PROJECT DOCUMENTATION

The theme I choose for the project is call Online Examination System. I stated building my code by first doing the html pages (Index), a php page (dashboard, database, login, register, start_exam, submit_exam), mysql (proj 2).

Index.html

This HTML code represents the login and registration page for an online exam system. It provides two separate forms: one for user login and another for user registration. Below is the detailed breakdown and explanation of the code:

- Specifies the document type as HTML5.
- Sets the language attribute to English (lang="en").

<meta charset="UTF-8">: Ensures proper encoding of characters, supporting a wide range of text.
<meta name="viewport" content="width=device-width, initial-scale=1.0">: Makes the page responsive, optimizing it for all screen sizes (especially mobile devices).
<title>: Sets the browser tab title as "Online Exam System."

Login Form

Purpose: Allows users to log in by submitting their credentials.

Structure:

<form>: Sends form data to login.php using the POST method for secure data transmission.

<input>:Text Input: Field for entering the username. Password Input: Field for entering the password, which hides the text entered. required: Ensures that both fields are mandatory.
<button>: Submits the login form.

Register Form

Purpose: Allows new users to create an account.

Structure: <form>: Sends form data to register.php using the POST method.
<input>: Text Input: Field for entering a desired username. Password Input: Field for entering a password, also masked for security. required: Ensures that both fields are mandatory.
<button>: Submits the registration form.

Features

1. **Responsive Design**: The use of the <meta viewport> tag ensures that the page is optimized for mobile and desktop screens.

2. **User Authentication**: Provides two entry points: Login for existing users, Registration for new users.
3. **Secure Data Submission**: Both forms use the POST method to transmit data securely, avoiding exposure of sensitive information in the URL.

Dashboard.php

This PHP code snippet displays an exam dashboard that lists available exams. The dashboard is accessible only to logged-in users. Below is the breakdown of the functionality and structure of the code:

session_start(); Initiates a session for the user. This allows tracking and storing session-specific data like user authentication.

Ensures the user is logged in by checking the user_id in the session. If user_id is not set, the user is redirected to the login page (index.html), and the script stops execution using exit.

Includes the db.php file to establish a connection to the database. This file should contain the database connection logic.

Executes a query to fetch all records from the exams table. Retrieves the result set as an associative array using fetchAll() and stores it in the $exams variable.

Loops through the $exams array using a foreach loop.

For each exam:

- Displays the exam title in an <h3> tag.
- Displays the duration in minutes in a <p> tag.
- Provides a hyperlink to start the exam, passing the exam_id as a query parameter to start_exam.php.

Ensure the db.php file is correctly set up for database connection. Log in as a user to access the exam_dashboard.php page. Select an exam by clicking the "Start Exam" link, which redirects to start_exam.php with the relevant exam_id.

Database.php

This PHP script establishes a connection to a MySQL database using PDO (PHP Data Objects). Below is the detailed explanation and functionality of the code:

$dsn (Data Source Name): Specifies the database type (mysql), the host (localhost), and the database name (online_exam).

$username: The username for connecting to the database. In this case, it's set to root (default for local servers).

$password: The password for the database user. It's empty here as is common for local development setups.

Creates a new PDO instance to establish a connection to the database using the provided DSN, username, and password.

Try Block: Sets the error handling mode to PDO::ERRMODE_EXCEPTION, ensuring that database errors throw exceptions. This makes it easier to catch and handle errors programmatically.

Catch Block: Catches any PDOException if the connection fails. Outputs an error message and terminates the script using die().

Database Connectivity: Establishes a secure and efficient connection to a MySQL database.

Error Handling: Utilizes exceptions to catch and handle connection errors gracefully. Provides detailed error messages for debugging purposes.

Configurable: Allows easy configuration of database credentials and connection parameters.

Login.php

This script handles user login by validating credentials against a database. It uses secure methods for input handling, database queries, and password verification. Below is a detailed breakdown of the code.

Authenticate users attempting to log in. Start a session upon successful authentication. Redirect authenticated users to a dashboard page.

Initializes or resumes a session. Enables storing user-related information (e.g., user_id) during the session.

Includes the db.php file to establish a connection with the database using the $pdo object.

Ensures the script executes only when the login form is submitted via the POST method, preventing direct access or unintended behavior.

trim(): Removes unnecessary spaces before and after the input. htmlspecialchars(): Converts special characters to HTML entities to prevent XSS (Cross-Site Scripting) attacks.

password_verify(): Compares the hashed password from the database with the plain text password provided by the user.

Session Storage: Stores the authenticated user's ID (user_id) in the session for subsequent use.

Redirect: Redirects the user to the dashboard page (dashboard.php) upon successful authentication.

Register.php

This PHP script handles user registration for a web application by securely storing user credentials in a database. Below is a detailed explanation of the code.

Accept user input from a registration form. Store the user's username and securely hashed password in the database. Provide feedback on successful or failed registration.

Includes the db.php file to establish a connection with the database using the $pdo object.

Ensures the script executes only when the form is submitted using the POST method.

trim(): Removes unnecessary whitespace from the input.

htmlspecialchars(): Converts special characters in the username to HTML entities, preventing XSS (Cross-Site Scripting) attacks.

password_hash(): Hashes the password using the bcrypt algorithm for secure storage in the database.

Prepared Statement: Prevents SQL injection by using placeholders (:username and :password).

Bind Parameters: Safely binds sanitized user input to the SQL query placeholders.

execute(): Executes the SQL statement to insert the new user into the database.

Success Message: Displays a success message with a link to the login page upon successful registration.

Failure Message: Displays a failure message if the query execution fails.

Start_exam.php

This PHP script generates an exam page for users to take an online exam. It checks if the user has already attempted the exam and fetches the questions from the database to display them dynamically.

Verify if a user has already taken a specific exam. Display the exam questions to the user if they are eligible. Allow the user to submit answers via a form.

Initializes or resumes the session. Ensures access to the session variables, such as user_id.

Includes the db.php file to establish a connection with the database using the $pdo object.

$exam_id: Retrieves the exam ID from the query string (GET method).

$user_id: Retrieves the logged-in user's ID from the session.

SQL Query: Checks the user_exams table for an entry matching the current user and exam.

Condition: If a record is found, it means the user has already taken the exam.

Response: Displays a message and exits to prevent further action.

SQL Query: Retrieves all questions associated with the given exam ID from the questions table.

Fetch: Retrieves the questions as an array.

Form Structure: Displays each question ($question['question_text']) and its corresponding options (A, B, C, D). Each option is an input of type radio grouped by name="answers[question_id]" to ensure one selection per question.

Hidden Field: Includes the exam_id as a hidden input field to be passed along with the form submission.

Submit Button: Submits the form data to submit_exam.php for processing.

This script dynamically generates a secure, user-friendly interface for taking exams while ensuring that users cannot retake exams or access invalid data.

Submit_exam.php

This PHP script processes the answers submitted by a user for an exam, calculates the score, and stores the results in the database. It ensures the operation is executed securely and transactionally.

Validate and evaluate user-submitted answers. Calculate the user's score for the exam. Store the exam result in the database. Provide feedback on successful or failed submission.

Initializes or resumes the session. Ensures access to session variables, such as user_id.

Includes the db.php file to establish a connection with the database using the $pdo object.

$user_id: Retrieves the logged-in user's ID from the session.

$exam_id: Exam ID from the form submission (POST method).

$answers: An associative array containing user-submitted answers, where the keys are question IDs and the values are the selected options.

$score: Initializes the user's score to 0.

Starts a database transaction to ensure atomicity. All operations within the transaction must succeed; otherwise, they are rolled back.

Loops through each submitted answer: Fetches the correct answer for the question from the questions table. Compares the correct answer with the user's selected answer. Increments the score for each correct match.

Inserts the user's result into the user_exams table, including:

user_id: The ID of the user who took the exam.

exam_id: The ID of the exam.

score: The calculated score.

This script is a robust solution for processing exam submissions. It adheres to best practices for security, transactional integrity, and dynamic handling of user-submitted data.

Proj 2.mysql

This documentation outlines the structure of the online_exam database, which supports an online examination system. It includes tables for managing users, exams, questions, and tracking user exam attempts and scores.

Creates a database named online_exam. Switches the context to use the newly created database.

The users table stores user information, including their credentials and roles.

Columns:

- id: Unique identifier for each user (Primary Key).
- username: Unique username for the user (e.g., john_doe).
- password: Hashed password for secure authentication.
- role: Defines the user's role, either student or admin, defaulting to student.

Constraints:

UNIQUE: Ensures no duplicate usernames. Default role is set to student.

The exams table stores details about the exams.

Columns:

- exam_id: Unique identifier for each exam (Primary Key).
- title: The name or title of the exam (e.g., Math Test).

- duration: Duration of the exam in minutes (e.g., 60 for 1 hour).
- total_marks: Total marks that can be achieved in the exam.

The questions table stores individual exam questions and their options.

Columns:

- question_id: Unique identifier for each question (Primary Key).
- exam_id: Foreign key linking the question to a specific exam.
- question_text: The text of the question.
- option_a, option_b, option_c, option_d: The multiple-choice options for the question.
- correct_answer: The correct option for the question (A, B, C, or D).

Constraints:

- FOREIGN KEY (exam_id): Links the question to the associated exam in the exams table. If the exam is deleted, its questions are also deleted (ON DELETE CASCADE).

The user_exams table tracks user attempts and scores for each exam.

Columns:

- user_exam_id: Unique identifier for each exam attempt (Primary Key).
- user_id: Foreign key linking the attempt to a user.
- exam_id: Foreign key linking the attempt to an exam.
- score: The score achieved by the user in the exam, defaulting to 0.
- timestamp: The date and time of the exam attempt.

Constraints:

- FOREIGN KEY (user_id): Links the attempt to a user in the users table.
- FOREIGN KEY (exam_id): Links the attempt to an exam in the exams table.
- UNIQUE (user_id, exam_id): Ensures that each user can attempt a specific exam only once.

This schema forms the backbone of an online exam system. It ensures seamless management of users, exams, and results while enforcing data integrity and supporting scalability.