Quiz Manager Design

# Summary

The project requires a front-end and a back-end. I will create these as separate applications.

I will create the front-end application as a React app. I will create the back-end application as an ASP.NET Core Web API.

Data will be persisted in a SQL database. I will use Entity Framework Core to interface with the database, rather than accessing the database directly.

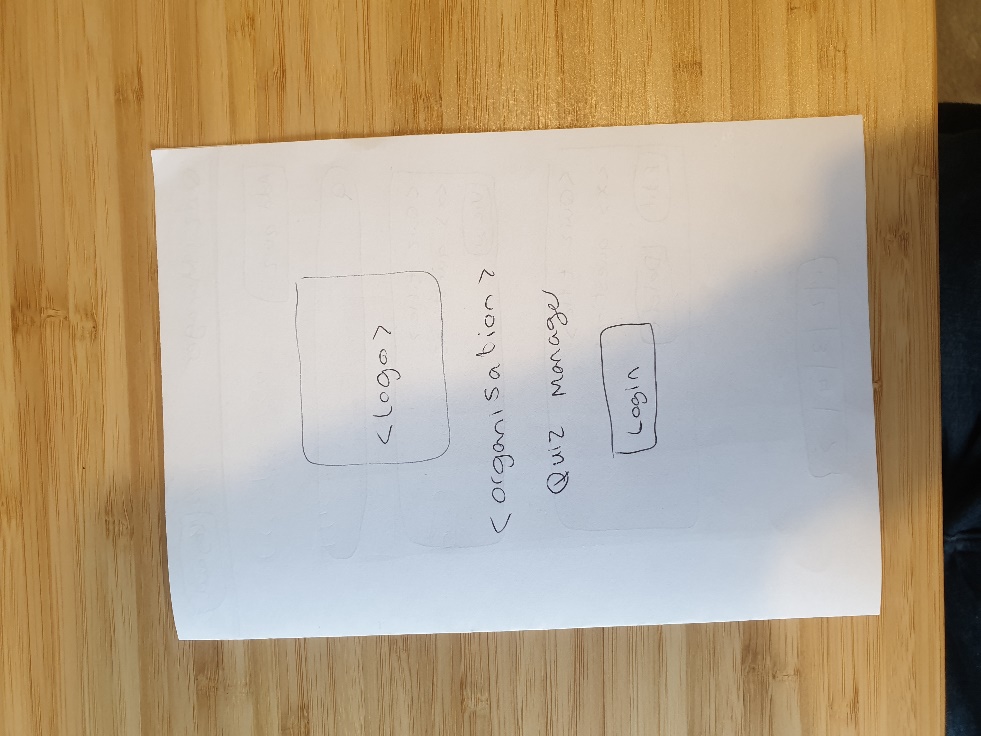
The project requires authentication and authorization. The spec states that the user passwords should be hashed but not that I need to develop the authentication system myself. I will use Auth0 as an authentication provider, as Auth0 certainly hash passwords and offer a mature authentication system for free.

The spec does not mention any requirement of recording which answer(s) is correct for any given question. It is my opinion that a quiz would be useless without a record of which answers are correct so I will allow users to record which answers are correct.

# Interface Design

This is a sketch of the login page. It will contain a logo, the name of the organisation that is using Quiz Manager and a login button.

The login button will redirect to Auth0.



This is a sketch of the interface to show a list of quizzes. At the top of the page will be a navigation bar, which will be visible on every page except the login page.

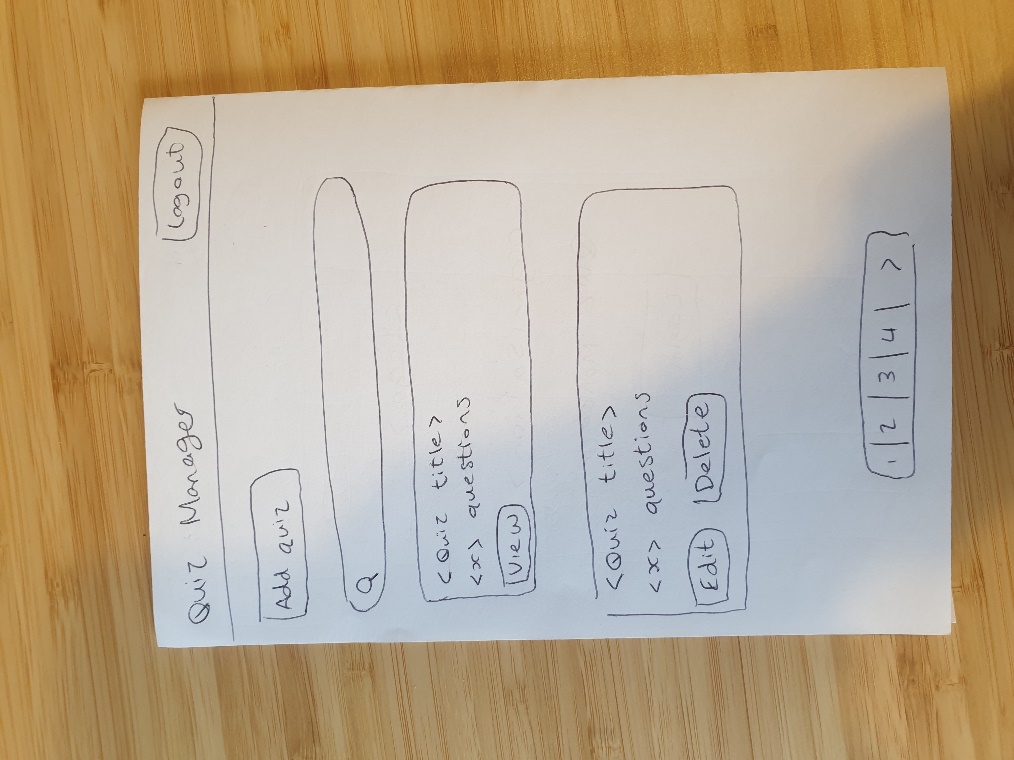
The navigation bar will have a link that always directs back to this page and a logout button.

The page will have an Add Quiz button if the user has the relevant permission.

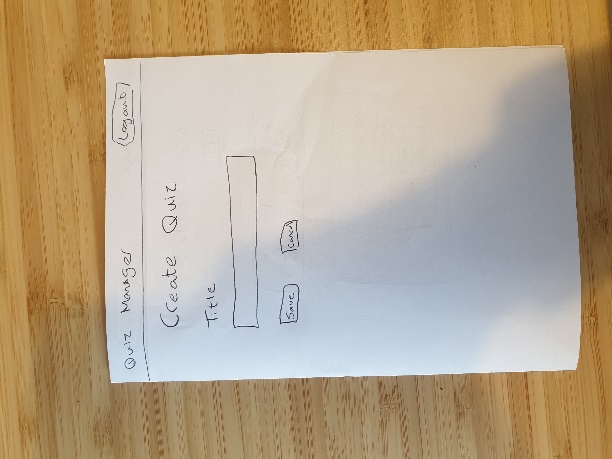
The page will have a search bar.

Each quiz will appear in a box. The box will contain the name of the quiz and the number of questions it has. The box will either contain a View button or contain Edit and Delete buttons, depending on the user’s permissions.

There will be a control at the bottom of the page for pagination.



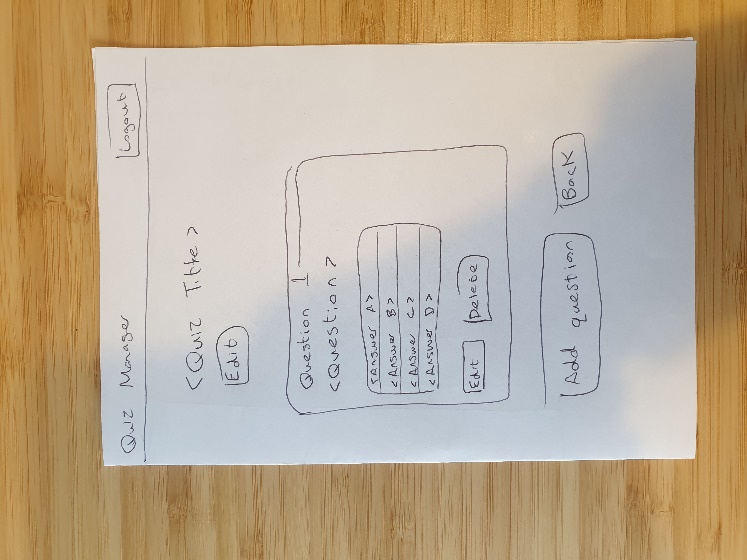
This is a sketch of the add quiz page. Here the user will only enter a title. Questions will be created separately.



This is a sketch of the page to view a quiz. The title of the quiz will be shown, with a button to edit the title if the user has the appropriate permissions.

Each question will appear in a box. If the user has the appropriate permissions, they will also see the answers for the question and Edit and Delete buttons.

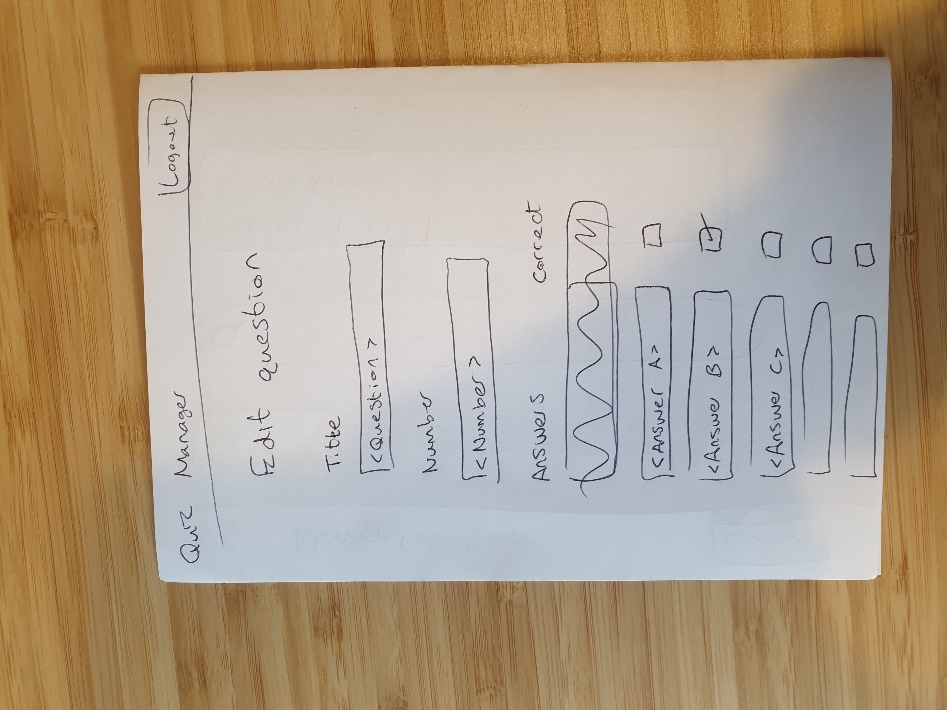
At the bottom of the page there will be a back button. There will also be a button to create a question if the user has the appropriate permissions



This is a sketch of the form to create or edit a question. The user must enter text for the question and a number for the question.

There will be 5 inputs for each potential answer. Each answer will have a checkbox to indicate whether the answer is correct.

There will be a save button and a back button at the bottom of the page (not shown in the sketch as I ran out of space).



# API Design

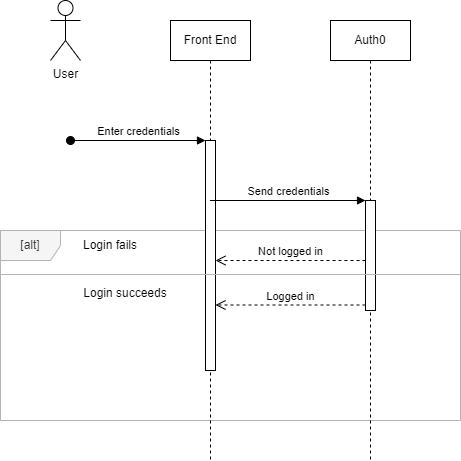
## Endpoints

The API will have the following endpoints

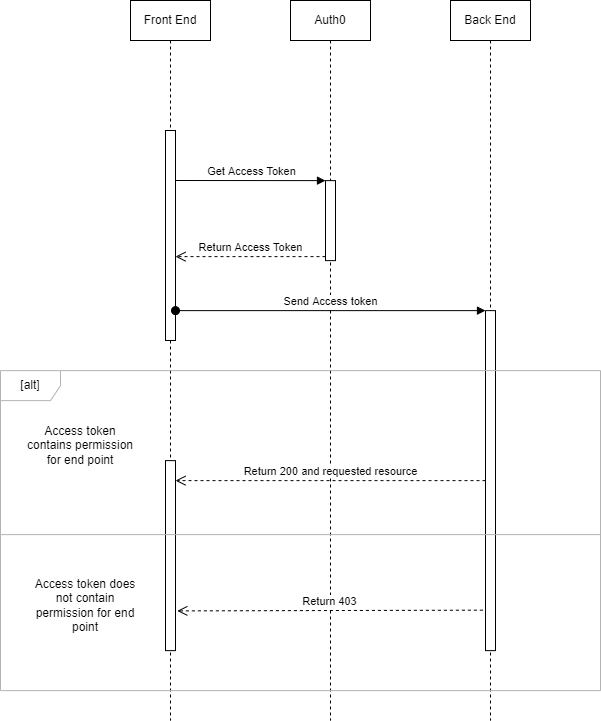
|  |  |  |
| --- | --- | --- |
| Endpoint | Method | Purpose |
| /question/id | GET | Get a single question with the provided ID and its answers |
| /question | POST | Create a question |
| /question/id | PATCH | Update a question with the provided ID |
| /question/id | DELETE | Delete a question with the provided ID |
| /quiz | GET | Get multiple quizzes with a summary of their questions |
| /quiz/id | GET | Get a single quiz with its questions and their answers |
| /quiz/summary/id | GET | Get a single quiz with its questions |
| /quiz | POST | Create a quiz |
| /quiz/id | PATCH | Update a quiz |
| /quiz/id | DELETE | Delete a quiz |

## Sequence Diagrams

This is a sequence diagram for the login process



This is a sequence diagram for the process of calling an API endpoint



# Database Design

This is an entity relationship diagram for the database

