Project 1: Build Web Application using PHP

Goal: ISTE341 Company wants to keep track of bugs in their various projects and who is responsible for fixing them.

Make sure you include a link to your project along with any username and passwords in the dropbox. There is a discussion area set up on MyCourses to ask questions regarding the project. Use this discussion area as in place of being able to ask the end-user about the requirements.

Requirements

- Download the SQL file to create the tables from MyCourses and create the tables on Solace. (There is also an EER diagram of the tables).
- You cannot modify or delete the tables created by the script.
- You will have to populate the tables with at least one user of each "Role" – Admin, Manager, User, 2 "projects" and 2 "bugs".
- The project will have 2 pages: an Admin Page and a Bug page (see below).
- Users need to be logged in to use the application (this can be done as a separate page or other method). If the user isn't logged in, you need to require them to login. Also, provide a logout option.
- You will use sessions to control access to the functions below based on role.

The application needs to track the following information:

- Detailed description of the bug (description – alphanumeric -required)
- A summary of the bug (summary – alphanumeric - required)
- Who identified the bug (owner – must be existing user - required)
- The date on which the bug was identified (dateRaised – valid date - required)
- Which project the bug is related to (projectId – must be existing project - required)
- Who the issue is assigned to (assignedToId – must be existing user – NULL if the person entering the bug can't assign it)
- A current status of the issue (statusId – must be one of the status ids provided – default to "unassigned" if the person entering the bug can't assign it)
- Priority of the issue (priorityId – must be one of the priority ids provided – default to "medium" if the person entering the bug can't assign it)
- Target resolution date (targetDate – valid date in the future – NULL if the person entering the bug can't assign it.)
- Actual resolution date (dateClosed – valid date – NULL when creating the bug.)
- Resolution summary (fixDescription – alphanumeric – NULL when creating the bug)

Because the company is concerned about everyone having access to all the information, the following access rules apply:

- Each team member (regular user) is only assigned to one project at a time (more than one user can be assigned to a project though).
- Managers are never assigned to a specific project.
- Only managers and Admin can define and maintain projects (create/update) and people (assign to project and/or bug).
- Everyone can enter new bugs.
- All passwords must be stored as hashes using a PHP hashing method (SHA256 or password_hash function).
- Only Admins can add users to the system. They can also delete users. When a user is deleted, they must be removed from any project and bugs that they are assigned to (the project and bug still remain).
- Once a bug is assigned, only the person assigned or a manager or the Admin can change data about the bugs.

Data Entry Functionality:

- Add/Update Bug functionality (Admin, Managers, Users) per above.
- Provide Admin Functions – User admin (Admin) and Project admin (Admin, Managers) per above.

Once the data is entered into the application, users need to view the data **according to the access rules** below. **All queries to create the datasets MUST BE DONE ON THE SERVER-SIDE**. (you should come up with a user-friendly way to do this on the client-side but doesn't have to be fancy):

1. All bugs by project (Admin, Managers), Users only for their assigned project
2. Open bugs by project (Admin, Managers), Users only for their assigned project
3. Overdue bugs (not completed by target date), by project and for all (Admin, Managers) Users only for their assigned project.
4. Unassigned bugs (Admin, Managers)

- **If not using MVC, the common elements for each page (navigation, etc) will NOT use include/required statements to include those elements. You need to achieve this via templates or functions.**
- **Your code should be structured to be re-usable and easily maintained. This means extensive use of functions and classes.**
- **All input must be validated and sanitized on the server-side.**
- **ALL DATABASE QUERIES MUST BE PARAMETERIZED QUERIES USING PREPARED STATEMENTS.**

- **DUE DATE: See dropbox on myCourses @ 11:59pm**
  - **Zip up entire project (file structure intact!).**
  - **Include links and all usernames and passwords in the comments of the dropbox.**

There is a separate grade and due date for a project plan and initial work:

- A system architecture diagram
  - https://aws.amazon.com/what-is/architecture-diagramming/
  - Tools:
    - https://whimsical.com/
    - https://miro.com
    - https://online.visual-paradigm.com
    - https://draw.io – https://app.diagrams.net/
    - https://www.taskade.com/
    - https://app.diagrams.net/
    - https://gliffy.com (has free student accounts)
    - https://lucidchart.com
    - https://excalidraw.com/
    - https://online.visual-paradigm.com/diagrams/solutions/free-aws-architecture-diagram-tool/

- Stubbed-in code for all your files and Database class(es).

- **Note**: Your final project may not match these exactly but should be close. You should put effort into designing your application before implementing it.

- **Zip up all of these including the diagram as a separate submission and submit it to the correct MyCourses dropbox by 11:59pm on the due date on the dropbox.**

Project Grading

- Meeting the above requirements will result in a 'B'. **See the rubric** for options required to get an 'A'.