Developer Documentation

Introduction - What is QuizEdu?

QuizEdu is a web application that allows teachers to create quizzes for their students. The application is designed to be used in a classroom setting, where the teacher can create a quiz and have a selected student answer the question in front of the class.

Used Technologies

1. MudBlazor - MudBlazor is a powerful component library designed for Blazor and Razor Components, providing a rich set of UI components for building interactive web applications. It simplifies the process of creating responsive and attractive user interfaces by offering a variety of pre-built components, themes, and utilities that seamlessly integrate with Blazor applications.

2. Blazor - Blazor is a cutting-edge framework that enables the development of interactive and dynamic client-side web applications using C# and .NET. Leveraging the power of WebAssembly, Blazor allows developers to write code in C# that runs directly in the browser, eliminating the need for JavaScript. This innovative approach provides a unified framework for both client and server-side development, streamlining the development process and promoting code reuse.

3. ASP.NET Core - ASP.NET Core is an open-source, cross-platform web framework developed by Microsoft. It empowers developers to build modern, scalable, and high-performance web applications and services using the .NET platform. ASP.NET Core supports cloud-based deployment, containerization, and microservices architecture, making it an ideal choice for building robust and flexible web solutions.

4. Entity Framework Core - Entity Framework Core is a sophisticated object-database mapper (ODM) for .NET, providing a seamless and efficient way to interact with databases using object-oriented programming. It simplifies data access by allowing developers to work with databases using C# objects, abstracting away the complexities of SQL queries. Entity Framework Core is designed to work seamlessly with various database providers, making it a versatile choice for data access in ASP.NET Core applications.

5. Microsoft SQL Server - Microsoft SQL Server is a robust relational database management system (RDBMS) developed by Microsoft. It is a powerful and scalable solution for storing, retrieving, and managing structured data. SQL Server integrates seamlessly with ASP.NET Core applications, providing a reliable and efficient data storage solution. With features such as advanced security, transaction support, and performance optimization, SQL Server is a preferred choice for building data-driven applications in the .NET ecosystem.

Project Structure

The root directory contains the following files and folders:

- `QuizEdu.sln` - The solution file for the QuizEdu project.
- `QuizEdu.csproj` - The project file for the QuizEdu application.

- `Properties/` - The properties folder contains the following files:
  - `launchSettings.json` - The launch settings file for the QuizEdu application.
- `wwwroot`
  - `app.css` - The CSS file for the QuizEdu application. This file contains some styles that are applied to the application's UI elements. Most of the styles are done inline or using the MadBlazor component library.
  - `sounds/` - The sounds folder contains the following files:
    - `correct.mp3` - The sound file that is played when the user answers a question correctly.
    - `incorrect.mp3` - The sound file that is played when the user answers a question incorrectly.
- `Src/`
  - `Components` - Some small reusable components that are used in the application.
    - `LinkButton.razor` - A button that navigates to a specified URL.
    - `MainTitle.razor` - The main title of the application.
    - `NoteText.razor` - A text that is displays some information to the user.
    - `OptionButton.razor` - A button which used in the quiz to select an answer.
    - `ScoreBoard.razor` - A component that displays the score of the user.
  - `Data/` - Contains the `ApplicationDbContext.cs` file which is used to connect to the database.
    - It also contains the repository files for the `Quiz`, `Question`, `Option`, `History` and `QuizCombination` models. These files are used to manipulate the data in the database using Entity Framework Core.
  - `Layout/` - Contains the main layout of the application: `MainLayout.razor`
  - `Models/` - Contains the models of the application:
    - `Quiz.cs` - Represents a quiz game with the following properties: ***Id***, ***Title***, ***Type***, ***RoundCount***
    - `Question.cs` - Represents a question in a quiz with the following properties: ***Id***, ***Text***, ***QuizId***
    - `Option.cs` - Represents an option in a question with the following properties: ***Id***, ***Text***, ***IsCorrect***, ***QuestionId***
    - `GameResult.cs` - Represents the result of a game with the following properties: ***Id***, ***QuizId***, ***UserName***, ***Score***, ***Date***
    - `QuizCombination.cs` - Represents the combination of a quiz and a question with the following properties: ***Id***, ***ParentId***, ***ChildId***

- `Pages/` - Contains the pages of the application:
  - `Home.razor` - The main page of the application It contains the start menu with a few interactive buttons.
  - `History.razor` - The page which contains the history of the games played.
  - `ManageQuiz.razor` - The page which contains the list of existing quizzes and the option to create a new quiz or edit an existing one.
  - `ManageQuestion.razor` - The page which contains the list of existing questions for a given quiz and the option to create and edit question.
  - `SelectQuiz.razor` - The page which contains the list of existing quizzes and the option to start a quiz game.
  - `PlayGame.razor` - The page which contains the dynamic quiz game with the questions and the options.
- `Services/` - Contains service files for data manipulation and for the main game-flow:
  - `QuizGamePlayService.cs` - The service file for the main game-flow. It contains the logic for the game and the methods for making the game dynamic.
  - `HistoryService.cs` - The service file for the history of the games. It contains the logic for the history and the methods for manipulating the history data.
  - `JsInteractionService` - The service file for the JavaScript interactions. It contains the logic for the JavaScript interactions and the methods for calling JavaScript functions from C#.
  - `ManageQuizService.cs` - This service file contains the logic for the quiz management. It contains the methods for manipulating the quiz data.
  - `ManageQuestionService.cs` - This service file contains the logic for the question management. It contains the methods for manipulating the question data.
  - `QuizCombinationService.cs` - This service file contains the logic for the quiz combination management.
  - `NavigationService.cs` - This service file contains the logic for navigating between the pages.
- `_Imports.razor` - Contains the global imports for the razer files
- `App.razor` - The main html file containing the **head** and ***body*** elements
- `MyMudProviders.razor` - Contains the global MudBlazor providers
- `MyMudThemeProvider.razor` - Contains the global MudBlazor theme provider
- `Routes.razor` - Contains the global routing for the application

- `appsettings.json` - the configuration file for the QuizEdu application. ***It contains the connection string for the database.***
- `appsettings.Development.json` - the configuration file used for development purposes
- `Program.cs` - The file serves as the entry point and configuration hub for the QuizEdu application, initializing services, establishing database connections, configuring middleware, and setting up the application for smooth execution.

Future Improvements

The QuizEdu application has a solid foundation, and several enhancements can be implemented to further improve its functionality and user experience:

Future Improvements

The QuizEdu application can be further enhanced with the following improvements:

1. **Timer for Each Question**: Implement timers for questions to add a time constraint, enhancing the quiz experience and making it more dynamic.
2. **Authentication and Authorization for Teachers**: Strengthen security with authentication and authorization mechanisms tailored for teachers, ensuring only authorized users can manage quiz-related functionalities.
3. **Animations and Transitions**: Enhance the user interface by incorporating animations and transitions. Adding visually appealing elements can elevate the overall user experience, making the application more engaging and polished. Consider animations during question transitions, score updates, or other key interactions for a more immersive environment.

These improvements aim to add new features, improve user interaction, and provide additional functionality for both quiz participants and teachers, contributing to a more dynamic and enjoyable educational experience.