Lab 6 Homework Exercise Report

Student: Lê Hoài Bảo
ID: ITCSIU22259
Exercise 1:
1.1

```
mysql> DESCRIBE users;
+------------+----------------------+------+-----+-------------------+-------------------+
| Field      | Type                 | Null | Key | Default           | Extra             |
+------------+----------------------+------+-----+-------------------+-------------------+
| id         | int                  | NO   | PRI | NULL              | auto_increment    |
| username   | varchar(50)          | NO   | UNI | NULL              |                   |
| password   | varchar(255)         | NO   |     | NULL              |                   |
| full_name  | varchar(100)         | NO   |     | NULL              |                   |
| role       | enum('admin','user') | YES  |     | user              |                   |
| is_active  | tinyint(1)           | YES  |     | 1                 |                   |
| created_at | timestamp            | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| last_login | timestamp            | YES  |     | NULL              |                   |
+------------+----------------------+------+-----+-------------------+-------------------+
8 rows in set (0.049 sec)
```

1.2

```java
package util;

import org.mindrot.jbcrypt.BCrypt;

public class PasswordHashGenerator {

    public static void main(String[] args) {
        String plainPassword = "password123";

        // Generate hash
        String hashedPassword = BCrypt.hashpw(plainPassword, BCrypt.gensalt());

        System.out.println("Plain Password: " + plainPassword);
        System.out.println("Hashed Password: " + hashedPassword);
        System.out.println("\nCopy the hashed password to your INSERT statement");

        // Test verification
        boolean matches = BCrypt.checkpw(plainPassword, hashedPassword);
        System.out.println("\nVerification test: " + matches);
    }
}
```

```
--- exec:3.1.0:exec (default-cli) @ student-management-mvc ---
Plain Password: password123
Hashed Password: $2a$10$.AMjybv1ReqdN6RrsGPssOMVWvwEKMSWDM4G/EJpNypyqtHV29LeK

Copy the hashed password to your INSERT statement

Verification test: true
------------------------------------------------------------------------
BUILD SUCCESS
```

1.3

```
mysql> INSERT INTO users (username, password, full_name, role) VALUES
    -> ('admin', '$2a$10$gkfZjtqXVt3uP3plRpcSjunypYV/UYyF1prCrqTzL8w79Kc.gPRsC', 'Admin User', 'admin
'
    -> ('john', '$2a$10$gkfZjtqXVt3uP3plRpcSjunypYV/UYyF1prCrqTzL8w79Kc.gPRsC', 'John Doe', 'user'),
    -> ('jane', '$2a$10$gkfZjtqXVt3uP3plRpcSjunypYV/UYyF1prCrqTzL8w79Kc.gPRsC', 'Jane Smith', 'user');
Query OK, 3 rows affected (0.012 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT id, username, role FROM users;
+----+----------+-------+
| id | username | role  |
+----+----------+-------+
|  4 | admin    | admin |
|  5 | john     | user  |
|  6 | jane     | user  |
+----+----------+-------+
3 rows in set (0.013 sec)
```

2.1

```java
package model;

import java.sql.Timestamp;

public class User {
    private int id;
    private String username;
    private String password;
    private String fullName;
    private String role;
    private boolean isActive;
    private Timestamp createdAt;
    private Timestamp lastLogin;

    // Constructors
    public User() {
    }

    public User(String username, String password, String fullName, String role) {
        this.username = username;
        this.password = password;
        this.fullName = fullName;
        this.role = role;
    }

    // Getters and Setters
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getUsername() {
```

```
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getFullName() {
        return fullName;
    }

    public void setFullName(String fullName) {
        this.fullName = fullName;
    }

    public String getRole() {
        return role;
    }

    public void setRole(String role) {
        this.role = role;
    }

    public boolean isActive() {
        return isActive;
    }
```

```
    public void setActive(boolean active) {
        isActive = active;
    }

    public Timestamp getCreatedAt() {
        return createdAt;
    }

    public void setCreatedAt(Timestamp createdAt) {
        this.createdAt = createdAt;
    }

    public Timestamp getLastLogin() {
        return lastLogin;
    }

    public void setLastLogin(Timestamp lastLogin) {
        this.lastLogin = lastLogin;
    }

    // Utility methods
    public boolean isAdmin() {
        return "admin".equalsIgnoreCase(this.role);
    }

    public boolean isUser() {
        return "user".equalsIgnoreCase(this.role);
    }

    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", username='" + username + '\'' +
                ", fullName='" + fullName + '\'' +
                ", role='" + role + '\'' +
                ", isActive=" + isActive +
                '}';
    }
}
```

2.2

```java
package dao;

import model.User;
import org.mindrot.jbcrypt.BCrypt;

import java.sql.*;

public class UserDAO {

    private static final String DB_URL = "jdbc:mysql://localhost:3306/student_manageme
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "Hb21082004";

    // SQL Queries
    private static final String SQL_AUTHENTICATE =
        "SELECT * FROM users WHERE username = ? AND is_active = TRUE";

    private static final String SQL_UPDATE_LAST_LOGIN =
        "UPDATE users SET last_login = NOW() WHERE id = ?";

    private static final String SQL_GET_BY_ID =
        "SELECT * FROM users WHERE id = ?";

    private static final String SQL_GET_BY_USERNAME =
        "SELECT * FROM users WHERE username = ?";

    private static final String SQL_INSERT =
        "INSERT INTO users (username, password, full_name, role) VALUES (?, ?, ?, ?)";

    // Get database connection
    private Connection getConnection() throws SQLException {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            return DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);
        } catch (ClassNotFoundException e) {
```

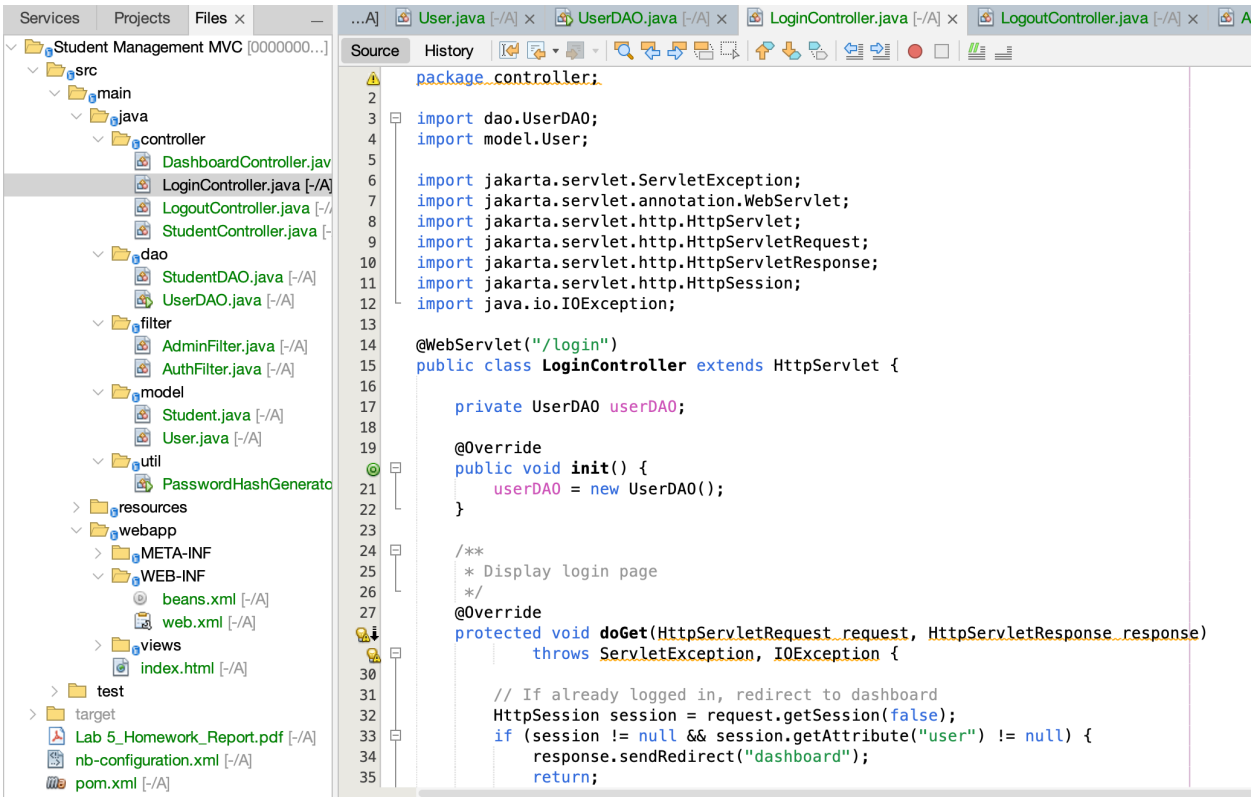
```
            throw new SQLException("MySQL Driver not found", e);
        }
    }

    /**
     * Authenticate user with username and password
     * @return User object if authentication successful, null otherwise
     */
    public User authenticate(String username, String password) {
        User user = null;

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

            pstmt.setString(1, username);

            try (ResultSet rs = pstmt.executeQuery()) {
                if (rs.next()) {
                    String hashedPassword = rs.getString("password");

                    // Verify password with BCrypt
                    if (BCrypt.checkpw(password, hashedPassword)) {
                        user = mapResultSetToUser(rs);

                        // Update last login time
                        updateLastLogin(user.getId());
                    }
                }
            }

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

```
        return user;
    }

    /**
     * Update user's last login timestamp
     */
    private void updateLastLogin(int userId) {
        try (Connection conn = getConnection();
             PreparedStatement pstmt = conn.prepareStatement(SQL_UPDATE_LAST_LOGIN)) {

            pstmt.setInt(1, userId);
            pstmt.executeUpdate();

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

    /**
     * Get user by ID
     */
    public User getUserById(int id) {
        User user = null;

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

            pstmt.setInt(1, id);

            try (ResultSet rs = pstmt.executeQuery()) {
                if (rs.next()) {
                    user = mapResultSetToUser(rs);
                }
            }
```

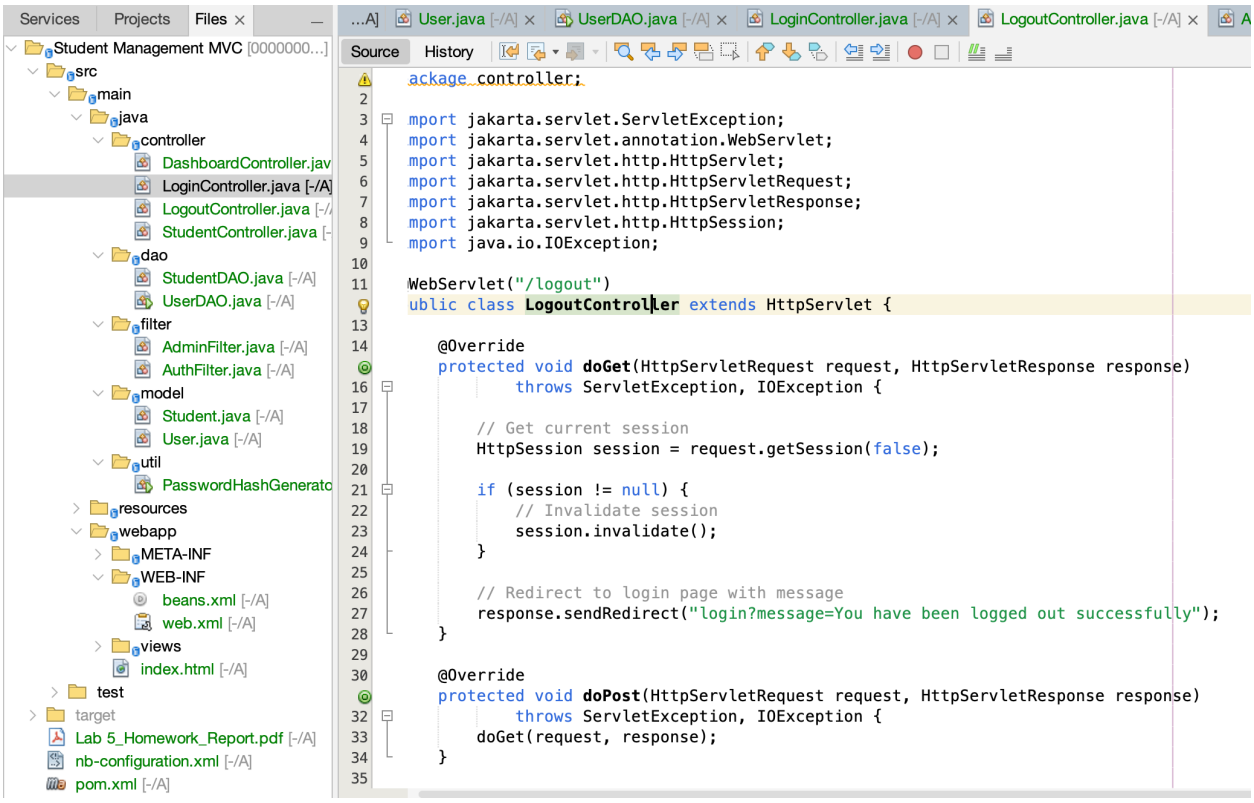
```
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return user;
    }

    /**
     * Get user by username
     */
    public User getUserByUsername(String username) {
        User user = null;

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

            pstmt.setString(1, username);

            try (ResultSet rs = pstmt.executeQuery()) {
                if (rs.next()) {
                    user = mapResultSetToUser(rs);
                }
            }

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

        return user;
    }

    /**
     * Create new user with hashed password
     */
```

```
    public boolean createUser(User user) {
        try (Connection conn = getConnection();
             PreparedStatement pstmt = conn.prepareStatement(SQL_INSERT)) {

            // Hash password before storing
            String hashedPassword = BCrypt.hashpw(user.getPassword(), BCrypt.gensalt());

            pstmt.setString(1, user.getUsername());
            pstmt.setString(2, hashedPassword);
            pstmt.setString(3, user.getFullName());
            pstmt.setString(4, user.getRole());

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

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

    /**
     * Map ResultSet to User object
     */
    private User mapResultSetToUser(ResultSet rs) throws SQLException {
        User user = new User();
        user.setId(rs.getInt("id"));
        user.setUsername(rs.getString("username"));
        user.setPassword(rs.getString("password"));
        user.setFullName(rs.getString("full_name"));
        user.setRole(rs.getString("role"));
        user.setActive(rs.getBoolean("is_active"));
        user.setCreatedAt(rs.getTimestamp("created_at"));
        user.setLastLogin(rs.getTimestamp("last_login"));
        return user;
    }

    /**
     * Test method - Generate hashed password
     */
    public static void main(String[] args) {
        // Generate hash for "password123"
        String plainPassword = "password123";
        String hashedPassword = BCrypt.hashpw(plainPassword, BCrypt.gensalt());
        System.out.println("Plain: " + plainPassword);
        System.out.println("Hashed: " + hashedPassword);

        // Test verification
        boolean matches = BCrypt.checkpw(plainPassword, hashedPassword);
        System.out.println("Verification: " + matches);
    }
}
```

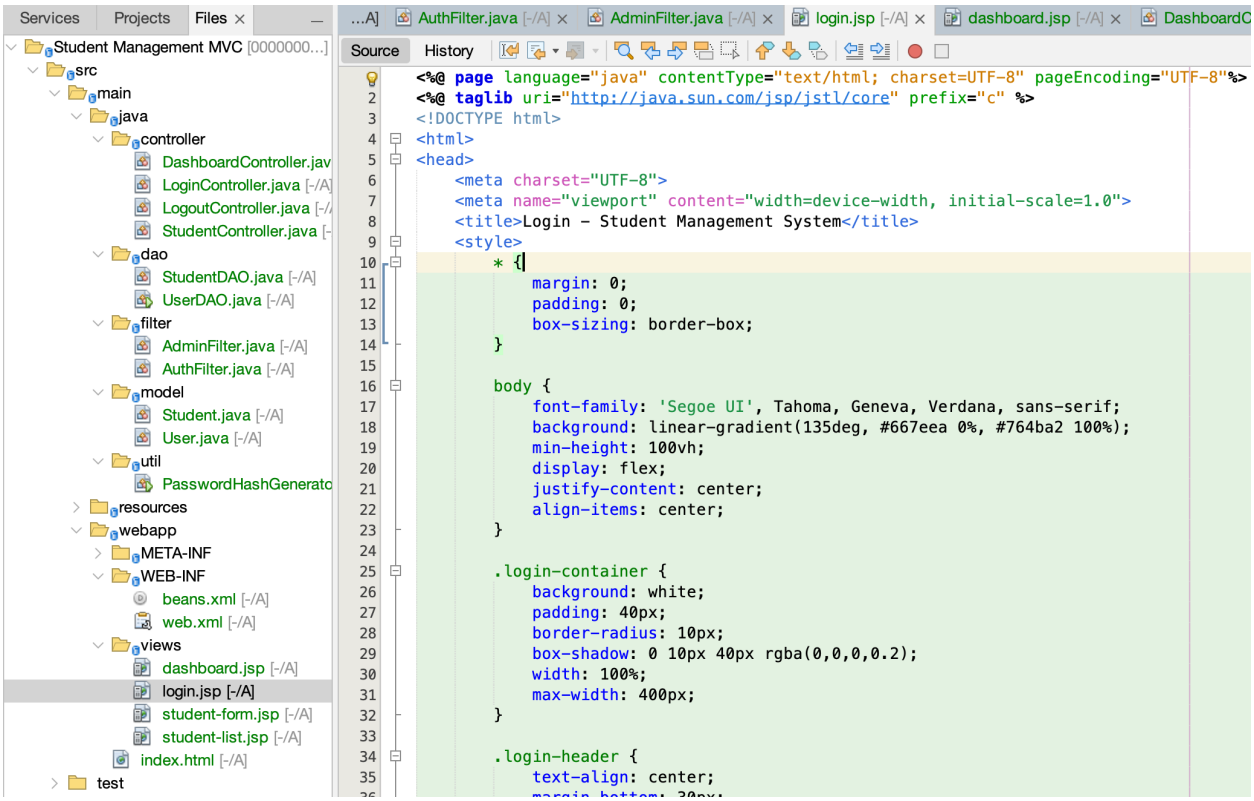
3.1


Services
Projects
Files
Student Management MVC [0000000...]
src
main
java
controller
DashboardController.jav
LoginController.java [-/A]
LogoutController.java [-/
StudentController.java [-
dao
StudentDAO.java [-/A]
UserDAO.java [-/A]
filter
AdminFilter.java [-/A]
AuthFilter.java [-/A]
model
Student.java [-/A]
User.java [-/A]
util
PasswordHashGenerato
resources
webapp
META-INF
WEB-INF
beans.xml [-/A]
web.xml [-/A]
views
index.html [-/A]
test
target
Lab 5_Homework_Report.pdf [-/A]
nb-configuration.xml [-/A]
pom.xml [-/A]
...A]
User.java [-/A]
UserDAO.java [-/A]
LoginController.java [-/A]
LogoutController.java [-/A]
Source
History
package controller;
import dao.UserDAO;
import model.User;
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;
@WebServlet("/login")
public class LoginController extends HttpServlet {
private UserDAO userDAO;
@Override
public void init() {
userDAO = new UserDAO();
}
/**
* Display login page
*/
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
// If already logged in, redirect to dashboard
HttpSession session = request.getSession(false);
if (session != null && session.getAttribute("user") != null) {
response.sendRedirect("dashboard");
return;

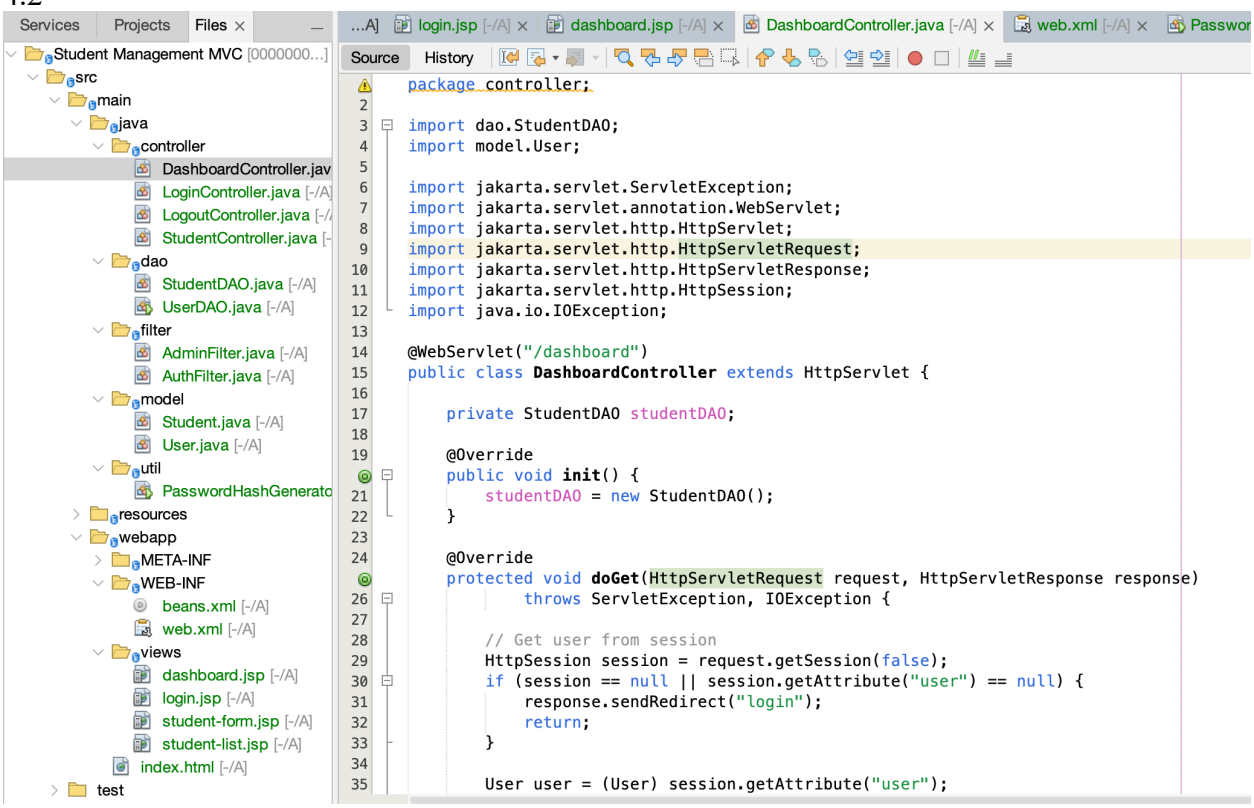
3.2

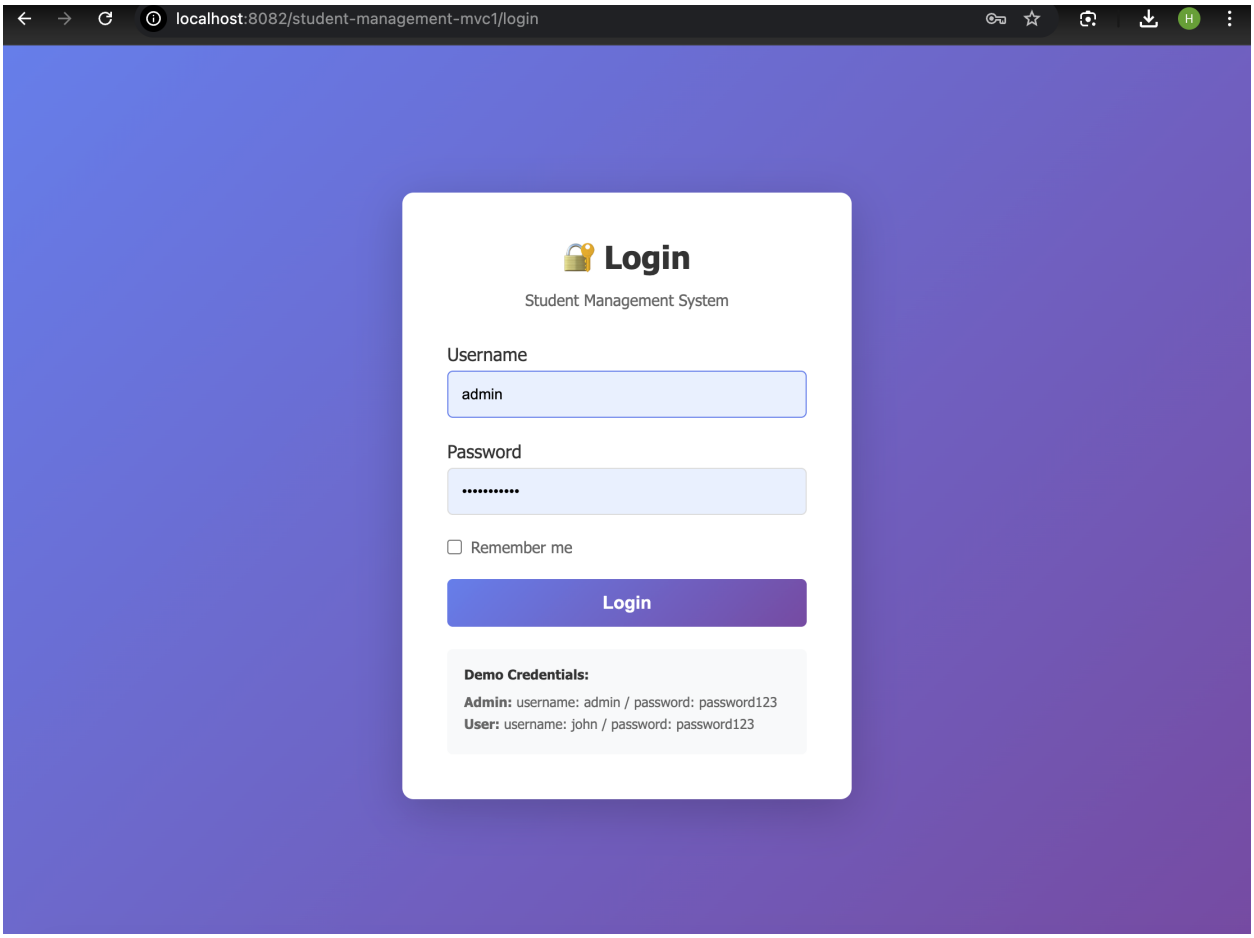

Services
Projects
Files
Student Management MVC [0000000...]
src
main
java
controller
DashboardController.jav
LoginController.java [-/A]
LogoutController.java [-/
StudentController.java [-
dao
StudentDAO.java [-/A]
UserDAO.java [-/A]
filter
AdminFilter.java [-/A]
AuthFilter.java [-/A]
model
Student.java [-/A]
User.java [-/A]
util
PasswordHashGenerato
resources
webapp
META-INF
WEB-INF
beans.xml [-/A]
web.xml [-/A]
views
index.html [-/A]
test
target
Lab 5_Homework_Report.pdf [-/A]
nb-configuration.xml [-/A]
pom.xml [-/A]
...A]
User.java [-/A]
UserDAO.java [-/A]
LoginController.java [-/A]
LogoutController.java [-/A]
Source
History
ackage controller;
mport jakarta.servlet.ServletException;
mport jakarta.servlet.annotation.WebServlet;
mport jakarta.servlet.http.HttpServlet;
mport jakarta.servlet.http.HttpServletRequest;
mport jakarta.servlet.http.HttpServletResponse;
mport jakarta.servlet.http.HttpSession;
mport java.io.IOException;
WebServlet("/logout")
ublic class LogoutController extends HttpServlet {
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
// Get current session
HttpSession session = request.getSession(false);
if (session != null) {
// Invalidate session
session.invalidate();
}
// Redirect to login page with message
response.sendRedirect("login?message=You have been logged out successfully");
}
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
doGet(request, response);
}

4.1

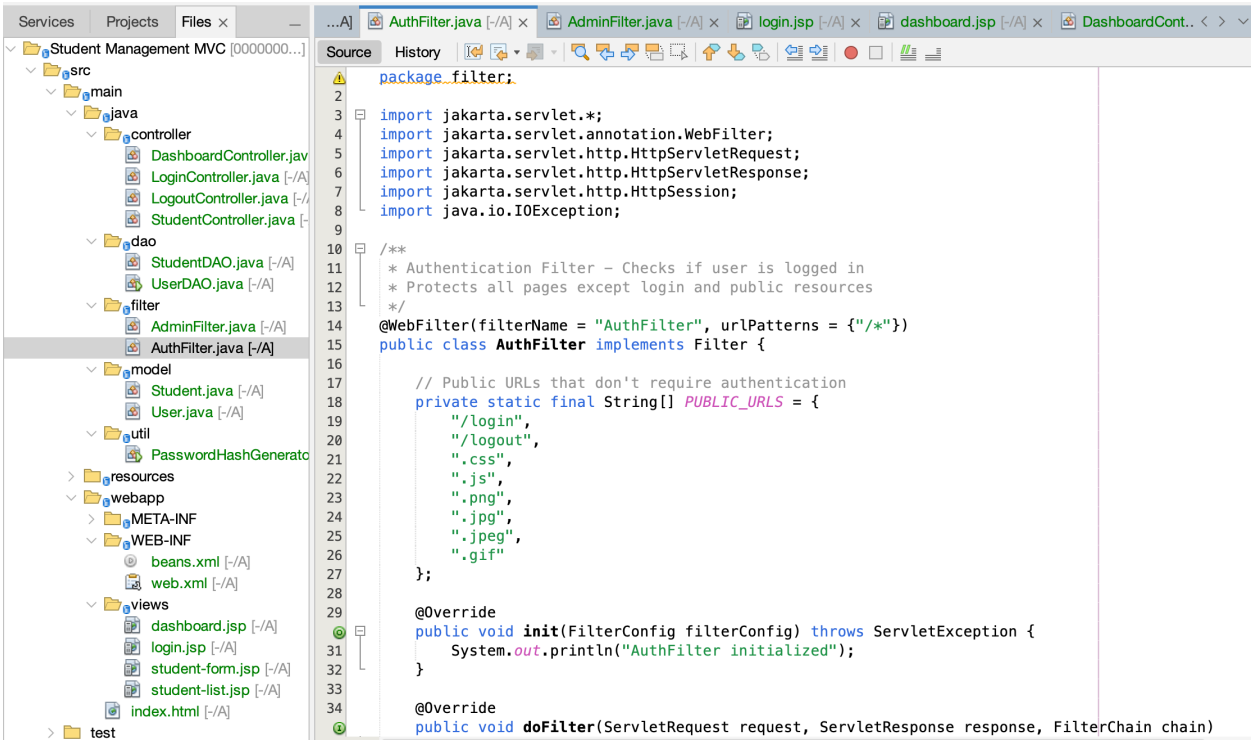

Services
Projects
Files
Student Management MVC [0000000...]
src
main
java
controller
DashboardController.jav
LoginController.java [-/A]
LogoutController.java [-/
StudentController.java [-
dao
StudentDAO.java [-/A]
UserDAO.java [-/A]
filter
AdminFilter.java [-/A]
AuthFilter.java [-/A]
model
Student.java [-/A]
User.java [-/A]
util
PasswordHashGenerato
resources
webapp
META-INF
WEB-INF
beans.xml [-/A]
web.xml [-/A]
views
dashboard.jsp [-/A]
login.jsp [-/A]
student-form.jsp [-/A]
student-list.jsp [-/A]
index.html [-/A]
test
...A]
AuthFilter.java [-/A]
AdminFilter.java [-/A]
login.jsp [-/A]
dashboard.jsp [-/A]
DashboardC
Source
History
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Login - Student Management System</title>
<style>
* {
margin: 0;
padding: 0;
box-sizing: border-box;
}
body {
font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
min-height: 100vh;
display: flex;
justify-content: center;
align-items: center;
}
.login-container {
background: white;
padding: 40px;
border-radius: 10px;
box-shadow: 0 10px 40px rgba(0,0,0,0.2);
width: 100%;
max-width: 400px;
}
.login-header {
text-align: center;

4.2

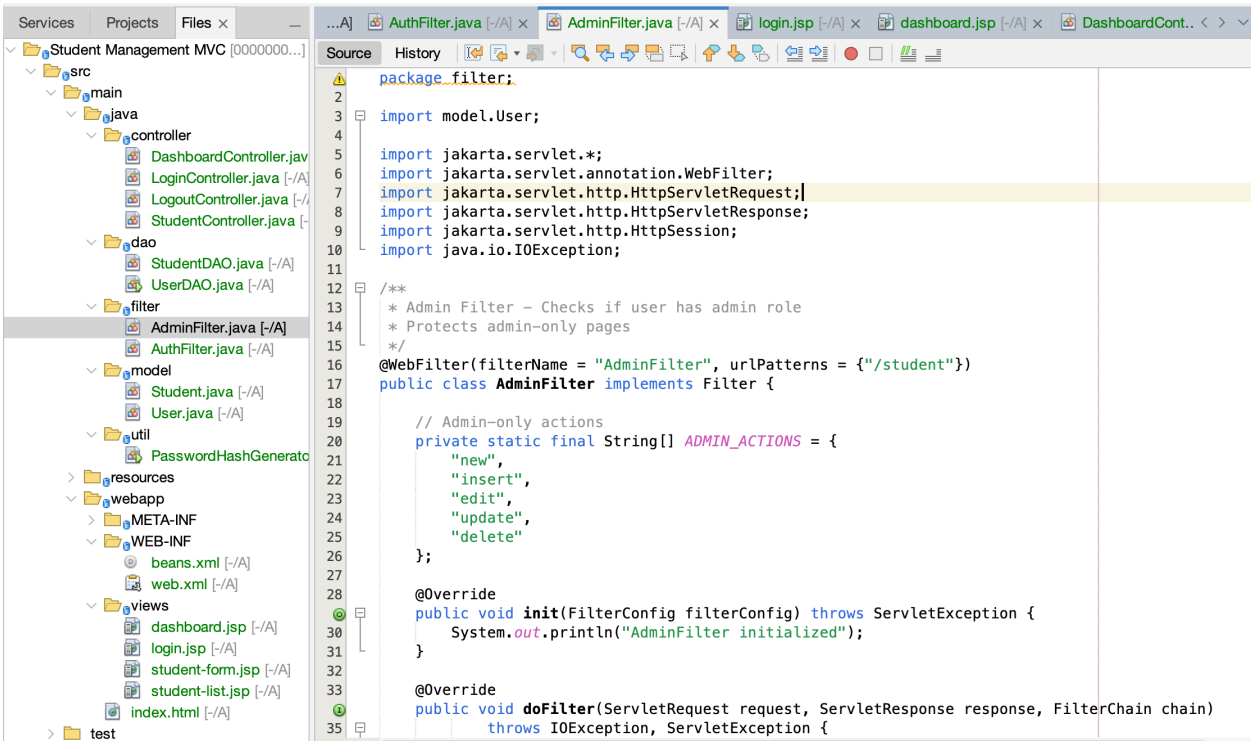

Services
Projects
Files
Student Management MVC [0000000...]
src
main
java
controller
DashboardController.jav
LoginController.java [-/A]
LogoutController.java [-/
StudentController.java [-
dao
StudentDAO.java [-/A]
UserDAO.java [-/A]
filter
AdminFilter.java [-/A]
AuthFilter.java [-/A]
model
Student.java [-/A]
User.java [-/A]
util
PasswordHashGenerato
resources
webapp
META-INF
WEB-INF
beans.xml [-/A]
web.xml [-/A]
views
dashboard.jsp [-/A]
login.jsp [-/A]
student-form.jsp [-/A]
student-list.jsp [-/A]
index.html [-/A]
test
...A]
login.jsp [-/A]
dashboard.jsp [-/A]
DashboardController.java [-/A]
web.xml [-/A]
Passwor
Source
History
package controller;
import dao.StudentDAO;
import model.User;
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;
@WebServlet("/dashboard")
public class DashboardController extends HttpServlet {
private StudentDAO studentDAO;
@Override
public void init() {
studentDAO = new StudentDAO();
}
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
// Get 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");


localhost:8082/student-management-mvc1/login
Login
Student Management System
Username
admin
Password
Remember me
Login
Demo Credentials:
Admin: username: admin / password: password123
User: username: john / password: password123

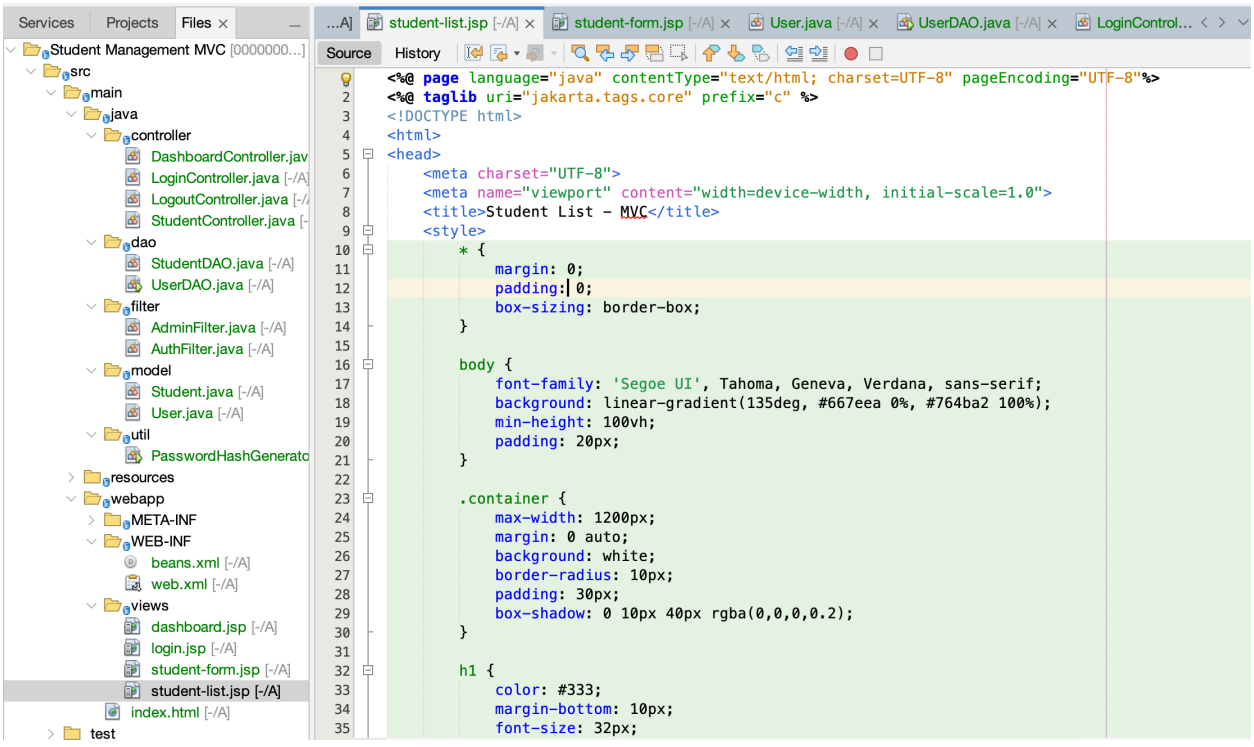
5.1


Services
Projects
Files
Student Management MVC
src
main
java
controller
DashboardController.jav
LoginController.java
LogoutController.java
StudentController.java
dao
StudentDAO.java
UserDAO.java
filter
AdminFilter.java
AuthFilter.java
model
Student.java
User.java
util
PasswordHashGenerato
resources
webapp
META-INF
WEB-INF
beans.xml
web.xml
views
dashboard.jsp
login.jsp
student-form.jsp
student-list.jsp
index.html
test
AuthFilter.java
AdminFilter.java
login.jsp
dashboard.jsp
DashboardCont..
Source
History
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");
}
@Override
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)

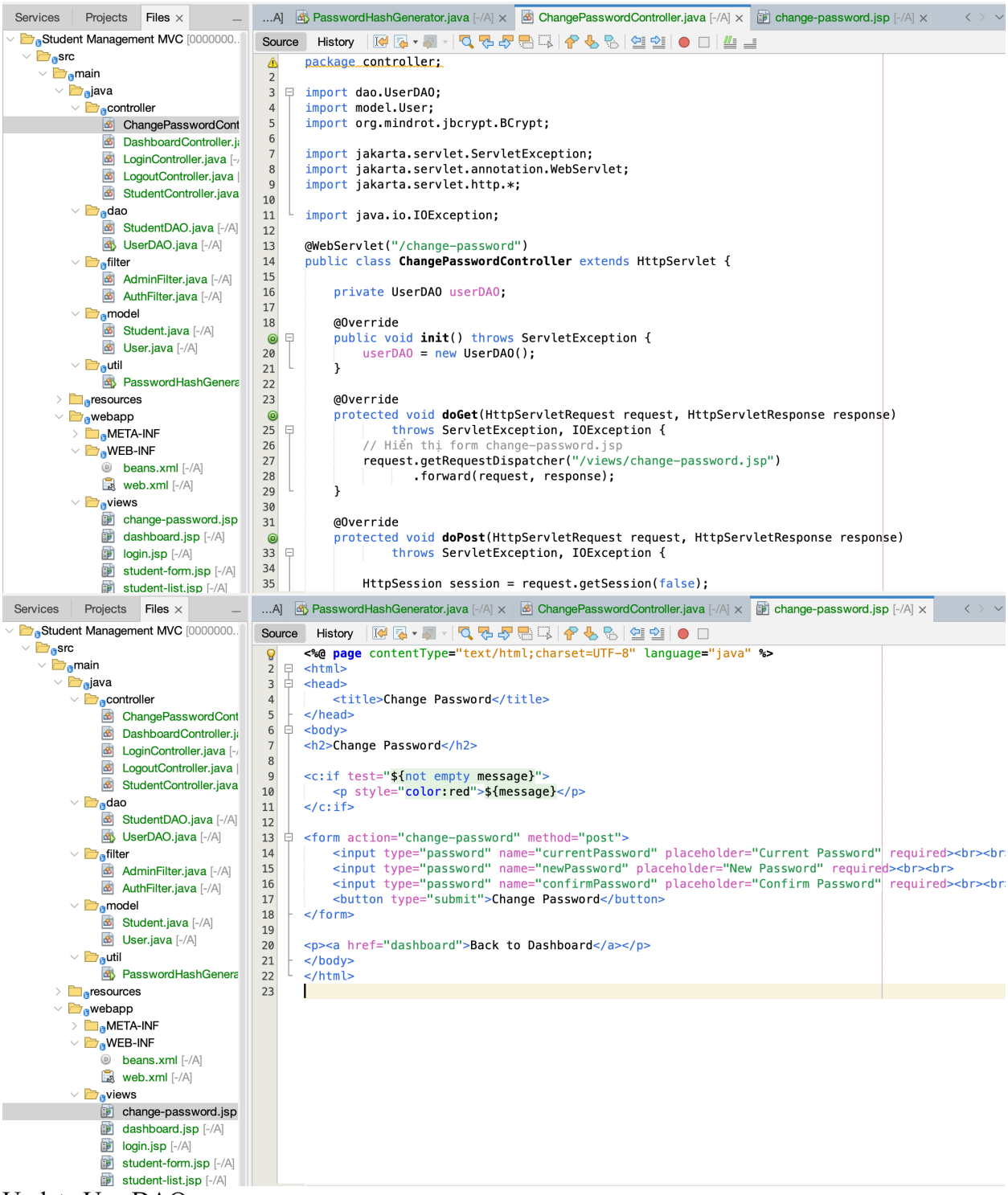
6.1


Services
Projects
Files
Student Management MVC [0000000...]
src
main
java
controller
DashboardController.jav
LoginController.java [-/A]
LogoutController.java [-/
StudentController.java [-
dao
StudentDAO.java [-/A]
UserDAO.java [-/A]
filter
AdminFilter.java [-/A]
AuthFilter.java [-/A]
model
Student.java [-/A]
User.java [-/A]
util
PasswordHashGenerato
resources
webapp
META-INF
WEB-INF
beans.xml [-/A]
web.xml [-/A]
views
dashboard.jsp [-/A]
login.jsp [-/A]
student-form.jsp [-/A]
student-list.jsp [-/A]
index.html [-/A]
test
...A]
AuthFilter.java [-/A]
AdminFilter.java [-/A]
login.jsp [-/A]
dashboard.jsp [-/A]
DashboardCont..
Source
History
package filter;
import 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"
};
@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 {

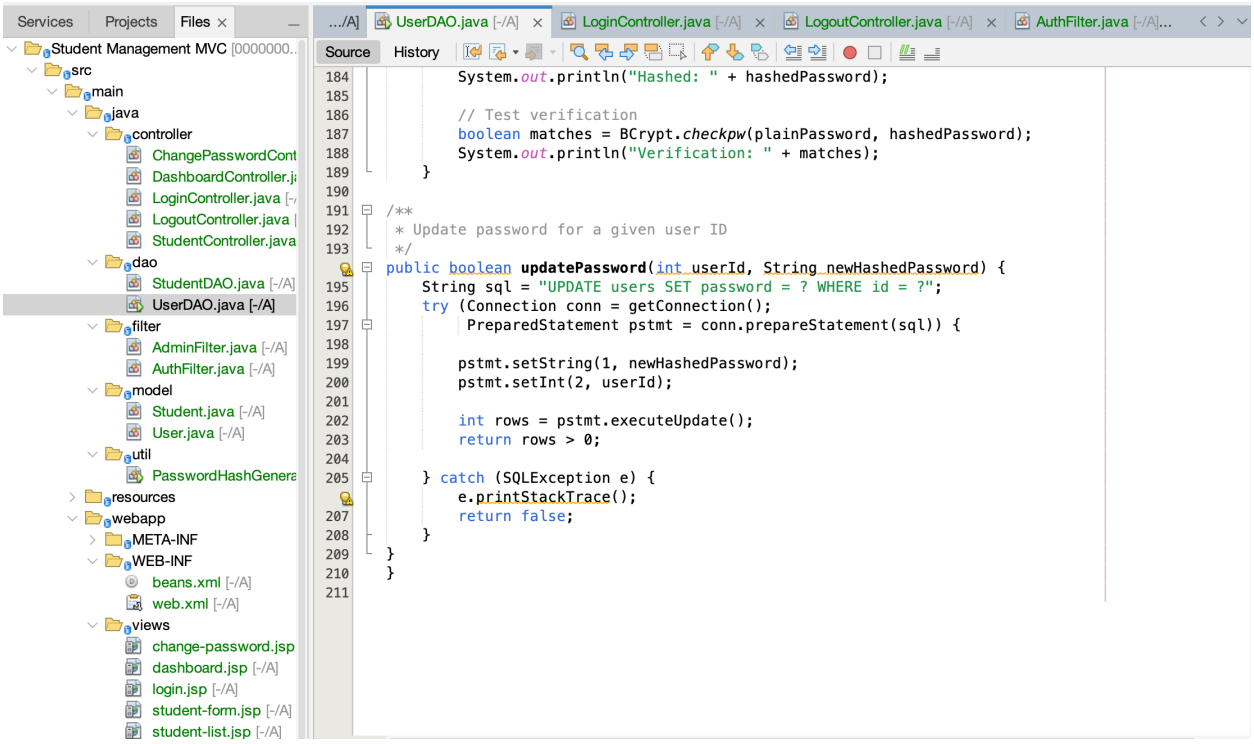
7.1


Services
Projects
Files
Student Management MVC [0000000...]
src
main
java
controller
DashboardController.jav
LoginController.java [-/A]
LogoutController.java [-/
StudentController.java [-
dao
StudentDAO.java [-/A]
UserDAO.java [-/A]
filter
AdminFilter.java [-/A]
AuthFilter.java [-/A]
model
Student.java [-/A]
User.java [-/A]
util
PasswordHashGenerato
resources
webapp
META-INF
WEB-INF
beans.xml [-/A]
web.xml [-/A]
views
dashboard.jsp [-/A]
login.jsp [-/A]
student-form.jsp [-/A]
student-list.jsp [-/A]
index.html [-/A]
test
...A]
student-list.jsp [-/A]
student-form.jsp [-/A]
User.java [-/A]
UserDAO.java [-/A]
LoginControl...
Source
History
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="jakarta.tags.core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Student List - MVC</title>
<style>
* {
margin: 0;
padding: 0;
box-sizing: border-box;
}
body {
font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
min-height: 100vh;
padding: 20px;
}
.container {
max-width: 1200px;
margin: 0 auto;
background: white;
border-radius: 10px;
padding: 30px;
box-shadow: 0 10px 40px rgba(0,0,0,0.2);
}
h1 {
color: #333;
margin-bottom: 10px;
font-size: 32px;

8.1



Update UserDAO


Services
Projects
Files
Student Management MVC [0000000..
src
main
java
controller
ChangePasswordCont
DashboardController.j
LoginController.java [-/
LogoutController.java
StudentController.java
dao
StudentDAO.java [-/A]
UserDAO.java [-/A]
filter
AdminFilter.java [-/A]
AuthFilter.java [-/A]
model
Student.java [-/A]
User.java [-/A]
util
PasswordHashGenera
resources
webapp
META-INF
WEB-INF
beans.xml [-/A]
web.xml [-/A]
views
change-password.jsp
dashboard.jsp [-/A]
login.jsp [-/A]
student-form.jsp [-/A]
student-list.jsp [-/A]
.../A]
UserDAO.java [-/A]
LoginController.java [-/A]
LogoutController.java [-/A]
AuthFilter.java [-/A]...
Source
History
System.out.println("Hashed: " + hashedPassword);
// Test verification
boolean matches = BCrypt.checkpw(plainPassword, hashedPassword);
System.out.println("Verification: " + matches);
}
/**
* Update password for a given user ID
*/
public boolean updatePassword(int userId, String newHashedPassword) {
String sql = "UPDATE users SET password = ? WHERE id = ?";
try (Connection conn = getConnection();
PreparedStatement pstmt = conn.prepareStatement(sql)) {
pstmt.setString(1, newHashedPassword);
pstmt.setInt(2, userId);
int rows = pstmt.executeUpdate();
return rows > 0;
} catch (SQLException e) {
e.printStackTrace();
return false;
}
}
}