Construction document

Contents

[Summary of build 2](#_Toc84594867)

[Technology choice 2](#_Toc84594868)

[Business Requirements achieved 2](#_Toc84594869)

[Users and Permissions 3](#_Toc84594870)

[Quizzes 3](#_Toc84594871)

[Viewing and editing 4](#_Toc84594872)

[Limitations 4](#_Toc84594873)

[Future improvements 5](#_Toc84594874)

# Summary of build

An ASP.Net MVC code first website was built with connections to a database consisting of 4 tables. C# programming language has been utilised with functions, variable and loops have been replaced with LINQ extension methods.

* ASP.NET website- front end
* SQL database backend– 4 tables;
  + Users
  + User Roles
  + Quiz
  + Questions & Answers

## Technology choice

Using C# ASP.Net MVC code first approach was chosen as I was familiar with the technology, and I had an idea that most requirements could be achieved.

I went with code first rather than database first because;

1. It enabled me to create the database and tables from the business objects.
2. I could specify which related collections were to be eager loaded, or not be serialized at all.
3. Database version control via migrations enabled traceability with dated, timed and titled changes.
4. Quick to set up for an initial minimum viable product (MVP).

## Business Requirements achieved

A database-driven website has been constructed to aid with the creation and management of quizzes consisting of multiple-choice questions.

The website has been designed to be straightforward and can be easily re-branded to include colour schemes and logos by adapting the html view pages. The shared view which is included on all pages can be easily rebranded where changing the style in this one page will be pushed to all views.

The website has been designed and built to production standards; Using version control via Azure Devops Repos. Graphical user interface, text, application, email

Description automatically generated

Figure 1 - Using Azure Devops Repos for version control of the code.

The build also adheres to industry best practices by using;

* Commenting- using comments to describe what the code is doing to help other understand the code and enable changes to be made easily in the future.
* Camel casing- Upper case naming of functions e.g. QuestionDetails.
* Appropriate naming conventions- Use descriptive names of methods and keeping them uniform throughout the code.
* Testing- The code has been tested as I have built it and once again at the end to check the overall program.
* Simple- The code has been kept as simplistic as possible.
* Scalability- The code has been designed to be reusable and scalable. For example NuGet packages have been used for ease of future maintainability which will be done by Microsoft. Code first entity framework has been chosen to enable updating the database easily with version control via migration folders including date, time and title.

## Users and Permissions

* Three user permissions have been pre-configured using a controller with views and inputting the roles manually into the website UI to be stored into the database.
* The password for the three user permissions have been hashed for security, using Microsoft.AspNetCore.Identity which uses a PBKDF2 algorithm hash. Microsoft.AspNetCore.Identity Nuget package was used for ease and future maintainability as Microsoft will update and maintain this package meaning less labour costs for us.
* The three Permission levels constructed are; {Edit, View, Restricted}, where Edit means the ability to add, delete and change questions and answers, View means the ability to view questions and answers, and Restricted means the ability to view questions only.
* Only known users pre-configured can login to the website. Once logged in, a user can only carry out the actions allowed by their permission level. Microsoft.AspNetCore.Identity has been used with mapping to certain views, to control what the different permission roles can see.
* The website maintains user session state while the user is logged in via ASP.NET Core maintaining session state by providing a cookie to the client that contains a session ID.
* A user can logout, which will take them back to homepage where they can access the login page.

## Quizzes

* A quiz has a title and a sequence of questions.
* Each question has been formulated as a text string.
* Each question has been associated with a set of between 3 and 5 answers. Each answer is a text string, shown in the user interface indexed by an uppercase character (‘A’, ‘B’, ‘C’ etc). First three answers are required and flag up a message if not filled in. Last two choices are optional and can be blank.

## Viewing and editing

* A user with Restricted permission can select a quiz from all available quizzes. Having selected the quiz, all the questions in that quiz can be viewed on the screen. If the quiz is too large to fit, the user can scroll up and down to see it.
* A user with View permission can select and view a quiz as above. They can also select a question to see the associated answers on another page.
* A user with Edit permission can select a quiz and view questions and answers as above. They can also make and save all the following changes:
  + Create new quizzes( there is validation where users can submit a blank quiz title)
  + Delete existing quizzes if there are no associated questions that would be orphaned;
  + Add and delete questions at any point in the sequence of a quiz;
  + Edit the text of any question and answer choices;
  + Add and delete answers to any question (which may cause the answers to be re-indexed);
  + Edit the text of any answer.

# Limitations

The limitations of the application are;

* Using code first has the following disadvantages;
  + Everything related to database has to be written in the visual studio code.
  + For stored procedures I have to map stored procedure using Fluent API and write Stored Procedure inside the code.
  + If I wanted to change anything in the database tables I would have to make changes in the entity classes in the code file and run the update-database from the package manager console.
  + If this application became Data intensive, then this may not be the best technology to use.
* Due to time constraints the business requirement request for numbered sequence of questions and their deletion or addition which has may cause the questions to be re-numbered has not been implemented yet but can be part of Version 2 release. Currently this doesn’t affect the application as the questions are structured in a table format, and it is clear the order of the questions in the quiz without the need for numbering. When deleting or adding a question currently there is no number so re-indexing is not required. I have looked at ways to implement this one example is to implement a for loop and using an array could be used to re indexing the items in the table and this would be automatically done.
* Due to time constraints the business requirement for adding and deleting answers to any question has been implemented but answers are not automatically re-indexed and will be a manual process currently. An automatic indexing can be implemented in a Version 2 release. As mentioned above a loop could be used to re-index but the issue is the optional last two answer choices can be blank so this will need to be considered and maybe changed.
* Quizzes can’t be deleted if there are questions associated to it. The questions need to be deleted first then the quiz can be deleted. This logic makes sense in case you want to transfer the questions to another quiz first. An error message needs to be added to explain this, due to time constraints this has not been implemented yet.
* The user interface can be improved and could be tested on users to gather feedback for further improvement in Version 2 of the application.
* The revealing of the answer is manual and not aesthetically pleasing, this can be improved in the future.
* Using the QuizID to associate the quiz to the questions in the user interface isn’t very user friendly.

# Future improvements

Some future considerations to improve the current build can include:

* Refactoring the code- By removing duplication and simplifying the code which will ensure it is easy to change in just one place for the future. This will also increase the scalability of the code.
* Implementing a numbered sequence of questions and deletion or addition of questions which cause re-numbering will be automatic. I have looked at ways to implement this, one example is to implement a for loop and using an array could be used to re indexing the items in the table and this would be automatically done.
* Implementing automatically re-indexing when adding and deleting answers to any question. As mentioned above a loop could be used to re-index but the issue is the optional last two answer choices can be blank so this will need to be considered and maybe changed.
* Working with the UX/UI designer would be recommended to ensure the applications interface is user friendly.
* Looking at user accessibility is another consideration, so the application adheres to W3 guidelines. This will future proof the application because in the near future accessibility will be lawful.
* Allow integration of separate websites- For example in the background information, a future request is to allow students to take quizzes and manage their grades and marks.
* Adding more data – For example more quizzes, questions and titles.
* The login system can be improved by add two factor authentication, confirmation of registration via email when registering, enabling logins using social media logins and reset password.
* Could make the application mobile compatible.
* The revealing/selecting of an answer could be more interactive and aesthetically pleasing.
* A error message could be added to explain that quizzes can’t be deleted if there are questions associated to it. The questions need to be deleted first then the quiz can be deleted.
* In the future instead of using the QuizID to associate the quiz to the questions in the user interface which isn’t very user friendly, I would investigate using the quiz title.