Documentation: Fitness Web App

Lecturer: Dr. FAZLIATY EDORA FADZLI

Prepared by:

Git Group team

NO.	NAME	Matrics Number
1.	**AHMED HANI AHMED GHALEB**	**A21EC9120**
2.	**ADAM ISMAIL HASSAN AMER ABOURAYA**	**A22EC0002**
3.	**ABU SAKIB BIN LUTFUL HASSAN**	**A20EC4007**
4.	**MUHAMMAD DAFA RAYHAN YASSER**	**A20EC0317**

Overview

This project implements a basic BMI and fitness management web application using Spring Boot, Spring Web MVC, and JSP. It consists of two main modules: **Client** and **Admin**, with dedicated JSP views for each. The application manages user registration, login, profile management, and fitness program administration.

Project Structure

```
src
├── main
│   ├── java
│   │   └── org
│   │    └── example
│   │             ├── controllers
│   │             │   ├── AdminController.java
│   │             │   ├── ClientController.java
│   │             │   └── MainController.java
│   │             ├── Main.java
│   │             ├── models
│   │             │   ├── AdminViewModel.java
│   │             │   ├── Client.java
│   │             │   ├── ClientViewModel.java
│   │             │   └── FitnessProgram.java
│   │             └── services
│   │             ├── AdminService.java
│   │             └── ClientService.java
│   ├── resources
│   │   └── application.properties
│   └── webapp
│        └── WEB-INF
│        ├── views
│        │   ├── admin
│        │   │   ├── addProgram.jsp
│        │   │   ├── clientList.jsp
│        │   │   └── manageRoles.jsp
│        │   ├── client
│        │   │   ├── login.jsp
│        │   │   ├── profile.jsp
│        │   │   └── register.jsp
│        │   └── index.jsp
│        └── web.xml
└── test
         └── java

16 directories, 19 files
```

Controllers

1. MainController

- **Purpose**: Handles navigation to the landing page.
- **Methods**:
  - `showHomePage()`:

        - **Description**: Maps `/` to the `index.jsp` view, which serves as the application's welcome page.
        - **Data Flow**: No data is passed; it simply renders the home page.
        - **Issues Encountered**: None.

---

2. ClientController

- **Purpose**: Manages client-related functionalities like registration, login, and profile management.
- **Methods**:
    - `showRegisterForm()`:
        - **Description**: Maps `/client/register` to `register.jsp`.
        - **Data Flow**: Displays a registration form.
        - **Issues Encountered**: None.
    - `processRegisterForm(Client client)`:
        - **Description**: Handles registration form submissions, validates input, and stores the data in a static list of clients.
        - **Data Flow**:
            - **Input**: `Client` object with `name`, `email`, `password`, and BMI-related fields.
            - **Validation Rules**:
                - Name: Required.
                - Email: Valid email format.
                - Password: Minimum strength check.
                - BMI: Positive numbers only.
            - **Output**: Redirects to the login page upon successful registration or displays error messages on failure.
        - **Issues Encountered**: Validation logic could not be easily displayed on the JSP pages without proper binding.
    - `showLoginForm()`:
        - **Description**: Maps `/client/login` to `login.jsp`.
        - **Data Flow**: Displays the login form.
        - **Issues Encountered**: None.
    - `processLoginForm(String email, String password)`:
        - **Description**: Validates login credentials against the stored client list.
        - **Data Flow**:
            - **Input**: Email and password.
            - **Output**: Redirects to the profile page on success or displays an error message on failure.
        - **Issues Encountered**: Errors for invalid credentials were not user-friendly initially.
    - `showProfile(String email)`:
        - **Description**: Maps `/client/profile` to `profile.jsp`. Retrieves the logged-in user's data for display.

    - **Data Flow**:
        - **Input**: User email (assumed to be retrieved from the session or a static variable).
        - **Output**: Displays user profile information with editable fields.
    - **Issues Encountered**: User session management not fully implemented.

---

3. AdminController

- **Purpose**: Manages admin functionalities like adding programs, viewing client lists, and managing roles.
- **Methods**:
    - **`showClientList()`**:
        - **Description**: Maps `/admin/clientList` to `clientList.jsp`. Displays all registered clients.
        - **Data Flow**:
            - **Input**: Static list of clients.
            - **Output**: List of all clients rendered in the JSP view.
        - **Issues Encountered**: None.
    - **`showAddProgramForm()`**:
        - **Description**: Maps `/admin/addProgram` to `addProgram.jsp`. Displays a form for creating new fitness programs.
        - **Data Flow**: No input required, serves the JSP view.
        - **Issues Encountered**: None.
    - **`processAddProgramForm(FitnessProgram program)`**:
        - **Description**: Handles submission of the add program form.
        - **Data Flow**:
            - **Input**: `FitnessProgram` object with `name`, `description`, and `duration`.
            - **Output**: Saves the program to a static list and redirects to the admin dashboard.
        - **Issues Encountered**: Validation errors were not user-friendly initially.
    - **`showManageRoles()`**:
        - **Description**: Maps `/admin/manageRoles` to `manageRoles.jsp`. Displays a list of clients and available roles.
        - **Data Flow**:
            - **Input**: Static list of clients and roles.
            - **Output**: Allows role management for each client.
        - **Issues Encountered**: Implementation of role assignment was incomplete.

Data Flow

1. **Client Registration**:
    - User submits data to `processRegisterForm()`.
    - Data is validated and saved to a static `List<Client>`.
    - Redirects to the login page on success.
2. **Client Login**:
    - User submits email and password to `processLoginForm()`.
    - Credentials are validated against the static list.
    - On success, the user's data is loaded into the profile view.
3. **Admin Operations**:
    - Admin accesses client data (`showClientList()`), adds programs (`processAddProgramForm()`), or manages roles (`showManageRoles()`).

Validation Rules

- **Client Registration**:
    - Name: Non-empty.
    - Email: Must match a valid email pattern.
    - Password: Minimum 8 characters with at least one special character.
    - BMI Fields: Numeric and greater than zero.
- **Fitness Program**:
    - Name: Non-empty.
    - Description: Non-empty.
    - Duration: Positive integer.

Issues Encountered

1. **JSP Support in Spring Boot**:
    - JSP rendering was problematic due to dependency issues (`javax.servlet` vs `jakarta.servlet`).
    - Solution: Added `tomcat-embed-jasper` and `javax.servlet:jstl` dependencies explicitly.
2. **Error Pages**:
    - The application showed Spring's default whitelabel error page on missing mappings.
    - Solution: Ensured all controller methods mapped properly to their respective JSP files.
3. **Validation Error Messages**:
    - Initial attempts to display validation errors on JSP pages were incomplete.
    - Solution: Used `model.addAttribute()` to pass error messages.
4. **Session Management**:
    - Client login and session management were not implemented fully.

    - Solution: Placeholder for session-based authentication planned but not implemented.