Logo, company name

Description automatically generated

|  |
| --- |
| Arden University |
| Computing |
| Web Application Development |
| Web Application Development |
| STU101292 |
| Geethu Krishnan Mohan |
| 1311 |

Table of Contents

[Introduction 3](#_Toc158600285)

[Video Presentation 3](#_Toc158600286)

[Application Structure 3](#_Toc158600287)

[Entity-Relationship Diagram (ERD) 4](#_Toc158600288)

[Directory Structure and Component Interaction: 4](#_Toc158600289)

[Root Directory: 4](#_Toc158600290)

[Layered Interaction: 4](#_Toc158600291)

[Business Logic Layer 4](#_Toc158600292)

[Data Access Layer: 5](#_Toc158600293)

[Data Storage Layer: 5](#_Toc158600294)

[Component Interaction: 5](#_Toc158600295)

[Review and Recommendations 5](#_Toc158600296)

[Architecture and Code Structure: 5](#_Toc158600297)

[Recommendation: 5](#_Toc158600298)

[Functionality 5](#_Toc158600299)

[Recommendation: 6](#_Toc158600300)

[Usability and Accessibility 6](#_Toc158600301)

[Recommendation: 6](#_Toc158600302)

[Security 6](#_Toc158600303)

[Recommendation: 6](#_Toc158600304)

[Scalability 6](#_Toc158600305)

[Recommendation: 6](#_Toc158600306)

[References 6](#_Toc158600307)

[Appendices 7](#_Toc158600308)

[Entity Relationship Diagram (ERD) Description: 7](#_Toc158600309)

[MySQL Tables Code: 8](#_Toc158600310)

[Connection to MySQL databse 9](#_Toc158600311)

[config.php: 9](#_Toc158600312)

[database.php: 10](#_Toc158600313)

[Admin Panel 10](#_Toc158600314)

[add\_answer.php: 10](#_Toc158600315)

[add\_exam.php: 11](#_Toc158600316)

[add\_question.php: 12](#_Toc158600317)

[delete\_answer.php: 13](#_Toc158600318)

[delete\_exam.php: 14](#_Toc158600319)

[delete\_question.php: 15](#_Toc158600320)

[update\_answer.php: 15](#_Toc158600321)

[update\_exam.php: 16](#_Toc158600322)

[update\_question.php: 17](#_Toc158600323)

[admin.html: 17](#_Toc158600324)

[Login and Registration Forms 25](#_Toc158600325)

[register.php: 25](#_Toc158600326)

[login.php: 28](#_Toc158600327)

[Exam Interface(Selection, Loading, Submission) 30](#_Toc158600328)

[select\_exam.php: 30](#_Toc158600329)

[start\_exam.php: 33](#_Toc158600330)

[exam\_interface.php: 34](#_Toc158600331)

[load\_answer.php: 40](#_Toc158600332)

[load\_exam.php: 42](#_Toc158600333)

[load\_question.php: 42](#_Toc158600334)

[submit\_answer.php: 43](#_Toc158600335)

[exam\_results.php: 44](#_Toc158600336)

[Leaderboard 47](#_Toc158600337)

[leaderboard.php: 47](#_Toc158600338)

[Exam History 51](#_Toc158600339)

[past\_exams.php: 51](#_Toc158600340)

[Index.html: 53](#_Toc158600341)

[Homepage 54](#_Toc158600342)

[welcome.php: 54](#_Toc158600343)

[Navigation menu 56](#_Toc158600344)

[navigation.html: 56](#_Toc158600345)

[Dark and Light Mode 57](#_Toc158600346)

[mode.js: 57](#_Toc158600347)

[mode.css: 58](#_Toc158600348)

[Some of the error logs recorded in error\_log.php: 59](#_Toc158600349)

# Introduction

The web application, accessible through:

<https://www.stu101292.webhosting.arden.ac.uk/COM6011/index.html>, serves as a tool to enhance students learning and assessment process. This innovative application combines features like registration and login systems, a dynamic leaderboard, exam history review and a flexible theme switcher between light and dark modes catered to users’ visual preferences for comfortable viewing (Zammetti, 2020). Upon logging in, users are welcomed with a fetched username and a dashboard that provides access to functionalities. One of the standout features is the exams module, which allows users to choose from a variety of exams. Each exam comes with its set of questions and multiple-choice answers, accompanied by a thirty-minute timer to replicate real exam conditions. This module not only tests user’s knowledge, but also offers immediate feedback by highlighting correct answers in green and incorrect ones in red after submission. This instant feedback mechanism plays a role in strengthening learning and understanding. Additionally, the application includes an exam history feature that enables users to review their performance on exams. It provides details such as the date and time of each exam along, with a breakdown of incorrect answers. Furthermore, the leaderboard feature showcases rankings based on users’ performance in exams encouraging engagement and development.

Administrators have access to an interface that is not directly linked from the user interface. This interface can be reached at <https://www.stu101292.webhosting.arden.ac.uk/COM6011/admin.html>. It empowers administrators to manage exams, questions and answers through CRUD operations ensuring that the content remains up to date and relevant. The application relies on a MySQL database with exams dynamically loaded onto the website using JavaScripts Fetch API. This creates an interactive user experience (Wagner, 2017). Navigation throughout the site is facilitated by an integrated navigation page allowing users to easily move between sections without any confusion.

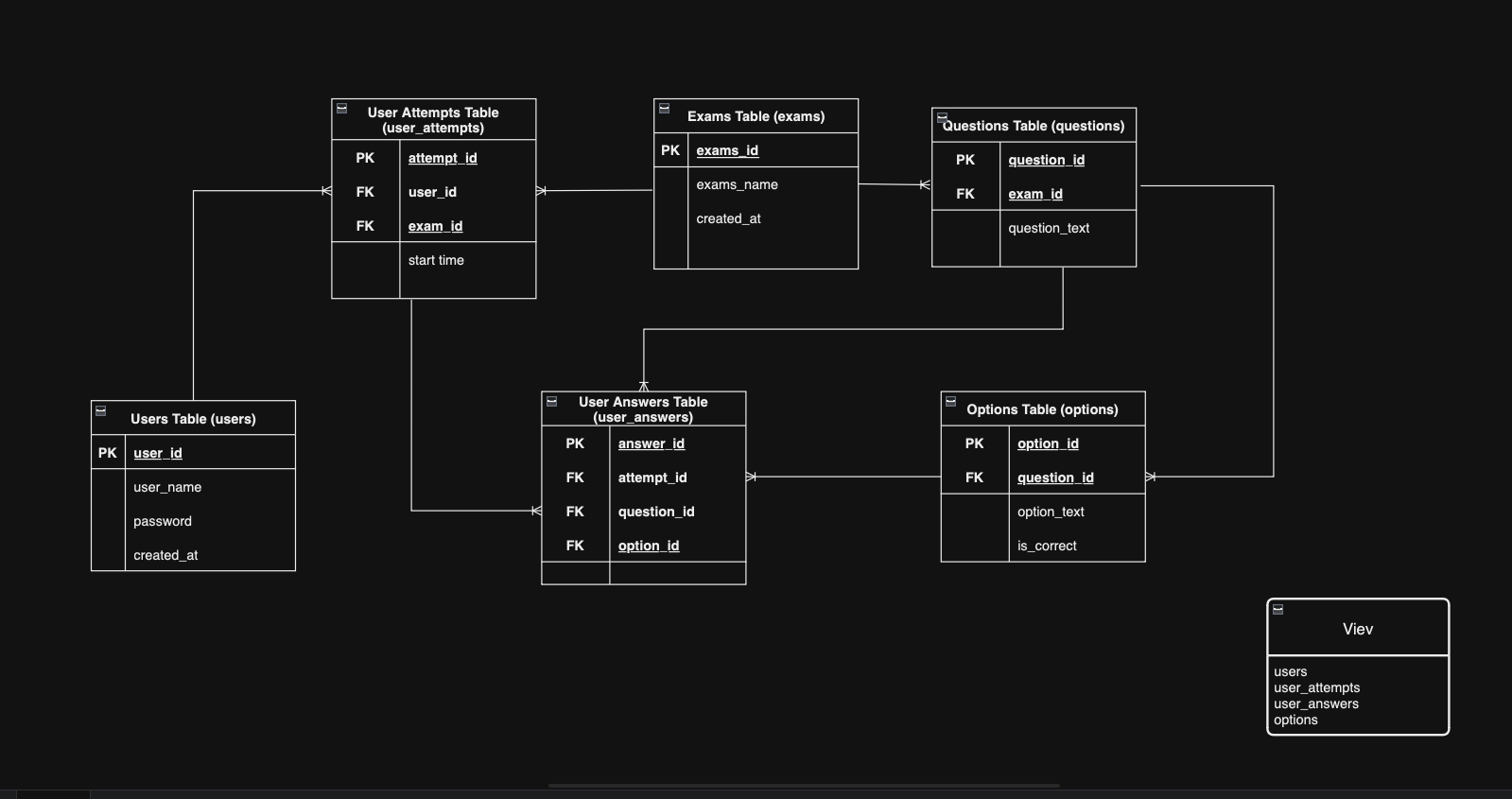
## Video Presentation

<https://www.youtube.com/watch?v=XGeHMb0CYNQ>

# Application Structure

The application follows an architecture based on the n-tier model carefully designed to separate presentation, business logic and data layers (Zammeti, 2020). This ensures a scalable and maintainable web solution. The file organisation has been specifically tailored to support functionality and enhance user experience. An interesting aspect of this application is its implementation of PHP based error logging mechanisms, which reflects an approach, towards application maintenance and security (Nixon, 2018). By runtime errors, exceptions and potentially harmful activities developers can promptly identify and resolve vulnerabilities to ensure that the application remains robust, against both operational challenges and security risks.

## Entity-Relationship Diagram (ERD)



## Directory Structure and Component Interaction:

The directory architecture of the application is simple yet effectively follows the n-tier design principle. It is primarily organised within a folder for management and simplicity. This setup reflects an approach to developing web applications, where the distinction between layers is maintained not in the code but also in physical file organization (Kleppmann, 2017).

## Root Directory:

It contains all PHP files that play roles such as rendering the user interface (presentation layer) handling business logic (business logic layer) and managing data persistence (data access layer) (Flanagan, 2020). This consolidation simplifies navigation and development for moderately complex applications. /css; This folder is dedicated to stylesheets for supporting light and dark mode options. It highlights the applications emphasis on user accessibility and preference (Meyer & Weyl, 2017). /js; Here you will find a JavaScript file specifically designed for toggling between dark modes. This demonstrates how the application utilises client-side scripting to enhance interactivity and improve user experience (Kalbag, 2021).

## Layered Interaction:

The Presentation Layer is the part that users directly interact with. It uses PHP files to display HTML structures, CSS for styling and JavaScript for features like the mode switch. The design approach of the application is focused on a single directory for simplicity, but with core functionality and separate folders for CSS and JavaScript related to the theme switch.

## Business Logic Layer

Embedded within the PHP files, the Business Logic Layer controls how the application operates. It takes care of tasks like validating user inputs, managing session states during exams and calculating scores (Nixon, 2018). This layer works closely with the presentation layer to provide a responsive user experience and interacts with the data layer to retrieve and store data (Zammetti, 2020).

## Data Access Layer:

Despite its file structure the application has concerns in how it handles data operations. PHP scripts are responsible for interacting with the MySQL database by executing queries (Flanagan, 2020). These queries save data needed for functionalities of the application, such as user registrations, exam history and leaderboard standings.

## Data Storage Layer:

The MySQL database serves as the foundation for storing information. User profiles, exam records and administrative data are all stored in this database. This particular layer plays a role, in enabling the applications interactive features, such as real time exam tracking and analysis of past performance (Kleppmann, 2017).

## Component Interaction:

The way JavaScript files interact with PHP scripts is an example of how the application loads data and updates the user interface especially for functions like exams loading. Through AJAX calls made with the help of the Fetch API data is dynamically requested from PHP scripts (Kalbag, 2021). These scripts then communicate with the MySQL database to retrieve or update information without having to reload the page. This architecture not only supports the functionality of the application but also provides a strong foundation for future enhancements.

# Review and Recommendations

## Architecture and Code Structure:

The applications adherence to an n-tier architecture ensures that concerns are well separated promoting scalability and ease of maintenance (Zammetti, 2020). By dividing it into layers for presentation, business logic and data access, a structured approach to web development is maintained. However, as more features are added and as the application grows in scale further optimisation may be required to maintain efficiency (Nixon, 2018).

### Recommendation:

Transitioning towards a microservices architecture could bring enhanced scalability and deployment flexibility (Chacon & Straub, 2014). This shift would support development of features while allowing components to scale based on demand. Ultimately this would ensure that the application remains resilient, under varying workloads.

## Functionality

The core features, including user registration, interactive quizzes with real time feedback and an admin panel for managing content make a solid foundation for a robust educational tool (Kleppmann, 2017). Furthermore, adding a time limit to complete the quizzes adds a challenge and enhances the user experience. However, to have an impact on education it would be beneficial to expand the functionality.

### Recommendation:

It would greatly improve outcomes by incorporating a learning model that adjusts quiz difficulty based on user performance. Additionally providing explanations for answers after completing the quiz would reinforce learning concepts and encourage user engagement.

## Usability and Accessibility

Regarding usability and accessibility, it is creditable that the application offers both dark and light themes to satisfy user preferences and ensure accessibility.

### Recommendation:

It is essential to prioritise compliance with Web Content Accessibility Guidelines (WCAG) for accommodating users with needs (Zammetti, 2020). This can be achieved by implementing features such as keyboard navigation, screen reader compatibility and customisable text sizes that promote inclusivity and provide a better user experience.

## Security

Considering security in digital age is crucial when handling user data and maintaining quiz content integrity. While there are already some security measures in place there is always room for strengthening these defenses.

### Recommendation:

Implementing encryption techniques to secure data storage and transmission along is crucial. It is also important to have protection against web vulnerabilities like SQL injection and XSS attacks. These measures will ensure the safety of user data and provide secure environment (Kleppmann, 2017).

## Scalability

Regarding scalability the current structure of the application supports initial user volumes and content. However, it is important to consider growth but also maintain performance and user satisfaction (Flanagan, 2021).

### Recommendation:

To address scalability concerns, leverage towards cloud services for dynamic resource allocation can solve scalability issues. In addition, using a Content Delivery Network (CDN) for static resources can reduce load times, and improve user experience globally (Wagner, 2017).

# References

*Chacon, S. and Straub, B. (2014) ‘Pro Git (2nd ed.)’, Apress.*

*Flanagan, D. (2020) ‘JavaScript: The Definitive Guide (7th ed.)’, O'Reilly Media.*

*Kalbag, L. (2021) ‘Accessible Web Design: A Guide for Creating Websites that Work for Everyone’, Smashing Magazine.*

*Kleppmann, M. (2017) ‘Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems’, O'Reilly Media.*

*Meyer, E.A. and Weyl, E. (2017) ‘CSS: The Definitive Guide (4th ed.)’, O'Reilly Media.*

*Nixon, R. (2018) ‘Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5 (5th ed.)’, O'Reilly Media.*

*Wagner, J.L. (2017) ‘Web Performance in Action: Building Faster Web Pages’, Manning Publications.*

*Zammetti, F. (2020) ‘Modern Full-Stack Development: Using TypeScript, React, Node.js, Webpack, and Docker’, Apress.*

# Appendices

## Entity Relationship Diagram (ERD) Description:

1. **User Table (‘users’):**

Primary Key: ‘user\_id’

Relationships: One-to-Many with ‘user\_attempts’: user can have multiple exam attempts

1. **Exams Table (‘exams’)**

Primary Key: ‘exam\_id’

Relationships: One-to-Many with ‘questions’: exam includes multiple questions

One-to-Many with ‘user\_attempts’: exam can be attempted by multiple users

1. **Question Table(‘questions’)**

Primary Key: ‘question\_id’

Foreign Key: ‘exam\_id’ references ‘exams.exam\_id’

Relationships: One-to-Many with ‘options’: A question can have multiple answer options

1. **Options Table(‘options’)**

Primary Key: ‘option\_id’

Foreign Key: ‘question\_id’ references ‘question.question\_id’

Attributes: Boolean ‘is\_correct’ to indicate correction option

1. **User Exam Attempts Table (‘user\_attempts’)**

Primary Key: ‘attempt\_id’

Foreign Keys: ‘user\_id’ references ‘users.user\_id’

‘exam\_id’ references ‘exams.exam\_id’

Relationships: Links users to their exam attempts, linking back to both ‘users’ and ‘exams’ tables

1. **User Answer Table (‘user\_answers’)**

Primary Key: ‘answer\_id’

Foreign Keys: ‘attempt\_id’ references ‘user\_attempts.attempt\_id’

‘question\_id’ references ‘questions.question\_id’

‘option\_id’ references ‘options.option\_id’

Relationships: Records the option selected by user for each question in exam attempts, linking through Many-to-One with relationship with ‘user\_attempts’, ‘questions’ and ‘options’

1. **Leaderboard View (‘leaderboard’)**

Aggregate data across ‘users’, ‘user\_attempts’, ‘user\_answers’ and ‘options’ to display ranking.

Joins ‘users’, ‘user\_attempts’ and ‘user\_answers’ with ‘options’ to calculate total and correct answers for each user per exam

## MySQL Tables Code:

-- Users Table

CREATE TABLE users (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Exams Table

CREATE TABLE exams (

exam\_id INT AUTO\_INCREMENT PRIMARY KEY,

exam\_name VARCHAR(100) NOT NULL,

duration INT NOT NULL, -- Duration in minutes

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Questions Table

CREATE TABLE questions (

question\_id INT AUTO\_INCREMENT PRIMARY KEY,

exam\_id INT NOT NULL,

question\_text TEXT NOT NULL,

FOREIGN KEY (exam\_id) REFERENCES exams(exam\_id)

);

-- Options Table

CREATE TABLE options (

option\_id INT AUTO\_INCREMENT PRIMARY KEY,

question\_id INT NOT NULL,

option\_text TEXT NOT NULL,

is\_correct BOOLEAN DEFAULT FALSE,

FOREIGN KEY (question\_id) REFERENCES questions(question\_id)

);

-- User Exam Attempts Table

CREATE TABLE user\_attempts (

attempt\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

exam\_id INT NOT NULL,

start\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(user\_id),

FOREIGN KEY (exam\_id) REFERENCES exams(exam\_id)

);

-- User Answers Table

CREATE TABLE user\_answers (

answer\_id INT AUTO\_INCREMENT PRIMARY KEY,

attempt\_id INT NOT NULL,

question\_id INT NOT NULL,

option\_id INT,

FOREIGN KEY (attempt\_id) REFERENCES user\_attempts(attempt\_id),

FOREIGN KEY (question\_id) REFERENCES questions(question\_id),

FOREIGN KEY (option\_id) REFERENCES options(option\_id)

);

-- Leaderboard View

CREATE VIEW leaderboard AS

SELECT

ua.exam\_id,

u.username,

COUNT(\*) AS exams\_taken,

SUM(CASE WHEN o.is\_correct = 1 THEN 1 ELSE 0 END) AS correct\_answers

FROM users u

JOIN user\_attempts ua ON u.user\_id = ua.user\_id

JOIN user\_answers a ON ua.attempt\_id = a.attempt\_id

JOIN options o ON a.option\_id = o.option\_id

GROUP BY ua.exam\_id, u.username;

## Connection to MySQL databse

### config.php:

<?php

// config.php: for connection to the database

return [

'mysql' => [

// The hostname of the database server.

'host' => 'localhost',

// The port number where the database server is listening.

'port' => '3306',

// The name of the database to connect to.

'name' => 'g46kbhv\_quiz\_app',

// The username used to authenticate with the database.

'user' => 'g46kbhv\_hrvoje',

// The password used to authenticate with the database.

'pass' => '10203040Hg',

// The character set to use for the database connection.

'charset' => 'utf8mb4'

]

];

?>

### database.php:

<?php

//databse.php

$config = require 'config.php';

// Extract the database configuration from $config

$dbConfig = $config['mysql'];

// Create a new mysqli connection

$con = new mysqli($dbConfig['host'], $dbConfig['user'], $dbConfig['pass'], $dbConfig['name']);

// Check connection

if ($con->connect\_error) {

die("Connection failed: " . $con->connect\_error);

}

?>

## Admin Panel

### add\_answer.php:

<?php

// add\_answer.php

// Include the database connection script.

include 'database.php';

// Check if the necessary POST data exists.

if (isset($\_POST['questionId']) && isset($\_POST['answerText']) && isset($\_POST['isCorrect'])) {

// Retrieve POST data and sanitise the 'isCorrect' flag as a boolean.

$questionId = $\_POST['questionId'];

$answerText = $\_POST['answerText'];

$isCorrect = filter\_var($\_POST['isCorrect'], FILTER\_VALIDATE\_BOOLEAN) ? 1 : 0;

// SQL query to insert a new answer into the 'options' table.

$query = "INSERT INTO options (question\_id, option\_text, is\_correct) VALUES (?, ?, ?)";

// Prepare the SQL statement for execution.

$stmt = $con->prepare($query);

// Check if statement preparation was successful.

if (!$stmt) {

// Log the error to the server's error log.

error\_log("Prepare failed: " . $con->error);

// Output an error message and exit the script.

echo "Error preparing statement: " . $con->error;

exit;

}

// Bind the input parameters to the prepared statement.

$stmt->bind\_param("isi", $questionId, $answerText, $isCorrect);

// Execute the prepared statement.

if ($stmt->execute()) {

// On successful execution, output a success message.

echo "Answer added successfully.";

} else {

// Log execution error and output an error message.

error\_log("Execute failed: " . $stmt->error);

echo "Error executing statement: " . $stmt->error;

}

// Close the statement and the database connection.

$stmt->close();

$con->close();

} else {

// Output an error message if the required POST data is not present.

echo "Required data not provided.";

}

?>

### add\_exam.php:

<?php

// add\_exam.php

// Include the database connection configuration.

include 'database.php';

// Check if the exam name has been provided via POST request.

if (isset($\_POST['examName'])) {

// Retrieve the exam name from POST data.

$examName = $\_POST['examName'];

// SQL query to insert a new exam into the 'exams' table.

$query = "INSERT INTO exams (exam\_name) VALUES (?)";

// Prepare the SQL statement for execution.

$stmt = $con->prepare($query);

// Check if the statement was prepared successfully, omitted for brevity.

// Bind the 'examName' variable to the prepared SQL statement.

$stmt->bind\_param("s", $examName);

// Execute the prepared statement.

if ($stmt->execute()) {

// Output a success message if the exam was added successfully.

echo "Exam added successfully.";

} else {

// Output an error message if the statement execution failed.

echo "Error: " . $stmt->error;

}

// Close the statement and the database connection to free up resources.

$stmt->close();

$con->close();

} else {

// Output an error message if no exam name was provided in the POST request.

echo "No exam name provided.";

}

?>

### add\_question.php:

<?php

// add\_question.php:

// Include the database connection script to establish a connection to the database.

include 'database.php';

// Check if both exam ID and question text have been provided through the POST request.

if (isset($\_POST['examId']) && isset($\_POST['questionText'])) {

// Assign POST data to variables for easier handling.

$examId = $\_POST['examId'];

$questionText = $\_POST['questionText'];

// Prepare an SQL query to insert the new question into the 'questions' table.

$query = "INSERT INTO questions (exam\_id, question\_text) VALUES (?, ?)";

// Prepare the SQL statement for execution, ensuring safe data handling.

$stmt = $con->prepare($query);

// Bind the variables to the prepared statement as parameters.

$stmt->bind\_param("is", $examId, $questionText);

// Execute the prepared statement and check if the execution was successful.

if ($stmt->execute()) {

// Output a success message if the question was successfully added.

echo "Question added successfully.";

} else {

// If execution fails, output an error message with the error detail.

echo "Error: " . $stmt->error;

}

// Close the statement and the database connection to free up resources.

$stmt->close();

$con->close();

} else {

// Output an error message if either the exam ID or question text was not provided.

echo "Exam ID or question text not provided.";

}

?>

### delete\_answer.php:

<?php

//delete\_answer.php

// Include database connection settings

include 'database.php';

// Check if an answer ID is provided in the POST request

if (isset($\_POST['answerId'])) {

// Assign the POSTed answerId to a variable

$answerId = $\_POST['answerId'];

// SQL query to delete the answer with the specified ID

$query = "DELETE FROM options WHERE option\_id = ?";

// Prepare the SQL statement for execution

$stmt = $con->prepare($query);

// Bind the answer ID to the prepared statement as an integer

$stmt->bind\_param("i", $answerId);

// Execute the prepared statement

if ($stmt->execute()) {

// Output success message if the answer is deleted successfully

echo "Answer deleted successfully.";

} else {

// Output error message if the deletion fails

echo "Error: " . $stmt->error;

}

// Close the statement and connection to free up resources

$stmt->close();

$con->close();

} else {

// Output error message if no answer ID is provided

echo "Answer ID not provided.";

}

?>

### delete\_exam.php:

<?php

//delete\_exam.php

// Include the database connection script

include 'database.php';

// Retrieve the exam ID from the POST data

$examId = $\_POST['examId'];

// SQL query to delete the exam with the specified ID

$query = "DELETE FROM exams WHERE exam\_id = ?";

// Prepare the SQL statement for execution

$stmt = $con->prepare($query);

// Bind the exam ID to the prepared statement as an integer

$stmt->bind\_param("i", $examId);

// Execute the prepared statement

if ($stmt->execute()) {

// Output success message if the exam is deleted successfully

echo "Exam deleted successfully.";

} else {

// Output error message if the deletion fails

echo "Error: " . $stmt->error;

}

// Close the statement and connection to free up resources

$stmt->close();

$con->close();

?>

### delete\_question.php:

<?php

//delete\_question.php

// Include the database connection configuration

include 'database.php';

// Check if a question ID has been provided through the POST request

if (isset($\_POST['questionId'])) {

// Retrieve the question ID from the POST data

$questionId = $\_POST['questionId'];

// SQL query to delete the question with the specified ID

$query = "DELETE FROM questions WHERE question\_id = ?";

// Prepare the SQL statement for safe execution

$stmt = $con->prepare($query);

// Bind the question ID to the prepared statement

$stmt->bind\_param("i", $questionId);

// Execute the prepared statement

if ($stmt->execute()) {

// Output success message if the question is deleted successfully

echo "Question deleted successfully.";

} else {

// Output error message if the deletion fails

echo "Error: " . $stmt->error;

}

// Close the statement and database connection to free up resources

$stmt->close();

$con->close();

} else {

// Output error message if no question ID is provided

echo "Question ID not provided.";

}

?>

### update\_answer.php:

<?php

// update\_answer.php

include 'database.php';

if (isset($\_POST['answerId']) && isset($\_POST['answerText']) && isset($\_POST['isCorrect'])) {

$answerId = $\_POST['answerId'];

$answerText = $\_POST['answerText'];

$isCorrect = $\_POST['isCorrect'] == 'true' ? 1 : 0;

$query = "UPDATE options SET option\_text = ?, is\_correct = ? WHERE option\_id = ?";

$stmt = $con->prepare($query);

$stmt->bind\_param("sii", $answerText, $isCorrect, $answerId);

if ($stmt->execute()) {

echo "Answer updated successfully.";

} else {

echo "Error: " . $stmt->error;

}

$stmt->close();

$con->close();

} else {

echo "Answer ID, text, or correctness not provided.";

}

?>

### update\_exam.php:

<?php

// update\_exam.php

include 'database.php';

if (isset($\_POST['examId']) && isset($\_POST['examName'])) {

$examId = $\_POST['examId'];

$examName = $\_POST['examName'];

$query = "UPDATE exams SET exam\_name = ? WHERE exam\_id = ?";

$stmt = $con->prepare($query);

$stmt->bind\_param("si", $examName, $examId);

if ($stmt->execute()) {

echo "Exam updated successfully.";

} else {

echo "Error: " . $stmt->error;

}

$stmt->close();

$con->close();

} else {

echo "Exam ID or name not provided.";

}

?>

### update\_question.php:

<?php

// update\_question.php

include 'database.php';

if (isset($\_POST['questionId']) && isset($\_POST['questionText'])) {

$questionId = $\_POST['questionId'];

$questionText = $\_POST['questionText'];

$query = "UPDATE questions SET question\_text = ? WHERE question\_id = ?";

$stmt = $con->prepare($query);

$stmt->bind\_param("si", $questionText, $questionId);

if ($stmt->execute()) {

echo "Question updated successfully.";

} else {

echo "Error: " . $stmt->error;

}

$stmt->close();

$con->close();

} else {

echo "Question ID or text not provided.";

}

?>

### admin.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Admin Panel</title>

<style>

/\* General styling for the page, using Arial font for consistency \*/

body {

font-family: Arial, sans-serif;

}

/\* Specific styling for headings to align text to the left and set the font size \*/

h2 {

text-align: left;

font-size: 40px;

}

/\* Container styling to centrally align the content with some margin \*/

.container {

width: 80%;

margin: auto;

}

/\* Section styling for logical grouping of content with margin for spacing \*/

.section {

margin-bottom: 20px;

}

/\* Styling for form elements to ensure they are fully width, padded for comfort, and margin for spacing \*/

select, .question, .answer {

width: 100%;

padding: 8px;

margin-bottom: 10px;

font-size: 15px;

}

/\* Cursor styling for question elements to indicate interactivity \*/

.questions {

cursor: pointer;

}

</style>

</head>

<body>

<div class="container">

<!-- Section for managing exams -->

<div class="section">

<h2>Manage Exams</h2>

<select id="examList" onchange="loadQuestions(this.value)">

<!-- Options for exams will be dynamically loaded here -->

</select>

<div class="center-buttons">

<button onclick="showAddExamForm()">Add Exam</button>

<button onclick="showEditExamForm()">Edit Exam</button>

<button onclick="deleteExam()">Delete Exam</button>

</div>

<!-- Hidden form for adding/editing exams -->

<div id="examForm" style="display: none;">

<input type="text" id="examName" placeholder="Exam Name">

<button onclick="addOrUpdateExam()">Submit</button>

</div>

</div>

<!-- Section for managing questions -->

<div class="section">

<h2>Manage Questions</h2>

<div id="questionManagement">

<button onclick="showAddQuestionForm()">Add Question</button>

<button onclick="showEditQuestionForm()">Edit Question</button>

<button onclick="deleteQuestion()">Delete Question</button>

<!-- Hidden form for adding/editing questions -->

<div id="questionForm" style="display: none;">

<textarea id="questionText" placeholder="Question Text"></textarea>

<button onclick="addOrUpdateQuestion()">Submit</button>

</div>

</div>

<div id="questionList" class="questions">

<!-- Dynamically loaded questions will appear here -->

</div>

</div>

<!-- Section for managing answers -->

<div class="section">

<h2>Manage Answers</h2>

<div id="answerManagement">

<button onclick="showAddAnswerForm()">Add Answer</button>

<button onclick="showEditAnswerForm()">Edit Answer</button>

<button onclick="deleteAnswer()">Delete Answer</button>

<!-- Hidden form for adding/editing answers -->

<div id="answerForm" style="display: none;">

<textarea id="answerText" placeholder="Answer Text"></textarea>

<label>

<input type="checkbox" id="isCorrect"> Correct Answer

</label>

<button onclick="addOrUpdateAnswer()">Submit</button>

</div>

</div>

<div id="answerList">

<!-- Dynamically loaded answers will be placed here -->

</div>

</div>

</div>

<script>

// JavaScript functions for CRUD operations on exams, questions, and answers

// Includes dynamic loading of exams, questions, and answers from the server

// and functions to show and process forms for adding, editing, and deleting entities.

function loadExams() {

fetch('load\_exams.php')

.then(response => response.json())

.then(data => {

const examList = document.getElementById('examList');

examList.innerHTML = data.map(exam => `<option value="${exam.id}">${exam.name}</option>`).join('');

})

.catch(error => console.error('Error:', error));

}

function loadQuestions(examId) {

fetch(`load\_questions.php?examId=${examId}`)

.then(response => response.json())

.then(data => {

const questionList = document.getElementById('questionList');

questionList.innerHTML = data.map(question =>

`<div class="question">

<input type="radio" name="selectedQuestion" value="${question.id}" onclick="selectQuestion(${question.id})">

${question.text}

</div>`

).join('');

})

.catch(error => console.error('Error:', error));

}

function selectQuestion(questionId) {

window.currentQuestionId = questionId;

loadAnswers(questionId);

}

function loadAnswers(questionId) {

const answerList = document.getElementById('answerList');

answerList.innerHTML = ''; // Clear existing answers

fetch(`load\_answers.php?questionId=${questionId}`)

.then(response => response.json())

.then(data => {

answerList.innerHTML = data.map(answer =>

`<div class="answer">

<input type="radio" name="selectedAnswer" value="${answer.id}" ${answer.correct ? 'data-is-correct="true"' : ''}>

${answer.text}

${answer.correct ? '<span>(Correct)</span>' : ''}

</div>`

).join('');

})

.catch(error => console.error('Error:', error));

}

document.addEventListener('DOMContentLoaded', function() {

loadExams();

});

// CRUD exams

function showAddExamForm() {

document.getElementById('examForm').style.display = 'block';

document.getElementById('examName').value = '';

window.currentExamId = null;

}

function showEditExamForm() {

var examId = document.getElementById('examList').value;

if (!examId) return alert('Please select an exam to edit.');

var examName = document.getElementById('examList').options[document.getElementById('examList').selectedIndex].text;

document.getElementById('examForm').style.display = 'block';

document.getElementById('examName').value = examName;

window.currentExamId = examId;

}

function addOrUpdateExam() {

var name = document.getElementById('examName').value;

var examId = window.currentExamId;

console.log('Adding/Updating Exam', { name, examId }); // Debugging line

var formData = new FormData();

formData.append('examName', name);

if (examId) {

formData.append('examId', examId);

}

var url = examId ? 'update\_exam.php' : 'add\_exam.php';

fetch(url, {

method: 'POST',

body: formData

}).then(response => response.text())

.then(result => {

console.log('Server Response:', result); // Debugging line

alert(result);

loadExams();

document.getElementById('examForm').style.display = 'none';

window.currentExamId = null;

}).catch(error => console.error('Error:', error));

}

function deleteExam() {

var examId = document.getElementById('examList').value;

if (!examId) return alert('Please select an exam to delete.');

if (confirm('Are you sure you want to delete this exam?')) {

var formData = new FormData();

formData.append('examId', examId);

fetch('delete\_exam.php', {

method: 'POST',

body: formData

}).then(response => response.text())

.then(result => {

alert(result);

loadExams();

}).catch(error => console.error('Error:', error));

}

}

// CRUD QUestions

function showAddQuestionForm() {

document.getElementById('questionForm').style.display = 'block';

document.getElementById('questionText').value = '';

window.currentQuestionId = null;

}

function showEditQuestionForm() {

var selectedQuestion = document.querySelector('input[name="selectedQuestion"]:checked');

if (!selectedQuestion) return alert('Please select a question to edit.');

var questionId = selectedQuestion.value;

document.getElementById('questionForm').style.display = 'block';

document.getElementById('questionText').value = selectedQuestion.nextSibling.textContent.trim();

window.currentQuestionId = questionId;

}

function addOrUpdateQuestion() {

var text = document.getElementById('questionText').value;

var questionId = window.currentQuestionId;

var examId = document.getElementById('examList').value;

var formData = new FormData();

formData.append('questionText', text);

formData.append('examId', examId);

if (questionId) {

formData.append('questionId', questionId);

}

var url = questionId ? 'update\_question.php' : 'add\_question.php';

fetch(url, {

method: 'POST',

body: formData

}).then(response => response.text())

.then(result => {

alert(result);

loadQuestions(examId);

document.getElementById('questionForm').style.display = 'none';

window.currentQuestionId = null;

}).catch(error => console.error('Error:', error));

}

function deleteQuestion() {

var selectedQuestion = document.querySelector('input[name="selectedQuestion"]:checked');

if (!selectedQuestion) return alert('Please select a question to delete.');

var questionId = selectedQuestion.value;

var examId = document.getElementById('examList').value;

if (confirm('Are you sure you want to delete this question?')) {

var formData = new FormData();

formData.append('questionId', questionId);

fetch('delete\_question.php', {

method: 'POST',

body: formData

}).then(response => response.text())

.then(result => {

alert(result);

loadQuestions(examId);

}).catch(error => console.error('Error:', error));

}

}

//CRUD Answers

function showAddAnswerForm() {

document.getElementById('answerForm').style.display = 'block';

document.getElementById('answerText').value = '';

document.getElementById('isCorrect').checked = false;

window.currentAnswerId = null;

}

function showEditAnswerForm() {

var selectedAnswer = document.querySelector('input[name="selectedAnswer"]:checked');

if (!selectedAnswer) return alert('Please select an answer to edit.');

var answerId = selectedAnswer.value;

document.getElementById('answerForm').style.display = 'block';

document.getElementById('answerText').value = selectedAnswer.nextSibling.textContent.trim();

document.getElementById('isCorrect').checked = selectedAnswer.dataset.isCorrect === 'true';

window.currentAnswerId = answerId;

}

function addOrUpdateAnswer() {

var text = document.getElementById('answerText').value;

var isCorrect = document.getElementById('isCorrect').checked;

var questionId = window.currentQuestionId;

var answerId = window.currentAnswerId;

// Basic validation

if (!questionId) {

alert("No question selected");

return;

}

var formData = new FormData();

formData.append('answerText', text);

formData.append('isCorrect', isCorrect);

formData.append('questionId', questionId); // Always send the question ID

// Determine if adding or updating based on if an answerId is present

var url = answerId ? 'update\_answer.php' : 'add\_answer.php';

if (answerId) {

formData.append('answerId', answerId); // Only append answerId if updating

}

fetch(url, {

method: 'POST',

body: formData

}).then(response => response.text())

.then(result => {

alert(result);

loadAnswers(questionId); // Refresh answers to show changes

document.getElementById('answerForm').style.display = 'none';

window.currentAnswerId = null; // Reset currentAnswerId

}).catch(error => console.error('Error:', error));

}

function deleteAnswer() {

var selectedAnswer = document.querySelector('input[name="selectedAnswer"]:checked');

if (!selectedAnswer) return alert('Please select an answer to delete.');

var answerId = selectedAnswer.value;

var questionId = window.currentQuestionId;

if (confirm('Are you sure you want to delete this answer?')) {

var formData = new FormData();

formData.append('answerId', answerId);

fetch('delete\_answer.php', {

method: 'POST',

body: formData

}).then(response => response.text())

.then(result => {

alert(result);

loadAnswers(questionId);

}).catch(error => console.error('Error:', error));

}

}

document.addEventListener('DOMContentLoaded', function() {

loadExams();

});

</script>

</body>

</html>

## Login and Registration Forms

### register.php:

<?php

session\_start(); // Start the session at the beginning of the script

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$config = include('config.php');

$dbConfig = $config['mysql'];

$pdo = new PDO("mysql:host={$dbConfig['host']};dbname={$dbConfig['name']};charset={$dbConfig['charset']}",

$dbConfig['user'],

$dbConfig['pass']);

$username = $\_POST['username'];

$password = password\_hash($\_POST['password'], PASSWORD\_DEFAULT);

// Check if username already exists

$stmt = $pdo->prepare("SELECT COUNT(\*) FROM users WHERE username = ?");

$stmt->execute([$username]);

if ($stmt->fetchColumn() > 0) {

echo "<p>Username already exists</p>";

} else {

$stmt = $pdo->prepare("INSERT INTO users (username, password) VALUES (?, ?)");

if ($stmt->execute([$username, $password])) {

// Set session variables and redirect to select\_exam.php

$\_SESSION['user\_id'] = $pdo->lastInsertId();

$\_SESSION['username'] = $username;

header("Location: welcome.php");

exit();

} else {

echo "<p>Error registering user</p>";

}

}

}

?>

<!DOCTYPE html>

<html>

<head>

<title>Register</title>

<link rel="stylesheet" href="css/mode.css">

<script src="js/mode.js"></script>

<style>

body {

font-family: Arial, sans-serif;

}

h2{

margin-top:10%;

text-align: center;

padding-bottom: 5%;

font-size: 40px;

}

form{

display: flex;

justify-content: center;

align-items: center;

flex-direction: column;

gap:10px;

}

label{

font-size:20px;

}

label[for="password"]{

padding-top:5%;

}

input{

padding: 10px;

font-size: 15px;

width:30%;

}

button{

cursor: pointer;

}

.submit-button{

font-size: 20px;

padding: 10px 20px;

border-radius: 8px;

border: 1px solid black;

color: black;

background-color: white;

cursor: pointer;

width:20%;

margin-top:5%;

}

.submit-button:active {

box-shadow: inset 0 3px 5px rgba(0, 0, 0, 0.125);

transform: translateY(2px);

}

/\* Media query for tablets \*/

@media only screen and (min-width: 768px) and (max-width: 1024px) {

h2{

font-size: 30px;

}

}

/\* Media query for mobile devices \*/

@media only screen and (max-width: 767px) {

.center-container{

flex-direction: column;

}

h2{

margin-top: 30%;

font-size: 30px;

}

input{

width:90%;

}

.submit-button{

font-size: 18px;

width: 80%;

}

}

</style>

</head>

<body>

<h2>Registration Form</h2>

<form method="post" action="register.php">

<label for="username">Username:</label>

<input type="text" id="username" name="username">

<label for="password">Password:</label>

<input type="password" id="password" name="password">

<input type="submit" value="Register" class="submit-button">

</form>

<button id="modeSwitch" class="switch">Switch to Dark Mode</button>

</body>

</html>

### login.php:

<?php

//login.php

session\_start(); // Start the session at the beginning of the script

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$config = include('config.php');

$dbConfig = $config['mysql'];

$pdo = new PDO("mysql:host={$dbConfig['host']};dbname={$dbConfig['name']};charset={$dbConfig['charset']}",

$dbConfig['user'],

$dbConfig['pass']);

$username = $\_POST['username'];

$password = $\_POST['password'];

$stmt = $pdo->prepare("SELECT user\_id, password FROM users WHERE username = ?");

$stmt->execute([$username]);

$user = $stmt->fetch(PDO::FETCH\_ASSOC);

if ($user && password\_verify($password, $user['password'])) {

// Set session variables and redirect to select\_exam.php

$\_SESSION['user\_id'] = $user['user\_id'];

$\_SESSION['username'] = $username;

header("Location: welcome.php");

exit();

} else {

echo "<p>Invalid username or password</p>";

}

}

?>

<!DOCTYPE html>

<html>

<head>

<title>Login</title>

<link rel="stylesheet" href="css/mode.css">

<script src="js/mode.js"></script>

<style>

body {

font-family: Arial, sans-serif;

}

h2{

margin-top:10%;

text-align: center;

padding-bottom: 5%;

font-size: 40px;

}

form{

display: flex;

justify-content: center;

align-items: center;

flex-direction: column;

gap:10px;

}

label{

font-size:20px;

}

label[for="password"]{

padding-top:5%;

}

input{

padding: 10px;

font-size: 15px;

width:30%;

}

button{

cursor: pointer;

}

.submit-button{

font-size: 20px;

padding: 10px 20px;

border-radius: 8px;

border: 1px solid black;

color: black;

background-color: white;

cursor: pointer;

width:20%;

margin-top:5%;

}

.submit-button:active {

box-shadow: inset 0 3px 5px rgba(0, 0, 0, 0.125);

transform: translateY(2px);

}

/\* Media query for tablets \*/

@media only screen and (min-width: 768px) and (max-width: 1024px) {

h2{

font-size: 30px;

}

}

/\* Media query for mobile devices \*/

@media only screen and (max-width: 767px) {

.center-container{

flex-direction: column;

}

h2{

margin-top: 30%;

font-size: 30px;

}

input{

width:90%;

}

.submit-button{

font-size: 18px;

width: 80%;

}

}

</style>

</head>

<body>

<h2>Login Form</h2>

<form method="post" action="login.php">

<label for="username">Username:</label>

<input type="text" id="username" name="username">

<label for="password">Password:</label>

<input type="password" id="password" name="password">

<input type="submit" value="Login" class="submit-button">

</form>

<button id="modeSwitch" class="switch">Switch to Dark Mode</button>

</body>

</html>

## Exam Interface(Selection, Loading, Submission)

### select\_exam.php:

<?php

session\_start();

// Check if the user is logged in

if (!isset($\_SESSION['user\_id'])) {

// Redirect to login page if not logged in

header("Location: login.php");

exit();

}

// Enable error reporting for debugging (remove/comment out in production)

ini\_set('display\_errors', 1);

error\_reporting(E\_ALL);

// Include the database configuration file

$config = include('config.php');

$dbConfig = $config['mysql'];

// Create a new PDO connection

try {

$pdo = new PDO("mysql:host={$dbConfig['host']};dbname={$dbConfig['name']};charset={$dbConfig['charset']}",

$dbConfig['user'],

$dbConfig['pass']);

} catch (PDOException $e) {

die("Could not connect to the database: " . $e->getMessage());

}

// Retrieve available exams

$stmt = $pdo->query("SELECT exam\_id, exam\_name FROM exams");

$exams = $stmt->fetchAll(PDO::FETCH\_ASSOC);

?>

<!DOCTYPE html>

<html>

<head>

<title>Select Exam</title>

<link rel="stylesheet" href="css/mode.css">

<script src="js/mode.js"></script>

<style>

body {

font-family: Arial, sans-serif;

}

h2{

margin-top:5%;

text-align: center;

padding-bottom: 5%;

font-size: 40px;

}

button{

cursor: pointer;

}

form{

display: flex;

justify-content: center;

align-items: center;

flex-direction: column;

}

/\* Style for the dropdown \*/

select {

width: 50%;

padding: 10px;

font-size: 20px;

margin-bottom: 5%;

border-radius: 8px;

border: 1px solid #ccc;

cursor:pointer;

}

/\* Style for the button \*/

input[type="submit"] {

padding: 10px 20px;

background-color: #4CAF50;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

font-size: 20px;

width:30%;

}

input[type="submit"]:active {

box-shadow: inset 0 3px 5px rgba(0, 0, 0, 0.125);

transform: translateY(2px);

}

/\* Media query for tablets \*/

@media only screen and (min-width: 768px) and (max-width: 1024px) {

h2{

font-size: 30px;

}

}

/\* Media query for mobile devices \*/

@media only screen and (max-width: 767px) {

.center-container{

flex-direction: column;

}

h2{

margin-top: 30%;

font-size: 30px;

}

select{

width: 90%;

}

input[type="submit"] {

width: 80%;

}

}

</style>

</head>

<body>

<?php include 'navigation.html'; ?>

<h2>Select an Exam</h2>

<form method='post' action='start\_exam.php'>

<select name='exam\_id'>

<?php foreach ($exams as $exam) {

echo "<option value='{$exam['exam\_id']}'>{$exam['exam\_name']}</option>";

} ?>

</select>

<input type='submit' value='Start Exam'>

</form>

<button id="modeSwitch" class="switch">Switch to Dark Mode</button>

</body>

</html>

### start\_exam.php:

<?php

session\_start();

//We need to ensure the user is logged in

if (!isset($\_SESSION['user\_id'])) {

header("Location: login.php"); // Redirect to login if not logged in

exit();

}

// Include the database configuration file

$config = include('config.php');

$dbConfig = $config['mysql'];

// Create a new PDO connection

try {

$pdo = new PDO("mysql:host={$dbConfig['host']};dbname={$dbConfig['name']};charset={$dbConfig['charset']}",

$dbConfig['user'],

$dbConfig['pass']);

} catch (PDOException $e) {

die("Could not connect to the database: " . $e->getMessage());

}

// Check if an exam is selected

if ($\_SERVER["REQUEST\_METHOD"] == "POST" && isset($\_POST['exam\_id'])) {

$exam\_id = $\_POST['exam\_id'];

$user\_id = $\_SESSION['user\_id'];

// Record the user's attempt

$stmt = $pdo->prepare("INSERT INTO user\_attempts (user\_id, exam\_id) VALUES (?, ?)");

$stmt->execute([$user\_id, $exam\_id]);

$attempt\_id = $pdo->lastInsertId();

// Redirect to exam interface with the first question

header("Location: exam\_interface.php?attempt\_id={$attempt\_id}");

exit();

} else {

die("No exam selected.");

}

?>

### exam\_interface.php:

<?php

//exam\_interface.php

// Start the session to enable user tracking across different pages.

session\_start();

// Check if the user is not logged in or if the attempt ID is not provided. Redirect to login if either is true.

if (!isset($\_SESSION['user\_id']) || !isset($\_GET['attempt\_id'])) {

header("Location: login.php");

exit();

}

// Assign the quiz attempt ID from the URL to the session for later use.

$\_SESSION['attempt\_id'] = $\_GET['attempt\_id'];

// Include the database configuration settings.

$config = include('config.php');

$dbConfig = $config['mysql'];

// Try connecting to the database with PDO.

try {

$pdo = new PDO("mysql:host={$dbConfig['host']};dbname={$dbConfig['name']};charset={$dbConfig['charset']}",

$dbConfig['user'],

$dbConfig['pass']);

} catch (PDOException $e) {

// If connection fails, end script and display error message.

die("Could not connect to the database: " . $e->getMessage());

}

// Retrieve the attempt ID from the URL to fetch questions.

$attempt\_id = $\_GET['attempt\_id'];

// Prepare and execute a query to get the next question based on the user's current exam attempt.

$stmt = $pdo->prepare("SELECT question\_id FROM questions WHERE exam\_id = (SELECT exam\_id FROM user\_attempts WHERE attempt\_id = ?) AND question\_id NOT IN (SELECT question\_id FROM user\_answers WHERE attempt\_id = ?) LIMIT 1");

$stmt->execute([$attempt\_id, $attempt\_id]);

// Fetch the result and determine the next question's ID.

$current\_question\_result = $stmt->fetch(PDO::FETCH\_ASSOC);

$current\_question\_id = $current\_question\_result ? $current\_question\_result['question\_id'] : null;

// If no next question is found, redirect to the exam results page.

if (!$current\_question\_id) {

header("Location: exam\_results.php");

exit();

}

// Retrieve the text for the current question.

$stmt = $pdo->prepare("SELECT question\_text FROM questions WHERE question\_id = ?");

$stmt->execute([$current\_question\_id]);

$question = $stmt->fetch(PDO::FETCH\_ASSOC);

// Retrieve all the options for the current question.

$stmt = $pdo->prepare("SELECT option\_id, option\_text FROM options WHERE question\_id = ?");

$stmt->execute([$current\_question\_id]);

$options = $stmt->fetchAll(PDO::FETCH\_ASSOC);

// Include the navigation bar HTML template.

include 'navigation.html';

?>

<!DOCTYPE html>

<html>

<head>

<title>Exam</title>

<link rel="stylesheet" href="css/mode.css">

<script src="js/mode.js"></script>

<style>

.correct { background-color: green; }

.incorrect { background-color: red; }

body {

font-family: Arial, sans-serif;

}

h2{

margin-top:5%;

text-align: center;

padding-bottom: 5%;

font-size: 40px;

}

.question-selection{

text-align: center;

font-size: 30px;

padding-bottom:1%;

}

button{

cursor: pointer;

}

form{

text-align: center;

font-size: 30px;

padding: 1%;

}

#questionForm {

display: flex;

flex-direction: column;

align-items: center; /\* Center the form items \*/

width: 100%;

}

.button-container, .next-button-container {

width: 100%;

display: flex;

justify-content: center;

margin-top: 1%;

}

.next-button-container {

justify-content: flex-end;

display: none;

}

#nextButton {

margin-left: auto;

width: auto;

}

input[type="radio"]{

cursor:pointer;

}

.button-container input[type="button"], .next-button-container input[type="button"]{

padding: 10px 20px;

background-color: #4CAF50;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

font-size: 20px;

width:20%;

width:30%;

}

input[type="button"]:active {

box-shadow: inset 0 3px 5px rgba(0, 0, 0, 0.125);

transform: translateY(2px);

}

#time{

text-align: center;

font-size: 20px;

}

</style>

</head>

<body>

<h2>Exam</h2>

<div class="question-selection">Question: <?php echo htmlspecialchars($question['question\_text']); ?></div>

<form id="questionForm">

<?php foreach ($options as $option): ?>

<label>

<input type="radio" name="answer" value="<?php echo $option['option\_id']; ?>">

<?php echo htmlspecialchars($option['option\_text']); ?>

</label><br>

<?php endforeach; ?>

<div class="button-container">

<input type="button" value="Submit" onclick="submitAnswer()">

</div>

<div class="next-button-container" style="display: none;">

<input type="button" id="nextButton" value="Next Question" onclick="loadNextQuestion()">

</div>

</form>

<div id='time'></div>

<button id="modeSwitch" class="switch">Switch to Dark Mode</button>

<script>

// Global variables

let remainingTime = sessionStorage.getItem('remainingTime') ? parseInt(sessionStorage.getItem('remainingTime')) : 60 \* 30; // 30 minutes

let timerInterval;

function isNewTest() {

let currentAttemptId = '<?php echo $\_SESSION['attempt\_id']; ?>';

let storedAttemptId = sessionStorage.getItem('lastAttemptId');

return !storedAttemptId || currentAttemptId !== storedAttemptId;

}

function startTimer() {

let remainingTime = sessionStorage.getItem('remainingTime') ? parseInt(sessionStorage.getItem('remainingTime')) : 60 \* 30; // 30 minutes in seconds

let timerInterval;

var display = document.querySelector('#time');

timerInterval = setInterval(function () {

let minutes = parseInt(remainingTime / 60, 10);

let seconds = parseInt(remainingTime % 60, 10);

minutes = minutes < 10 ? '0' + minutes : minutes;

seconds = seconds < 10 ? '0' + seconds : seconds;

display.textContent = minutes + ':' + seconds;

if (--remainingTime < 0) {

clearInterval(timerInterval);

alert("Time's up!");

window.location.href = 'exam\_results.php'; // Redirect to results page

}

sessionStorage.setItem('remainingTime', remainingTime);

}, 1000);

}

window.onload = function () {

if (isNewTest()) {

sessionStorage.setItem('remainingTime', 60 \* 30); // Reset timer for new test

sessionStorage.setItem('lastAttemptId', '<?php echo $\_SESSION['attempt\_id']; ?>');

}

startTimer();

};

function submitAnswer() {

var formData = new FormData(document.getElementById('questionForm'));

formData.append('question\_id', '<?php echo $current\_question\_id; ?>');

formData.append('attempt\_id', '<?php echo $\_SESSION['attempt\_id']; ?>');

fetch('submit\_answer.php', {

method: 'POST',

body: formData

})

.then(response => response.json())

.then(data => {

updateFeedback(data.is\_correct, data.correct\_option\_id);

if (data.status === 'more\_questions') {

document.getElementById('nextButton').style.display = 'block';

document.getElementById('nextButton').onclick = function() { loadNextQuestion(data.next\_question\_id); };

} else if (data.status === 'exam\_completed') {

setTimeout(() => window.location.href = 'exam\_results.php', 2000);

}

})

.catch(error => console.error('Error:', error));

document.querySelector('.next-button-container').style.display = 'flex';

}

function loadNextQuestion(nextQuestionId) {

window.location.href = 'exam\_interface.php?attempt\_id=' + '<?php echo $\_SESSION['attempt\_id']; ?>' + '&question\_id=' + nextQuestionId;

}

function updateFeedback(isCorrect, correctOptionId) {

let selectedOption = document.querySelector('input[name="answer"]:checked');

if (selectedOption) {

let parentLabel = selectedOption.parentNode;

parentLabel.style.backgroundColor = isCorrect ? 'green' : 'red';

if (!isCorrect) {

// Highlight the correct answer in green

let correctOption = document.querySelector('input[value="' + correctOptionId + '"]');

if (correctOption) {

let correctLabel = correctOption.parentNode;

correctLabel.style.backgroundColor = 'green';

}

}

}

}

</script>

</body>

</html>

### load\_answer.php:

<?php

//load\_answer.php

// Load the PHP file for database connection settings

include 'database.php';

// Check if a question ID is provided in the query string

if (isset($\_GET['questionId'])) {

// Store the provided question ID from the query string into a variable

$questionId = $\_GET['questionId'];

// SQL query to select options related to the question ID

$query = "SELECT option\_id, option\_text, is\_correct FROM options WHERE question\_id = ?";

// Prepare the SQL query for execution

$stmt = $con->prepare($query);

// Bind the question ID to the placeholder in the SQL query

$stmt->bind\_param("i", $questionId);

// Execute the prepared statement

$stmt->execute();

// Get the result of the query execution

$result = $stmt->get\_result();

// Initialize an empty array to store the answers

$answers = [];

// Fetch each row from the result set as an associative array

while ($row = $result->fetch\_assoc()) {

// Add the option details to the answers array

$answers[] = ['id' => $row['option\_id'], 'text' => $row['option\_text'], 'is\_correct' => $row['is\_correct']];

}

// Encode the answers array as JSON and output it

echo json\_encode($answers);

// Close the statement

$stmt->close();

// Close the database connection

$con->close();

} else {

// Output an error message if the question ID was not provided

echo "Question ID not provided.";

}

?>

<?php

//load\_answer.php

// Load the PHP file for database connection settings

include 'database.php';

// Check if a question ID is provided in the query string

if (isset($\_GET['questionId'])) {

// Store the provided question ID from the query string into a variable

$questionId = $\_GET['questionId'];

// SQL query to select options related to the question ID

$query = "SELECT option\_id, option\_text, is\_correct FROM options WHERE question\_id = ?";

// Prepare the SQL query for execution

$stmt = $con->prepare($query);

// Bind the question ID to the placeholder in the SQL query

$stmt->bind\_param("i", $questionId);

// Execute the prepared statement

$stmt->execute();

// Get the result of the query execution

$result = $stmt->get\_result();

// Initialize an empty array to store the answers

$answers = [];

// Fetch each row from the result set as an associative array

while ($row = $result->fetch\_assoc()) {

// Add the option details to the answers array

$answers[] = ['id' => $row['option\_id'], 'text' => $row['option\_text'], 'is\_correct' => $row['is\_correct']];

}

// Encode the answers array as JSON and output it

echo json\_encode($answers);

// Close the statement

$stmt->close();

// Close the database connection

$con->close();

} else {

// Output an error message if the question ID was not provided

echo "Question ID not provided.";

}

?>

### load\_exam.php:

<?php

//load\_exam.php

// Include the database connection settings

include 'database.php';

// SQL query to select all exams from the 'exams' table

$query = "SELECT exam\_id, exam\_name FROM exams";

// Execute the query and store the result

$result = $con->query($query);

// Initialise an empty array to store exams

$exams = [];

// Loop through each row in the result set

while($row = $result->fetch\_assoc()) {

// Add each exam's ID and name to the exams array

$exams[] = ['id' => $row['exam\_id'], 'name' => $row['exam\_name']];

}

// Encode the exams array as JSON and output it

echo json\_encode($exams);

// Close the database connection

$con->close();

?>

### load\_question.php:

<?php

// load\_questions.php

include 'database.php';

$examId = isset($\_GET['examId']) ? $\_GET['examId'] : 0;

$query = "SELECT question\_id, question\_text FROM questions WHERE exam\_id = ?";

$stmt = $con->prepare($query);

// Bind the exam ID to the query parameter

$stmt->bind\_param("i", $examId);

// Execute the prepared statement

$stmt->execute();

// Get the result of the query

$result = $stmt->get\_result();

// Initialize an empty array to store questions

$questions = [];

// Loop through each row in the result set

while($row = $result->fetch\_assoc()) {

// Add each question's ID and text to the questions array

$questions[] = ['id' => $row['question\_id'], 'text' => $row['question\_text']];

}

// Encode the questions array as JSON and output it

echo json\_encode($questions);

// Close the prepared statement

$stmt->close();

// Close the database connection

$con->close();

?>

### submit\_answer.php:

<?php

session\_start();

header('Content-Type: application/json');

$response = ['status' => 'error', 'message' => 'An unknown error occurred'];

if (!isset($\_SESSION['user\_id']) || !isset($\_POST['question\_id']) || !isset($\_POST['answer']) || !isset($\_POST['attempt\_id'])) {

$response['message'] = 'Invalid access.';

echo json\_encode($response);

exit();

}

$config = include('config.php');

$dbConfig = $config['mysql'];

try {

$pdo = new PDO("mysql:host={$dbConfig['host']};dbname={$dbConfig['name']};charset={$dbConfig['charset']}",

$dbConfig['user'],

$dbConfig['pass']);

$question\_id = htmlspecialchars($\_POST['question\_id']);

$answer = htmlspecialchars($\_POST['answer']);

$attempt\_id = htmlspecialchars($\_POST['attempt\_id']);

// Record the answer

$stmt = $pdo->prepare("INSERT INTO user\_answers (attempt\_id, question\_id, option\_id) VALUES (?, ?, ?)");

$stmt->execute([$attempt\_id, $question\_id, $answer]);

// Check if the answer is correct and fetch the correct answer's ID

$checkAnswerStmt = $pdo->prepare("SELECT option\_id, is\_correct FROM options WHERE question\_id = ?");

$checkAnswerStmt->execute([$question\_id]);

$isCorrect = false;

$correctOptionId = null;

while ($row = $checkAnswerStmt->fetch(PDO::FETCH\_ASSOC)) {

if ($row['is\_correct']) {

$correctOptionId = $row['option\_id'];

$isCorrect = $row['option\_id'] == $answer;

}

}

// Fetch next question ID

$nextQuestionStmt = $pdo->prepare("SELECT question\_id FROM questions WHERE exam\_id = (SELECT exam\_id FROM user\_attempts WHERE attempt\_id = ?) AND question\_id NOT IN (SELECT question\_id FROM user\_answers WHERE attempt\_id = ?) LIMIT 1");

$nextQuestionStmt->execute([$attempt\_id, $attempt\_id]);

$next\_question = $nextQuestionStmt->fetch(PDO::FETCH\_ASSOC);

if ($next\_question) {

$response = [

'status' => 'more\_questions',

'next\_question\_id' => $next\_question['question\_id'],

'is\_correct' => $isCorrect,

'correct\_option\_id' => $correctOptionId

];

} else {

$response = [

'status' => 'exam\_completed',

'is\_correct' => $isCorrect,

'correct\_option\_id' => $correctOptionId

];

}

} catch (PDOException $e) {

$response['message'] = 'Database connection failed: ' . $e->getMessage();

}

echo json\_encode($response);

?>

### exam\_results.php:

<?php

session\_start();

//exam\_results.php

// Check if attempt\_id is set in the session

if (!isset($\_SESSION['attempt\_id'])) {

die("No exam attempt ID found in session.");

}

$attempt\_id = $\_SESSION['attempt\_id'];

// Include the database configuration file

$config = include('config.php');

$dbConfig = $config['mysql'];

try {

// Create a new PDO connection

$pdo = new PDO("mysql:host={$dbConfig['host']};dbname={$dbConfig['name']};charset={$dbConfig['charset']}",

$dbConfig['user'],

$dbConfig['pass']);

// Prepare and execute the query

$stmt = $pdo->prepare("SELECT ua.question\_id, ua.option\_id, o.is\_correct

FROM user\_answers ua

JOIN options o ON ua.option\_id = o.option\_id

WHERE ua.attempt\_id = ? AND ua.question\_id = o.question\_id");

$stmt->execute([$attempt\_id]);

$answers = $stmt->fetchAll(PDO::FETCH\_ASSOC);

// Count correct and incorrect answers

$correct = 0;

$incorrect = 0;

foreach ($answers as $answer) {

if ($answer['is\_correct']) {

$correct++;

} else {

$incorrect++;

}

}

} catch (PDOException $e) {

die("Database error: " . $e->getMessage());

}

include 'navigation.html';

?>

<!DOCTYPE html>

<html>

<head>

<title>Exam Results</title>

<link rel="stylesheet" href="css/mode.css">

<script src="js/mode.js"></script>

<style>

body {

font-family: Arial, sans-serif;

}

h2{

margin-top:5%;

text-align: center;

padding-bottom: 5%;

font-size: 40px;

}

.results {

margin-top: 20px;

}

.correct {

color: green;

}

.incorrect {

color: red;

}

p{

text-align: center;

font-size: 30px;

}

/\* Media query for tablets \*/

@media only screen and (min-width: 768px) and (max-width: 1024px) {

h2{

font-size: 30px;

}

p{

font-size: 25px;

}

}

/\* Media query for mobile devices \*/

@media only screen and (max-width: 767px) {

h2{

margin-top: 30%;

}

p{

font-size: 20px;

}

}

</style>

</head>

<body>

<h2>Exam Results</h2>

<div class="results">

<p class="correct">Correct Answers: <?php echo $correct; ?></p>

<p class="incorrect">Incorrect Answers: <?php echo $incorrect; ?></p>

</div>

<button id="modeSwitch" class="switch">Switch to Dark Mode</button>

</body>

</html>

## Leaderboard

### leaderboard.php:

<!DOCTYPE html>

<html>

<head>

<title>Leaderboard</title>

<!-- Link to external CSS file for styling -->

<link rel="stylesheet" href="css/mode.css">

<!-- Link to external JavaScript file for functionality -->

<script src="js/mode.js"></script>

<style>

/\* Basic styling for the body to use Arial font \*/

body {

font-family: Arial, sans-serif;

}

/\* Styling for the heading, including its alignment and spacing \*/

h2 {

margin-top:5%;

text-align: center;

padding-bottom: 5%;

font-size: 40px;

}

/\* Ensures the cursor changes to pointer on button hover \*/

button {

cursor: pointer;

}

/\* Flexbox layout for table for center alignment \*/

table {

display: flex;

justify-content: center;

align-items: center;

font-size: 30px;

}

/\* Flexbox layout for forms for center alignment \*/

form {

display: flex;

justify-content: center;

align-items: center;

font-size: 30px;

}

/\* Styles for submit input, including button appearance \*/

input[type="submit"] {

cursor: pointer;

font-size: 20px;

padding: 10px 20px;

background-color: #4CAF50;

color: white;

border: none;

border-radius: 5px;

margin-bottom: 1%;

}

/\* Additional styling for submit input when active/pressed \*/

input[type="submit"]:active {

box-shadow: inset 0 3px 5px rgba(0, 0, 0, 0.125);

transform: translateY(2px);

}

/\* Styling for dropdown select elements \*/

select {

width: 30%;

padding: 10px;

font-size: 20px;

margin-bottom: 5%;

border-radius: 8px;

border: 1px solid #ccc;

cursor:pointer;

margin-bottom: 1%;

margin-right: 1%;

}

/\* Basic table styling, including border and text alignment \*/

td, th {

border: 1px solid;

border-collapse: collapse;

padding: 5px;

text-align: left;

}

/\* Media query adjustments for tablet devices \*/

@media only screen and (min-width: 768px) and (max-width: 1024px) {

h2{

font-size: 30px;

}

table {

font-size: 25px;

}

}

/\* Media query adjustments for mobile devices \*/

@media only screen and (max-width: 767px) {

h2{

margin-top: 30%;

}

select{

width: 90%;

}

input[type="submit"] {

width: 80%;

}

table {

font-size: 20px;

}

}

</style>

</head>

<body>

<!-- Include navigation HTML snippet -->

<?php include 'navigation.html'; ?>

<!-- Heading indicating this page is a leaderboard -->

<h2>Leaderboard</h2>

<!-- PHP script to fetch configuration and establish database connection -->

<?php

$config = include('config.php');

$dbConfig = $config['mysql'];

$pdo = new PDO("mysql:host={$dbConfig['host']};dbname={$dbConfig['name']};charset={$dbConfig['charset']}", $dbConfig['user'], $dbConfig['pass']);

$selected\_exam\_id = isset($\_GET['exam\_id']) ? $\_GET['exam\_id'] : 1;

$examStmt = $pdo->query("SELECT exam\_id, exam\_name FROM exams");

$exams = $examStmt->fetchAll(PDO::FETCH\_ASSOC);

?>

<!-- Form for selecting an exam to view its leaderboard -->

<form action="" method="get">

<select name="exam\_id">

<!-- Dynamically generate options based on available exams -->

<?php foreach ($exams as $exam): ?>

<option value="<?php echo $exam['exam\_id']; ?>"<?php echo $exam['exam\_id'] == $selected\_exam\_id ? " selected" : ""; ?>><?php echo htmlspecialchars($exam['exam\_name']); ?></option>

<?php endforeach; ?>

</select>

<input type="submit" value="Show Leaderboard">

</form>

<!-- PHP script to prepare and execute query for leaderboard data -->

<?php

$stmt = $pdo->prepare("SELECT

u.username AS name,

COUNT(DISTINCT ua.attempt\_id) AS exams\_taken,

SUM(CASE WHEN o.is\_correct = 1 THEN 1 ELSE 0 END) AS correct\_answers,

SUM(CASE WHEN o.is\_correct = 0 THEN 1 ELSE 0 END) AS incorrect\_answers

FROM users u

JOIN user\_attempts ua ON u.user\_id = ua.user\_id

JOIN user\_answers a ON ua.attempt\_id = a.attempt\_id

JOIN options o ON a.option\_id = o.option\_id

WHERE ua.exam\_id = :exam\_id

GROUP BY u.username

ORDER BY exams\_taken DESC, correct\_answers DESC, incorrect\_answers ASC");

$stmt->execute(['exam\_id' => $selected\_exam\_id]);

$leaderboard = $stmt->fetchAll(PDO::FETCH\_ASSOC);

?>

<!-- Displaying the leaderboard in a table format -->

<table>

<tr>

<th>Name</th>

<th>Exams Taken</th>

<th>Correct Answers</th>

<th>Incorrect Answers</th>

</tr>

<!-- Loop through each leaderboard entry and display its data -->

<?php foreach ($leaderboard as $entry): ?>

<tr>

<td><?php echo htmlspecialchars($entry['name']); ?></td>

<td><?php echo htmlspecialchars($entry['exams\_taken']); ?></td>

<td><?php echo htmlspecialchars($entry['correct\_answers']); ?></td>

<td><?php echo htmlspecialchars($entry['incorrect\_answers']); ?></td>

</tr>

<?php endforeach; ?>

</table>

<!-- Button to switch the page to dark mode, functionality handled by external JS -->

<button id="modeSwitch" class="switch">Switch to Dark Mode</button>

</body>

</html>

## Exam History

### past\_exams.php:

<?php

//past\_exams.php

session\_start();

// Include the database configuration file

$config = include('config.php');

$dbConfig = $config['mysql'];

$pdo = new PDO("mysql:host={$dbConfig['host']};dbname={$dbConfig['name']};charset={$dbConfig['charset']}",

$dbConfig['user'],

$dbConfig['pass']);

$user\_id = $\_SESSION['user\_id'];

// Modified SQL query to fetch exam name, start time, and count of correct/incorrect answers

$stmt = $pdo->prepare("SELECT e.exam\_name, ua.start\_time,

SUM(o.is\_correct = 1) AS correct\_answers,

SUM(o.is\_correct = 0) AS incorrect\_answers

FROM user\_attempts ua

JOIN exams e ON ua.exam\_id = e.exam\_id

LEFT JOIN user\_answers a ON ua.attempt\_id = a.attempt\_id

LEFT JOIN options o ON a.option\_id = o.option\_id

WHERE ua.user\_id = ?

GROUP BY ua.attempt\_id, e.exam\_name, ua.start\_time");

$stmt->execute([$user\_id]);

$attempts = $stmt->fetchAll(PDO::FETCH\_ASSOC);

include 'navigation.html';

?>

<!DOCTYPE html>

<html>

<head>

<title>Past Exams</title>

<link rel="stylesheet" href="css/mode.css">

<script src="js/mode.js"></script>

<style>

body {

font-family: Arial, sans-serif;

}

h2{

margin-top:5%;

text-align: center;

padding-bottom: 5%;

font-size: 40px;

}

.center-exam{

font-size: 30px;

text-align: center;

}

button {

cursor: pointer;

}

/\* Media query for tablets \*/

@media only screen and (min-width: 768px) and (max-width: 1024px) {

h2{

font-size: 30px;

}

.center-exam{

font-size: 25px;

}

}

/\* Media query for mobile devices \*/

@media only screen and (max-width: 767px) {

h2{

margin-top: 30%;

}

.center-exam{

font-size: 20px;

}

}

</style>

</head>

<body>

<h2>Past Exams</h2>

<div class=center-exam>

<?php foreach ($attempts as $attempt): ?>

<p>

Exam: <?php echo htmlspecialchars($attempt['exam\_name']); ?> <br>

Started on: <?php echo htmlspecialchars($attempt['start\_time']); ?> <br>

Correct Answers: <?php echo $attempt['correct\_answers']; ?> <br>

Incorrect Answers: <?php echo $attempt['incorrect\_answers']; ?>

</p>

<?php endforeach; ?>

</div>

<button id="modeSwitch" class="switch">Switch to Dark Mode</button>

</body>

</html>

## Index.html:

<!DOCTYPE html>

<html>

<head>

<title>Quiz Application</title>

<link rel="stylesheet" href="css/mode.css">

<script src="js/mode.js"></script>

<style>

body {

font-family: Arial, sans-serif;

}

h1{

text-align: center;

margin-top: 10%;

padding-bottom: 5%;

font-size: 60px;

}

.center-container {

display: flex;

justify-content: center;

align-items: center;

flex-direction: column;

}

.center-container a {

display: inline-block;

width: 20%;

margin: 10px;

}

button{

cursor: pointer;

}

.reg-button{

font-size: 20px;

padding: 10px 20px;

border-radius: 8px;

border: 1px solid black;

color: black;

background-color: white;

width: 100%;

}

.reg-button:active {

box-shadow: inset 0 3px 5px rgba(0, 0, 0, 0.125);

transform: translateY(2px);

}

/\* Media query for tablets \*/

@media only screen and (min-width: 768px) and (max-width: 1024px) {

h1{

font-size: 40px;

}

}

/\* Media query for mobile devices \*/

@media only screen and (max-width: 767px) {

.center-container{

flex-direction: column;

}

h1{

margin-top: 30%;

font-size: 30px;

}

.center-container a {

width: 80%;

}

.reg-button{

font-size: 18px;

}

}

</style>

</head>

<body>

<h1>Welcome to the Quiz Application</h1>

<div class="center-container">

<a href="login.php"><button class="reg-button">Login</button></a>

<a href="register.php"><button class="reg-button">Register</button></a>

</div>

<button id="modeSwitch" class="switch">Switch to Dark Mode</button>

</body>

</html>

## Homepage

### welcome.php:

<?php

session\_start();

// Redirect user to login page if not logged in

if (!isset($\_SESSION['username'])) {

header("Location: login.php");

exit();

}

$username = $\_SESSION['username'];

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Welcome</title>

<link rel="stylesheet" href="css/mode.css">

<script src="js/mode.js"></script>

<style>

body {

font-family: Arial, sans-serif;

}

.welcome-message {

margin-top: 10%;

text-align: center;

font-size: 40px;

}

.container-boxes {

display: flex;

justify-content: space-around;

flex-wrap: wrap;

margin-top: 10%;

}

.box {

width: 30%;

margin: 10px;

padding: 20px;

text-align: center;

cursor: pointer;

border: 1px solid;

border-radius: 8px;

box-sizing: border-box;

box-shadow: 2px 2px 2px 1px rgb(0 0 0 / 20%);

}

.box:active {

box-shadow: inset 0 3px 5px rgba(0, 0, 0, 0.125);

transform: translateY(2px);

}

button{

cursor: pointer;

}

/\* Media query for mobile devices \*/

@media only screen and (max-width: 767px) {

.welcome-message{

font-size: 30px;

margin-top: 30%;

}

.box {

width: 100%;

}

}

</style>

</head>

<body>

<div class="welcome-message">

Hello, <?php echo htmlspecialchars($username); ?>!<br>Please choose what you want to do today!

</div>

<div class="container-boxes">

<div class="box" onclick="navigateToPage('select\_exam.php')">

<h2>Exams</h2>

</div>

<div class="box" onclick="navigateToPage('past\_exams.php')">

<h2>Exam History</h2>

</div>

<div class="box" onclick="navigateToPage('leaderboard.php')">

<h2>Leaderboard</h2>

</div>

</div>

<button id="modeSwitch" class="switch">Switch to Dark Mode</button>

<script>

function navigateToPage(pageUrl){

window.location.href = pageUrl;

}

</script>

</body>

</html>

## Navigation menu

### navigation.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<link rel="stylesheet" href="css/mode.css">

<style>

nav ul {

list-style-type: none;

margin: 0;

padding: 0;

overflow: hidden;

}

nav ul li {

float: left;

}

nav ul li a {

display: block;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

</style>

</head>

<body>

<nav>

<ul>

<li><a href="welcome.php">Home</a></li>

<li><a href="select\_exam.php">Exams</a></li>

<li><a href="leaderboard.php">Leaderboard</a></li>

<li><a href="past\_exams.php">Exams History</a></li>

<!-- Add more links as needed -->

</ul>

</nav>

</body>

</html>

## Dark and Light Mode

### mode.js:

document.addEventListener('DOMContentLoaded', function () {

const body = document.body;

const modeSwitch = document.getElementById('modeSwitch');

// Check if dark mode was selected previously

if (localStorage.getItem('darkMode') === 'true') {

body.classList.add('dark-mode');

modeSwitch.innerText = 'Switch to Light Mode';

}

modeSwitch.addEventListener('click', function () {

body.classList.toggle('dark-mode');

let isDarkMode = body.classList.contains('dark-mode');

localStorage.setItem('darkMode', isDarkMode);

modeSwitch.innerText = isDarkMode ? 'Switch to Light Mode' : 'Switch to Dark Mode';

});

});

### mode.css:

body {

background-color: white;

color: black;

}

/\* Dark mode styles \*/

body.dark-mode {

background-color: black;

color: white;

}

body nav ul{

background-color: #333;

}

body nav ul li a {

color: white;

}

body nav ul li a:hover {

background-color: #111;

}

/\* Dark mode styles nav \*/

body.dark-mode nav ul {

background-color: white;

}

body.dark-mode nav ul li a {

color: black;

}

body.dark-mode nav ul li a:hover {

background-color: #444;

color: white;

}

.box {

border-color: black;

}

/\* Dark mode styles welcome \*/

body .dark-mode .box{

border-color: white;

}

td, th{

border-color: black;

}

/\* Dark mode styles table \*/

.dark-mode td, th{

border-color:white;

}

/\* Styling for switch button \*/

.switch {

position: fixed;

top: 20px;

right: 20px;

}

## Some of the error logs recorded in error\_log.php:

[08-Feb-2024 23:43:06 UTC] PHP Warning: Trying to access array offset on value of type null in /home/g46kbhv/public\_html/STU101292/app/leaderboard.php on line 119

[08-Feb-2024 23:43:06 UTC] PHP Warning: Trying to access array offset on value of type null in /home/g46kbhv/public\_html/STU101292/app/leaderboard.php on line 119

[08-Feb-2024 23:43:06 UTC] PHP Warning: Trying to access array offset on value of type null in /home/g46kbhv/public\_html/STU101292/app/leaderboard.php on line 119

[08-Feb-2024 23:43:06 UTC] PHP Warning: Trying to access array offset on value of type null in /home/g46kbhv/public\_html/STU101292/app/leaderboard.php on line 119

[08-Feb-2024 23:43:06 UTC] PHP Fatal error: Uncaught PDOException: SQLSTATE[HY000] [2019] Unknown character set in /home/g46kbhv/public\_html/STU101292/app/leaderboard.php:119

Stack trace:

#0 /home/g46kbhv/public\_html/STU101292/app/leaderboard.php(119): PDO->\_\_construct('mysql:host=;dbn...', NULL, NULL)

#1 {main}

thrown in /home/g46kbhv/public\_html/STU101292/app/leaderboard.php on line 119