MathWhiz: Documentation

Outline

The application, MathWhiz is an interactive and engaging quiz platform designed to test and enhance one’s mathematical skills. Users can register, log in, and access a variety of quizzes categorized by difficulty levels. Starting off with basic topics like addition and subtractions, users can unlock higher-level quizzes by achieving a score of 80% of or more on preceding levels. The app ensures personalized experience with progress tracking and a user profile. Built using Node.js, Express.js, and MySQL, MathWhiz is designed with a user-friendly interface, engaging animations, and real-time feedback to ensure a fun and educational experience.

Links

Deployed app: <https://www.doc.gold.ac.uk/usr/206/>

GitHub Repository: <https://github.com/anushkaishana/math_quiz>

Architecture

The app follows a client-server architecture with the application tier using Node.js and Express.js to handle user sessions, routing and all the backend logic. MySQL is relied on by the data tier to store quiz questions, user information and user quiz progress. Communication between these tiers is established via SQL queries that are used throughout and session management is being handled using express-session.

Client (browser) (Browser

MySQL Database (quiz data, users, and progress)

Server (Express and Node for routing and logic)

Data Model

The database includes 3 primary tables:

* questions - Stores quiz questions, answer options, and the correct answer. Each question is associated with a specific quiz type through the quiz\_id.
* users - Stores user credentials, including their first name, last name, email, and passwords.
* user\_progress - Tracks the user's progress through quiz levels. It links to the users table via a foreign key, ensuring data integrity.

A one-to-many relationship exists between Questions and quiz levels.

A one-to-one relationship between user and user progress also exists (1 user can have only one record in user progress).

|  |
| --- |
| user\_progress |
| id (PK) |
| user\_id (FK) |
| level |

many to one

one to one



|  |
| --- |
| questions |
| id (PK) |
| quiz\_id |
| question |
| options |
| answer |

|  |
| --- |
| users |
| id (PK) |
| first\_name |
| last\_name |
| email |
| password |



User functionality

The home page consists of a basic introduction with the options to learn more about the application or proceed to begin the quiz.

For doing so, users are provided with the functionality to either create and account or sign in, leading them to their respective quiz-list pages. This displays available quizzes, the user’s progress, and levels unlocked based on performance.

Each quiz is categorized by topics such as Addition, Subtraction, Multiplication, and more advanced topics like Algebra and Geometry. Users can attempt quizzes sequentially, as higher levels unlock only after achieving a minimum score of 80% on the current level. This gamified progression encourages continuous learning and mastery of topics.

During a quiz, users answer multiple-choice questions with options dynamically retrieved from the database. At the end of the quiz, their score is calculated and displayed, along with motivational feedback. If a user scores 80% or higher, the next level is unlocked, and their progress is updated in the database. Users failing to meet this threshold are encouraged to retry and improve.

The quiz-list page also allows users to either head back to the homepage or view their profile that shows their current progress and allows user to sign out of their session.

The application is designed to be user-friendly, with seamless navigation, clear instructions, and responsive interfaces for a smooth experience across devices.

Advanced Techniques

1. **Dynamic progress tracking**: Implemented by querying the user\_progress table and updating levels based on user performance.

A screenshot of a computer program

Description automatically generated

1. **Functionality disabling to avoid illogical signing in**: Sign out functionalities are disabled once a user has already logged in.

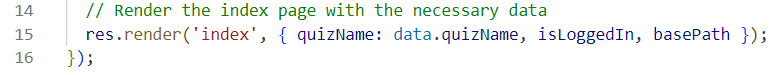
A screen shot of a computer code

Description automatically generated

1. **Conditional Navigation Links Based on Session:** If a user has logged in, an additional link shows on the home page that says head back to quizzing.

A screenshot of a computer code

Description automatically generated



1. **Secure Session Management**: User sessions are securely handled using express-session. Sessions store user ID, name, and progress for personalization.

A computer screen shot of a code

Description automatically generated

1. **Base Path Redirection for Virtual Servers**: Dynamically redirecting users to quiz-list and logout functionalities using base path of “usr/206/” to properly work on the virtual server once deployed.

A computer screen shot of a quiz

Description automatically generated with medium confidence

1. **Level-Based Access Control**: Higher quiz levels unlocked on achieving a minimum score of 80% on the current quiz level.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer code

Description automatically generated

1. **Dynamic Rendering of Quiz Levels and Questions:** Quiz questions and options are stored in the database as JSON. Data is parsed and dynamically rendered for the user.

A screenshot of a computer code

Description automatically generated

1. **Prevent Duplicate User Registrations: The register route validates new users by ensuring no duplicate email addresses are registered.**

**A computer screen shot of text

Description automatically generated**

References

1. <https://stackoverflow.com/questions/5573256/how-to-end-a-session-in-expressjs>
2. <https://www.geeksforgeeks.org/mysql-deleting-rows-when-there-is-a-foreign-key/>
3. <https://www.npmjs.com/package/express-session>
4. <https://medium.com/@jrudlong/what-are-express-user-sessions-and-how-can-i-utilize-them-with-javascript-for-controlling-routes-6d6feb943b40>