PHP assignment (home project)

Submit Assignment

Due Jan 15, 2023 by 11:59pm **Points** 20 **Submitting** a file upload **File Types** zip
Available until Jan 29, 2023 at 11:59pm

In this home assignment you need to create a web application where logged-in users can cast their votes on polls (questionnaires/forms). Admin users can create polls for which users can vote by selecting one or more options. On the main page, all polls in the system are listed. Polls that have already ended are at the bottom, while the ongoing polls appear at the top of the page. We have put the task together so that you don't have to use sessions or authentication to complete the minimum requirements, thus making things easier for you during this exceptional exam period. If you have any questions, you can ask in the General Teams channel of the subject or contact your practice teacher!

Tasks

Preparation

- The application will need to store the polls, the users, and the votes that have been submitted by the users on a poll. We have included one possible storage format below as an example, but you may freely choose any other format.
- Your submission also needs to contain some already prepared polls that the users can vote on. Each user can cast only one vote per poll but can change it before the deadline expires. If your site does not include authentication, a user can cast an unlimited number of votes.
- A special administrative user must be created whose login details are fixed. See the description for admin functions below.

Main page

- On the *listing page* (a.k.a. the *main/index page*) a creative title and a short description about the application should be visible as static text.
- The main page is also accessible to unauthenticated users who are free to browse the polls displayed here.
- The most recently created poll should always appear at the top of the page. Following it must be the other polls ordered descending by the date of creation. The polls should be listed in two sections on the page:
  - Polls whose deadline has not yet expired are displayed in the top section.
  - Polls that have already been closed are displayed in the bottom section. Their results are displayed as well.
- The following elements should appear for each poll:
  - the ID / number of the poll;
  - the time of creation;
  - the voting deadline;
  - a button to vote.

- A button belongs to each poll where votes can be submitted to that poll on the **voting page**. If your site contains session management and authentication but the user is not logged in yet, redirect the user to the login page instead.

Voting page

- The *voting page* should display the following information about a given poll:
  - the text (description) of the poll;
  - the possible options/choices;
  - the voting deadline;
  - the time of creation.
- A submit button should appear below the possible options. By clicking on it, your vote will be stored persistently. Verify that the user has selected a (valid) option and notify them if they haven't. Also notify the user if the vote has successfully been submitted.

Poll creation page

- Show a form on the *poll creation page* where the following fields are present and can be saved:
  - the text of the poll (text);
  - the possible options/choices (textarea - with one option per line for simplicity);
  - whether multiple options can be selected (radio);
  - the voting deadline (date);
  - the time of creation (date - not a required field but needs to be stored);
  - submit button (submit).
- If your solution contains authentication and authorization, then poll creation should only be accessible to the admin user. (See: admin features)

Authentication pages

- The login and registration page should be accessible from the main page.
- During **registration** users must enter their username, email address, and their password. The password must be entered twice. All fields are required, the email must be a valid email format, and the two passwords must match. In case of an error, display appropriate error messages! The form must be persistent, so after an error, previously filled data should remain in the form. Upon successful registration, save the data and redirect the user to the login page!
- On the **login** page users can identify themselves with their username and password. If there was an error logging in, display a message about it under the login form! After successful login, redirect the user to the main page.

Admin features

- Create a special user named **admin** with password `admin` who can:
  - create new polls;
  - delete polls;
  - (for extra points) edit the already existing polls.

Additional expectations and help

Design is important. Your submission doesn't have to be too pretty and filled with frills, but it should look nice on a screen of at least 1024×768 pixels. You can use minimalist design, custom CSS with extra graphical elements and background images or even a proper CSS framework.

Add the `novalidate` attribute to your `<form>` elements to disable browser-level validation (you have to validate everything on the server side)!

```
<form action="" novalidate>
</form>
```

Data

- The are two sets of data in the assignment: polls and users.
- For each poll you need to store its unique ID, the question, the possible choices, whether multiple options can be selected, the date of creation and the voting deadline. Verify whether all fields have been properly filled when saving or modifying.
- It is also necessary to somehow store who submitted answers to the given poll and how many people voted for each option. Due to anonymity, it is important not to store which vote was cast by whom! You can achieve this by creating the array containing the votes within the data structure of the polls; but you can even work with them in a separate data file or structure by storing the number of votes per option and the users who have submitted them separately.
- The users are described by three required properties: the username, the email address, and password. You should also store whether the user has admin priviledges.
- Based on the description above, the following is a valid method of representation (but you are not required to use this):

```
$polls = [
    'poll1' => [
        'id' => 'poll1',
        'question' => 'Would you like FREE snacks on the university campus?',
        'options' => ['Yes', 'No'],
        'isMultiple' => False,
        'createdAt' => '2022-12-04',
        'deadline' => '2022-12-12',
        'answers' => ['Yes' => 2, 'No' => 0]
        'voted' => ['userid1', 'userid2']
      ],
    'poll2' => [
        'id' => 'poll2',
        'question' => 'What kind of snacks would you like?',
        'options' => ['Potato chips', 'Peanuts', 'Chocolate bars', 'Cookies'],
        'isMultiple' => True,
        'createdAt' => '2022-12-05',
        'deadline' => '2024-12-13',
        'answers' => ['Potato chips' => 3, 'Peanuts' => 3, 'Chocolate bars' => 3, 'Cookies' => 2]
        'voted' => ['userid1', 'userid2', 'userid3']
        ]
];
$users = [
    'userid1' => [
        'id' => 'userid1',
        'username' => 'admin',
        'email' => 'email1@email.hu',
        'password' => 'admin'
        'isAdmin' => True,
    ],
    'userid2' => [
        'id' => 'userid2',
        'username' => 'user2',
        'email' => 'email2@email.hu',
        'password' => 'user2'
        'isAdmin' => False,
    ],
    'userid1' => [
        'id' => 'userid1',
        'username' => 'user3',
        'email' => 'email3@email.hu',
        'password' => 'user3'
        'isAdmin' => False,
    ],
];
```

Breaking down the development into steps

We would like to help those who have a harder time seeing and planning a bigger task. You can plan the whole task in advance and improve it later, but the following steps can also be used to solve smaller subtasks:

1. First, create a **static HTML prototype** for your application! This means that as a first step, you should use only HTML and CSS to design the main/listing page, the voting page, and so on. There are pages where there is conditional information such as whether the poll is closed or still open. Design these as well and hide them later from PHP. You can also test CSS statically, e.g. how individual polls or results will be displayed, and later fill in the exact properties dynamically. You can connect the pages with hyperlinks.
2. Think about what **data** you will need. What kind of data structure can store it, with what fields?
   - Where do you store the users?
   - Where do you store the polls?
   - Where do you store the votes that belong to a poll?
   - How do you store if the user submits a vote?
   - How can you store multiple votes per poll, how can you store multiple choice polls at a user level?
3. Think about how you will extract the **right data format** from the data structures in the previous stage! You can create functions, or even better, Storage class methods to implement transformations.
   - How do you get data of a given poll?
   - How do you get newest polls?
   - How do you check if a poll has been voted for by the user?
4. For **forms** you need to think about two things.
   - What happens on a success?
   - How do you detect errors and how will the form become persistent?

Tasks for extra points

1. Group creation: Create an additional page that can only be accessed by the admin user. On this page the admin user can create groups and, in this case, only the members of that group can see the poll. Show a form on the page that contains a field for the group's name and for example a `<select name='' multiple>` element where you can select the users that will be included in the group. It is recommended to store the groups as a new dataset, and the poll needs to have a new property that describes which group it has been assigned to. It is also possible to assign a poll to every user. Polls that are assigned to a group should only be visible to that group.

2. Editing existing polls: The admin user can modify the properties of polls. Think about what can happen if you change the possible options, how will that affect the votes that have already been submitted. For example, you can implement that if you change the voting options, then voting is reset; but if you only touch the text or the voting deadline, then the votes are kept.

Grading

By solving the assignment, 20 points can be obtained. There are minimum requirements without which the submission will not be acceptable. An additional 5 points can be earned for extra tasks. That is, if someone does everything, they can get to 25 points.

Criteria related to the PHP home assignment for getting you final course grade: it is required to meet all minimum requirements AND have at least 8 points (40%) after the deduction of points due to late submission. (Please consider what is easier for you: solve the minimum requirements by the deadline; or hand in an almost perfect version two weeks later and still receive barely more than the minimum required points.)

Minimum requirements (must be completed, 8 pts)

- Main page: displayed (0 pts)
- Main page: all created polls are displayed (1 pt)
- Main page: the data about the polls can be seen - poll ID, time of creation, voting deadline (1 pt)
- Main page: expired polls are displayed separately, both sections are ordered by the date of creating in descending order (newest poll is at the top) (1 pt)
- Main page: clicking on a poll/button redirects to the voting page (1 pt)
- Voting page: the details of the given poll are shown - the question text, the possible options, and the submit button saves the vote (2 pts)
- Voting page: if the voting form is filled properly (an option has been chosen) the user is notified about the successful vote. An error is displayed if the form is sent without selecting an option. (0.5 pts)
- Poll creation page: the question, the possible options (two or more), the voting deadline can be entered on the page and the poll (including the time of creation) can be saved (1.5 pts)

Basic tasks (12 pts)

- Login: error handling (1 pt)
- Login: successful login (1 pt)
- Registration form: contains appropriate elements (0.5 pts)
- Registration form: validation, error messages, keeping the form state (1.5 pt)
- Registration form: successful registration (0.5 points)
- Main page: the voting button redirects to the login page if the user is not logged in. The button redirects to the voting page if the user is logged in. (1 pt)
- Main page: the button's caption shows whether a vote has been submitted by the user to that poll. (e.g., if there was a vote then change to text to *Update/edit vote*) (0.5 pt)
- Main page: the vote cannot be changed after the deadline and the results are only shown after the deadline. (1 pt)
- Admin feature: the admin can log in with the given credentials (0.5 pts)
- Admin feature: a new poll can ONLY be created by the admin user (0.5 pts)
- Admin feature: when creating the poll, the admin can determine (using radio or checkbox input) whether selecting multiple options is allowed (1 pt)

- Admin feature: the admin user can delete an existing poll (1 pt)
- Voting page: the voting page shows whether selecting multiple options is allowed and if it is, then multiple options can actually be selected, and the votes are saved correctly (2 pts)
- **Late submission: -0.5 pts / started day!**

Extra tasks (extra 5 pts)

- Admin feature: the admin user can create groups from the existing users and can assign polls to the groups with one click at the creation of the poll (3 pts)
- Admin feature: when the admin user is logged in, an additional button is shown on the main page where the admin can edit the data of the poll which is updated in the saved data (2 pts)

Additional requirements

- The assignment should be compressed into an archive containing all necessary files AND the `README.md` file in the root folder of the program and uploaded to Canvas by the deadline (plus two weeks with point deduction).
- **You CANNOT use any PHP framework OR external, third-party PHP library!** Only CSS frameworks are permitted.
- The `README.md` file has the following requirements:
  - You fill in your own data at the start of the file (marked by `< >` signs)
  - You mark all (even partially) finished subtasks with an `x` in place of the space between the square brackets `[ ]` in the file.

```
<student's name>
<student's NEPTUN code>
Web-programming - PHP home assignment
This solution was submitted by the stundent named above for a Web-programming assignment.
Hereby I declare that the solution is my own work. I did not copy or use solutions from a third party. I did not share this solution with fellow students, and I did not publish it.
According to the Academic Regulations for Students (Eötvös Loránd University Organisational and Operational Regulations – Volume 2, Section 74/C), a student purporting the intellectual property of others as their own [...] is committing a disciplinary offence.
The worst result of a disciplinary offence can be the expulsion of the student.

## Minimum requirements (must be completed, 8 pts)

[ ] Main page: displayed (0 pts)
[ ] Main page: all created polls are displayed (1 pt)
[ ] Main page: the data about the polls can be seen - poll ID, time of creation, voting deadline (1 pt)
[ ] Main page: expired polls are displayed seperately, both sections are ordered by the date of creating in descending order (newest poll is at the top) (1 pt)
[ ] Main page: clicking on a poll/button redirects to the voting page (1 pt)
[ ] Voting page: the data about the given poll are shown - the question text, the possible options, and the submit button saves the vote (2 pts)
[ ] Voting page: if the voting form is filled properly (an option has been chosen) the user is notified about the successful vote. An error is displayed if the form is sent without selecting an option. (0.5 pts)
[ ] Poll creation page: the question, the possible options (two or more), the voting deadline can be entered on the page and the poll (including the time of creation) can be saved (1.5 pts)

## Basic tasks (12 pts)

[ ] Login: error handling (1 pt)
[ ] Login: successful login (1 pt)
[ ] Registration form: contains appropriate elements (0.5 pts)
[ ] Registration form: validation, error messages, keeping the form state (1.5 pt)
[ ] Registration form: successful registration (0.5 points)
[ ] Main page: the voting button redirects to the login page if the user is not logged in. The button redirects t
o the voting page if the user is logged in. (1 pt)
[ ] Main page: the button's caption shows whether a vote has been submitted by the user to that poll. (e.g. if th
ere was a vote then change to text to *Update/edit vote*) (0.5 pt)
[ ] Main page: the vote cannot be changed after the deadline and the results are only shown after the deadline.
(1 pt)
[ ] Admin feature: the admin can log in with the given credentials (0.5 pts)
[ ] Admin feature: a new poll can ONLY be created by the admin user (0.5 pts)
[ ] Admin feature: when creating the poll, the admin can determine (using radio or checkbox input) whether select
ing multiple options is allowed (1 pt)
[ ] Admin feature: the admin user can delete an existing poll (1 pt)
[ ] Voting page: the voting page shows whether selecting multiple options is allowed and if it is, then multiple
options can actually be selected and the votes are saved correctly (2 pts)
[ ] **Late submission: -0.5 pts / started day!**

## Extra tasks (extra 5 pts)

[ ] Admin feature: the admin user can create groups from the existing users and can assign polls to the groups wi
th one click at the creation of the poll (3 pts)
[ ] Admin feature: when the admin user is logged in, an additional button is shown on the main page where the adm
in can edit the data of the poll which is updated in the saved data (2 pts)
```

A submission without a properly filled `README.md` file is not eligible to be graded!