

Web Application Development Lab 06

AUTHENTICATION & SESSION MANAGEMENT

Part A – INCLASS PRACTICE & EXERCISES

I. DATABASE & USER MODEL

The screenshot shows the MySQL Workbench interface. In the Navigator pane, the 'student_management' schema is selected, showing its tables ('Tables'), views ('Views'), stored procedures ('Stored Procedures'), and functions ('Functions'). The 'Query 1' editor contains the SQL query: 'SELECT * FROM users'. The Result Grid displays the following data:

	id	username	password	full_name	role	is_active	created_at	last_login
1	admin	\$2a\$10\$kjvJQbfFeExgW3Be.OoLJSfrp.CDogP...	Admin User	admin	1	2025-11-22 09:24:23	2025-11-22 09:56:50	NULL
2	john	\$2a\$10\$kjvJQbfFeExgW3Be.OoLJSfrp.CDogP...	John Doe	user	1	2025-11-22 09:24:23	2025-11-22 09:52:36	NULL
3	jane	\$2a\$10\$F4BmgSDxujBg3n8EBmUOyOhmh8tRc...	Jane Smith	user	1	2025-11-22 09:24:23	2025-11-22 09:52:36	NULL

The Output pane shows the execution of the query: 'SELECT * FROM users LIMIT 0, 1000'. The message indicates '3 row(s) returned'.

II. USER MODEL & DAO

1. User model

```
package com.student.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;  
}  
  
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;  
    }  
  
    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. User DAO

```
package com.student.dao;

/**
 *
 * @author Admin
 */

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

import java.sql.*;

public class UserDAO {
```

```
private static final String DB_URL = "jdbc:mysql://localhost:3306/student_management";  
private static final String DB_USER = "lvp";  
private static final String DB_PASSWORD = "050904";  
  
// 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
 * @param username
 * @param 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
 * @param id
*/
```

```
* @return
*/
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  
     */  
  
    public static void main(String[] args) {  
        // Generate hash for "password123"  
  
        String plainPassword = "thisIsASecretPassword";  
        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);  
    }  
}
```

Authentication testing:

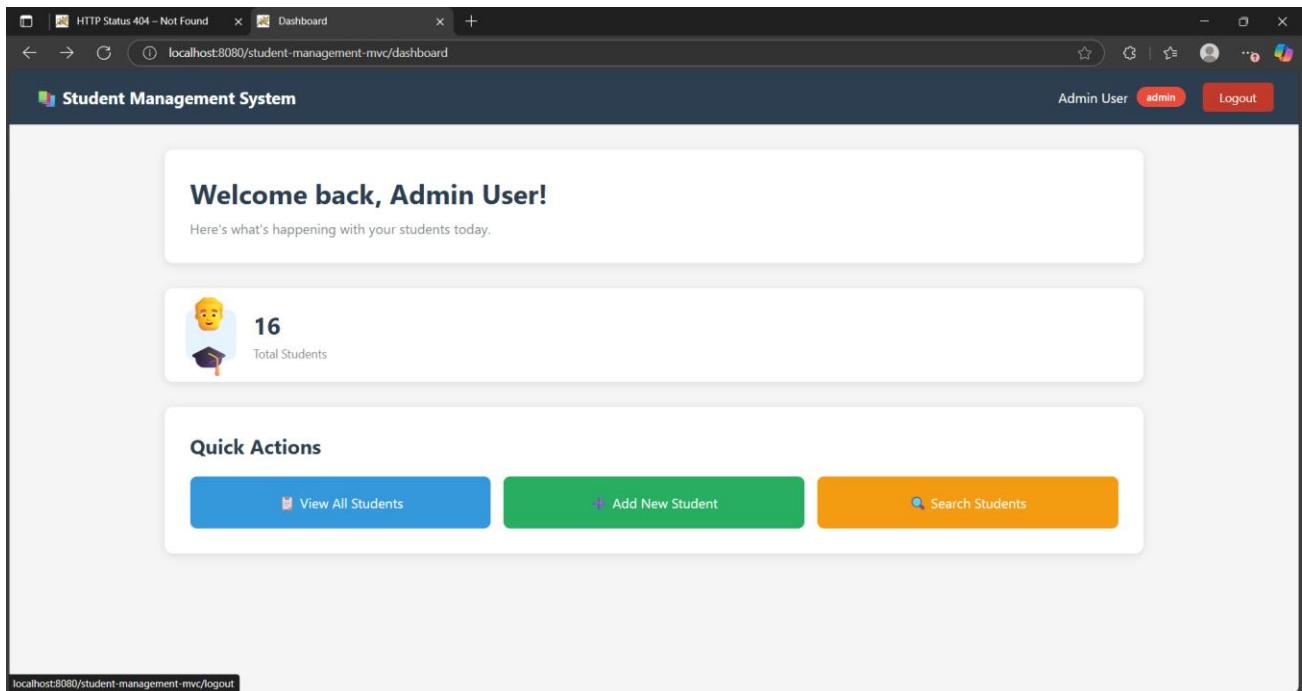
```
--- exec:3.1.0:exec (default-cli) @ student-management-mvc ---  
Authentication successful!  
User{id=1, username='admin', fullName='Admin User', role='admin', isActive=true}  
Invalid auth: Correctly rejected  
-----
```

First line: - Testing a valid authentication

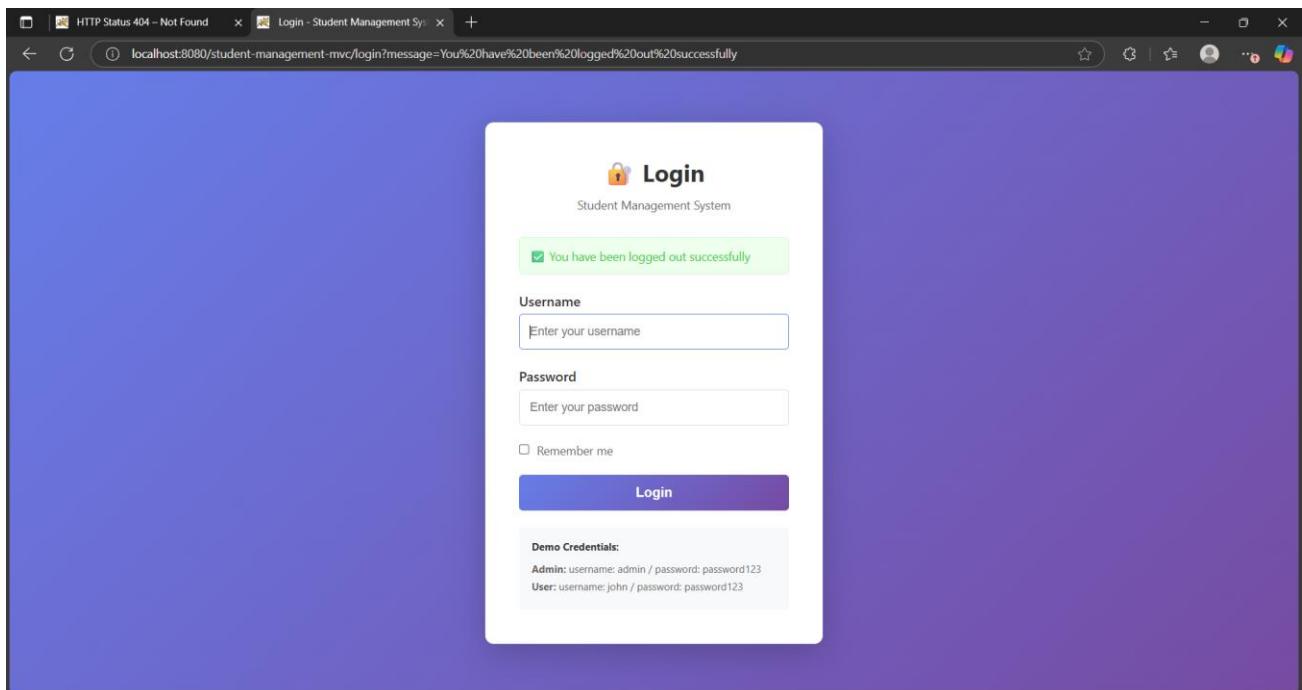
Second line: - Printing the user information

Third line: - Testing an invalid authentication

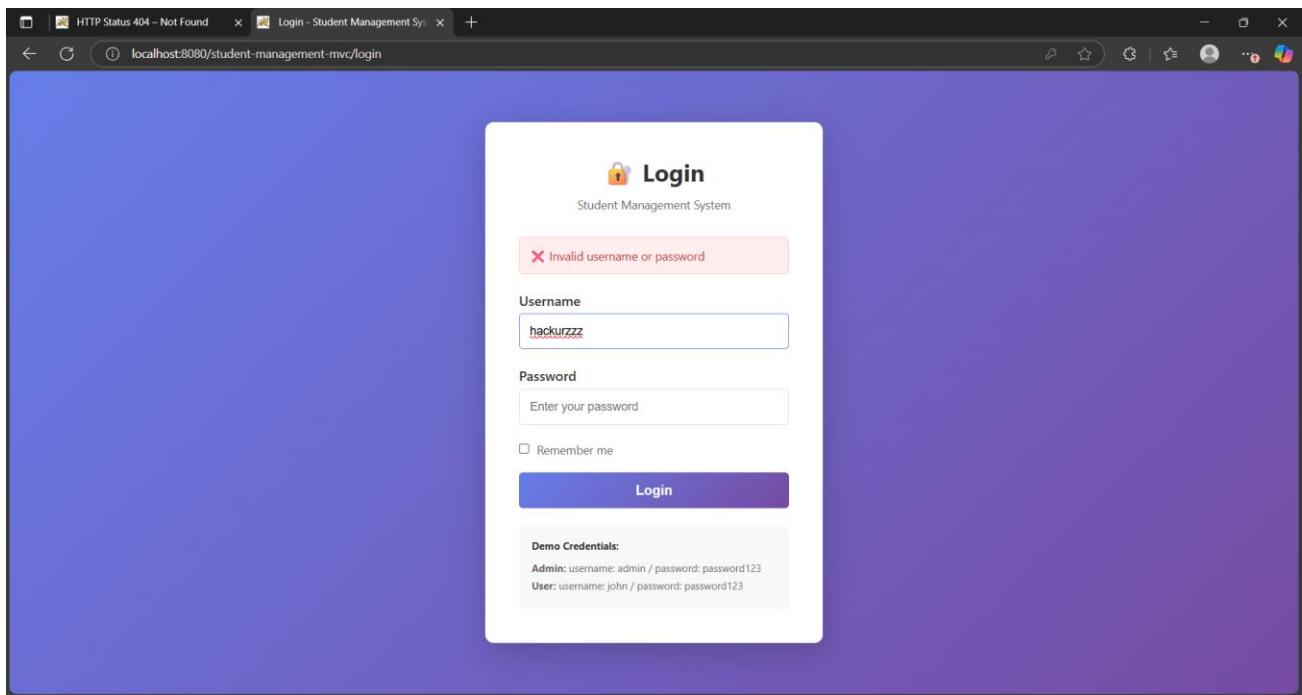
III. LOGIN/LOGOUT CONTROLLERS



Logging out



Logging out successfully



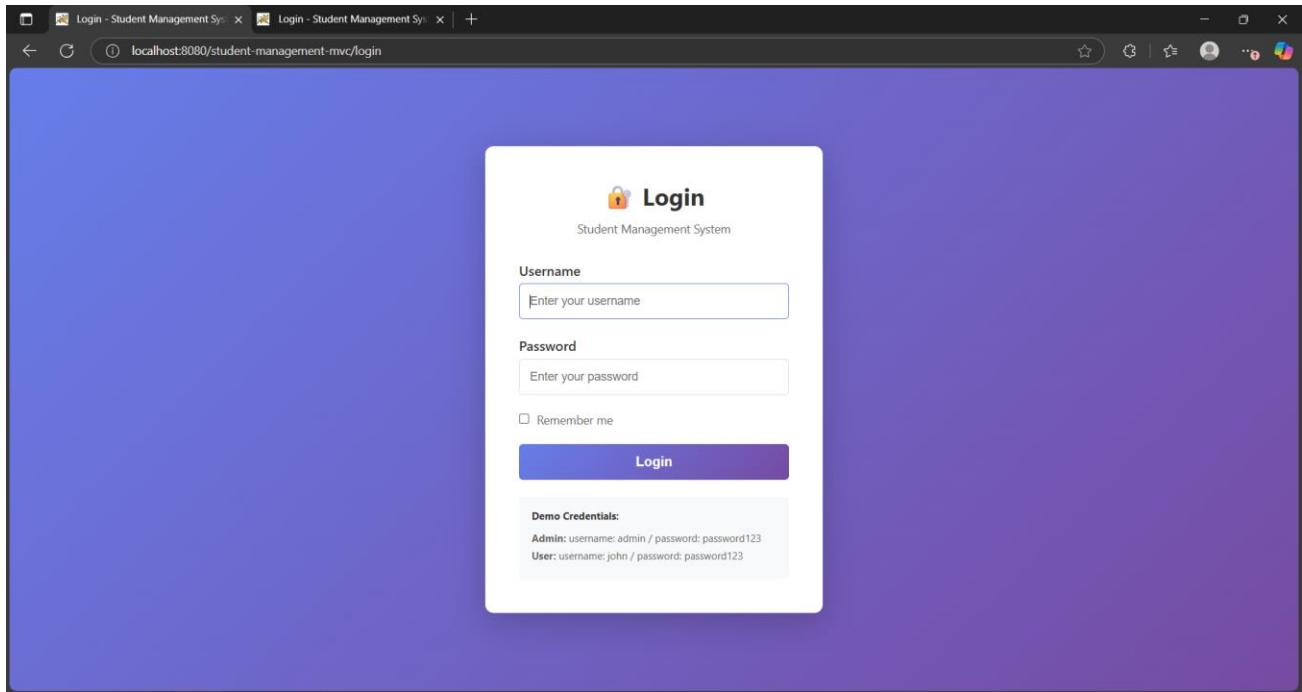
Error message

*Issue:

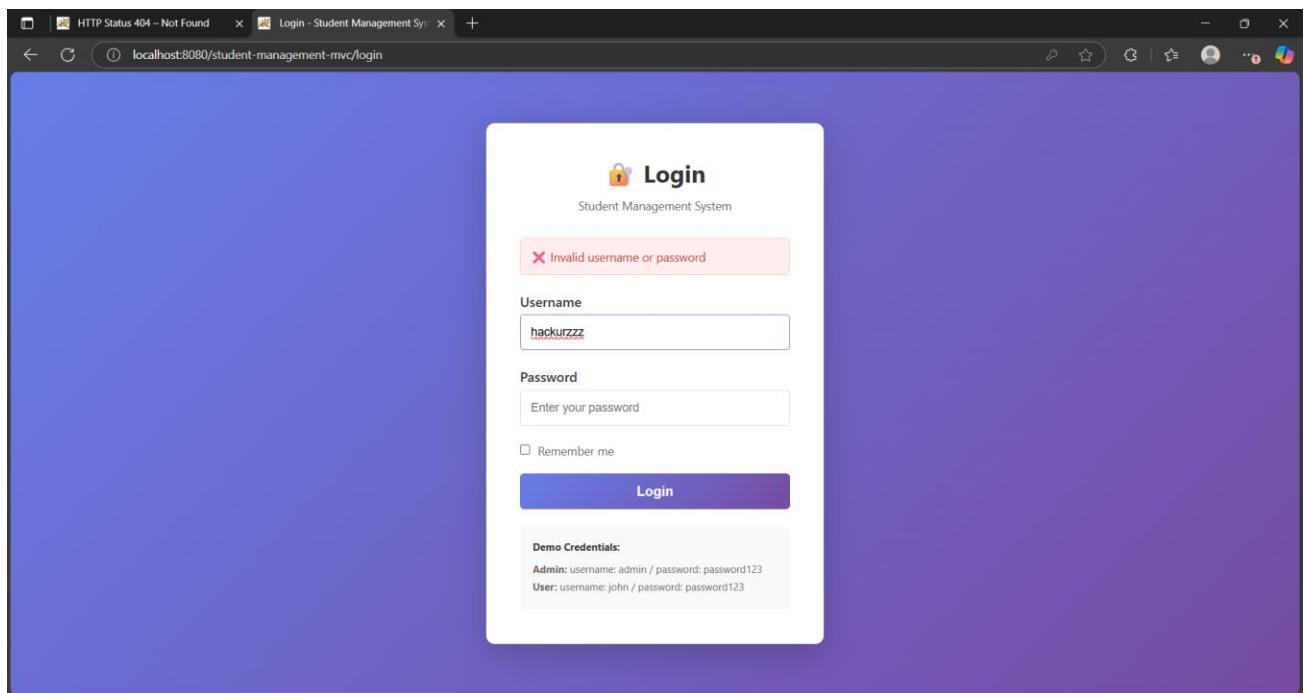
- Login also takes space ‘ ‘ into account when validating users, even though trim() is applied [BUG]. (This could be that the string is not padded with traditional white space)

IV. VIEWS & DASHBOARD

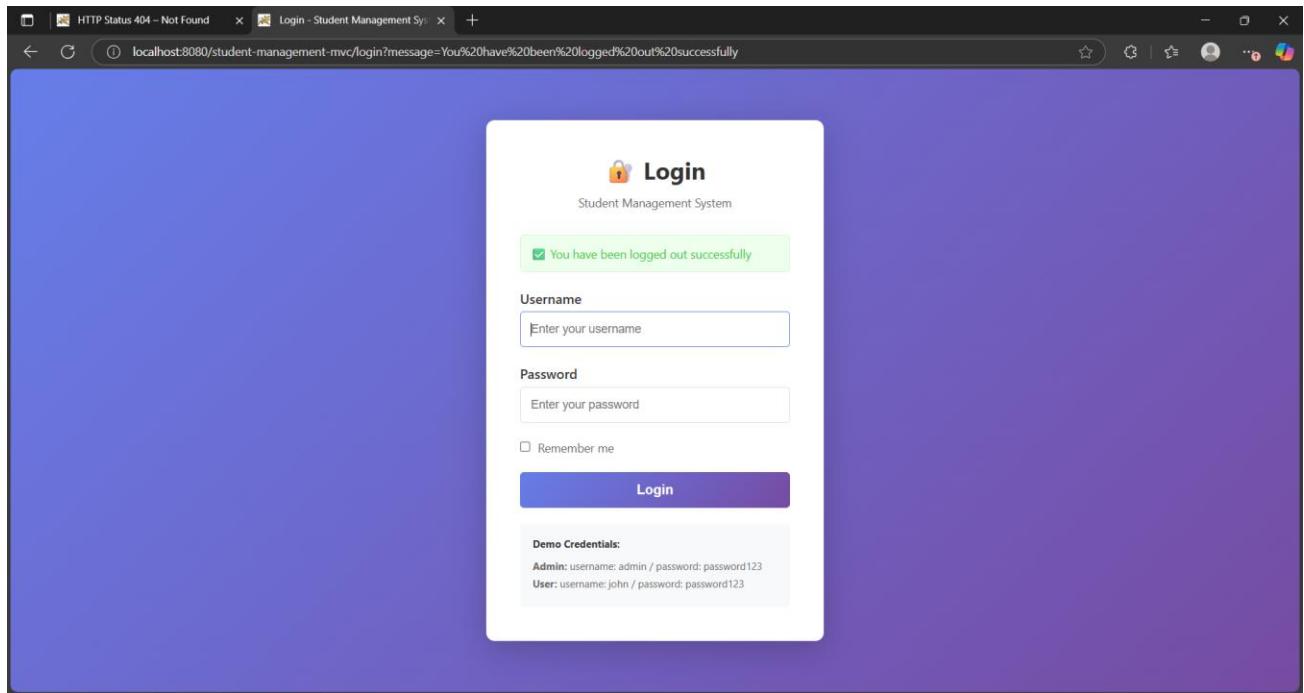
1. Login



Default login page



Error message



Logs out successfully

2. Dashboard

A screenshot of a web browser window showing the 'Dashboard' of the 'Student Management System'. The address bar shows 'localhost:8080/student-management-mvc/dashboard'. The main header is 'Student Management System'. A large white box displays 'Welcome back, Admin User!' and a message 'Here's what's happening with your students today.' Below this is a stats card with a graduation cap icon, '16' (Total Students), and a 'Logout' button. A modal window titled 'Save your password?' is open, asking if the password will be saved to a Microsoft account. It has fields for 'Username' (admin) and 'Password' (redacted), and buttons for 'Save' and 'Not now'. At the bottom, there is a 'Quick Actions' section with three buttons: 'View All Students' (blue), 'Add New Student' (green), and 'Search Students' (orange).

Default dashboard

The screenshot shows a web application titled "Student Management System" running on a local host. The interface includes a header with the title and a subtitle "MVC Pattern with Jakarta EE & JSTL". Below the header is a navigation bar with buttons for "Add New Student" and "Export to Excel". There is also a search bar with fields for "Search..." and "All Majors", and a "Search" button.

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	Johning Science	<button>Edit</button> <button>Delete</button>
22	JJ001	thisisJohn	john@email.com	Johning Science	<button>Edit</button> <button>Delete</button>

Quick actions take you to /student