Individual Project Assignment for the
PHP MVC Frameworks Course @ SoftUni

Design and implement a **Shopping cart/CMS/RPG Game/Conference Scheduler using PHP (Symfony) and HTML / CSS / JavaScript**. Your project must meet all the requirements listed below.

General Requirements

- **Use PHP** – the major part of your work should be PHP written
  - You **must use Symfony Framework**
    - **The application must have at least 12 web pages (views)**
    - **The application must have at least 4 independent entity models**
    - **The application must have at least 4 controllers**
  - You have to additionally use **HTML5, CSS3** to create the content and to stylize your web application
  - You may optionally use **JavaScript, jQuery, Bootstrap**
  - Use **PHP 7**
- **User source control system**
  - **Use GitHub** or other source control system as project collaboration platform and commit your daily work
- **Valid and high-quality PHP, HTML and CSS**
  - Follow the best practices for PHP development: http://www.phptherightway.com, https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-2-coding-style-guide.md, http://symfony.com/doc/current/best_practices/index.html
  - Validate (when possible) your HTML (http://validator.w3.org) and CSS code (http://css-validator.org)
  - Follow the best practices for **high-quality PHP, HTML and CSS**: good formatting, good code structure, consistent naming etc.
- **Usability, UX and browser support**
  - Your web application should be easy-to-use, with intuitive UI, with good usability (usability != beauty)
  - Ensure your web application works correctly in the latest HTML5-compatible browsers: Chrome, Firefox, IE, Opera, Safari (latest versions, desktop and mobile versions)
  - You do not need to support old browsers like IE9

Forbidden Techniques and Tools

- Using **CMS / blog systems** (like WordPress, Drupal and Joomla) is forbidden.
- Using **Shopping cart systems** (like OpenCart) is forbidden.

Additional Requirements

Your Project **MUST** have a well-structured **Architecture** and a well-configured **Control Flow**.

- Follow the best practices for Object Oriented design and **high-quality code** for the Web application:
  - Use the OOP principles properly: data encapsulation, inheritance, abstraction and polymorphism
  - Use exception handling properly
  - Follow the principles of strong cohesion and loose coupling

    - Correctly format and structure your code, name your identifiers and make the code readable
- Make the user interface (UI) good-looking and easy to use
    - If you provide a broken design, your Functionality Points will be sanctioned
- Support all major modern Web browsers
    - Optionally, make the site as responsive as possible – think about tablets and smartphones
- Use Caching where appropriate

Source Control

Use a **source control system** by choice, e.g. **GitHub**, **BitBucket**

- Submit a link to your public source code repository
- You should have **commits** in at least **3 DIFFERENT** days
- You should have at least **10 commits**

IMPORTANT: The **Source Control Requirements** are **ABSOLUTELY MANDATORY**.
IMPORTANT: NOT following the **Source Control Requirements** will result in your **DIRECT DISQUALIFICATION** from the **Project Defenses**.

Public Project Defense

Each student will have to deliver a **public defense** of its work in front of a trainer.
Students will have **only 10-15 minutes** for the following:

- **Demonstrate** how the application works (very shortly)
- Show the **source code** and explain how it works
- Answer questions related to the project (and best practices in general)

Please be **strict in timing**! On the 15th minute you **will be interrupted**! It is good idea to leave **the last 2-3 minutes for questions** from the trainers.

Be **well prepared** for presenting maximum of your work for minimum time. Bring your **OWN LAPTOP**. Test it preliminarily with the multimedia projector. Open the project assets beforehand to save time.

Bonuses

- Anything that is not described in the assignment is a bonus if it has some practical use
- Examples
    - Use **Front-End Frameworks** (like **Angular**, **React**, **Vue**)
    - Host the application in a **cloud environment**, e.g. in **Google Cloud** or **AWS**
    - 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.

Assessment Criteria

- **Functionality** – **0…20**
- **Implementing controllers correctly** (controllers should do only their work) **– 0...10**
- **Implementing views correctly** (using display and editor templates) **– 0…10**
- **Unit tests** (unit test 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**
- **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**