FPT Software

JAVA SPRING FRAMEWORK

Lab Guides

Document Code	25e-BM/HR/HDCV/FSOFT
Version	1.0
Effective Date	01/09/2024

Hanoi, 08/2024

RECORD OF CHANGES

No	Effective Date	Change Description	Reason	Reviewer	Approver
1	06/08/2024	Create a new Lab	Create new		VinhNV

Contents

CODE:	JSFW_Lab_05_Opt1
TYPE:	LONG
LOC:	200
DURATION:	180 MINUTES

Java Spring Framework Introduction

Objectives:

- Understand DAO Pattern: Learn how to use the Data Access Object (DAO) pattern to separate data access logic from business logic in a Spring MVC application.
- Configure Spring MVC: Learn to configure and use Spring MVC to manage entities with database interactions.
- Implement Login and Subject Management: Implement and test login functionality and CRUD operations for managing subjects using DAO patterns and PostgreSQL for persistence.

Lab Specifications:

In this lab, you will build a University Management System where the User and Subject entities interact with a PostgreSQL database using DAO classes. You will implement CRUD operations for Subject management and a login feature for user authentication.

Problem Description:

Trainees are required to implement and test the following functionalities:

1. **Login Feature**: Implement a login mechanism that allows users to authenticate and access restricted pages.
2. **Subject Management**: Implement CRUD operations for managing subjects using DAO patterns and PostgreSQL for persistence.

Prerequisites:

- Using Java SDK version 8.0 at least.
- Using Maven.
- Using Spring Framework 5.0 or higher version.

Guidelines:

Step 1: Extend the previous project, add these entity classes:

```
package com.example.model;

import org.springframework.stereotype.Component;

@Component
public class Subject {
```

```
    private int subjectIdd;
    private String subjectName;
    private String description;

// Getters and Setters

    public int getSubjectId() {
        return subjectIdd;
    }

    public void setSubjectId(int subject_id) {
        this.subjectIdd = subject_id;
    }

    public String getSubjectName() {
        return subjectName;
    }

    public void setSubjectName(String subject_name) {
        this.subjectName = subject_name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }


}
```

Step 2: Implement DAO Pattern:

- **Create DAO Interfaces**: Define interfaces for SubjectDAO with methods for CRUD operations.

```
package com.example.dao;


import com.example.model.Subject;
import java.util.List;

public interface SubjectDAO {

    void save(Subject subject);

    void update(Subject subject);

    boolean delete(Integer subjectId);

    Subject findById(Integer subjectId);

    List<Subject> findAll();
}

package com.example.dao;


import com.example.model.Subject;
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Component;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;

@Component
public class SubjectDAOImpl implements SubjectDAO {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    @Override
    public void save(Subject subject) {
        String sql = "INSERT INTO Subjects (subject_name, description) VALUES
(?, ?)";
        jdbcTemplate.update(sql, subject.getSubjectName(),
subject.getDescription());
    }

    @Override
    public void update(Subject subject) {
        String sql = "UPDATE Subjects SET subject name = ?, description = ?
WHERE subject_id = ?";
        jdbcTemplate.update(sql, subject.getSubjectName(),
subject.getDescription(), subject.getSubjectId());
    }

    @Override
    public boolean delete(Integer subjectId) {
        String sql = "DELETE FROM Subjects WHERE subject_id = ?";
        int rowsAffected = jdbcTemplate.update(sql, subjectId);
        return rowsAffected > 0;
    }

    @Override
    public Subject findById(Integer subjectId) {
        String sql = "SELECT * FROM Subjects WHERE subject_id = ?";
        return jdbcTemplate.queryForObject(sql, this::mapRowToSubject,
subjectId);
    }

    @Override
    public List<Subject> findAll() {
        String sql = "SELECT * FROM Subjects";
        return jdbcTemplate.query(sql, this::mapRowToSubject);
    }

    private Subject mapRowToSubject(ResultSet rs, int rowNum) throws
SQLException {
        Subject subject = new Subject();
        subject.setSubjectId(rs.getInt("subject_id"));
        subject.setSubjectName(rs.getString("subject_name"));
        subject.setDescription(rs.getString("description"));
        return subject;
    }
}
```

Update UserDAOImp class with these method for login feature:

```
//for login feature
@Override
```

```
public User findByUsernameAndPassword(String username, String password) {
    String sql = "SELECT * FROM Users WHERE username = ? AND password = ?";
    Role role = new Role();
    try {
        return jdbcTemplate.queryForObject(sql, new Object[]{username,
password}, (rs, rowNum) -> {
            User user = new User();
            user.setUserId(rs.getInt("user_id"));
            user.setUsername(rs.getString("username"));
            user.setPassword(rs.getString("password"));
            user.setEmail(rs.getString("email"));
            user.setFullName(rs.getString("full_name"));

            // Populate Role
            role.setRoleId(rs.getInt("role_id"));
            user.setRole(role); // Assuming Role is set by ID
            return user;
        });
    } catch (Exception e) {
        return null;
    }
}

@Override
public void save(User user) {
    // Retrieve the role_id for the role_name "student" (case-insensitive)
    String getRoleIdSql = "SELECT role_id FROM Role WHERE LOWER(role_name) =
LOWER(?)";
    Integer roleId = jdbcTemplate.queryForObject(getRoleIdSql, new
Object[]{"Student"}, Integer.class);

    // Insert the new user with the retrieved role_id
    String sql = "INSERT INTO Users (username, password, email, full_name,
role_id) VALUES (?, ?, ?, ?, ?)";

    jdbcTemplate.update(sql, user.getUsername(), user.getPassword(),
user.getEmail(), user.getFullName(), roleId);
}
```

Step7: Implement Spring MVC Controllers:

- **HomeController: Handle login and logout operations.**

```
package com.example.controller;


import com.example.dao.UserDAO;
import com.example.model.Role;
import com.example.model.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import javax.servlet.http.HttpSession;

@Controller
public class HomeController {
```

```
    private final UserDAO userDAO;

    @Autowired
    public HomeController(UserDAO userDAO) {
        this.userDAO = userDAO;
    }

    @GetMapping("/")
    public String index() {
        return "index";
    }

    @GetMapping("/signup")
    public String showSignUpForm(Model model) {
        //Integer userId, String username, String password, String email, String
fullName
        User mockUser = new User(null, "test", "test", "test@gmail.com",
"TEST");

        mockUser.setRole(new Role());

        model.addAttribute("user", mockUser);
        return "user/signup";
    }

    @PostMapping("/signup")
    public String signUp(@ModelAttribute("user") User user, RedirectAttributes
redirectAttributes) {
        try {
            userDAO.save(user);
            redirectAttributes.addFlashAttribute("message", "User registered
successfully! Please log in.");
            return "redirect:/login";
        } catch (Exception e) {
            redirectAttributes.addFlashAttribute("error", "An error occurred
during registration. Please try again.");
            return "redirect:/signup";
        }
    }

    @GetMapping("/login")
    public String showLoginForm(Model model) {
        model.addAttribute("user", new User()); // Bind form data
        return "login";
    }

    @PostMapping("/login")
    public String login(@ModelAttribute("user") User user, HttpSession session,
RedirectAttributes redirectAttributes) {
        User existingUser =
userDAO.findByUsernameAndPassword(user.getUsername(), user.getPassword());

        if (existingUser != null) {
            session.setAttribute("loggedInUser", existingUser);
            return "redirect:/";
        } else {
            redirectAttributes.addFlashAttribute("error", "Invalid username or
password. Please try again.");
            return "redirect:/login";
        }
    }

    @GetMapping("/logout")
    public String logout(HttpSession session) {
```

```
        session.invalidate();
        return "redirect:/login";
    }
}
```

SubjectController: Handle CRUD operations for Subject.

```
package com.example.controller;

import com.example.dao.SubjectDAO;
import com.example.model.Subject;
import com.example.model.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import javax.servlet.http.HttpSession;
import java.util.List;

@Controller
public class SubjectController {

    private final SubjectDAO subjectDAO;

    @Autowired
    public SubjectController(SubjectDAO subjectDAO) {
        this.subjectDAO = subjectDAO;
    }

    private boolean isLoggedIn(HttpSession session) {
        User loggedInUser = (User) session.getAttribute("loggedInUser");
        return loggedInUser != null;
    }

    @GetMapping("/subjects")
    public String listSubjects(Model model, HttpSession session,
RedirectAttributes redirectAttributes) {
        if (!isLoggedIn(session)) {
            //redirectAttributes.addFlashAttribute("error", "You must be logged
in to access this page.");
            return "redirect:/login";
        }

        List<Subject> subjects = subjectDAO.findAll();
        model.addAttribute("subjects", subjects);
        return "subject/subjectList";
    }

    @GetMapping("subject/create")
    public String showCreateForm(Model model, HttpSession session,
RedirectAttributes redirectAttributes) {
        if (!isLoggedIn(session)) {
            redirectAttributes.addFlashAttribute("error", "You must be logged in
to access this page.");
            return "redirect:/login";
        }

        model.addAttribute("subject", new Subject());
        return "subject/createSubject";
    }
```

```
    @PostMapping("subject/create")
    public String createSubject(@ModelAttribute Subject subject,
RedirectAttributes redirectAttributes, HttpSession session) {
        if (!isLoggedIn(session)) {
            redirectAttributes.addFlashAttribute("error", "You must be logged in
to access this page.");
            return "redirect:/login";
        }

        subjectDAO.save(subject);
        redirectAttributes.addFlashAttribute("message", "Subject created
successfully!");
        return "redirect:/subjects";
    }

    @GetMapping("subject/edit/{id}")
    public String showEditForm(@PathVariable("id") Integer id, Model model,
RedirectAttributes redirectAttributes, HttpSession session) {
        if (!isLoggedIn(session)) {
            redirectAttributes.addFlashAttribute("error", "You must be logged in
to access this page.");
            return "redirect:/login";
        }

        Subject subject = subjectDAO.findById(id);
        if (subject != null) {
            model.addAttribute("subject", subject);
            return "subject/editSubject";
        } else {
            redirectAttributes.addFlashAttribute("error", "Subject not found!");
            return "redirect:/subjects";
        }
    }

    @PostMapping("subject/update/{id}")
    public String updateSubject(@PathVariable("id") Integer id, @ModelAttribute
Subject subject, RedirectAttributes redirectAttributes, HttpSession session) {
        if (!isLoggedIn(session)) {
            redirectAttributes.addFlashAttribute("error", "You must be logged in
to access this page.");
            return "redirect:/login";
        }

        subject.setSubjectId(id);
        subjectDAO.update(subject);
        redirectAttributes.addFlashAttribute("message", "Subject updated
successfully!");
        return "redirect:/subjects";
    }

    @GetMapping("subject/delete/{id}")
    public String deleteSubject(@PathVariable("id") Integer id,
RedirectAttributes redirectAttributes, HttpSession session) {
        if (!isLoggedIn(session)) {
            redirectAttributes.addFlashAttribute("error", "You must be logged in
to access this page.");
            return "redirect:/login";
        }

        boolean isDeleted = subjectDAO.delete(id);

        if (isDeleted) {
            redirectAttributes.addFlashAttribute("message", "Subject deleted
successfully!");
```

```
        } else {
            redirectAttributes.addFlashAttribute("error", "Subject not found!");
        }

        return "redirect:/subjects";
    }

    @GetMapping("subject/{id}")
    public String viewSubjectDetail(@PathVariable("id") Integer id, Model model,
RedirectAttributes redirectAttributes, HttpSession session) {
        if (!isLoggedIn(session)) {
            redirectAttributes.addFlashAttribute("error", "You must be logged in
to access this page.");
            return "redirect:/login";
        }

        Subject subject = subjectDAO.findById(id);
        if (subject != null) {
            model.addAttribute("subject", subject);
            return "subject/subjectDetail";
        } else {
            redirectAttributes.addFlashAttribute("error", "Subject not found!");
            return "redirect:/subjects";
        }
    }
}
```

Step 8: Create Thymeleaf Templates:

- **Index (index.html)**

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
<div th:replace="fragments :: navbar"></div>

<div class="container mt-5">
    <h1>Welcome to the System</h1>
    <div class="mt-4">
        <p>Use the navigation bar to access Department and Employee management
functionalities.</p>
    </div>
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.j
s"></script>
</body>
</html>
```

- **Menu (fragments.html)**

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Common Fragments</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
<div th:fragment="navbar">
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <a class="navbar-brand" href="/">  Home</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false"
aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav">
                <li class="nav-item">
                    <a class="nav-link" href="/roles">Role</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/users">User</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/subjects">Subject</a>
                </li>
            </ul>
            <ul class="navbar-nav ms-auto">
                <li class="nav-item">
                    <a class="nav-link" th:href="@{/login}"
th:if="${session.loggedInUser == null}">Login</a>
                    <a class="nav-link" th:href="@{/logout}"
th:if="${session.loggedInUser != null}">Logout</a>
                </li>
            </ul>
        </div>
    </nav>
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.j
s"></script>
</body>
</html>
```

- **Login Page (login.html)**

```


<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-6">
            <div class="card mt-5">
```

```
                <div class="card-header text-center">
                    <h2>Login</h2>
                </div>
                <div class="card-body">
                    <form th:action="@{/login}" method="post">
                        <div th:if="${error}" class="alert alert-danger"
role="alert">
                            <p th:text="${error}">
                        </div>

                        <div class="mb-3">
                            <label for="username" class="form-
label">Username</label>
                            <input type="text" class="form-control"
id="username" name="username" required>
                        </div>

                        <div class="mb-3">
                            <label for="password" class="form-
label">Password</label>
                            <input type="password" class="form-control"
id="password" name="password" required>
                        </div>

                        <div class="d-grid">
                            <button type="submit" class="btn btn-
primary">Login</button>
                        </div>
                    </form>
                </div>
                <div class="card-footer text-center">
                    <p>Don't have an account? <a th:href="@{/signup}">Sign
up</a></p>
                </div>
            </div>
        </div>
    </div>
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.j
s"></script>
</body>
</html>
```

- **Subject List Page (subjectList.html)**

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Subject List</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>

<div th:replace="fragments :: navbar"></div>

<div class="container mt-5">
    <h1 class="mb-4">Subject List</h1>

    <a href="/subject/create" class="btn btn-primary mb-3">Create New
```

```
Subject</a>

    <table class="table table-bordered table-striped">
        <thead class="table-dark">
        <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Description</th>
            <th>Actions</th>
        </tr>
        </thead>
        <tbody>
        <tr th:each="subject : ${subjects}">
            <td th:text="${subject.subjectId}"></td>
            <td th:text="${subject.subjectName}"></td>
            <td th:text="${subject.description}"></td>
            <td>
                <a th:href="@{'/subject/' + ${subject.subjectId}}" class="btn
btn-info btn-sm">View</a>
                <a th:href="@{'/subject/edit/' + ${subject.subjectId}}"
class="btn btn-warning">Edit</a>
                <a th:href="@{'/subject/delete/' + ${subject.subjectId}}"
class="btn btn-danger" onclick="return confirm('Are you sure?');">Delete</a>
            </td>
        </tr>
        </tbody>
    </table>

    <div th:if="${message}" class="alert alert-success mt-4">
        <p th:text="${message}"></p>
    </div>
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.j
s"></script>
</body>
</html>
```

- **Create Subject Page (createSubject.html)**

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Create Subject</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="/">  Home</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-
label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
            <li class="nav-item">
                <a class="nav-link" href="/subjects">Subject List</a>
            </li>
```

```
        </ul>
    </div>
</nav>

<div class="container mt-5">
    <h1>Create Subject</h1>

    <form th:action="@{/subject/create}" th:object="${subject}" method="post">
        <div class="mb-3">
            <label for="subjectName" class="form-label">Name:</label>
            <input type="text" id="subjectName" th:field="*{subjectName}"
class="form-control" required>
        </div>

        <div class="mb-3">
            <label for="description" class="form-label">Description:</label>
            <textarea id="description" th:field="*{description}" class="form-
control" rows="4" required></textarea>
        </div>

        <button type="submit" class="btn btn-primary">Create</button>
        <a href="/subjects" class="btn btn-secondary">Cancel</a>
    </form>
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.j
s"></script>
</body>
</html>
```

- **Edit Subject Page (editSubject.html)**

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Edit Subject</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="/">  Home</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-
label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
            <li class="nav-item">
                <a class="nav-link" href="/subjects">Subject List</a>
            </li>
        </ul>
    </div>
</nav>

<div class="container mt-5">
    <h1>Edit Subject</h1>

    <form th:action="@{/subject/update/{id}(id=${subject.subjectId})}"
```

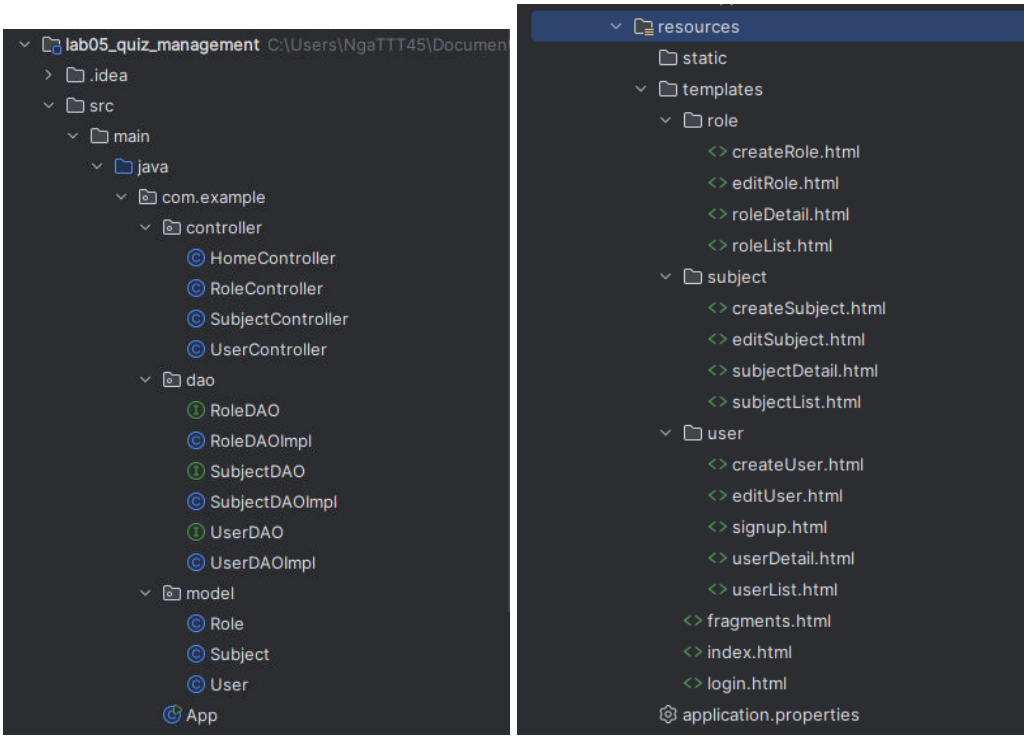
```
th:object="${subject}" method="post">
        <div class="mb-3">
            <label for="subjectName" class="form-label">Name:</label>
            <input type="text" id="subjectName" th:field="*{subjectName}"
class="form-control" required>
        </div>

        <div class="mb-3">
            <label for="description" class="form-label">Description:</label>
            <textarea id="description" th:field="*{description}" class="form-
control" rows="4" required></textarea>
        </div>

        <button type="submit" class="btn btn-primary">Update</button>
        <a href="/subjects" class="btn btn-secondary">Cancel</a>
    </form>
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.j
s"></script>
</body>
</html>
```

- **Subject Detail Page (subjectDetail.html)**

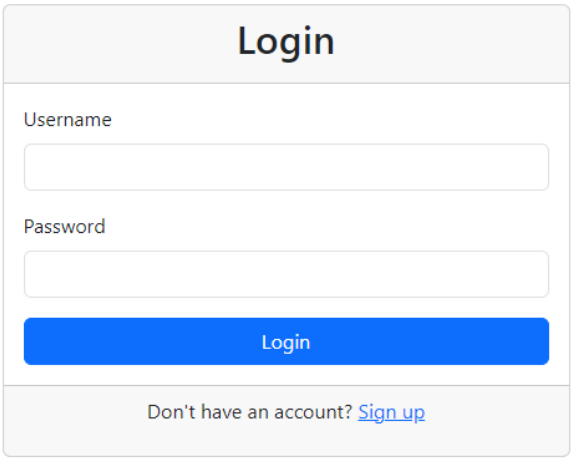
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Subject Details</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="/">  Home</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-
label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
            <li class="nav-item">
                <a class="nav-link" href="/subjects">Subject List</a>
            </li>
        </ul>
    </div>
</nav>

<div class="container mt-5">
    <h1>Subject Details</h1>

    <div class="card mt-3">
        <div class="card-body">
            <p><strong>ID:</strong> <span
th:text="${subject.subjectId}"></span></p>
            <p><strong>Name:</strong> <span
th:text="${subject.subjectName}"></span></p>
            <p><strong>Description:</strong> <span
th:text="${subject.description}"></span></p>
```

```
        </div>
    </div>

    <a href="/subjects" class="btn btn-secondary mt-3">Back to List</a>
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.j
s"></script>
</body>
</html>
```

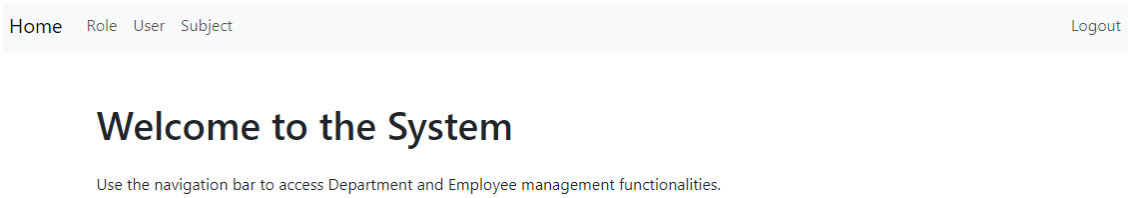
Here is the structure of the program:



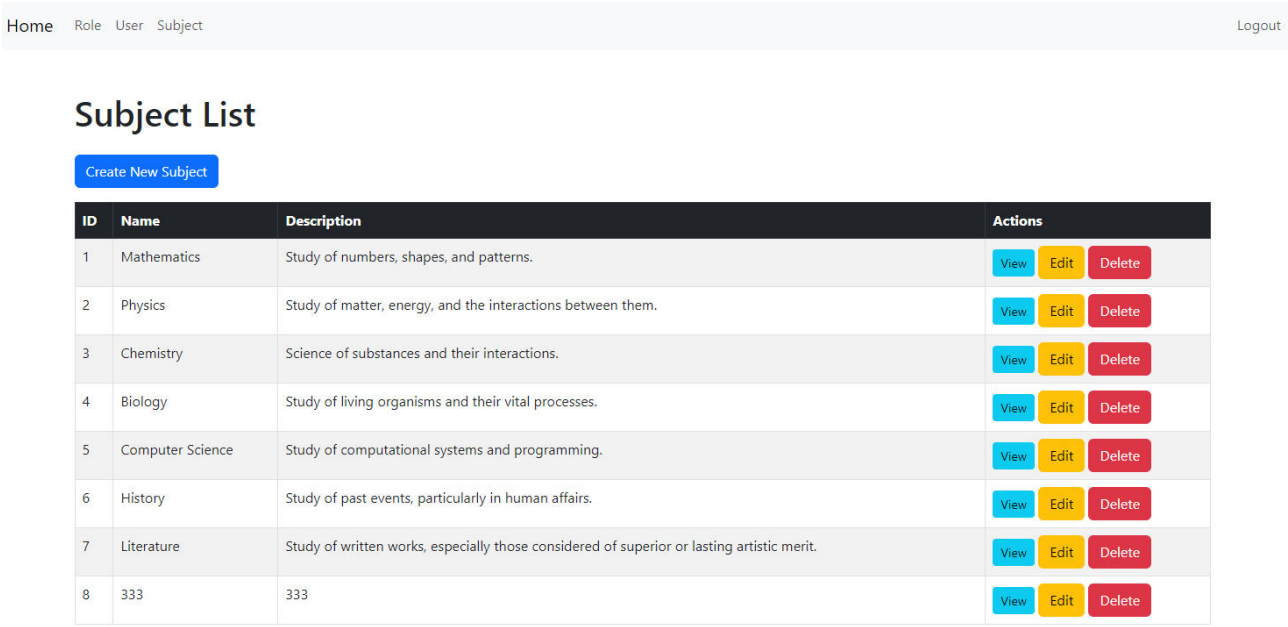
Step 7: Run the application:



After user login, the [Login] menu turn into [Logout]:



When user click to Subject:



This example provides a clear understanding of how Spring MVC organizes the interaction between controllers, data access layers, and models to create a modular and maintainable web application.

----oOo-----

THE END