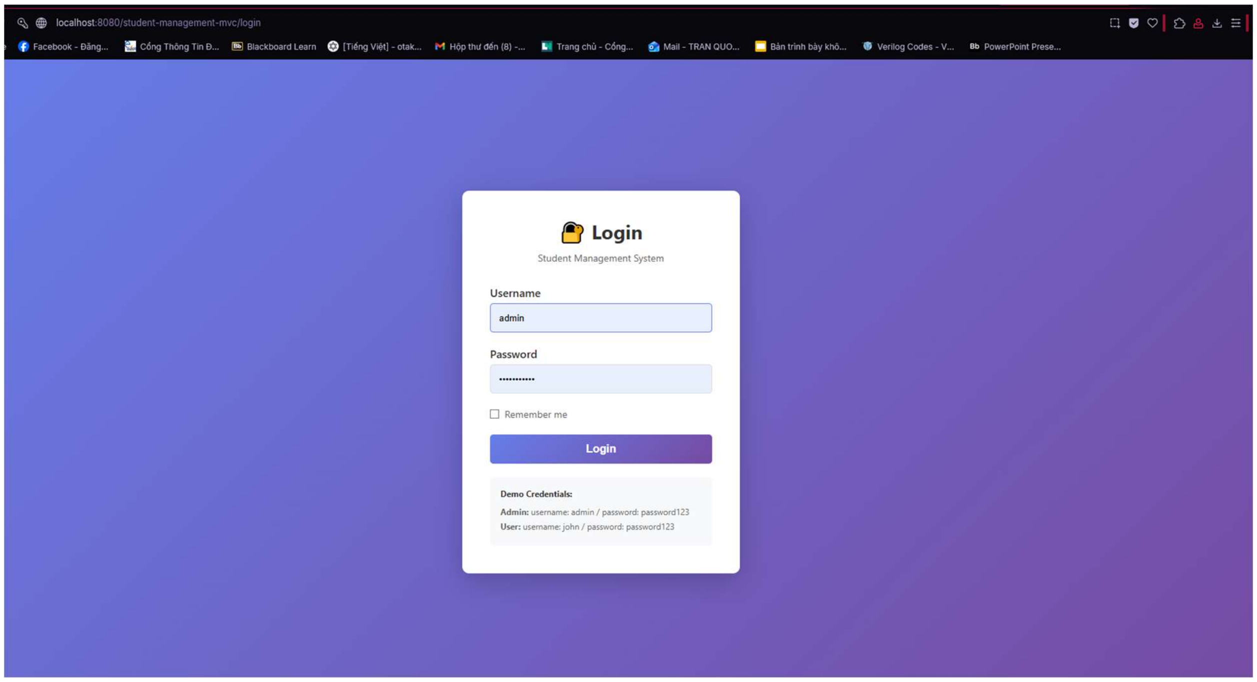
SUGGESTIONS
localhost:8080/student-management-mvc/student?action=delete&id=1
http://localhost:8080/student-management-mvc/student?action=delete&id=1 - Google Search
Login
Student Management System

You have been logged out successfully

Username: admin

Password: ••••••••••

Demo Credentials:
Admin: username: admin / password: password123
User: username: john / password: password123



localhost:8080/student-management-mvc/login

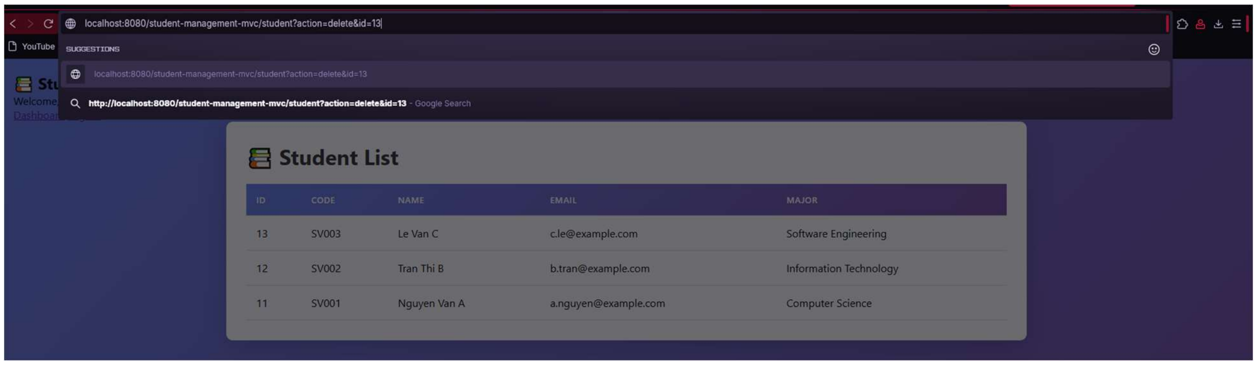
Username: admin

Password: ••••••••••

Demo Credentials:
Admin: username: admin / password: password123
User: username: john / password: password123

Case user is logged in with user role:

User tries to perform admin exclusive actions -> isAdminAction() return true ->dofilter() get session -> user logged in -> user role != admin -> deny action


localhost:8080/student-management-mvc/student?action=delete&id=13

SUGGESTIONS

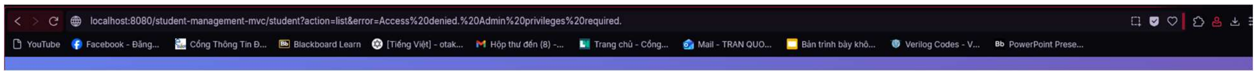
localhost:8080/student-management-mvc/student?action=delete&id=13

http://localhost:8080/student-management-mvc/student?action=delete&id=13 - Google Search

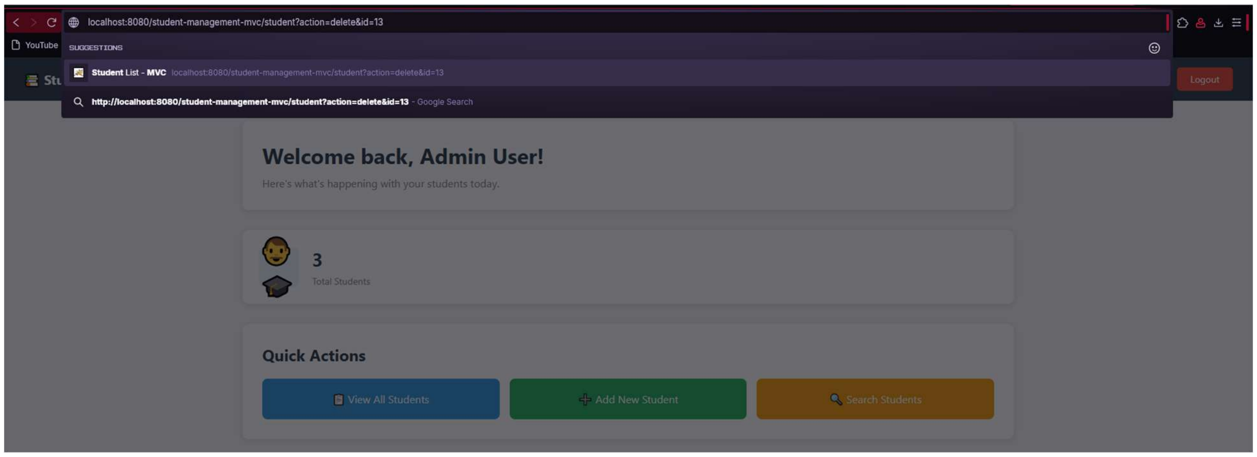
Student List

ID	CODE	NAME	EMAIL	MAJOR
13	SV003	Le Van C	c.le@example.com	Software Engineering
12	SV002	Tran Thi B	b.tran@example.com	Information Technology
11	SV001	Nguyen Van A	a.nguyen@example.com	Computer Science

http://localhost:8080/student-management-mvc/student?action=list&error=Access%20denied.%20Admin%20privileges%20required.


localhost:8080/student-management-mvc/student?action=list&error=Access%20denied.%20Admin%20privileges%20required.

Case user is logged in with admin role :
User tries to perform admin exclusive actions -> isAdminAction() return true ->dofilter() get session -> user logged in -> user role = admin -> execute action


localhost:8080/student-management-mvc/student?action=delete&id=13

SUGGESTIONS

Student List - MVC localhost:8080/student-management-mvc/student?action=delete&id=13

http://localhost:8080/student-management-mvc/student?action=delete&id=13 - Google Search

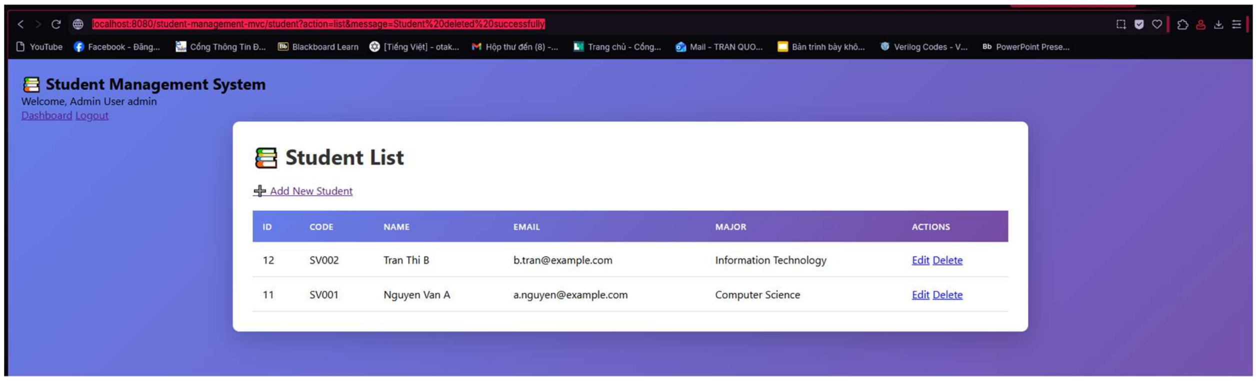
Welcome back, Admin User!

Here's what's happening with your students today.

3
Total Students

Quick Actions

http://localhost:8080/student-management-mvc/student?action=list&message=Student%20deleted%20successfully



localhost:8080/student-management-mvc/student?action=list&message=Student%20deleted%20successfully

YouTube | Facebook - Đăng... | Cổng Thông Tin Đ... | Blackboard Learn | [Tiếng Việt] - otak... | Hộp thư đến (8) -... | Trang chủ - Cổng... | Mail - TRAN QUO... | Bản trình bày khô... | Verilog Codes - V... | PowerPoint Prese...

Student Management System
Welcome, Admin User admin
Dashboard Logout

Student List

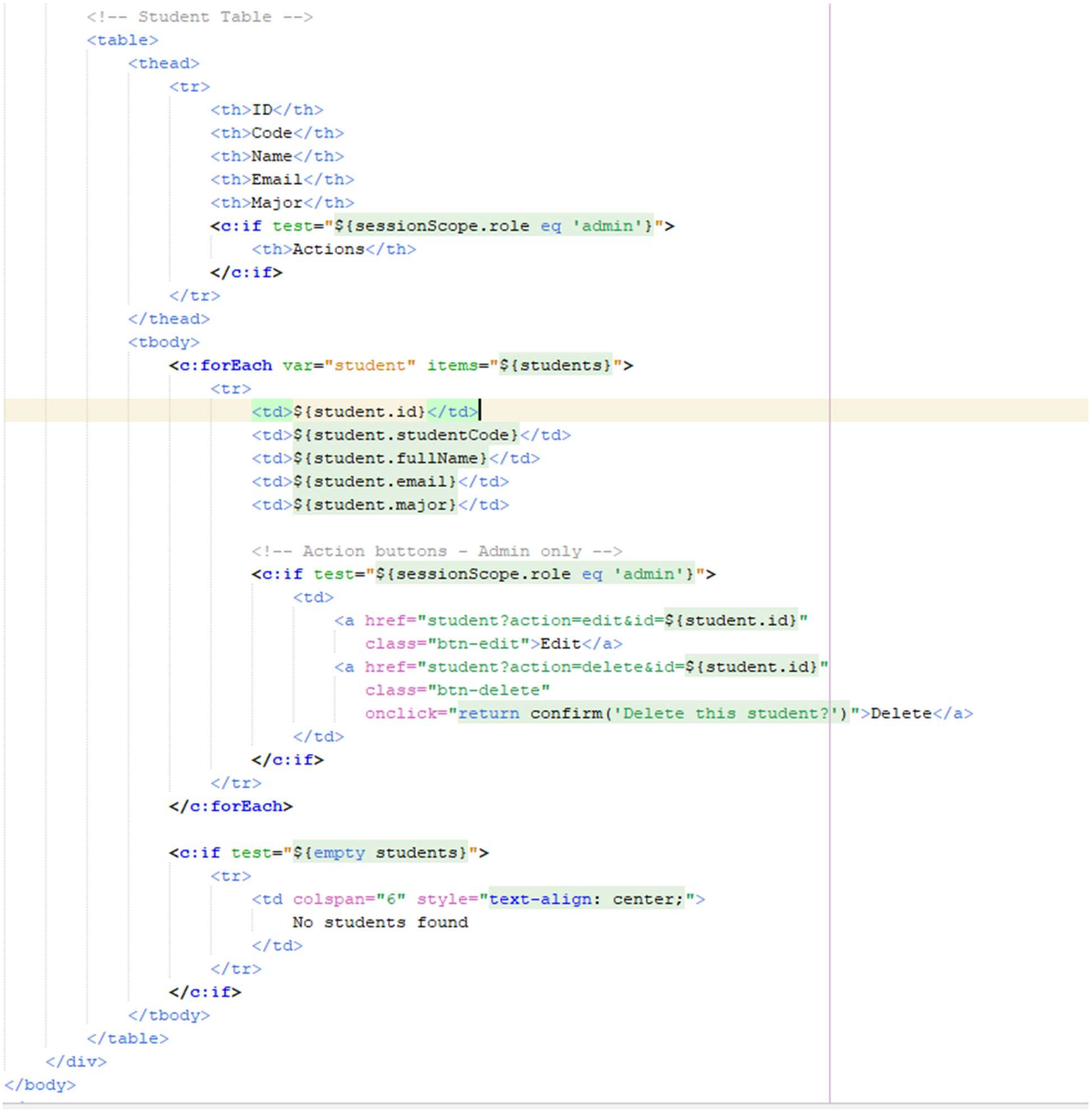
➕ Add New Student

ID	CODE	NAME	EMAIL	MAJOR	ACTIONS
12	SV002	Tran Thi B	b.tran@example.com	Information Technology	Edit Delete
11	SV001	Nguyen Van A	a.nguyen@example.com	Computer Science	Edit Delete

7. Role-Based UI

Code:

```
<body>
<!-- Navigation Bar -->
<div class="navbar">
    <h2>📚 Student Management System</h2>
    <div class="navbar-right">
        <div class="user-info">
            <span>Welcome, ${sessionScope.fullName}</span>
            <span class="role-badge role-${sessionScope.role}">
                ${sessionScope.role}
            </span>
        </div>
        <a href="dashboard" class="btn-nav">Dashboard</a>
        <a href="logout" class="btn-logout">Logout</a>
    </div>
</div>

<div class="container">
    <h1>📚 Student List</h1>

    <!-- Add button - Admin only -->
    <c:if test="${sessionScope.role eq 'admin'}">
        <div style="margin: 20px 0;">
            <a href="student?action=new" class="btn-add">➕ Add New Student</a>
        </div>
    </c:if>
```



```
        <!-- Student Table -->
        <table>
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Code</th>
                    <th>Name</th>
                    <th>Email</th>
                    <th>Major</th>
                    <c:if test="${sessionScope.role eq 'admin'}">
                        <th>Actions</th>
                    </c:if>
                </tr>
            </thead>
            <tbody>
                <c:forEach var="student" items="${students}">
                    <tr>
                        <td>${student.id}</td>
                        <td>${student.studentCode}</td>
                        <td>${student.fullName}</td>
                        <td>${student.email}</td>
                        <td>${student.major}</td>

                        <!-- Action buttons - Admin only -->
                        <c:if test="${sessionScope.role eq 'admin'}">
                            <td>
                                <a href="student?action=edit&id=${student.id}"
                                   class="btn-edit">Edit</a>
                                <a href="student?action=delete&id=${student.id}"
                                   class="btn-delete"
                                   onclick="return confirm('Delete this student?')">Delete</a>
                            </td>
                        </c:if>
                    </tr>
                </c:forEach>

                <c:if test="${empty students}">
                    <tr>
                        <td colspan="6" style="text-align: center;">
                            No students found
                        </td>
                    </tr>
                </c:if>
            </tbody>
        </table>
    </div>
</body>
```

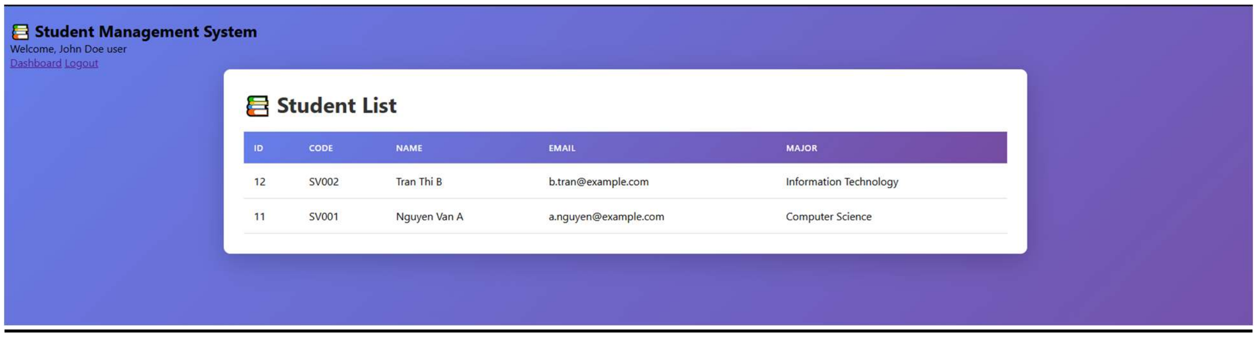
Explanation:

Role-Based UI declare that how the UI will be displayed based on user role. Since the browser can be accessed easily in both client side and server side. We can use browser session to get the user role to display UI. However, role-based UI can only works for client side, it only **hide/show** the actions/pages based on **User Role**. This means that without filter and authorization steps from server, user with user role can still performs action that should be admin exclusive.

Conclusion:

Case user is logged in with user role:

User logged in -> create a session for user -> send login request to server -> server response with User role to session if user is exist -> redirect to page with user role:



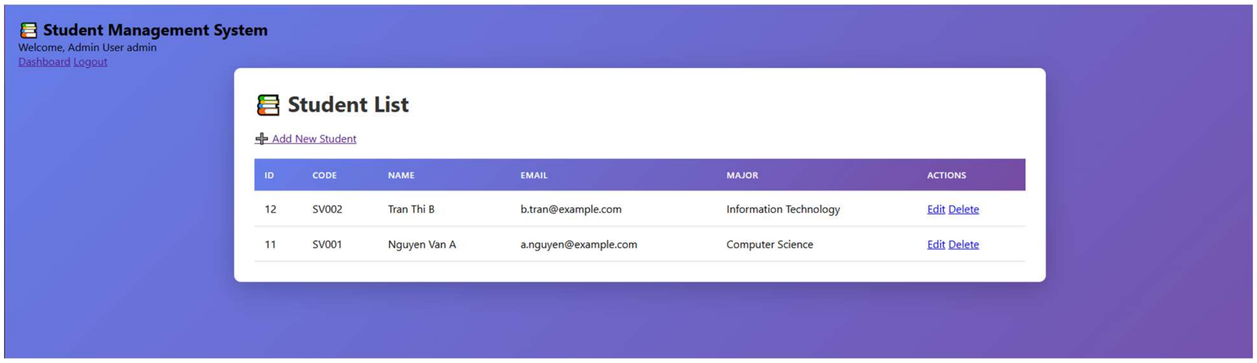
Student Management System
Welcome, John Doe user
Dashboard Logout

Student List

ID	CODE	NAME	EMAIL	MAJOR
12	SV002	Tran Thi B	b.tran@example.com	Information Technology
11	SV001	Nguyen Van A	a.nguyen@example.com	Computer Science

Case user is logged in with admin role:

User logged in -> create a session for user -> send login request to server -> server response with Admin role to session if user is exist -> redirect to page with admin role:



Student Management System
Welcome, Admin User admin
Dashboard Logout

Student List

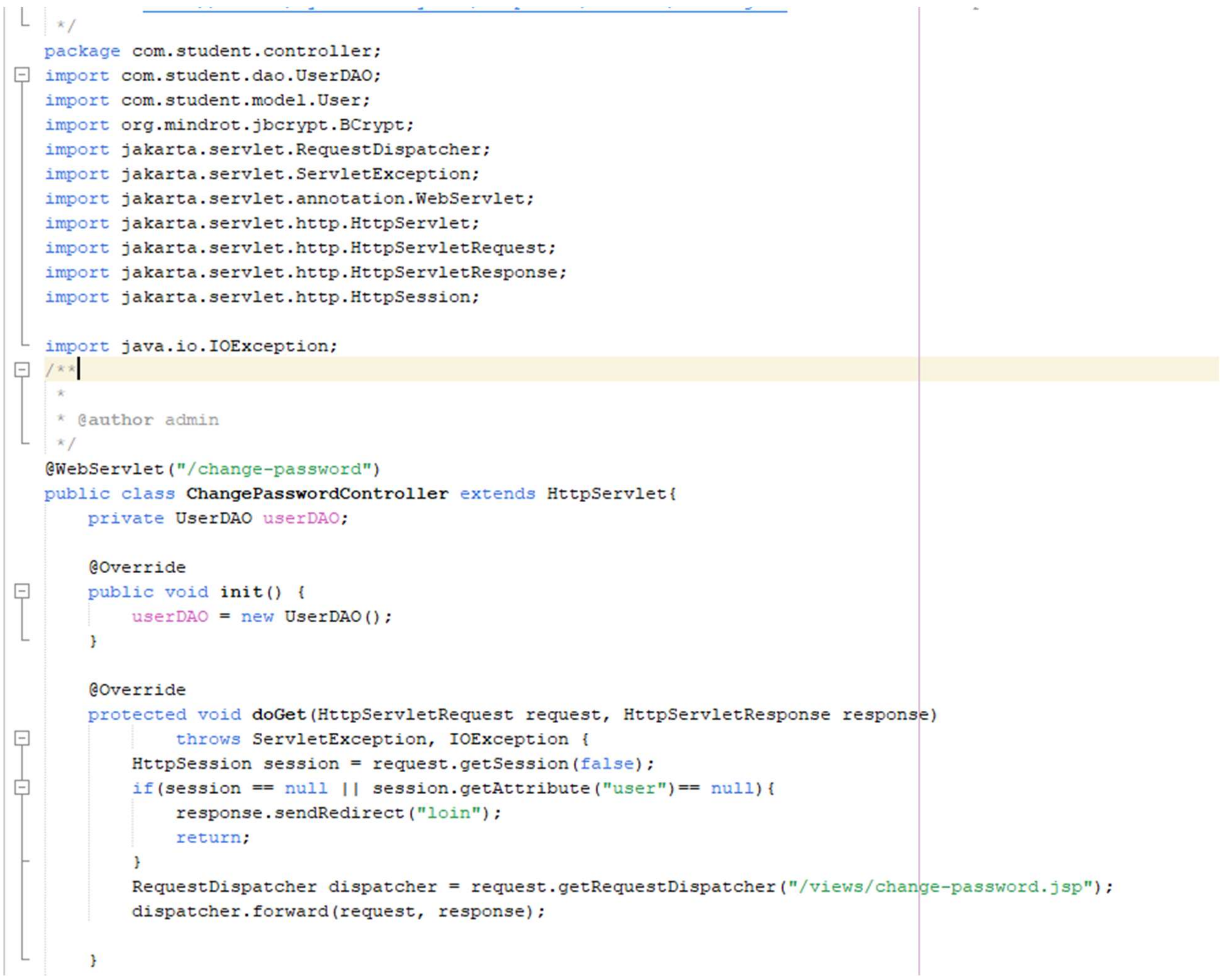
Add New Student

ID	CODE	NAME	EMAIL	MAJOR	ACTIONS
12	SV002	Tran Thi B	b.tran@example.com	Information Technology	Edit Delete
11	SV001	Nguyen Van A	a.nguyen@example.com	Computer Science	Edit Delete

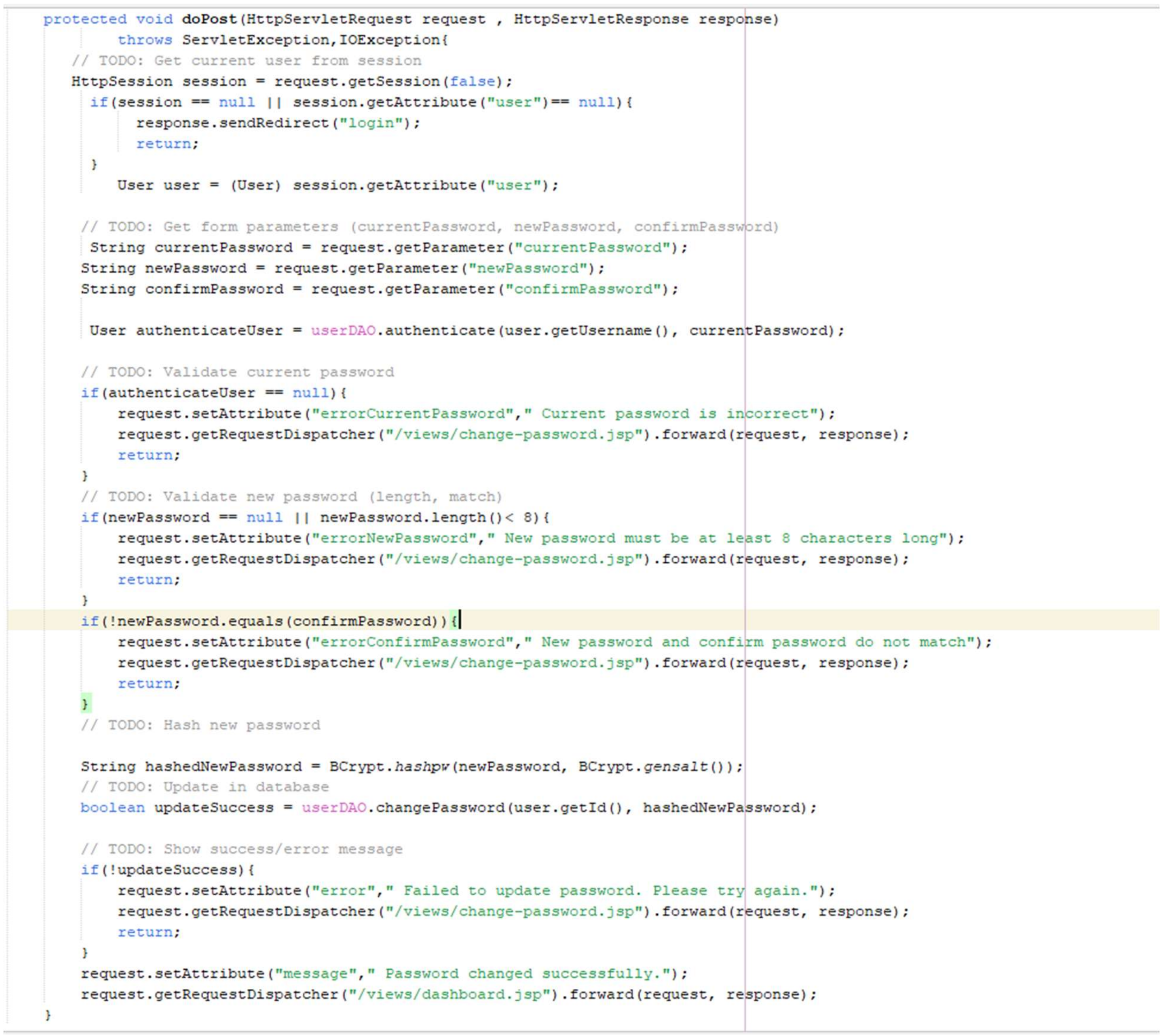
8. Change password

Code:

ChangePasswordController.java:

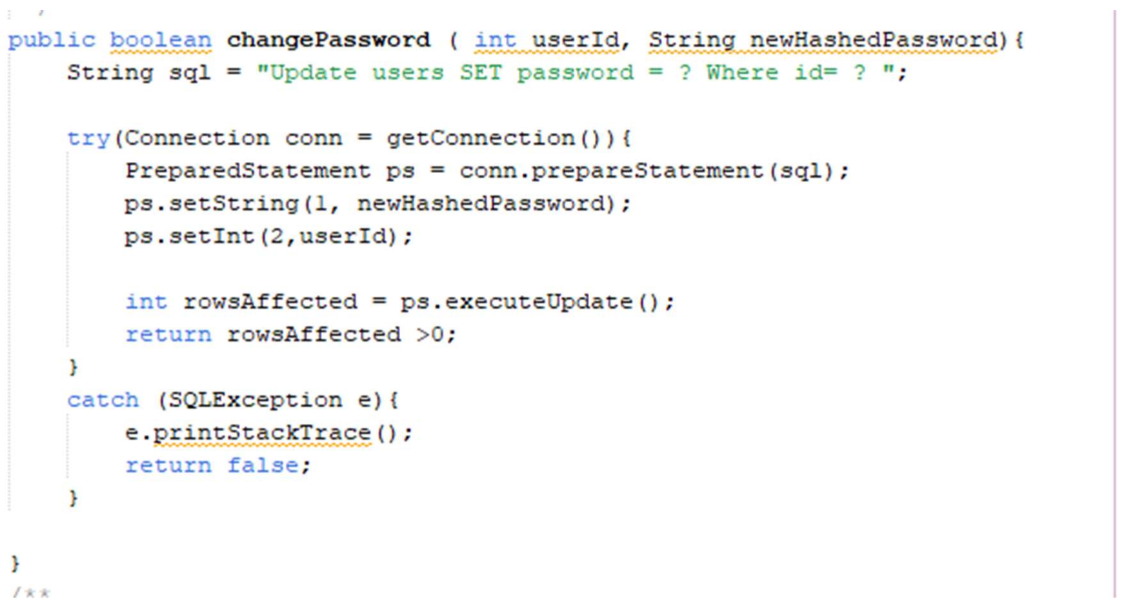


```java
 */
package com.student.controller;
import com.student.dao.UserDAO;
import com.student.model.User;
import org.mindrot.jbcrypt.BCrypt;
import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;

import java.io.IOException;
/**
 *
 * @author admin
 */
@WebServlet("/change-password")
public class ChangePasswordController extends HttpServlet{
    private UserDAO userDAO;

    @Override
    public void init() {
        userDAO = new UserDAO();
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        HttpSession session = request.getSession(false);
        if(session == null || session.getAttribute("user")== null){
            response.sendRedirect("loin");
            return;
        }
        RequestDispatcher dispatcher = request.getRequestDispatcher("/views/change-password.jsp");
        dispatcher.forward(request, response);

    }
```



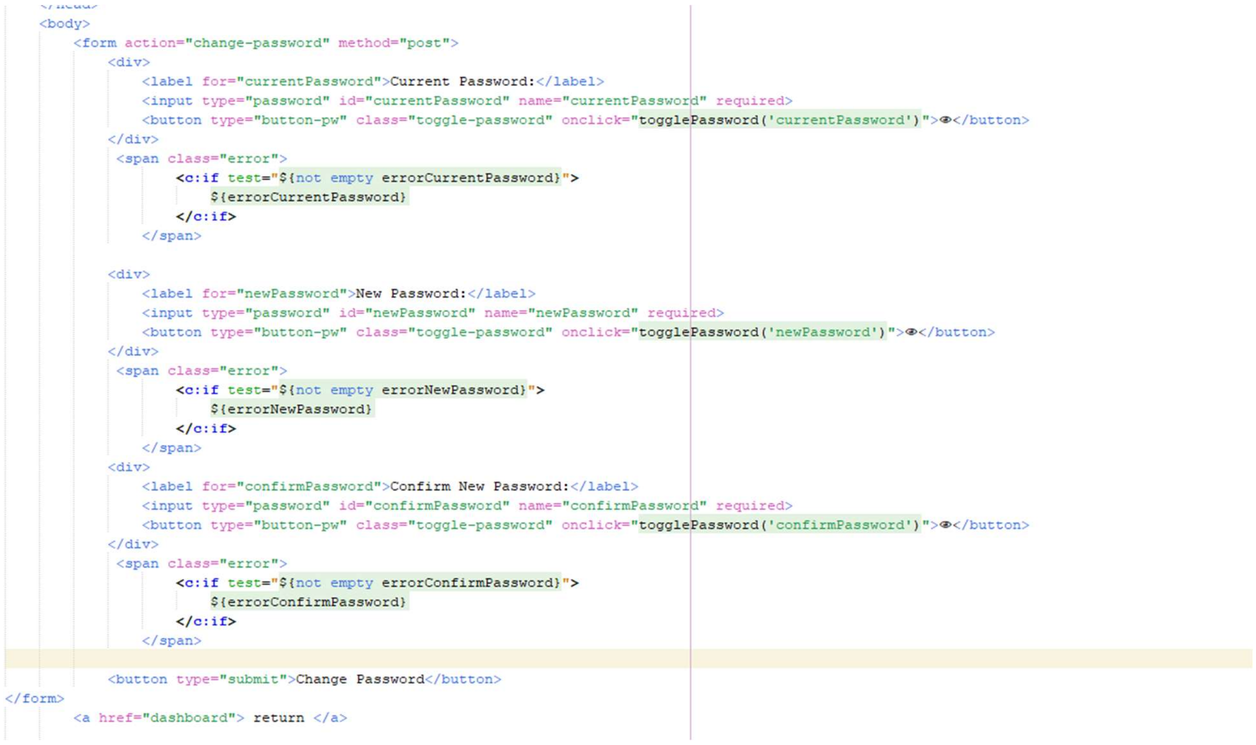
```
protected void doPost(HttpServletRequest request , HttpServletResponse response)
        throws ServletException,IOException{
   // TODO: Get current user from session
   HttpSession session = request.getSession(false);
    if(session == null || session.getAttribute("user")== null){
        response.sendRedirect("login");
        return;
    }
      User user = (User) session.getAttribute("user");

    // TODO: Get form parameters (currentPassword, newPassword, confirmPassword)
     String currentPassword = request.getParameter("currentPassword");
    String newPassword = request.getParameter("newPassword");
    String confirmPassword = request.getParameter("confirmPassword");

     User authenticateUser = userDAO.authenticate(user.getUsername(), currentPassword);

    // TODO: Validate current password
    if(authenticateUser == null){
        request.setAttribute("errorCurrentPassword"," Current password is incorrect");
        request.getRequestDispatcher("/views/change-password.jsp").forward(request, response);
        return;
    }
    // TODO: Validate new password (length, match)
    if(newPassword == null || newPassword.length()< 8){
        request.setAttribute("errorNewPassword"," New password must be at least 8 characters long");
        request.getRequestDispatcher("/views/change-password.jsp").forward(request, response);
        return;
    }
    if(!newPassword.equals(confirmPassword)){
        request.setAttribute("errorConfirmPassword"," New password and confirm password do not match");
        request.getRequestDispatcher("/views/change-password.jsp").forward(request, response);
        return;
    }
    // TODO: Hash new password

    String hashedNewPassword = BCrypt.hashpw(newPassword, BCrypt.gensalt());
    // TODO: Update in database
    boolean updateSuccess = userDAO.changePassword(user.getId(), hashedNewPassword);

    // TODO: Show success/error message
    if(!updateSuccess){
        request.setAttribute("error"," Failed to update password. Please try again.");
        request.getRequestDispatcher("/views/change-password.jsp").forward(request, response);
        return;
    }
    request.setAttribute("message"," Password changed successfully.");
    request.getRequestDispatcher("/views/dashboard.jsp").forward(request, response);
}
```

Update UserDAO.java:



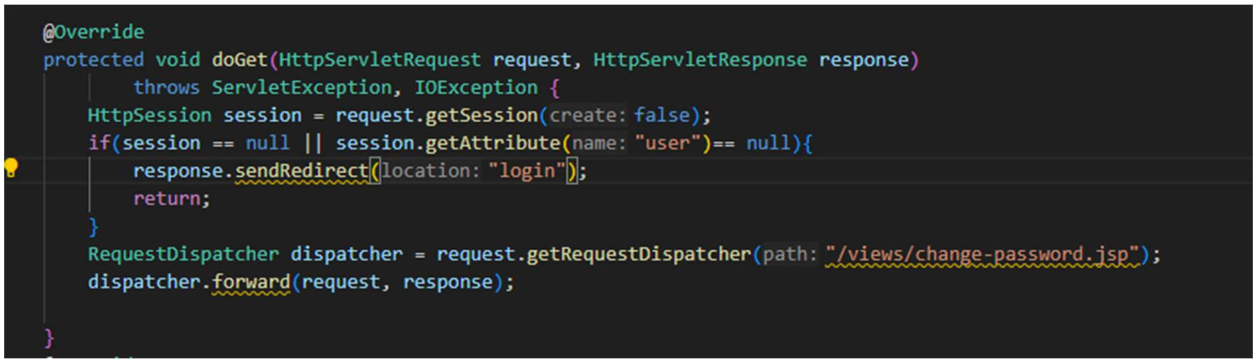
```java
public boolean changePassword ( int userId, String newHashedPassword){
    String sql = "Update users SET password = ? Where id= ? ";

    try(Connection conn = getConnection()){
        PreparedStatement ps = conn.prepareStatement(sql);
        ps.setString(1, newHashedPassword);
        ps.setInt(2,userId);

        int rowsAffected = ps.executeUpdate();
        return rowsAffected >0;
    }
    catch (SQLException e){
        e.printStackTrace();
        return false;
    }

}
/**
```

Change-password.jsp:



```html
    <body>
        <form action="change-password" method="post">
            <div>
                <label for="currentPassword">Current Password:</label>
                <input type="password" id="currentPassword" name="currentPassword" required>
                <button type="button-pw" class="toggle-password" onclick="togglePassword('currentPassword')">👁</button>
            </div>
             <span class="error">
                    <c:if test="${not empty errorCurrentPassword}">
                        ${errorCurrentPassword}
                    </c:if>
                </span>

            <div>
                <label for="newPassword">New Password:</label>
                <input type="password" id="newPassword" name="newPassword" required>
                <button type="button-pw" class="toggle-password" onclick="togglePassword('newPassword')">👁</button>
            </div>
             <span class="error">
                    <c:if test="${not empty errorNewPassword}">
                        ${errorNewPassword}
                    </c:if>
                </span>
            <div>
                <label for="confirmPassword">Confirm New Password:</label>
                <input type="password" id="confirmPassword" name="confirmPassword" required>
                <button type="button-pw" class="toggle-password" onclick="togglePassword('confirmPassword')">👁</button>
            </div>
             <span class="error">
                    <c:if test="${not empty errorConfirmPassword}">
                        ${errorConfirmPassword}
                    </c:if>
                </span>

            <button type="submit">Change Password</button>
</form>
        <a href="dashboard"> return </a>
```

<u>Explanation:</u>

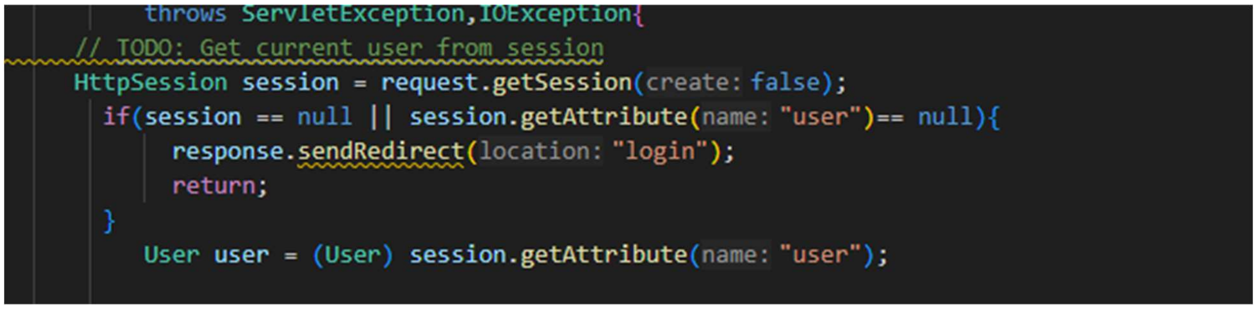


```java
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    HttpSession session = request.getSession(create: false);
    if(session == null || session.getAttribute(name: "user")== null){
        response.sendRedirect(location: "login");
        return;
    }
    RequestDispatcher dispatcher = request.getRequestDispatcher(path: "/views/change-password.jsp");
    dispatcher.forward(request, response);

}
```

When user perform action change password through URL or through UI , redirect to change-password.jsp if user is logged in , else redirect to login page.

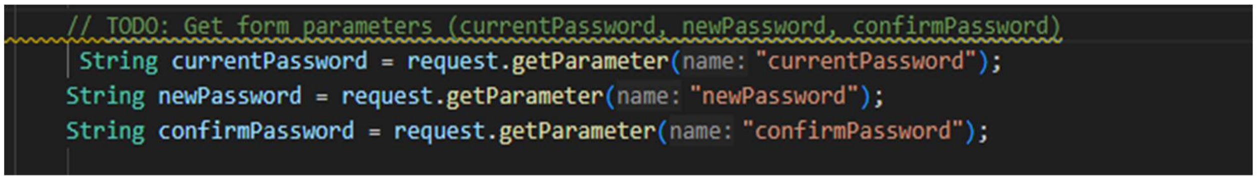
changePasswordController.java doPost(…) method:

initialize user object by gettAttribute from session. We now have access to user object for data validation .



```java
        throws ServletException,IOException{
    // TODO: Get current user from session
    HttpSession session = request.getSession(create: false);
     if(session == null || session.getAttribute(name: "user")== null){
         response.sendRedirect(location: "login");
         return;
     }
        User user = (User) session.getAttribute(name: "user");
```

Next we will take form parameter for password validation.

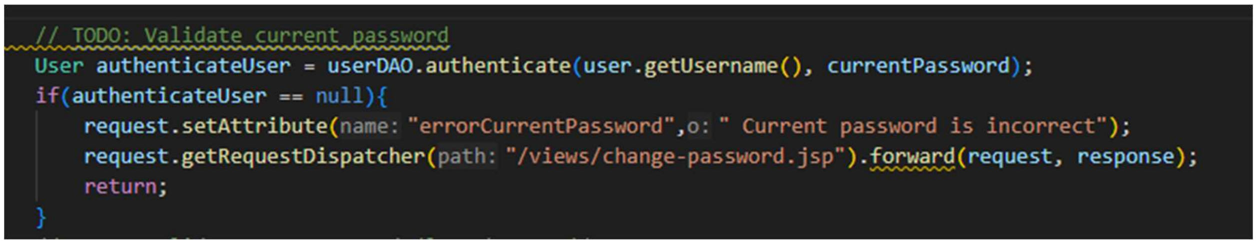


```java
   // TODO: Get form parameters (currentPassword, newPassword, confirmPassword)
    String currentPassword = request.getParameter(name: "currentPassword");
   String newPassword = request.getParameter(name: "newPassword");
   String confirmPassword = request.getParameter(name: "confirmPassword");
```

For password validations will require some constraints:

CurrentPassword requirement: must match the password stored in Database

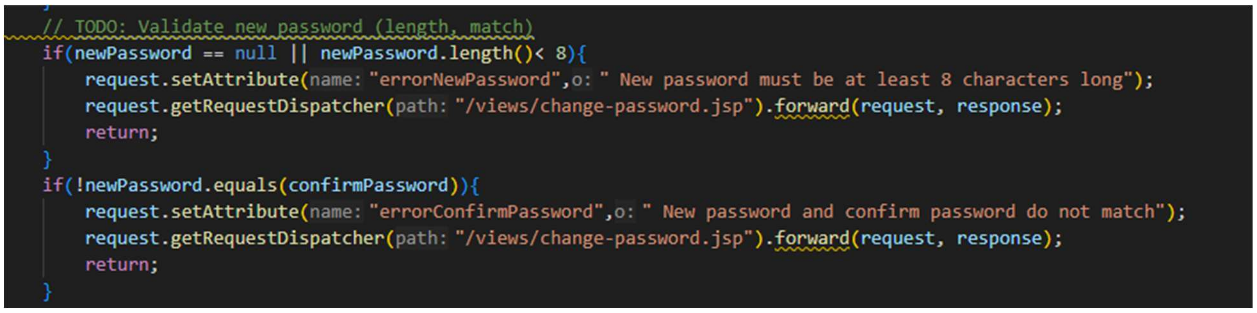
By using method getUsername() from user class and UserDAO.authenticate() method from UserDAO with input parameters user.getUsername() and currentPassword. We can check if the input currentPassword is matched with password we store database



```
// TODO: Validate current password
User authenticateUser = userDAO.authenticate(user.getUsername(), currentPassword);
if(authenticateUser == null){
    request.setAttribute(name: "errorCurrentPassword",o: " Current password is incorrect");
    request.getRequestDispatcher(path: "/views/change-password.jsp").forward(request, response);
    return;
}
```

NewPassword requirement : must at least 8 characters

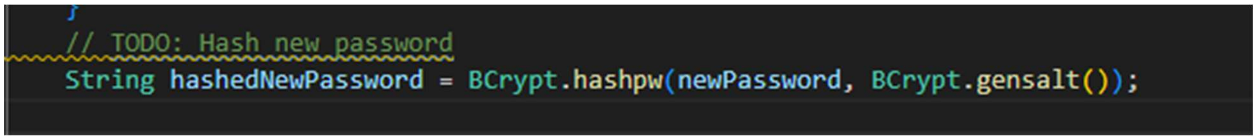
ConfirmPassword requirement : must match with NewPassword



```
// TODO: Validate new password (length, match)
if(newPassword == null || newPassword.length()< 8){
    request.setAttribute(name: "errorNewPassword",o: " New password must be at least 8 characters long");
    request.getRequestDispatcher(path: "/views/change-password.jsp").forward(request, response);
    return;
}
if(!newPassword.equals(confirmPassword)){
    request.setAttribute(name: "errorConfirmPassword",o: " New password and confirm password do not match");
    request.getRequestDispatcher(path: "/views/change-password.jsp").forward(request, response);
    return;
}
```

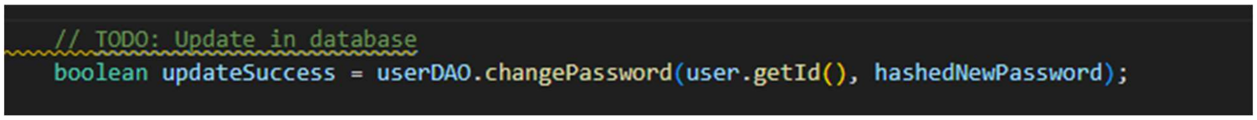
If one of those password validation is not satisfied show error and require user to take perform input password again

When all requirements are satisfied, server perform hashing newPassword



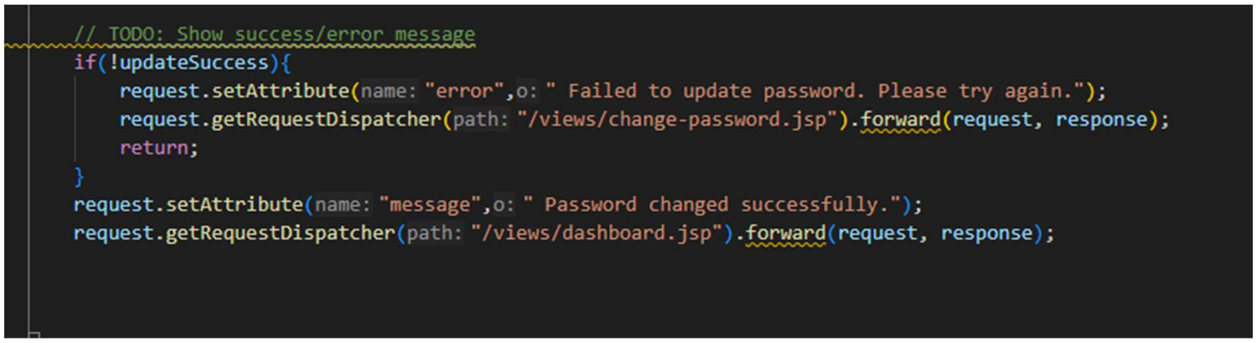
```
// TODO: Hash new password
String hashedNewPassword = BCrypt.hashpw(newPassword, BCrypt.gensalt());
```

Then update password in database



```
// TODO: Update in database
boolean updateSuccess = userDAO.changePassword(user.getId(), hashedNewPassword);
```

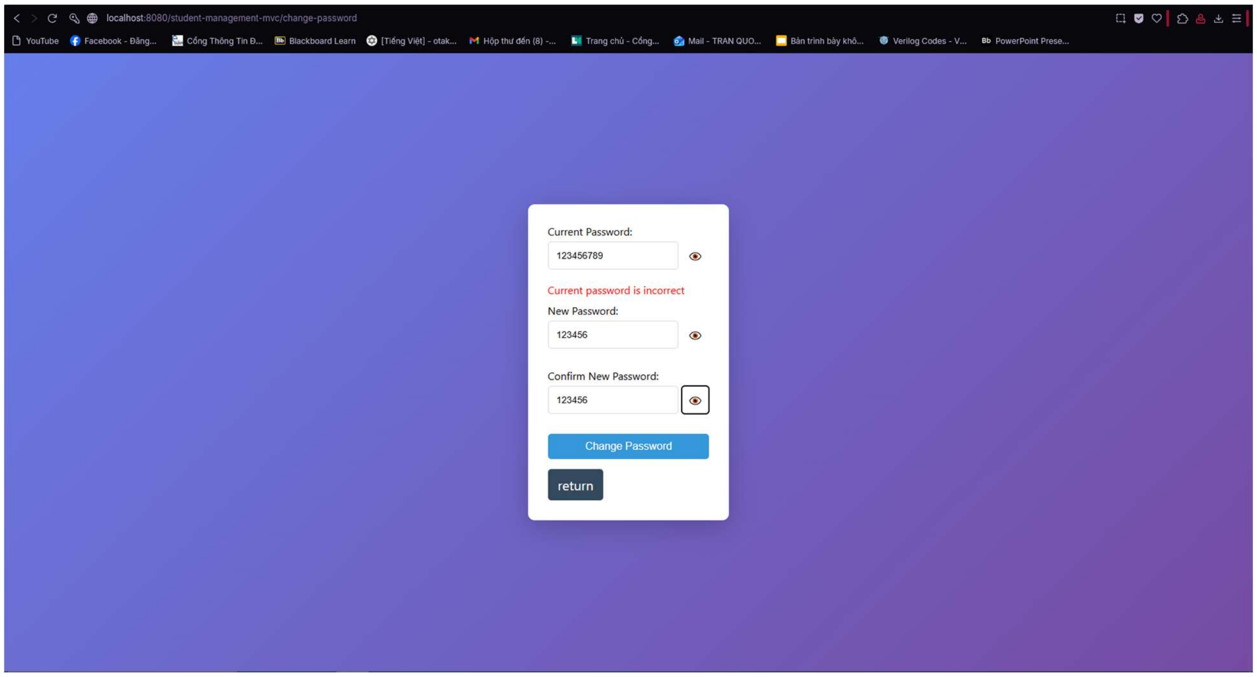
Finally will notify user if password is changed successfully.



```
// TODO: Show success/error message
if(!updateSuccess){
    request.setAttribute(name: "error",o: " Failed to update password. Please try again.");
    request.getRequestDispatcher(path: "/views/change-password.jsp").forward(request, response);
    return;
}
request.setAttribute(name: "message",o: " Password changed successfully.");
request.getRequestDispatcher(path: "/views/dashboard.jsp").forward(request, response);
```

Conclusion

User click change password -> navigate to change-password.jsp -> page take User input -> send to server -> server validate input -> if errors exist -> redirect to change-password with error name -> If no error -> execute password changes -> notify user


localhost:8080/student-management-mvc/change-password

Current Password:

123456789

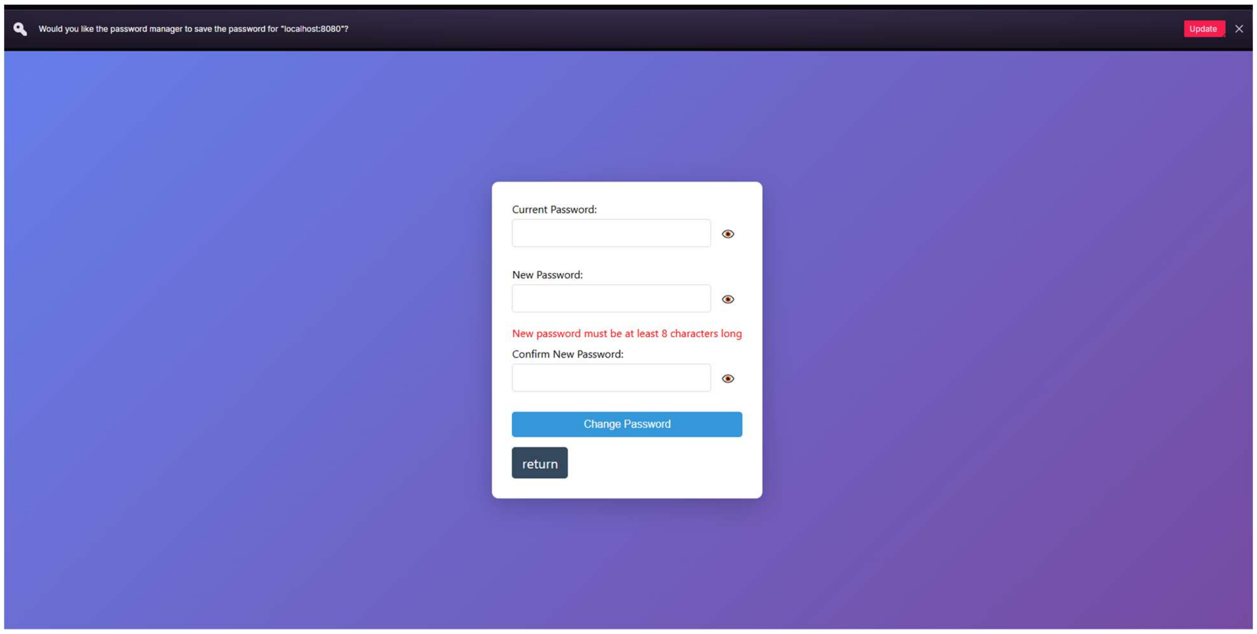
Current password is incorrect

New Password:

123456

Confirm New Password:

123456

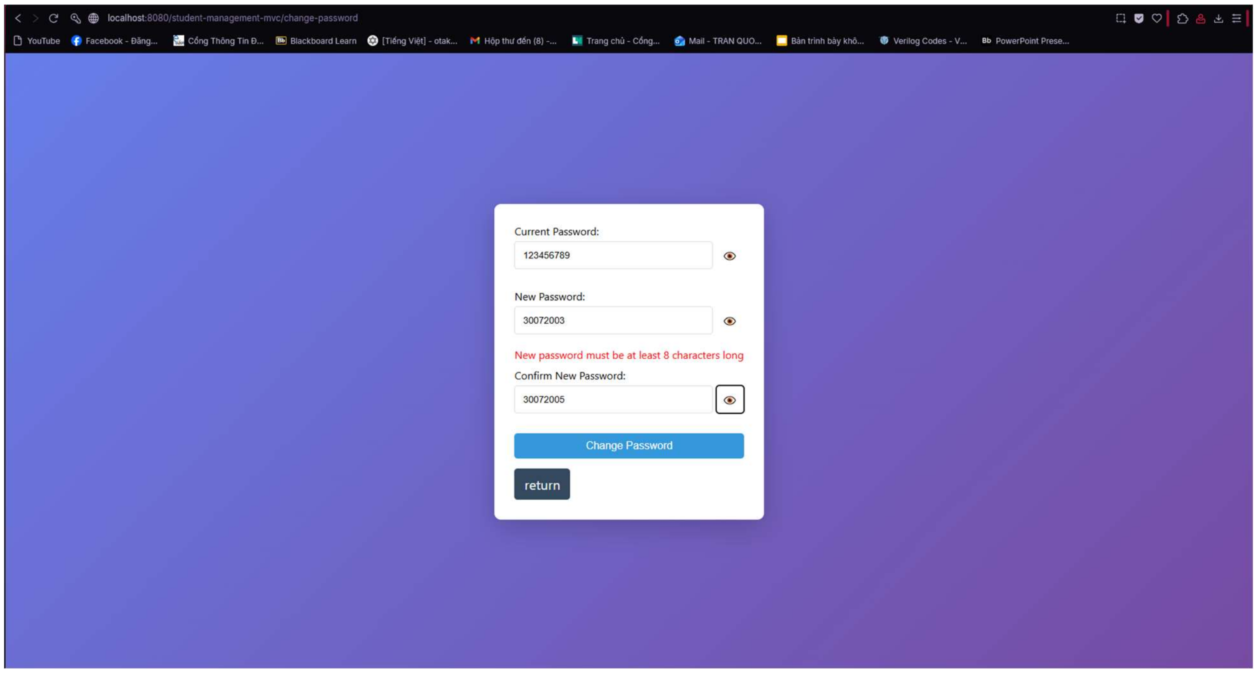

Would you like the password manager to save the password for "localhost:8080"?

Current Password:

New Password:

New password must be at least 8 characters long

Confirm New Password:



localhost:8080/student-management-mvc/change-password

Current Password: 123456789

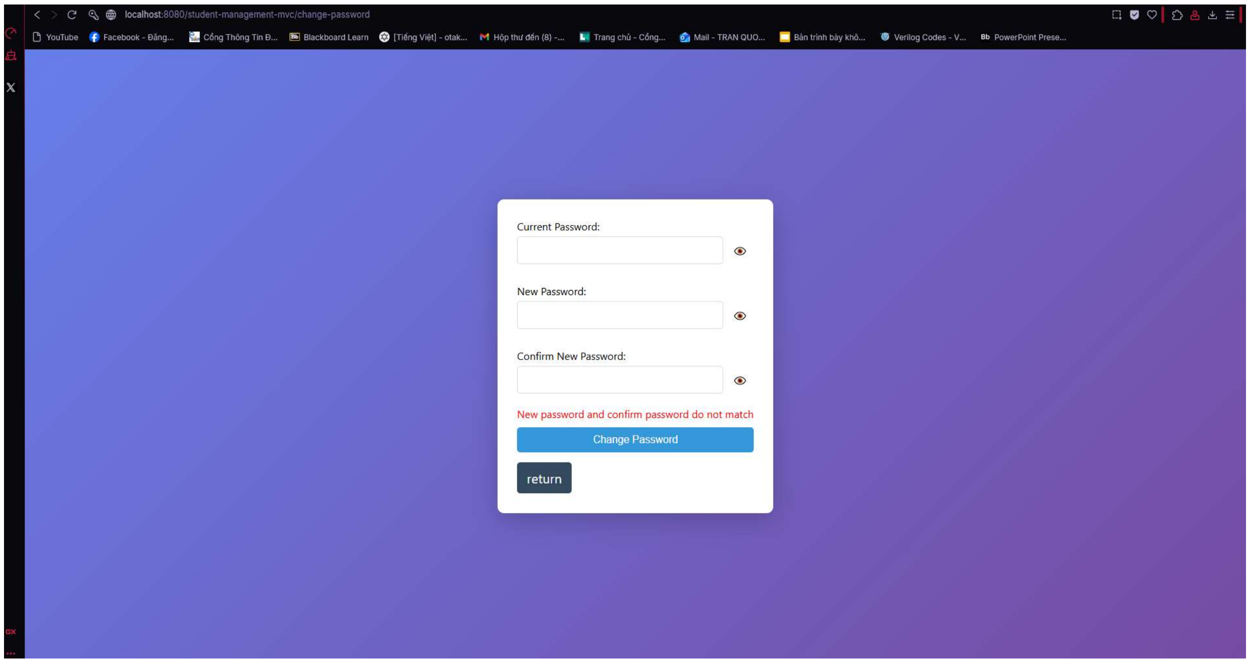
New Password: 30072003

New password must be at least 8 characters long

Confirm New Password: 30072005

Change Password

return



localhost:8080/student-management-mvc/change-password

Current Password:

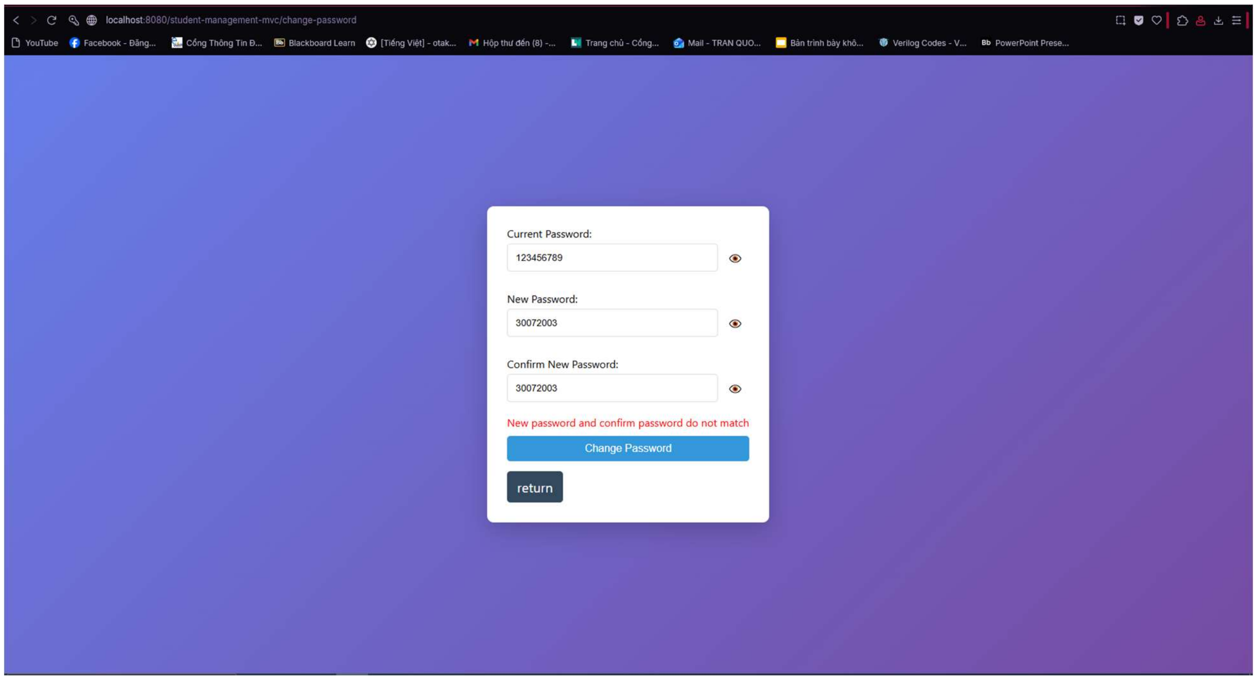
New Password:

Confirm New Password:

New password and confirm password do not match

Change Password

return



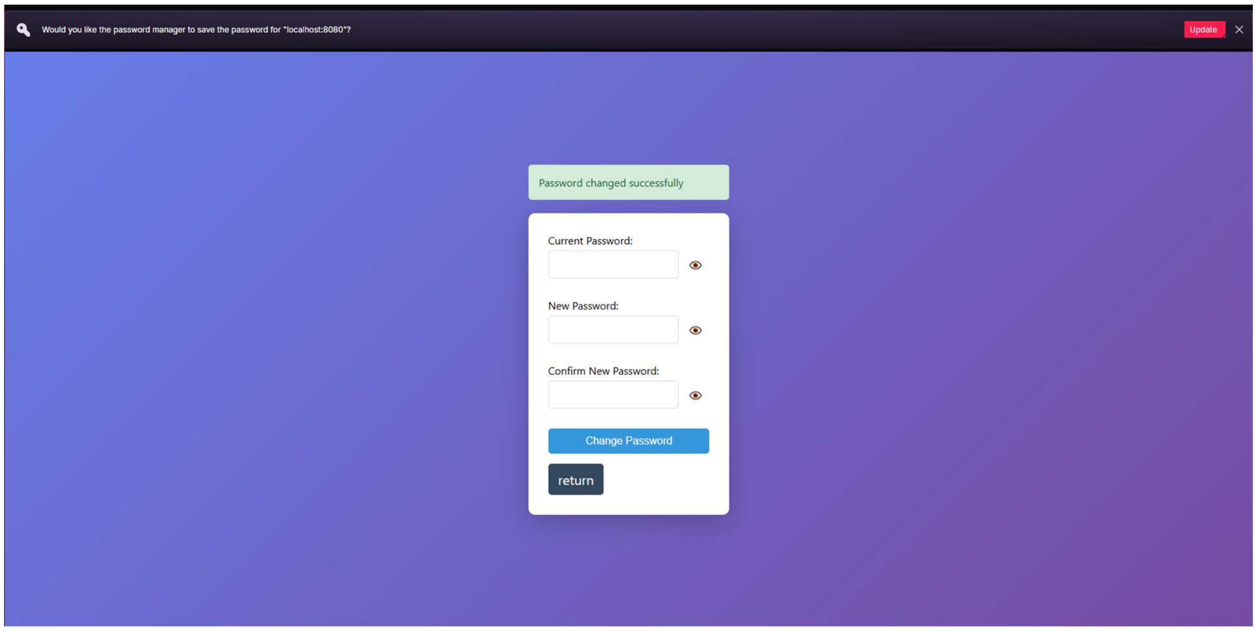
localhost:8080/student-management-mvc/change-password

Current Password: 123456789

New Password: 30072003

Confirm New Password: 30072003

New password and confirm password do not match



Would you like the password manager to save the password for "localhost:8080"?

Password changed successfully

Current Password:

New Password:

Confirm New Password: