Challenge PHP No.5 - Specification

Description
After the presentation of the challenge and receiving the starter files, each student is required to complete the assignment within the given deadline. The solution should be uploaded to GitLab, with the assessor added as a maintainer to the repository. Additionally, the student is instructed to add the solution link to the learning platform.

Requirements

Part 1:
You need to create 4 pages.

The **first page** will be a sign up/login page. It should have a background image and two buttons. (sign up and login). The forms are not on this page, only 2 links (styled as buttons) that when clicked will take the user to a page dedicated to either login or register action.

The sign-up link/button should take the user to the **second page** where a form with two inputs for username and password is shown. At the top of this page the text 'Sign up form' is printed.
When the user submits the form, the data should be written into a users.txt file (format: username=password). The password should be hashed, either using md5 or password_hash (preferred option).
Every new user should be inserted in a new row in the users.txt file.

After form submission, the user should be redirected to a new page (**third page**) where the message 'Welcome $username' will be shown.

The login link/button should take the user to a new page (**fourth page**) where a form with 2 inputs for username and password is also shown. At the top of this page the text 'Login form' is printed.
When a user submits the form – a check should be done if the data the user entered matches some of the users in the users.txt file. Remember, the passwords need to be hashed.
If the user is not found in users.txt file – show the message 'Wrong username/password combination '.
If everything is ok – Redirect the user to the third page where the same message 'Welcome $username' is shown.

Part 2:
Add another input to the sign-up page: email.

Create a new file where instead of storing only the username=password, you will save the email too, in the following format: 'email, username=password'.

When registering add logic to check if the username already exists in the users.txt file. If it does, print a message 'Username taken'.

Also, check if the email already exists in the users.txt file. That means that when submitting the form, the user has to provide a unique email. If the email already exists in the users.txt file, print the following message: 'A user with this email already exists'. Use yellow font color for this message.

Level 1: Beginner
Page 1 + Page 2

Level 2: Intermediate
Page 1 + Page 2 + Page 3 + Page 4

Level 3: Advanced
Page 1 + Page 2 + Page 3 + Page 4 + Adding a unique email field: 1 point

Evaluation system

Criteria	Excellent 2.5p	Proficient 2.0p	Good 1.5p	Fair 1.0p	Poor 0.5p
Solution Correctness	*The solution is flawless, meeting all requirements and functionalities.*	*The solution is correct for the most part, with minor errors that do not significantly impact the functionality.*	*The solution is mostly correct, but there are some major errors affecting the overall functionality.*	*The solution has several major errors, making it partially functional, but shows a basic understanding of the problem.*	*The solution is incorrect, incomplete and lacks understanding of the problem.*
Code Quality	*Code is exceptionally well-organized, follows best practices, and demonstrates a deep understanding of all required concepts.*	*Code is mostly clear and well-organized and follows good practices, with only minor improvements needed.*	*Code is mostly organized, but there are notable areas that could be improved for better readability and maintainability.*	*Code lacks organization, readability, and structure, and could be significantly improved.*	*Code is messy, unreadable, poorly organized, and does not adhere to coding standards.*
Adherence to Specifications	*The solution precisely follows all specified challenge requirements.*	*The solution mostly adheres to the specifications, with only minor deviations or oversights.*	*The solution deviates from specifications in some aspects but still fulfills the main requirements.*	*The solution deviates significantly from the specifications, but some elements are implemented correctly.*	*The solution disregards most or all of the specified requirements.*
Documentation, Comments, Cleanliness	*The code is well-documented, clear and exceptionally clean. Comments explain the logic behind any complex parts.*	*Good documentation and comments. Code is generally clean with a few areas for improvement.*	*Basic documentation and comments, with room for improvement in clarity. Code cleanliness is acceptable but needs improvement.*	*Limited documentation and comments. Code lacks clarity and cleanliness and is not well-organized.*	*No documentation or comments. Code is disorganized and challenging to understand.*

Deadline

1 week after its presentation, at 23:59 (end of the day).

Assessment Rules

- Fair Assessment: ethical considerations
  - Assessors should ensure that the assessments are conducted in a fair and ethical manner, respecting the principles of academic integrity and honesty.
- Reliability and Validity: enabling consistency
  - Assessments should be consistent and reliable, meaning that they yield consistent results when applied repeatedly to the same task or performance.
  - Assessments must accurately gauge the knowledge, skills, or abilities they are intended to evaluate, ensuring their validity as indicators.
- Feedback: as a method for continuous improvement
  - Assessors should offer constructive feedback that identifies strengths and areas for improvement. The Feedback should be specific and actionable, it should include thought provoking guides and should challenge the student to become better at a specific task.
- Late Submission Policy: assessing assignments after their deadline
  - Students should be allowed a grace period of 3 days (72 hours) to make a late submission on any assignment, with the notice that they will be deducted 20% from the total possible points.
- Plagiarism Policy: assessing assignments with matching solutions
  - In the event of suspected plagiarism, the assessor is required to promptly collaborate with the Student Experience Coordinator/Team as the initial step. Together, they will draft a notice to remind students of the strict prohibition against plagiarism, with potential repercussions for recurrent violations. The following actions will be considered in cases of repeated plagiarism:
    - If a submitted solution exhibits substantial similarity, exceeding 60%, with another student's work (individually or within a group), the respective challenge will incur a 50% reduction from the maximum points attainable.
    - In cases where a solution is identified as more than 90% identical to another student's work (individually or within a group), the challenge in question will receive a score of 0 points.
  - The use of generative AI is encouraged as a learning tool in our educational programs; however, students must engage with the material and contribute with original thought. Reliance on AI for complete content generation is strictly prohibited and will result in point deductions, official warning, or other academic penalties.
  - Upon completion of the assessment process for each challenge/project, the assessor is tasked with selecting the most complete, optimal, and creative solution and to showcase it by publishing it on the platform together with the assessment results
- Timeliness: timeframe for delivering results and feedback to students
  - Feedback on challenges should be provided within 7 days after the deadline has passed.