Java MVC Frameworks

Individual Project Assignment

This is the Individual Project Assignment for the Spring MVC Frameworks Course @ SoftUni.

General Requirements

Your Web application should use the following technologies, frameworks and development techniques:

- The application must be implemented using **Spring Framework**.
  - The application must have at least **12 web pages** (views).
  - The application must have at least **5 independent entity models**.
  - The application must have at least **5 controllers**.
  - The application must have at least **5 services**.
  - The application must have at least **5 repositories**.
  - Optionally, you may use **Spring Data REST**.
- Use **IntelliJ** or **Eclipse**.
  - Use **Thymeleaf** template engine for generating the **UI**.
    - Use **fragments.**
  - You could also make the **Front-End** using **JavaScript**, **consuming REST services** from a **Web API**.
- Use **MySQL / Oracle / PostgreSQL** as a **database**.
- Use **Spring Data** to access your database.
  - User **Hibernate** / **EclipseLink** or any other provider as a **JPA implementation**.
- Implement **Responsive Web Page Design** based on **Bootstrap / Google Material Design**.
- Use the standard **Spring Security** for managing **users** and **roles**.
  - Your **registered** users should have at least **these roles**: **user** and **administrator**.
  - User **roles** should be **managable** from the application.
  - Make sure the **role management** is **secured** and **error-safe**.
- Use **AJAX** to asynchronously **load** and **display data** somewhere in your application.
- Write **Unit Tests** for your **logic**, **services**, **repository query methods**, **helpers**, etc.
  - You should have at least **80% coverage** on your **business logic**.
- Implement **Error Handling** and **Data Validation** to avoid **crashes** when **invalid data** is entered (both **client-side** and **server-side**).
- Handle correctly the **special HTML characters** and tags like **<br />** and **<script>** (**escape** special characters).
- Use at least **2 Interceptors**.
- Run **asynchronous tasks** for jobs that do not need to run sequential or for jobs in the background.
- **Schedule jobs** that impact the whole application running e.g. once/twice a day.
- Use **ModelMapper** or other mapping library.

Additional Requirements

- Follow the **best practices** for **OO design** and **high-quality code** for the **Web application**:
  - Use **data encapsulation**.
  - Use **exception handling** properly.
  - Use **inheritance**, **abstraction** and **polymorphism** properly.

  - Follow the **principles** of **strong cohesion** and **loose coupling**.
  - Correctly **format** and **structure** your **code**, name your **identifiers** and make the code **readable**.
  - Follow the concept of **thin controllers**.

- Well looking **user interface** (**UI**).
- **Good usability** (easy to use **UI**).
- Supporting of **all modern Web browsers**.
- Use **caching** where appropriate.
- Use a **source control system** by choice, e.g. **GitHub**, **BitBucket**.
  - Submit a link to your public source code repository.

Public Project Defense

Each student will have to deliver a **public defense** of its work in front of a trainer jury.
Students will have **only 15 minutes** for the following:

- **Demonstrate** how the application works (very shortly).
- Show the **source code** and explain how it works.

Please be **strict in timing**! On the **15th** minute you **will be interrupted**! It is good idea to leave **the last 4-5 minutes for questions** from the jury.

Be **well prepared** for **presenting maximum** of **your work** for **minimum time**:

- Bring **your own laptop**!
- **Open** the **project assets beforehand** to save time!

Bonuses

- Use **Spring Social** to connect with **Software-as-a-Service** (**SaaS**) **API providers**.
- Host the application in a **cloud environment**, e.g. in **Amazon Web Services**.
- Use a **file storage cloud API**, e.g. **Dropbox**, **Google Drive** or other for storing the files.
- Use of features of **HTML5** like **Geolocation**, **Local Storage**, **SVG**, **Canvas**, etc.
- Implement **Microservice architecture** in your application.
- Anything that is not described in the assignment is a bonus if it has some practical use.

Assessment Criteria

- **Functionality** – **0…20**
- **Implementing controllers correctly** (controllers should do only their work) **– 0...5**
- **Implementing views correctly** (using display and editor templates) **– 0…5**
- **Testing** (unit test and integration tests for some of the controllers using mocking) **– 0…10**
- **Security** (prevent SQL injection, XSS, CSRF, parameter tampering, etc.) **– 0…5**
- **Data validation** (validation in the models and input models) **– 0…10**
- **Using model mapper and inversion of control – 0…5**
- **Using layers with multiple layouts – 0…10**
- **Code quality** (well-structured code, following the MVC pattern, following SOLID principles, etc.) – **0…10**

- **Bonus** (bonus points are given for exceptional project) – **0…25**