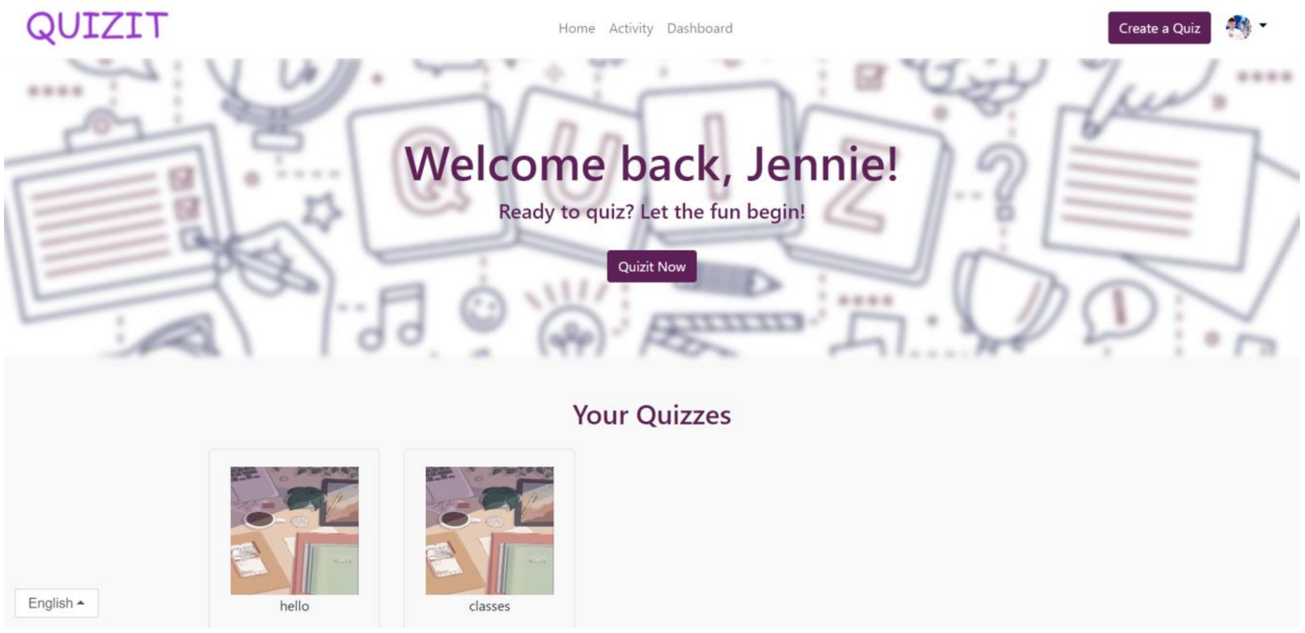
WEB DATABASE PROGRAMMING (61355_006)

Project Work _ Quiz Platform

Group 3	
Submitted by:	AMBER YEO EN (202399156) GOH CHOON MENG, JEREN (202399157) JAVERINE TAN JING XUAN (202399158) NICHOLAS TAN JUNRONG (202399159)
Project Name:	QuizIT
Major:	Exchange Student (From Singapore Nanyang Polytechnic)
Professor:	MAJEED ABDUL
Submission Date:	12/11/2023
Due Date:	12/11/2023

Table of Contents

Description of the Project



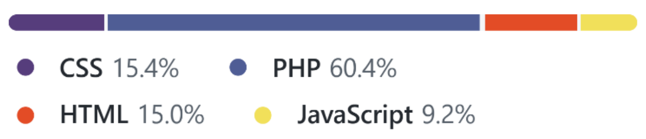
Description:
We have designed and developed an online quiz platform called "QuizIT" with functions such user registration, quiz creation, participation and many more. The focus of this website is to allow users to create and attempt quiz for educational purposes and to the build a community of quiz lovers, sharing their quizzes with one another. By leveraging on modern website development frameworks, we made the website scalable and responsive to improve efficiency and functionality of our website and improve overall user experience.

Here is the breakdown of roles in this project by our group members:

Group Member	Roles
Amber	• Account Management • Quiz Section • Database Designer
Javerine	• Quiz Section • Majority of the JavaScript • Navigation Bar • Account Page • Database
Jeren	• Home Page • Activity Page • Majority of the CSS
Nicholas	• Dashboard Page
Overall	• Everyone helped each other whenever we could

Tools and Technologies used

Languages



Languages	Purpose
HTML & CSS	Front End Design
JavaScript	Client-side interactivity, like model pop ups and alerts
PHP	Server-Side Scripting
Language APIs for language changes	To enhance accessibility and user experience for individuals using different languages, consider integrating Language APIs. These tools enable dynamic language changes on your website, allowing users to navigate and interact with content in their preferred language. By incorporating Language APIs, you ensure that your website is more inclusive and user-friendly for a diverse audience, promoting a seamless experience regardless of language preferences.
MySQL	Data Storage
XAMPP	Run the website

Overall, the tools and technologies used have enabled us to design and develop a dynamic website for our users.

What was in the project?

- We scraped the homepage of an existing quiz website (quizizz.com) into our Index.php
- Continued to build on and further develop the rest of the website ourselves.

Why did we choose this project?

This project was chosen based on how effective it could showcase our knowledge on how much we have learnt from our Web Database Programming classes. Our primary focus would be to show the CRUD (Create, Read, Update, Delete) operations as well as using a database to store our data. We believe that by creating a dynamic quiz website, it hits all the criteria needed as well as challenging ourselves to add additional functionalities such as charts. Overall, this project was chosen to apply what we have learnt from the class.

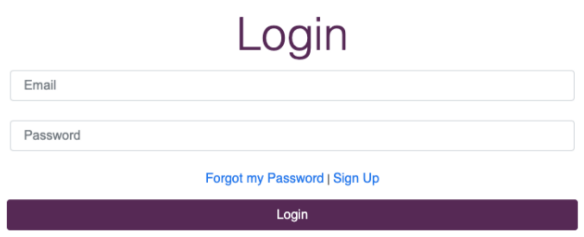
What we developed [Overview]

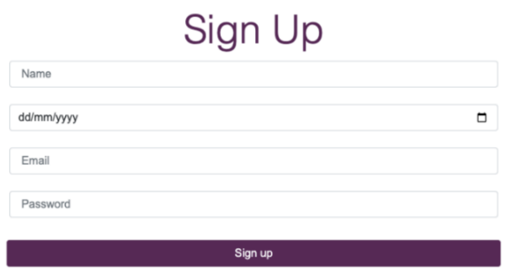
Unique features

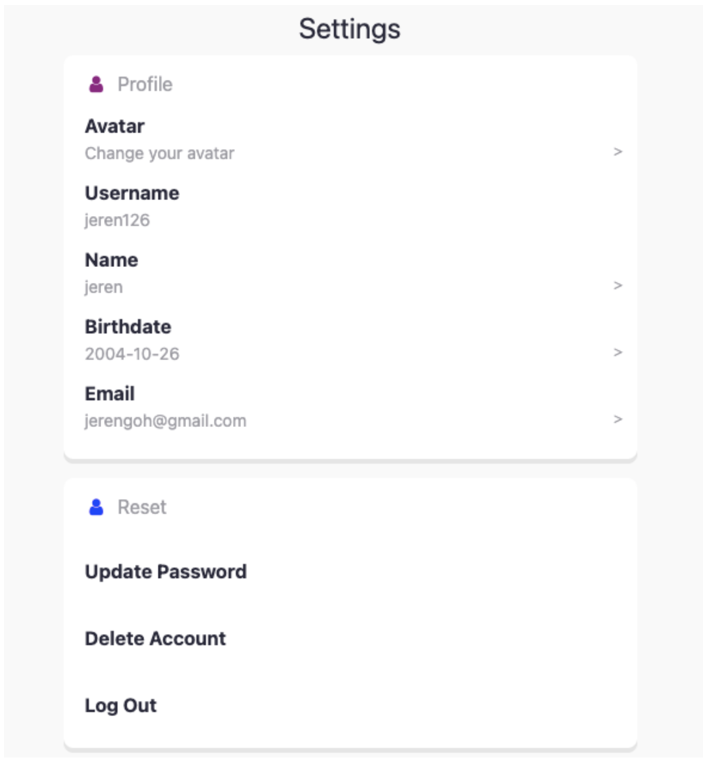
- Language change using existing language API
- Flexible and fit different screen size

Account management (CRUD)

We have developed an account management system where we can create, update, read and delete accounts. The account is used as sessions when the user logs in to track user activities on the website. Admin accounts can also manage accounts, activities, and quizzes. Account management allows users to create an account, sign in using that account, update the details of the account and delete the account entirely.







Avatar
Change your avatar >

Username
jeren126

Name
jeren >

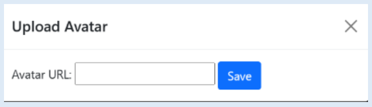
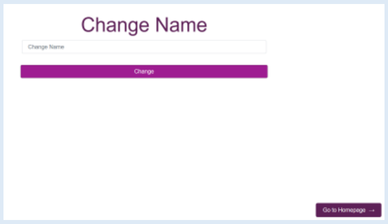
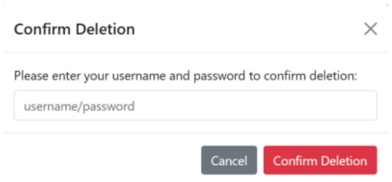
Birthdate
2004-10-26 >

Email
jerengoh@gmail.com >

Update Password

Delete Account

Log Out

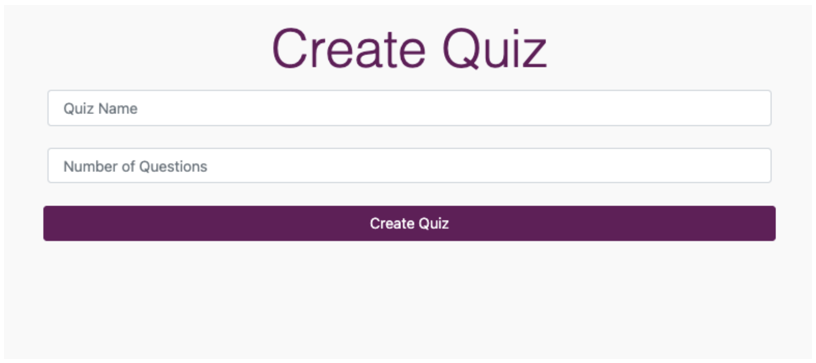
	Description	CRUD Used
Avatar 	Enable users to personalize their profile pictures effortlessly by allowing them to input image links directly, eliminating the need for downloading and saving images to their desktops.	Retrieve and Update
Username	Display the username prominently to facilitate easy identification for users who may need to delete their accounts later.	Retrieve
Name, Birthdate, Email, Password 	Facilitate easy updates for users by allowing corrections to their name, birthdate, email, and password. Redirect them to a dedicated page for modifications, with the option to return to the main page using the "Back to Homepage" button if they decide not to proceed with the changes.	Retrieve and Update
Delete Account Confirm Deletion Please enter your username and password to confirm deletion: 	When users choose to delete their account, implementing a pop-up that requires entering their username and password adds an extra layer of security to ensure the account owner initiates the deletion. Deleting all details, including logs and created quizzes, is done to uphold user privacy and maintain a clean database. This practice aligns with data protection principles, allowing users to have control over their personal information and ensuring that their footprint within the system is completely erased upon account deletion. It also helps in minimizing potential data retention issues and adhering to privacy regulations.	Retrieve and delete

Quiz (CRUD)

The quiz selection also uses all the CRUD operation. It allows users to create their own quizzes, edit their quizzes by adding more questions or updating existing questions and deleting questions or the quiz. The quiz they created will be displayed on the home page of the website where they can navigate through the quizzes they created, or community quizzes created by the community. They can then attempt the quizzes that they created themselves or by the community.

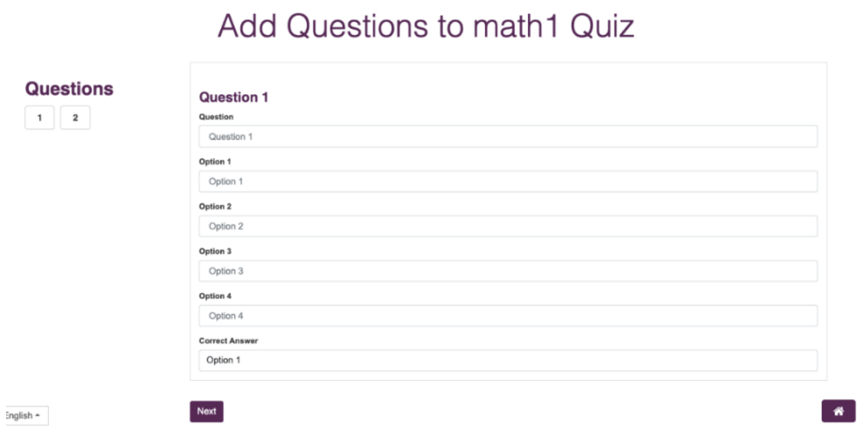
Create Quiz

The Create Quiz page allows users to enter in the quiz name and the number of questions the quiz should have. This data is then saved into the Quiz table in our database where it stores the quiz_id, quiz_name, creator_id (user who create the quiz), the number of questions and the time created. This data can be used to reference other quiz tables when retrieving data. This part uses the create operation in CRUD.



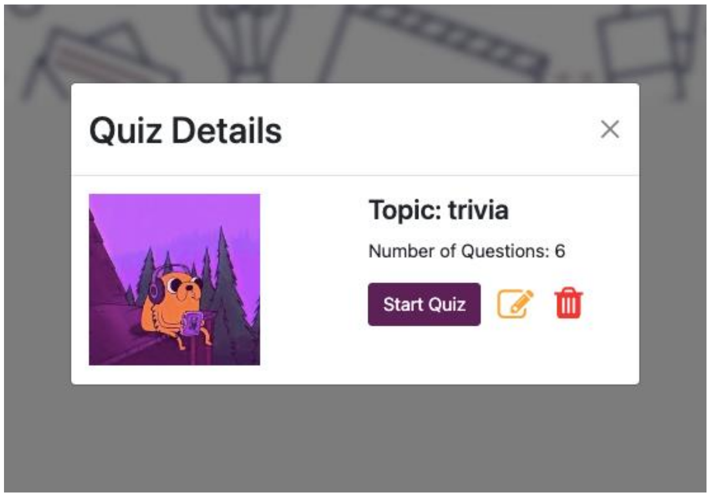
Add Quiz Questions

The Add Quiz Questions page is the next part after specifying the name and number of questions the quiz should have. In the page, users can set the questions, options and answers of the question. This data is then stored in a separate table in our database as the questions for each quiz may vary. Hence, we store quizzes individually by using the quiz_id as the table name. The Quiz table can then be use as a reference for the quiz_id table to get data such as the quiz_name when needed. This part uses the create operation in CRUD.



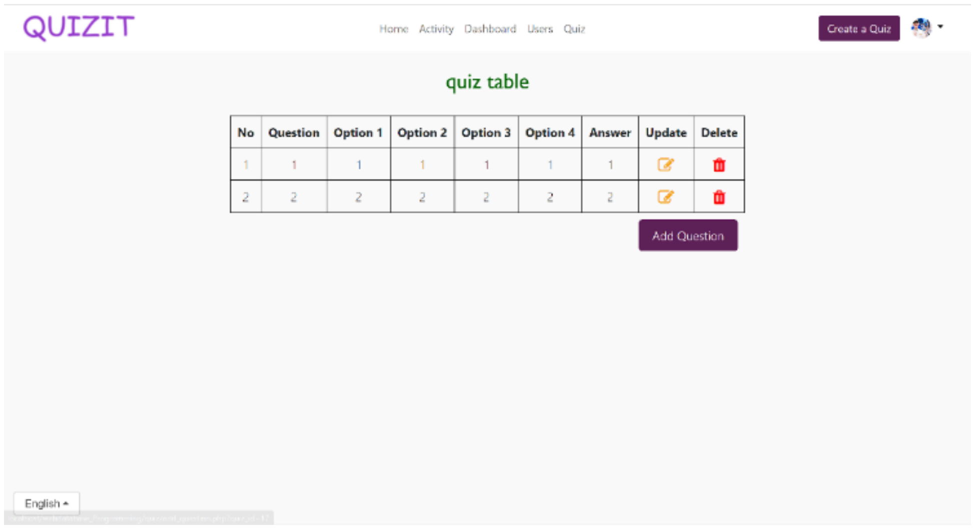
Quiz Details

The Quiz Details modal allows users to view the details of the quiz, start the quiz, edit the quiz or delete the quiz. It fetches data from the database by using the quiz_id on the quiz table. Users can choose to start the quiz and it will redirect them to the attempt_quiz page using the quiz_id. The edit quiz button also uses similar logic where it takes the quiz_id and redirect to the edit_quiz page using the quiz_id to retrieve the relevant data from the quiz_id table. This part used the retrieve operation in CRUD.



Edit Quiz

This is the edit quiz page where it allows users to add, edit or delete questions from the quiz. This page fetches data from the quiz_id table in our database to display all the information about the quiz the user wants to edit. Users can edit specific questions, delete unwanted questions or add new questions into the quiz. All these changes will then be updated and stored back into our database where it overwrites old data and updates it with the newly changed one. This part uses all 4 operations in CRUD.

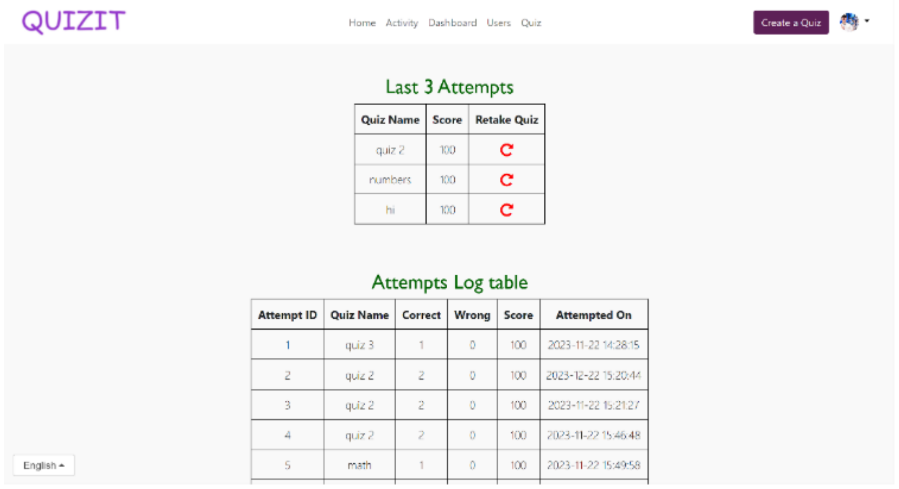


quiz table

No	Question	Option 1	Option 2	Option 3	Option 4	Answer	Update	Delete
1	1	1	1	1	1	1	📝	🗑
2	2	2	2	2	2	2	📝	🗑

Activity

This is the activity page allows users to see their past activities. They can view their last 3 attempts of quizzes and the logs of all the activities they have done. This data is all taken directly from the quiz_attempt_logs table in our database. This table stores all activities on the website. We then filter out the activities by using the session id of the user to display each specific users' activities. This page allows users to re-attempt the quizzes also to try and improve their score. This part uses the retrieve operation in CRUD



Last 3 Attempts

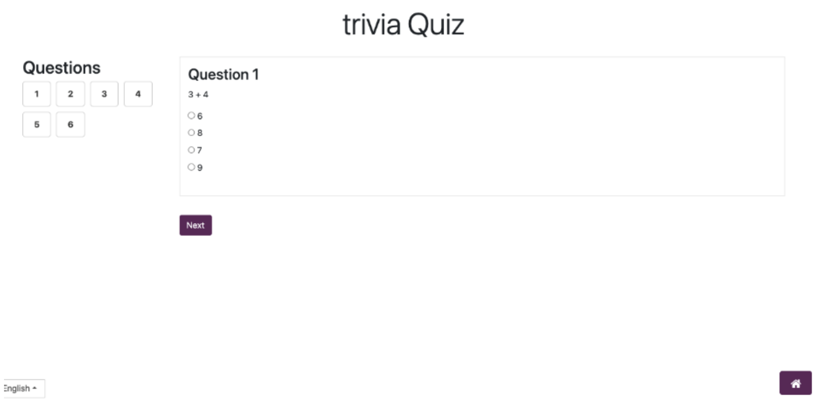
Quiz Name	Score	Retake Quiz
quiz 2	100	↻
numbers	100	↻
hi	100	↻

Attempts Log table

Attempt ID	Quiz Name	Correct	Wrong	Score	Attempted On
1	quiz 3	1	0	100	2023-11-22 14:28:15
2	quiz 2	2	0	100	2023-12-22 15:20:44
3	quiz 2	2	0	100	2023-11-22 15:21:27
4	quiz 2	2	0	100	2023-11-22 15:46:48
5	math	1	0	100	2023-11-22 15:49:58

Attempt Quiz

This is the attempt quiz page where users can attempt the quizzes that was created by them or the community. The option and questions are fetched from the database, and we have randomized the questions and options to make it more realistic. If users were to refresh the page or start the quiz again, the options and question will all be randomized again. If users chose to not do the quiz already, they can click on the back to home button on the bottom corner of the screen, and it will return them back to the home page without saving the scores and answer of the attempted quiz. This part uses the create and retrieve operation in CRUD as it will create a new row on the quiz_attempt_logs storing the score gotten by the user.



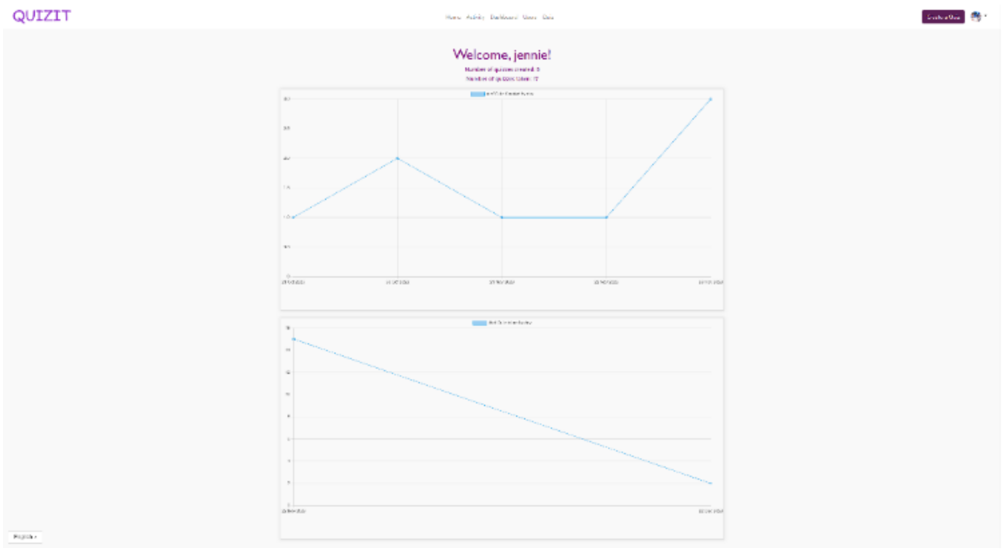
Question 1

3 + 4

- ○ 6
- ○ 8
- ○ 7
- ○ 9

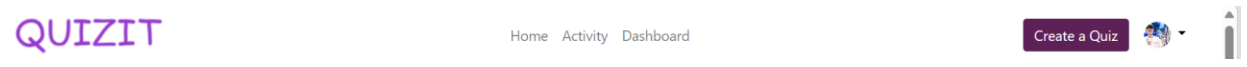
Dashboard

This is the dashboard part where users can see their progression and statistics of the activities such as number of quizzes created, and number of quizzes attempted. This will help users track their progression. The data is directly being retrieved from the quiz table and quiz_attempt_logs table in our database. It is then being used in the labels for the dates for the y-axis and also the count for the x-axis and also the short description of the webpage. This part uses the retrieve operation in CRUD.

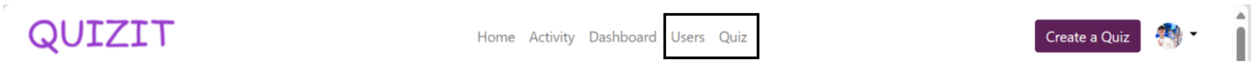


Navbar

- User



- Admin

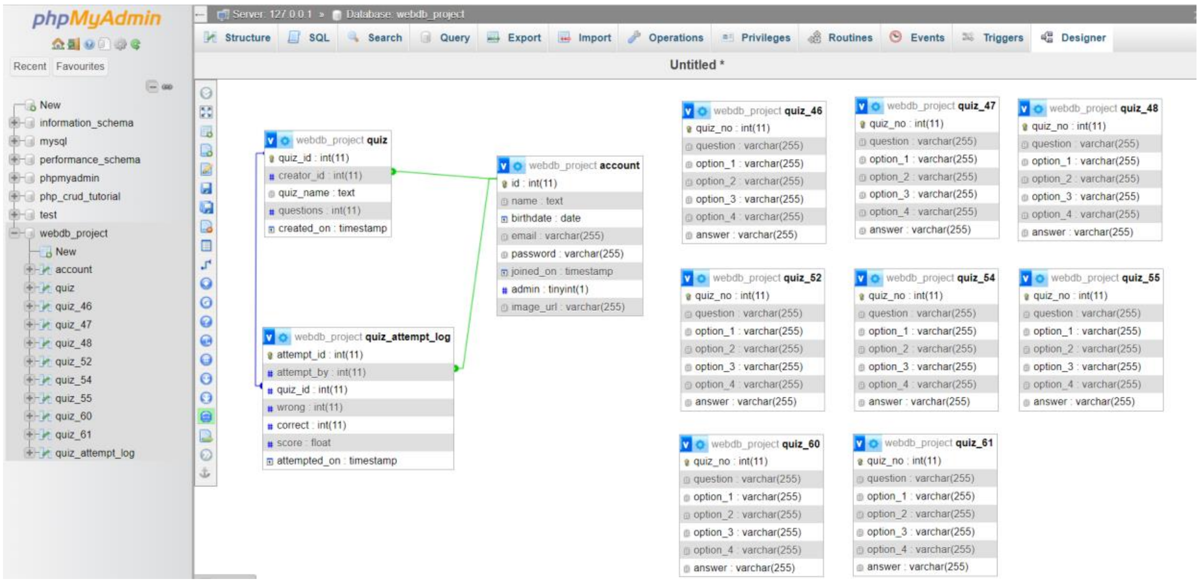


The administration panel has been augmented with two additional features on the navbar. These functionalities provide administrators with a comprehensive view of the user database and the creators behind each quiz. Specifically, administrators can access a list of users stored in the database, offering insights into the user base. Additionally, they can conveniently track and identify the creators responsible for crafting each quiz, enhancing their oversight and management capabilities.

By incorporating these additional elements into the navbar, administrators gain quick and direct access to crucial information. The inclusion of user and quiz tracking features underscores the commitment to providing administrators with a robust and efficient toolset for managing the platform. These enhancements empower administrators to make informed decisions and maintain a thorough understanding of the platform's user dynamics and quiz creation activities.

Project Structure

Database Structure

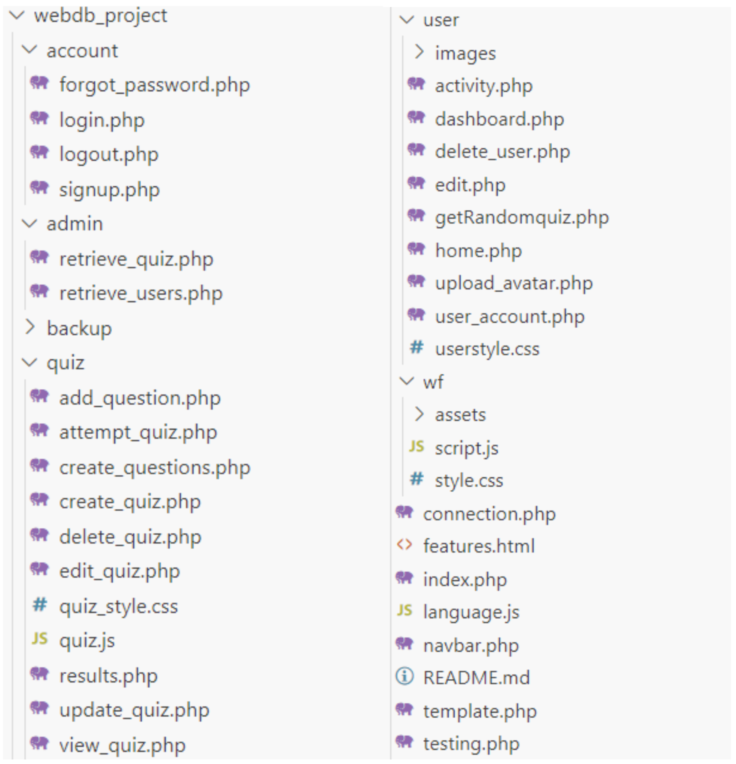


Each quiz question is a table, with its table name being quiz_(quiz_id)

quiz table has 1 foreign key, *creator_id* referencing *id* in **account** table

quiz_attempt_log table has 2 foreign keys, *quiz_id* referencing *quiz_id* in **quiz** table, and *attempt_by* referencing *id* in **account** table

Code Structure



```
webdb_project
  account
    forgot_password.php
    login.php
    logout.php
    signup.php
  admin
    retrieve_quiz.php
    retrieve_users.php
  backup
  quiz
    add_question.php
    attempt_quiz.php
    create_questions.php
    create_quiz.php
    delete_quiz.php
    edit_quiz.php
    quiz_style.css
    quiz.js
    results.php
    update_quiz.php
    view_quiz.php
  user
    images
    activity.php
    dashboard.php
    delete_user.php
    edit.php
    getRandomquiz.php
    home.php
    upload_avatar.php
    user_account.php
    userstyle.css
  wf
    assets
    script.js
    style.css
  connection.php
  features.html
  index.php
  language.js
  navbar.php
  README.md
  template.php
  testing.php
```

Challenges/ Problems that we faced.

Challenges/ Problem	Solution
Unfamiliar programming language	We had no experience in coding in PHP, so we had to use many online resources on how to code and debug our issues.
Deleting and adding of questions into quiz that are already created.	Initially, the process of deleting a question from the system involved a straightforward removal of the question. However, upon closer inspection, it became evident that the deletion process left behind a gap in the question numbering sequence. This issue prompted a reevaluation of the deletion mechanism to ensure the sequential integrity of question numbers. To address this challenge, an enhancement was introduced to the deletion process. Now, when a question is deleted, the system checks if there are subsequent questions. If there are, it recalculates the total number of original questions. In the presence of questions following the deleted one, the system adjusts their question numbers by decrementing them by one. This meticulous step ensures that the sequential order of question numbers is maintained, eliminating any gaps or missing numbers. By implementing this modification, the system guarantees that the question numbering remains continuous and orderly even after the removal of a question. Consequently, this streamlined process sets the stage for seamlessly adding new questions, as the numbering system remains coherent without any interruptions.
Scrapping of the website	When I first undertook the task of scraping the `index.php` file, I encountered a series of challenges, particularly related to security measures implemented on the website. The process of coding a Python function to retrieve the content of the `index.php` file proved to be somewhat daunting, especially given my limited prior experience in this specific area. To overcome these challenges, I turned to various educational resources, including multiple tutorial videos on platforms such as YouTube. Watching these tutorials provided valuable insights into the intricacies of web scraping, guiding me through the process and helping me understand the nuances involved. After assimilating knowledge from these tutorials and leveraging the information gleaned, I was eventually able to successfully scrape the contents of the first page of the official website. This hands-on experience not only allowed me to overcome the initial hurdles but also equipped me with a more profound understanding of web scraping techniques in the context of Python. In retrospect, while the journey involved overcoming a learning curve, the acquired skills have proven invaluable, laying the foundation for more adept and confident handling of similar tasks in the future.
Quizit Now Button – randomizing function	When initially faced with a non-functional Quizit Now button, I strategized a solution by conceptualizing the logic required for its implementation. The primary goal was to enable the button to randomly select a quiz from the database upon being clicked.

	The logical progression began with the realization that the first step was to retrieve quiz information from the database. To achieve this, I formulated the necessary SQL query to fetch all relevant quiz IDs. Once I had this query in place, I proceeded to create an empty array, anticipating the need to store the retrieved quiz IDs. In the array creation process, each quiz ID obtained from the database query was systematically appended to the array. This step was crucial for establishing a pool from which the Quizit Now functionality could randomly draw a quiz. With a populated array of quiz IDs at my disposal, I employed a random function—leveraging assistance from my more proficient groupmates—to select a single quiz ID dynamically. This step was pivotal in ensuring that the Quizit Now feature lived up to its promise of unpredictability, offering users a unique and engaging experience each time they clicked the button. Collaborating with my groupmates proved invaluable, as they contributed their coding expertise to bring the envisioned functionality to life. While I may not have been as adept at coding, my ability to articulate the underlying theory and logic facilitated effective collaboration and allowed us to collectively implement the Quizit Now button's functionality. This collaborative effort not only resolved the initial challenge but also fostered a conducive learning environment within the team.

Proof of Development

	Links
GitHub	https://github.com/javerinetan/Quiz-Website-using-database
Slides	https://docs.google.com/presentation/d/1zeckgNf4VofXEGKjdCb6sxICIlYXJvMYjvbFskBBRrc/edit?usp=sharing

Lesson Learned

- Learned to Use GitHub, PHP, and JavaScript to code a Web Application using a Database:
    - GitHub:
        - Version Control: GitHub is a platform for version control using Git. We've learned how to create repositories, commit changes, and merge changes. This ensures that our codebase is well-managed, and changes can be tracked when needed
        - Collaboration: GitHub facilitates collaboration among team members. We have learned how to handle merge conflicts, pull requests, and code reviews, ensuring a smooth collaboration process.
    - PHP:
        - Server-Side Scripting: We learned how to write server-side logic, handle forms, and interact with databases. PHP enables dynamic content generation and data processing on the server side.
        - Integration with Databases: We also learn how to connect to a database, perform CRUD (Create, Read, Update, Delete) operations.
    - JavaScript:
        - Client-Side Scripting: We learned how to enhance user interfaces, handle asynchronous requests (AJAX), and validate user inputs on the client side.
        - JavaScript Libraries: We explored JavaScript frameworks like Chart.js for building the dashboard section of the project.
    - Database Integration:
        - Database Design: We've learned the principles of database design, normalization, and structuring data for efficient storage and retrieval.
        - SQL: You may have used SQL queries to interact with databases, including selecting, updating, and deleting records. Understanding database transactions and ensuring data integrity is crucial.
    - Teamwork:
        - Task Distribution: We learnt the how to distribute tasks among team members based on their strengths and expertise. This involves creating a balanced workload and ensuring everyone's skills are utilized effectively.
        - Code Reviews: Engaging in and learning from code reviews. This includes providing constructive feedback and incorporating suggestions to improve code quality.
    - Communication:
        - Clear Documentation: We learnt that having effective communication which often involves documenting code, processes, and project details comprehensively ensures that team members can understand and work with the codebase efficiently.
        - Regular Updates: Keeping the team informed about progress, challenges, and achievements. We did this through regular meetings at the café and also calling each other to ensure that everyone knows what's happening.
        - Problem Resolution: We also learn how to communicate and collaborate when issues arise. This includes discussing potential solutions, seeking input from team members, and collectively addressing challenges.