Building a User Registration System using PHP and JSON

Objective:
The objective of this workshop task is to guide participants through the process of creating a user registration system using PHP and JSON. Participants will learn how to handle form data, implement data validation and security measures, and store user information in a JSON file.

Task Description:
1. Create a new PHP file named "registration.php" and set up an HTML form for user registration. The form should include the following fields: name, email address, password, and a confirm password field. Use appropriate HTML form elements such as `<input>` and `<label>`. Also, ensure that the form has a proper submit button.

2. Implement form validation using PHP to ensure that all required fields are filled, the email address is in a valid format, and the password meets certain criteria (e.g., minimum length, special characters). You can use PHP's built-in functions, regular expressions, or custom validation functions to perform the validation. If any validation fails, display appropriate error messages next to the respective fields to guide the user on what needs to be corrected.

3. Create a JSON file named "users.json" to store user information. The file should initially contain an empty array, representing the collection of registered users. Make sure the file has the necessary read and write permissions.

4. When the registration form is submitted and the data is valid, use PHP's file handling functions to read the contents of "users.json" and decode it into a PHP array. For example, use `file_get_contents()` and `json_decode()` functions.

5. Hash the user's password using PHP's password_hash() function to enhance security. Store the hashed password along with the user's name and email in an associative array.

6. Add the new user's associative array to the existing array obtained from "users.json".

7. Write the updated array back to "users.json" using PHP's file handling functions. For example, use `json_encode()` to convert the PHP array back to JSON format, and `file_put_contents()` to write it to the file. Make sure to overwrite the previous contents of the file.

8. Provide user feedback after successful registration by displaying a success message on the page, informing the user that their registration was successful. You can use a simple HTML `<div>` element to display the message.

9. Implement error handling for potential issues, such as file read/write failures or any other exceptional scenarios. Display appropriate error messages to the user when necessary. For example, if there is an error while reading or writing the JSON file, display an error message indicating the issue.

10. Test the registration system by submitting valid and invalid data through the form. Make sure to test scenarios such as empty fields, invalid email addresses, mismatched passwords, and successful registrations. Ensure that the validation and error handling mechanisms are functioning correctly. Verify that the user's information is being stored accurately in the "users.json" file.