2019

Retrospective Document

WEBBISKOOLS QUIZ MANAGER

AIMEE CRAIG

CONTENTS

OVERVIEW

The project was completed within the deadline and all requested functionality was implemented. A demo of the functionality can be seen by viewing the “Demo.mp4” video in the “Documentation” folder.

GALLERY

Below are screenshots from the web application, highlighting the homepage, create quiz, read quiz, edit quiz, and delete quiz pages.

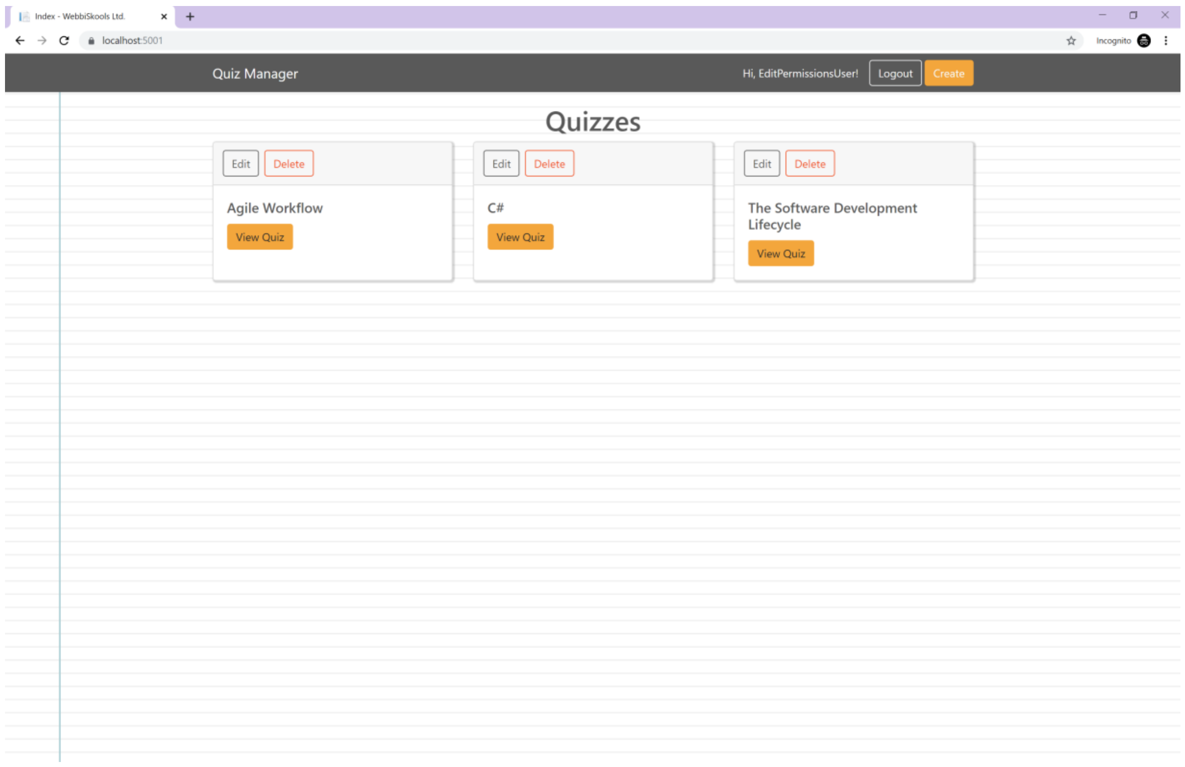
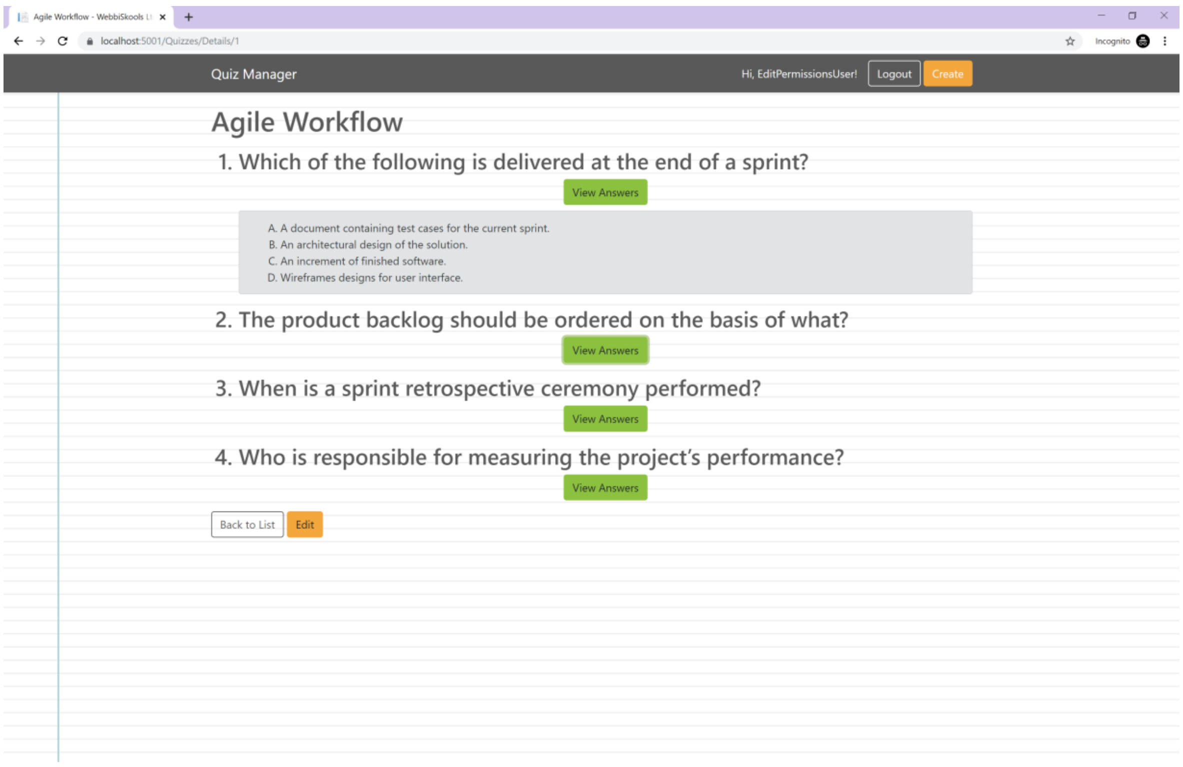


Figure 1: Home page

The home page allows users to view all available quizzes and then view each quiz individually by clicking on their respective **View Quiz** button. The user currently signed in has the Edit permission level so they are able to see the **Create** button in the navbar as well as the **Edit** and **Delete** buttons above each quiz.


Agile Workflow - WebbiSkools L
localhost:5001/Quizzes/Details/1
Incognito
Quiz Manager
Hi, EditPermissionsUser!
Logout
Create

Agile Workflow

1. Which of the following is delivered at the end of a sprint?

A. A document containing test cases for the current sprint.
B. An architectural design of the solution.
C. An increment of finished software.
D. Wireframes designs for user interface.

2. The product backlog should be ordered on the basis of what?

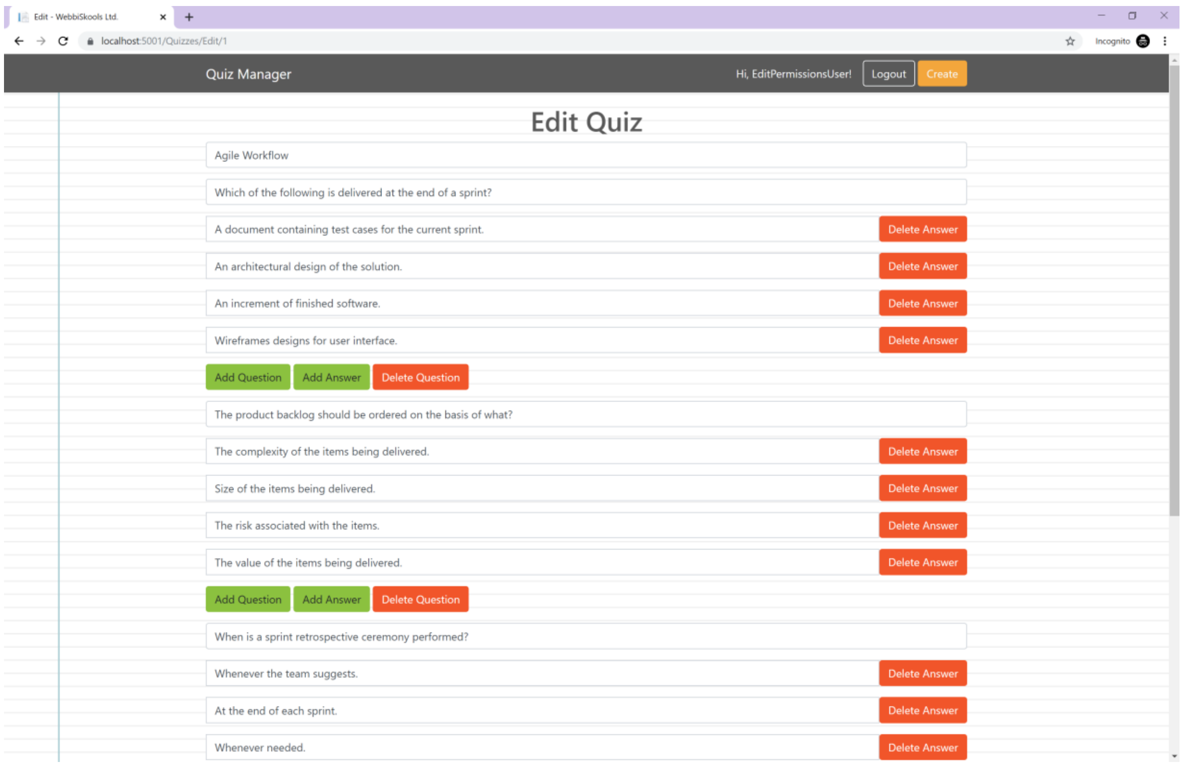
3. When is a sprint retrospective ceremony performed?

4. Who is responsible for measuring the project's performance?

Figure 2: View quiz

All users are able to view a quiz, but only users with the View or Edit permission level can see the **View Answers** buttons below each question. When these buttons are clicked on, the area below the question is expanded to reveal all possible answers for that question. If the button is clicked on again then the possible answers are hidden again.

From this page, the user is also able to edit the quiz they are currently viewing by clicking on the **Edit** button at the bottom of the page.



Edit Quiz

Agile Workflow

Which of the following is delivered at the end of a sprint?

A document containing test cases for the current sprint.

An architectural design of the solution.

An increment of finished software.

Wireframes designs for user interface.

The product backlog should be ordered on the basis of what?

The complexity of the items being delivered.

Size of the items being delivered.

The risk associated with the items.

The value of the items being delivered.

When is a sprint retrospective ceremony performed?

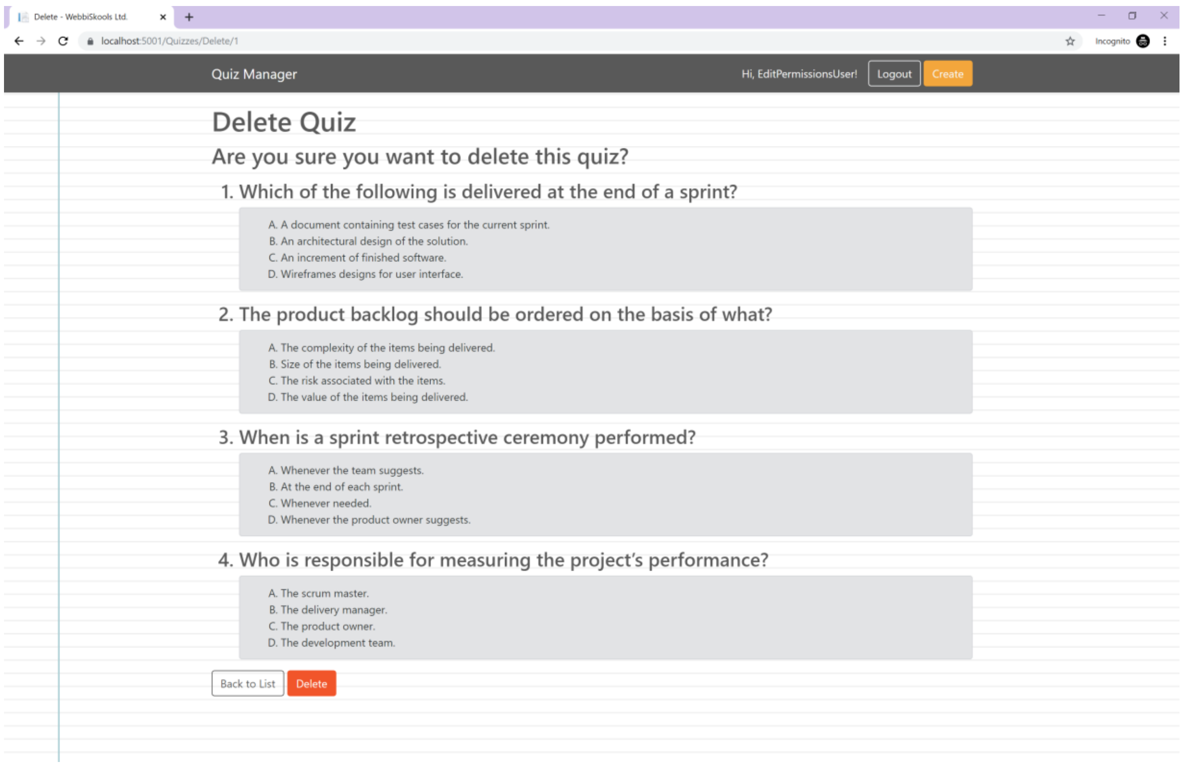
Whenever the team suggests.

At the end of each sprint.

Whenever needed.

Figure 3: Edit quiz

Users editing a quiz will see a page similar to that above. The page is displayed just like the create quiz page, however, all the input fields are already populated with data from the quiz. Users can make changes to the values of any of these input boxes as well as add a question or answers, and similarly, delete a question or answer. Clicking on the **Reset** button at the bottom of the page will reset all the values of the input boxes to their original state.


Delete - WebbiSkools Ltd.

localhost:5001/Quizzes/Delete/1

Quiz Manager

Hi, EditPermissionsUser! Logout Create

Delete Quiz

Are you sure you want to delete this quiz?

1. Which of the following is delivered at the end of a sprint?

A. A document containing test cases for the current sprint.
B. An architectural design of the solution.
C. An increment of finished software.
D. Wireframes designs for user interface.

2. The product backlog should be ordered on the basis of what?

A. The complexity of the items being delivered.
B. Size of the items being delivered.
C. The risk associated with the items.
D. The value of the items being delivered.

3. When is a sprint retrospective ceremony performed?

A. Whenever the team suggests.
B. At the end of each sprint.
C. Whenever needed.
D. Whenever the product owner suggests.

4. Who is responsible for measuring the project's performance?

A. The scrum master.
B. The delivery manager.
C. The product owner.
D. The development team.

Back to List Delete

Figure 4: Delete quiz

When a user clicks on the **Delete** button they will be taken to a page asking for confirmation to delete the quiz. On this page, all questions and their possible answers are shown without the **View Answers** buttons to make the information easy to read over. Clicking the **Delete** button removes the quiz and its associated questions and answers from the database before returning the user to the home page where they should now see that their quiz is missing from the list.

If the user decides against deleting the quiz, they can return to the home page by clicking the **Back to List** button.

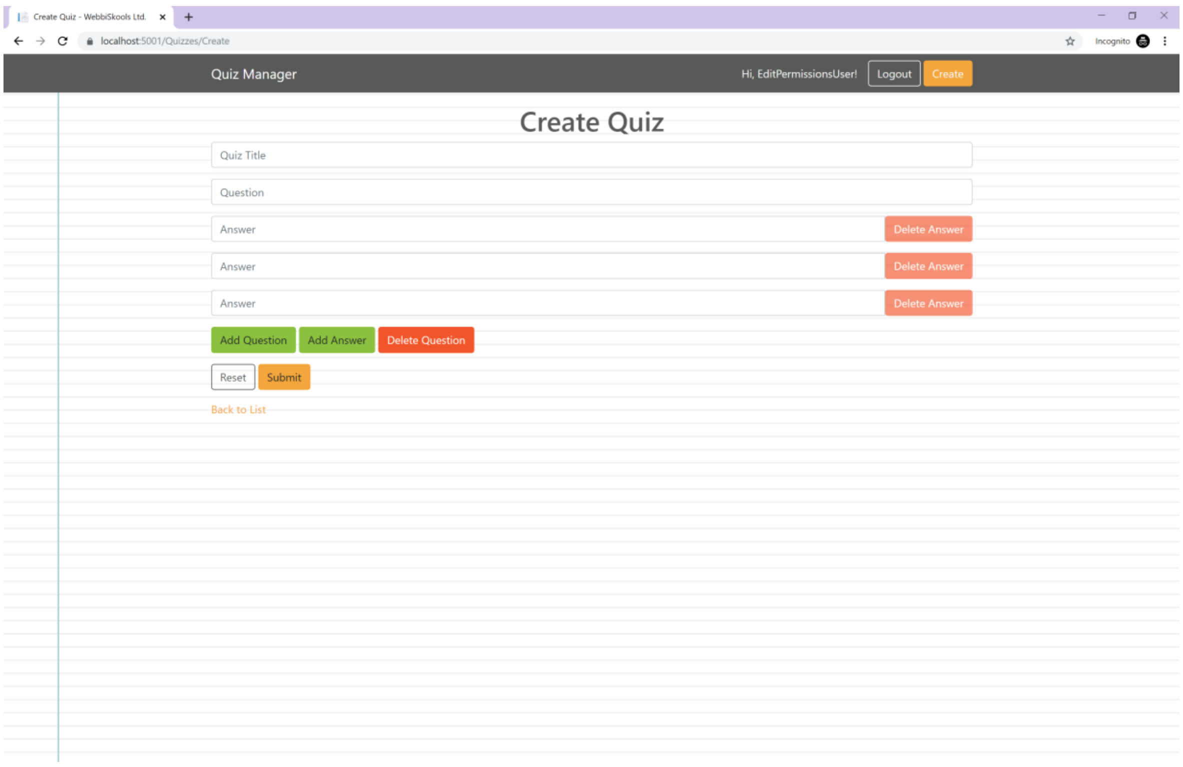


Figure 5: Create quiz

When creating a quiz, the user starts with an input for quiz title, one question, and three possible answers. The user can add a question by clicking the **Add Question** button. This will add another set of one question and three possible answers to the page.

Clicking on the **Add Answer** button will add another answer to its respective question. Similarly, clicking the **Delete Answer** button will delete the respective answer from its respective question.

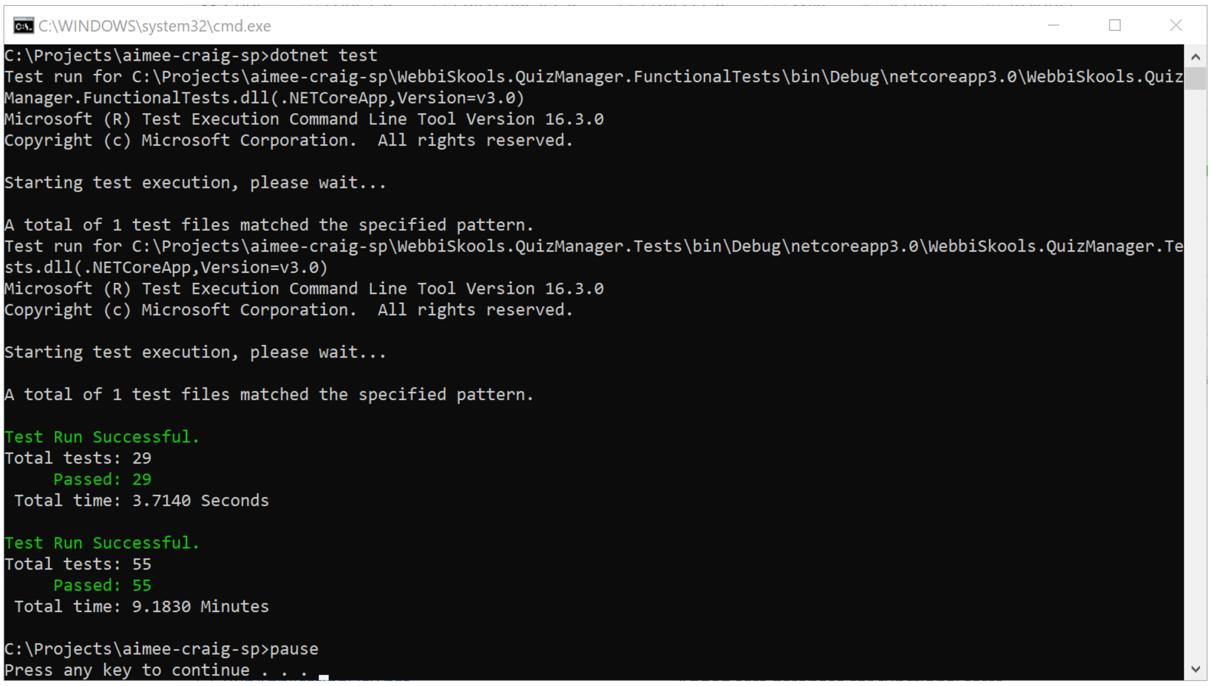
A user can delete a question by clicking its respective **Delete Question** button. This will remove the question and any associated answers from the page entirely.

Each question can only have between three and five possible answers.

TESTS

A copy of the unit and functional tests can be found in the Documentation\Tests folder and can be opened with any text editor. The manual test plan is available in the Documentation folder.

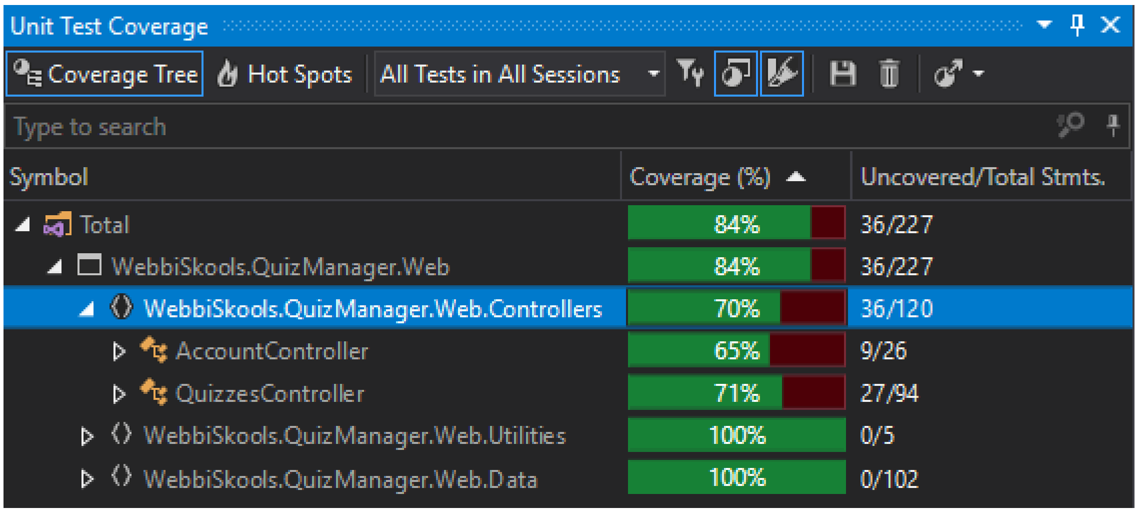
At the time of writing, all 143 tests (29 unit, 55 functional, 59 manual) passed when run against the development environment.



```
C:\WINDOWS\system32\cmd.exe
C:\Projects\aimee-craig-sp>dotnet test
Test run for C:\Projects\aimee-craig-sp\WebbiSkools.QuizManager.FunctionalTests\bin\Debug\netcoreapp3.0\WebbiSkools.Quiz
Manager.FunctionalTests.dll(.NETCoreApp,Version=v3.0)
Microsoft (R) Test Execution Command Line Tool Version 16.3.0
Copyright (c) Microsoft Corporation.  All rights reserved.

Starting test execution, please wait...

A total of 1 test files matched the specified pattern.
Test run for C:\Projects\aimee-craig-sp\WebbiSkools.QuizManager.Tests\bin\Debug\netcoreapp3.0\WebbiSkools.QuizManager.Te
sts.dll(.NETCoreApp,Version=v3.0)
Microsoft (R) Test Execution Command Line Tool Version 16.3.0
Copyright (c) Microsoft Corporation.  All rights reserved.

Starting test execution, please wait...

A total of 1 test files matched the specified pattern.

Test Run Successful.
Total tests: 29
     Passed: 29
 Total time: 3.7140 Seconds

Test Run Successful.
Total tests: 55
     Passed: 55
 Total time: 9.1830 Minutes

C:\Projects\aimee-craig-sp>pause
Press any key to continue . . .
```

Figure 6: Unit and functional test results

The overall test coverage for the application is 84% at the time of writing, this is mostly due to instances of HttpContext which are difficult to mock and are not present when the tests are running as the HttpContext class is only accessible when a request is made to the server.



Symbol	Coverage (%)	Uncovered/Total Stmts.
Total	84%	36/227
WebbiSkools.QuizManager.Web	84%	36/227
WebbiSkools.QuizManager.Web.Controllers	70%	36/120
AccountController	65%	9/26
QuizzesController	71%	27/94
WebbiSkools.QuizManager.Web.Utilities	100%	0/5
WebbiSkools.QuizManager.Web.Data	100%	0/102

Figure 7: Test coverage

FEATURES

Below are features that I am particularly proud of.

OFFLINE USE

This project is built in such a way that it can be used completely offline. The database containing the quizzes, questions, answers and users is created and run from the user's machine. This does mean that quizzes created on one machine won't be accessible to someone running the application on another machine but this can be resolved by deploying the application to a server if need be.

FUNCTIONAL AND UNIT TESTS

I was able to include functional and unit tests within the project by creating these as I developed the product. This meant that my test coverage stayed high throughout the project and I was able to implement new features quicker as I could be confident that my changes had not inadvertently broken anything outside of what I was working on.

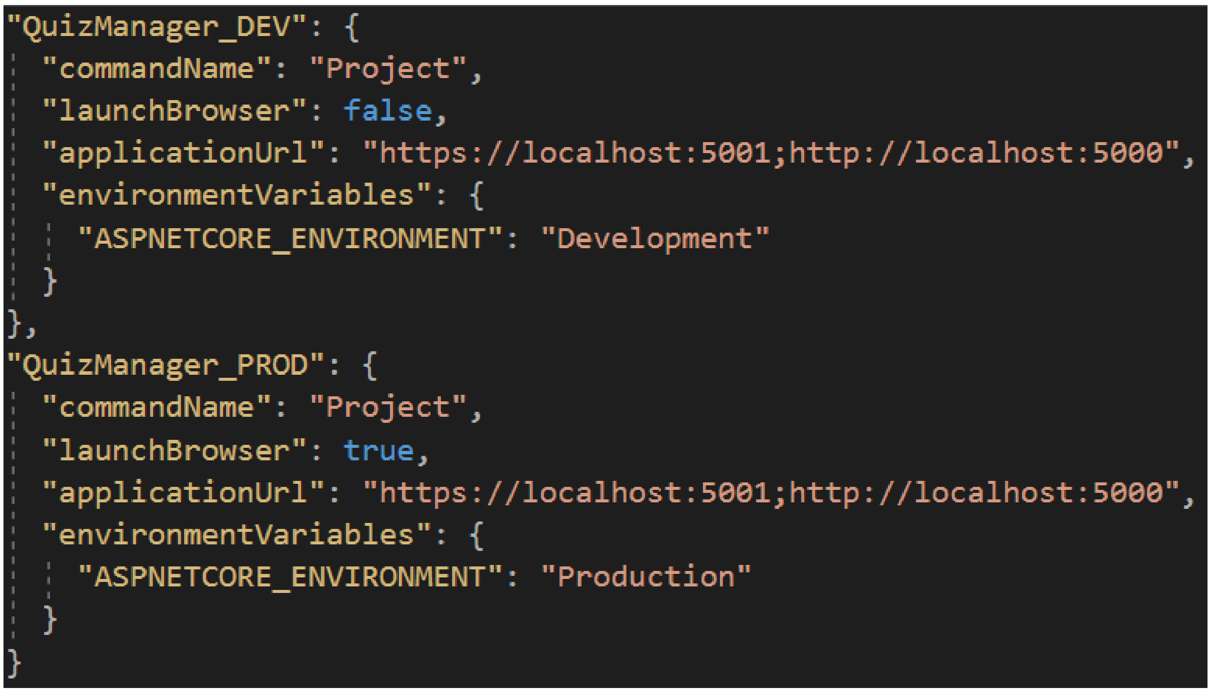
TEST DATABASE

My application has two environments from which to run from, a development environment and a production environment. This means that when the user is running the application in the development environment, the application is connecting to a different database altogether. Because of this, the user can confidently make changes to the database without touching live data.

This also means that my functional tests can run against a test database. I can be sure of the expected outcome of my functional tests as I know that users won't be able to add or delete quizzes in the database I am testing against.

ENVIRONMENTS

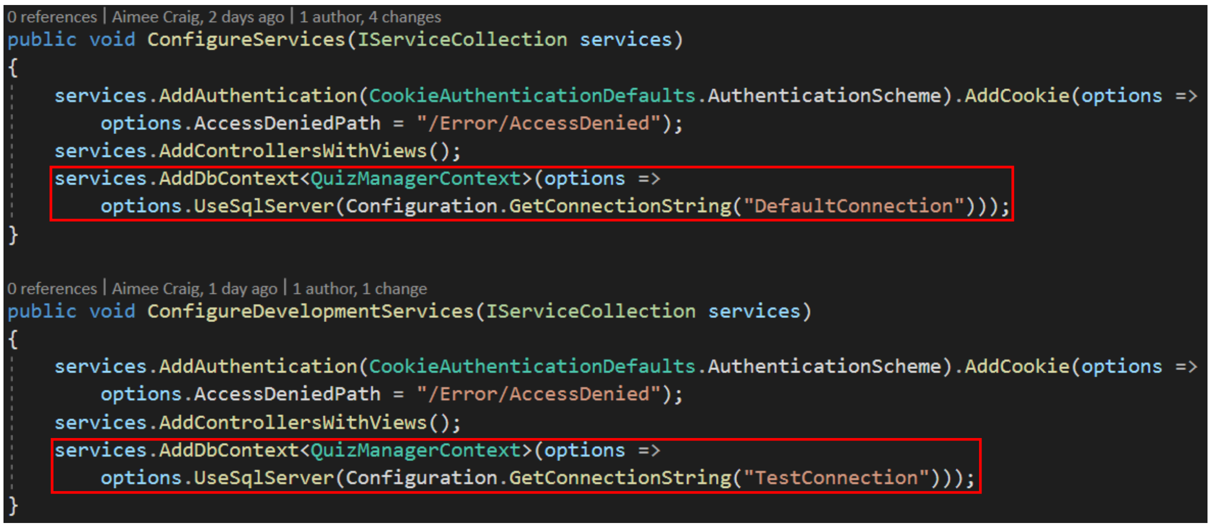
The application can currently run from one of two environments, production or development. The development environment connects to a different database when running the application. I was able to implement this using launch profiles.



```
"QuizManager_DEV": {
  "commandName": "Project",
  "launchBrowser": false,
  "applicationUrl": "https://localhost:5001;http://localhost:5000",
  "environmentVariables": {
    "ASPNETCORE_ENVIRONMENT": "Development"
  }
},
"QuizManager_PROD": {
  "commandName": "Project",
  "launchBrowser": true,
  "applicationUrl": "https://localhost:5001;http://localhost:5000",
  "environmentVariables": {
    "ASPNETCORE_ENVIRONMENT": "Production"
  }
}
```

Figure 8: Launch profiles

Each launch profile has an attribute called "ASPNETCORE_ENVIRONMENT" that can have a value of either "Development", "Staging", or "Production". .NET Core will query this attribute on application start and then decide which ConfigureServices method to use based on its value.

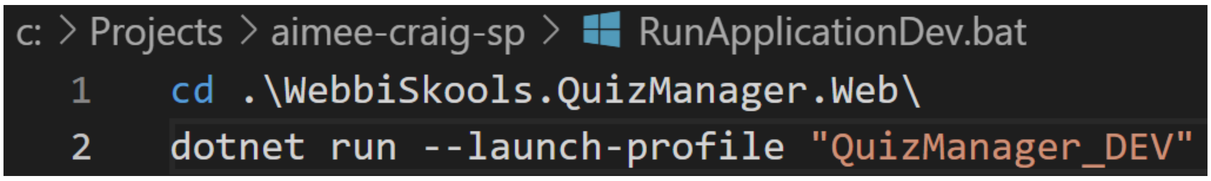


```
0 references | Aimee Craig, 2 days ago | 1 author, 4 changes
public void ConfigureServices(IServiceCollection services)
{
    services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme).AddCookie(options =>
        options.AccessDeniedPath = "/Error/AccessDenied");
    services.AddControllersWithViews();
    services.AddDbContext<QuizManagerContext>(options =>
        options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));
}

0 references | Aimee Craig, 1 day ago | 1 author, 1 change
public void ConfigureDevelopmentServices(IServiceCollection services)
{
    services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme).AddCookie(options =>
        options.AccessDeniedPath = "/Error/AccessDenied");
    services.AddControllersWithViews();
    services.AddDbContext<QuizManagerContext>(options =>
        options.UseSqlServer(Configuration.GetConnectionString("TestConnection")));
}
```

Figure 9: ConfigureServices methods

When the application is run in the Development environment, the ConfigureDevelopmentServices method is called instead of the ConfigureServices method and this, in turn, connects to a different database.

To run the application in the Development environment the name of the profile just needs to be passed as a command-line argument. This is what I have done in the **RunApplicationDev.bat** file.



```
c: > Projects > aimee-craig-sp > RunApplicationDev.bat
cd .\WebbiSkools.QuizManager.Web\
dotnet run --launch-profile "QuizManager_DEV"
```

Figure 10: RunApplicationDev.bat file

LIMITATIONS

REORDERING QUESTIONS

Currently, the user is unable to reorder questions in either the create or edit pages. The user can add a question at any point in the sequence of questions but say for instance the user wants to move the second question to the end of the list of questions, they would have to delete all the questions after the second question and then re-add them after the first question.

I have proposed a solution to this issue in the “Drag and Drop” sub-section of the “Future Improvements” section.

DELETE QUESTION/ANSWER CONFIRMATION

When a user clicks on the **Delete Question** or **Delete Answer** button, the question/answer is deleted immediately. The user is not given any chance to confirm the deletion so this could lead to the user accidentally deleting a question/answer.

To resolve this, I could instead set the event for the **Delete Question** and **Delete Answer** buttons to show an alert on the page with a button asking the user to confirm that they wish to delete the question/answer.

FRONT END VALIDATION

The user is able to create a quiz without a quiz title, questions or answers as there is no front-end validation present. This means that blank quizzes could be added to the database and then subsequently be available in the list of quizzes.

I have detailed the issue and possible resolution further in the “Front End Validation” sub-section in the “Future Improvements” section below.

FUTURE IMPROVEMENTS

DRAG AND DROP

I would like to add the functionality to drag and drop questions and answers within the create and edit pages as I feel this would improve the experience for the end-user. This would also resolve the limitation on reordering questions.

Fortunately, the functionality to drag and drop elements already exists within the jQuery UI framework which is already present within my solution. To add this functionality I would need to call “.sortable()” and “.disableSelection()” on the div containing the elements I would like to be able to drag and drop (in my case, this would be the div with an Id of “all-questions-and-answers” for dragging and dropping questions and the divs with the class “answers” for dragging and dropping answers). More information on jQuery UI Sortable can be found here.

FRONT END VALIDATION

.NET Core 3 MVC already includes the functionality to be able to add front end validation to forms with its jQuery Unobtrusive Validation framework. Unfortunately, this gets trickier the more complex the model you are trying to validate. Each input you want to add front end validation for requires an “asp-for” tag helper or a “data-val” attribute (as well as a “data-val-required” or similar attribute depending on what validation you are adding). A span is then required to hold the validation error message and this requires an “asp-validation-for” tag helper or “data-valmsg-for” and “data-valmsg-replace” attributes. The “data-valmsg-for” attribute value needs to match the name attribute value of the input to be validated in order to correctly validate and display any error messages.

The issue is that my inputs on the page can be dynamically generated and as such, I cannot use the “asp-for” tag helper. This is because the “asp-for” tag helper gets changed to name and Id attributes on page render which only occurs once. I would have to instead add the “data-val”, “data-val-required” (if used), “name”, and “id” attributes dynamically as well as the “data-valmsg-for” and “data-valmsg-replace” attributes on the input’s respective span.

However, I could still take advantage of the tag helpers by creating a partial view that contains a new set of question and answers inputs that use the “asp-for” tag helper and then add this to the page via an AJAX request as this would be rendered by the back-end upon return to the page. This would ensure that all attributes required for the front-end validation framework to work would be included on the page.

TEST DATABASE TEARDOWN

Although my functional tests are running against a separate database, there is currently no functionality in place for me to clear the database before each test. This means that I cannot be sure of the number of quizzes in the database to make assertions against. To combat this, I have added manual tests to my test plan that will check that when a quiz is created/edited/deleted that it is displayed appropriately in the application.

As the database is set up on application start and the application must be started before the tests run, I cannot replace the database with an in-memory database as I have done for the unit tests.

REGISTRATION

Although registration wasn't a requirement of this project, I feel as though including a registration system of some kind would be useful in the future. Currently, new users must be added to the database initialisation method and they will only be added if the database is empty at the time of running.

By adding a registration system, users could be added to the database without having to amend the DbInitialiser.cs file and then running commands in the command-line interface to delete the database first. This also means that users will be able to keep their current list of quizzes when adding a new user.

EDIT QUIZ RESET BUTTON

Currently, the **Reset** button on the edit quiz page will only reset the values that were originally present when the page first loads. If the user has added any questions/answers to the page then clicks on the **Reset** button, those question/answers won't be removed. Similarly, if the user has deleted any questions/answers and then clicks on the **Reset** button, those questions/answer won't be restored to the page.

To resolve this, I would use JavaScript and jQuery to set a new event on click of the **Reset** button that would make an AJAX call to the server that would then return the HTML for the edit page that would be populated with the data currently in the database. I could then delete the HTML currently on the page and replace it with the HTML that is returned from the server.

A quick fix for this issue would be to set a new event on the **Reset** button that refreshes the page as this also achieves the same thing. However, using an AJAX request to reset the HTML on the page would be less distracting for the end-user as the page would not need to be refreshed.