Synoptic Project - Quiz Manager

Ed Reeve - Software Engineer, Spektrix Ltd

This document is intended to act as a development diary, but will probably also contain user documentation. The structure will probably change on an hourly/daily basis.

Improvements / Further Development

- Deleting an empty answer option should be instant, but currently makes the user cycle through the confirmation stage.
- Most of the styling was done inline, as it was the easiest way to quickly adjust things on the fly during development. In order to make this an easier app to skin for different clients, this styling would need to be extracted to a CSS file.
- I had hoped to have time to create and add images, both as icons for actions like ‘Delete’ and ‘Edit’, to make the app more user friendly, and as e.g. Logos, to demonstrate that the site could easily be skinned and branded. Unfortunately I ran out of time. The MVC framework already makes use of Bootstrap and partial views, which are in place in this application, so that would be straightforward to do.
- I didn’t have time to work on inserting questions at a specified point, or reordering questions in a Quiz. However, the position of each question is being correctly set and recorded, so the ground work is there :-/
- Similarly, I didn’t have time to work on logging users out automatically and resetting session state, so users will stay logged in indefinitely. The management of this is an existing Identity feature, so again, with a little more time this should be relatively straightforward.
- I wrote a bunch of javascript in the cshtml pages, and would have liked to extract this into the wwwroot folder at some point
- It would have been nice to put a search field on the Quiz Database page :-(
- The data model and application structure was intended to allow for further feature expansion, e.g.;
    - The creation of random / pot luck quizzes based on the existing Question database; a user could provide parameters to generate a selection of questions at random.
    - Categorisation of quizzes.
    - Extra information displayed against quizzes, e.g. number of times taken, total number of questions
    - The ability to take quizzes in the app
    - Full user management; registration, profile management, two factor authentication etc
    - The ability for users to see their past results
    - The use of results to indicate how ‘hard’ particular quizzes are

User Guide

The Quiz Manager has 3 users preconfigured with the following details;

Username : ‘Edit’

Username : ‘View’

Username : ‘Restricted’

All three users have the same password: ‘P@ssword1’. They each have different permission levels, allowing them to perform certain actions.

All Users

Once you’ve logged in, you should see a list of dummy Quizzes. As any user, click on a quiz to be taken to a page showing you all of the questions it contains.

While logged in, you can return to the main Quiz Database at any point by either clicking the ‘Quiz Manager’ logo in the top left corner, or the ‘Quizzes’ link in the menu bar. There are also navigation links on most pages, and you can use the browser’s forward/back buttons, too.

At any point, you can click Logout in the top right corner to return to the main screen.

Edit and View Users only

If you are logged in as either the ‘Edit’ or ‘View’ role, you will be able to click on any question when viewing a quiz, in order to see all available answers for it. Click the question again, or select another, in order to hide the questions again.

Edit Users only

Creating a quiz

From the main Quiz Database page, you can create a new Quiz at any point by clicking the ‘Create new’ button. On the following page, enter a name for your quiz and click ‘Create’. You’ll be returned to the Quiz Database, which will include your newly created quiz.

Click your quiz. You’ll be taken to the quiz details page, ready to add your questions. Click the ‘Add question’ button at the top of the page to get started.

Adding Questions

Each question should have some text to begin with, and then between 3 and 5 potential answers. You can select which of the answers is correct using the Checkboxes next to each answer. Once you've filled in as many answers as you'd like, click 'Create', and you'll be taken back to the main quiz details page. Any blank answers will be automatically deleted.

You can add as many questions as you like. Once you're finished, simply navigate back to the home page using the menu links, or the 'Back to Quiz Database' link at the bottom of the quiz page.

Editing Quizzes and Question

You can rename quizzes from the main Quiz Database page by clicking the 'Edit' option next to any quiz. Simply enter the new name on the following page and click 'Save'. If you've already clicked on a quiz, you can rename it using the 'Rename Quiz' button at the top of the page.

To edit any question, simply click the 'edit' button next to the question, and you'll be taken to a page where you can update the question text, and the text of any answers, as well as deleting answers or adding new ones, and changing which is marked as correct.

Once an item has been deleted, it's gone for good, so you'll always be asked to confirm any time you try to delete something.

Once you've finished editing the question, just click 'Save' to return the Quiz's list of questions.

Quizzes and Questions can also be deleted using the links next to them. Remember, the deleting a Question will also delete its Answers, and deleting a Quiz will also delete its Questions, along with their answers.

Decision Log

Day 1

Tooling / Admin - Metro Retro I used Metro Retro as the planning tool. It's extremely flexible, allowing users to create their own 'boards' in whatever layout they choose, and acts as a huge whiteboard where you can stick as many virtual post-its as you like, annotating them with emojis etc

Tooling / Admin - Trello In addition to Metro Retro, I wanted a tool which easily provided more control over tasks, and the ability to lay them out in a clear and logical manager. Trello serves that purpose, using a List/Card structure that can clearly map to a workflow, with cards able to contain checklists and so on, for tracking tasks in a more granular way.

Admin To ensure I keep on top of the documentation side of the project, I'll add two daily calendar reminders in my Google Calendar. The first will be a 15 minute period before lunch, to see if there's anything from that morning I neglected to document, and the second will be a 30 minute end of day recap, in which to record a mini-retro as well as catching anything I missed in the afternoon. This should serve to not only ensure documentation is up to scratch, but give me space to check how progress is coming along and readjust my approach and/or priorities as required.

Language - C# The language I want to use is C#. I've worked almost exclusively in C# during my placement, since working with Ruby, Javascript and C# during the initial 3 month course at Makers Academy. I feel that my familiarity with the language will speed up development, and given the short length of the project, this seems crucial

Framework - .NET Core 3.1 Working with C#, I'll use the .NET Core 3.1 framework. This provides cross-platform support, and has many infrastructure platform options, not least Azure. It plays nicely in containerised services with tools like Docker and Kubernetes, and due to its cross-platform nature is a good choice for a distributed environment. The project brief outlined that the actual taking of quizzes will be managed by another service, so this fits well.

Tooling / Admin - Draw.IO, GSuite For the rest of the planning, which will be more to do with the technical aspects of the application, I'll use a mix of Draw.IO and Google Suite apps. I've used both of these tools a fair amount in my placement job, and they cover most kinds of document creation, from text to detailed diagrams.

Tooling - Visual Studio 2019 I'll use Visual Studio 2019 as my IDE for this project. It's what I use as my principal IDE during my placement, so doesn't require any additional learning to use. I may also use Visual Studio code to supplement it for certain tasks, as it's more lightweight and has a very wide array of extensions that can make some tasks quicker.

Application - MVC To improve the speed of development, I'll create a standard .NET Core MVC application. There are scaffold/templating options which make it very quick to build up the bones of the main application, especially within Visual Studio

Styling - Bootstrap I'll use Bootstrap for the styling of the application; it's a widely adopted, easy to use framework, and is included in scaffolded MVC applications by default

Tooling - Identity Framework As the project requires that the app supports users with varying permissions, I'll include the Identity framework when creating the MVC app. This provides a fully-featured user management platform, which will actually provide more functionality than is required by the project outline, however it's possible to disable things as required while still retaining the robust framework. This means that in future if the app's user functionality were to be expanded, it could be done quickly and easily.

Tooling - Database Management The Identity Framework makes use of Entity Framework core by default, an ORM (Object relational mapper) that provides an easy to use interface between the code and the database. By default I believe it uses a SQL Server / LocalDB, but allows connections to be made to a number of DB types and providers. For the purposes of the project, which doesn't require the app to be fully hosted, I'll try to use a LocalDB, so that whoever runs the application has a locally stored DB or DB file. This would be easily swapped out in production for a properly hosted independent database.

DAY 2

Working Practices Before starting a task, I'll estimate how much time needs to be spent on it, given the other tasks lined up for the day. At the end of that time I'll either move on, or identify blockers and timebox their completion.

Tooling - Testing Framework Again, due to familiarity from my placement job, I'll use NUnit and Fluent Assertions for testing my code. Fluent Assertions will allow me to write code in more easily legible syntax, while NUnit is a widely used testing framework.

Tooling - Testing Framework Led by the Microsoft / Entity Framework documentation, I decided to use an in-memory SQLite database for my test. As another relational database type, it would closely mirror the functionality of SQL Server, the DB type I was using for the main application. Having it run in memory means an instance of the database can be spun up, configured and then disposed of by every test that needs it.

Working Practices 'Program Code and Testing' will comprise the bulk of the project, fingers crossed. As such, I will try to use Trello to break User Stories and Project Tasks down into their component parts, or 'Tickets', and keep track of them in separate Trello Lists. This will allow me to clearly scope out what is

involved in each piece of functionality, feature or testing, and work through each identified task sequentially.

DAY 3

Working Practices Clarifying the decision made yesterday; I'll use Trello to track the coding and testing work. There'll be one list per previously defined user story, with the option to add more as required. Each list will contain one card per task, and each task will have acceptance criteria I'll define during planning.

Working Practices - Testing The bulk of my testing will be manual, and comprise of satisfying Acceptance Criteria on tickets, with some end to end and exploratory testing thrown in. Automatic / Code tests will only be written when the code complexity demands it.

Development Diary

To keep a record of the development process, I'll be adding a page for each of the 5 days of the project. Each page may contain sub-pages; time will tell as I figure out the best format.

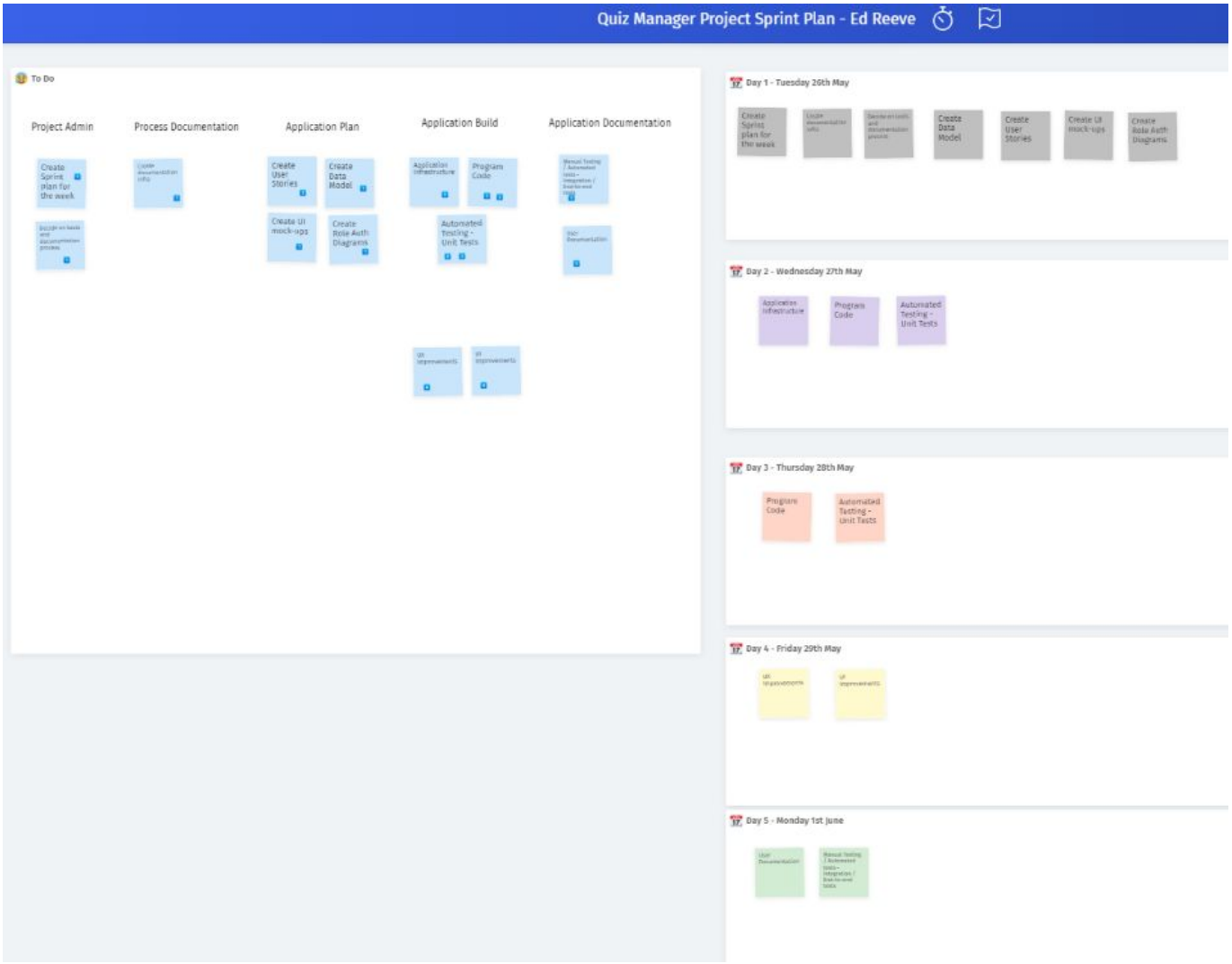
The aim of the diary is to keep a centralised record of the overall process, and track decisions etc, while also providing a hub to add content from other tools used in my work.

Day 1 Tuesday 26th May

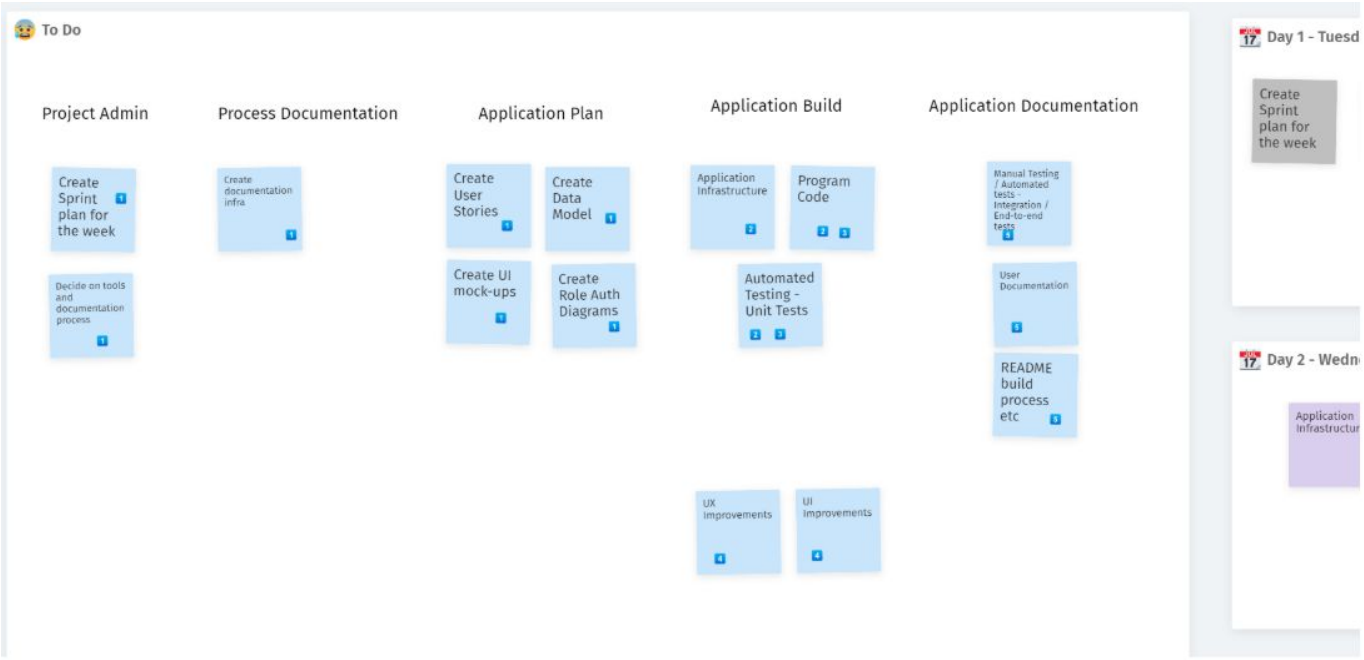
Upon receiving the full brief for the Quiz Manager project, I first wanted to plan out the following week in order to have a structure to follow.

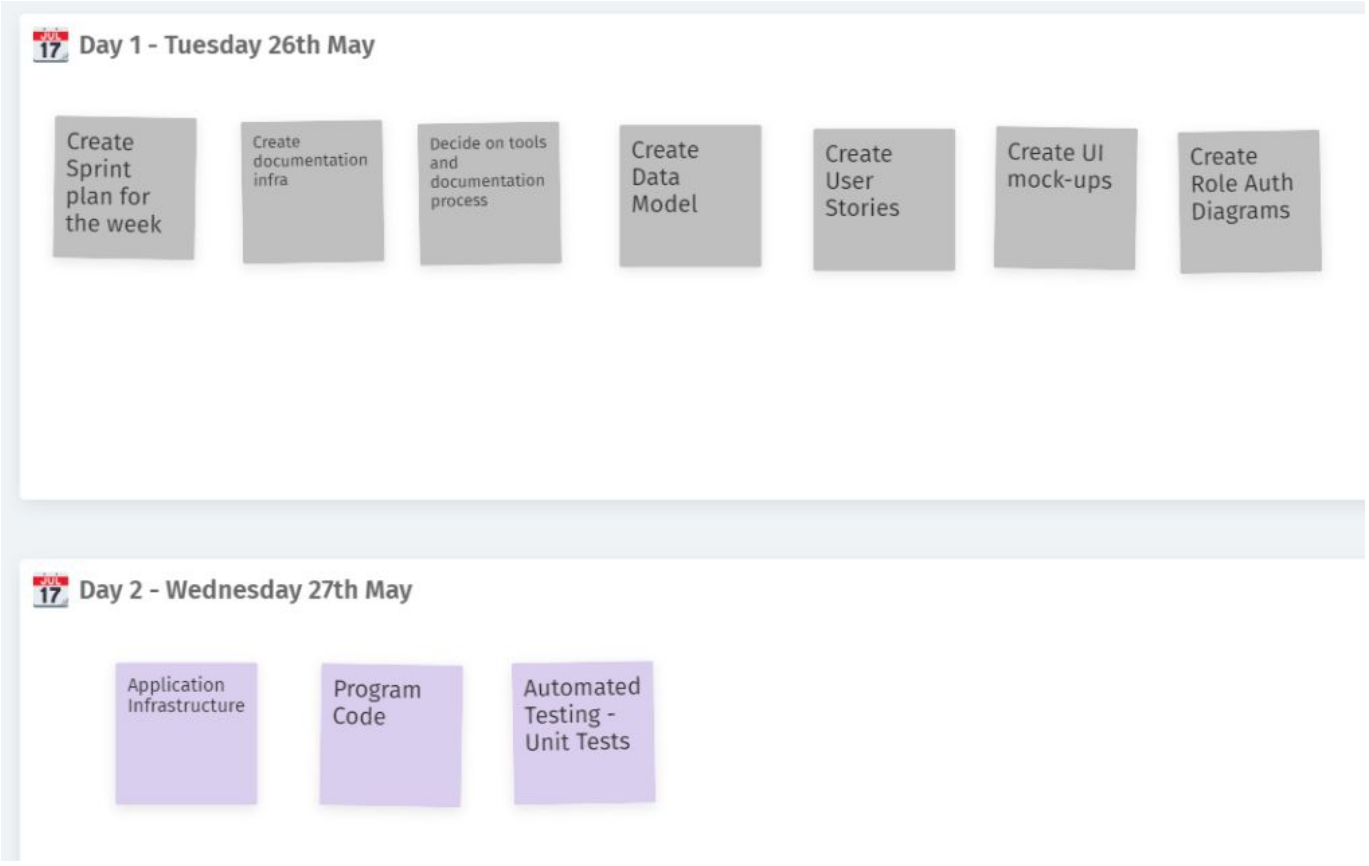
- ***Decision*** *I'll use Metro Retro as the planning tool. It's extremely flexible, allowing users to create their own 'boards' in whatever layout they choose, and acts as a huge whiteboard where you can stick as many virtual post-its as you like, annotating them with emojis etc*

Here's the initial plan;



Having started with a general ToDo section, I split the project into 5 high level stages, then added post-its for the main steps that would comprise each. These were then copied across to days, so I could loosely plan which bits of the work I should be concentrating on at any particular time;



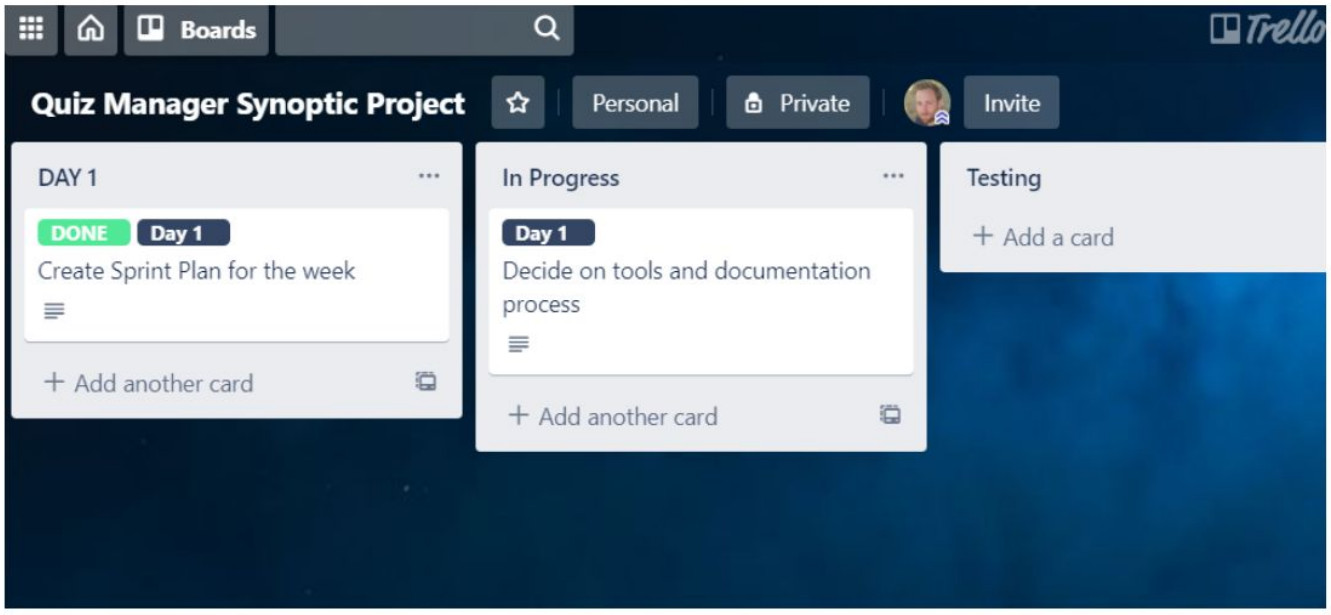


Once the plan was roughly in place, I started working through the tasks in order.

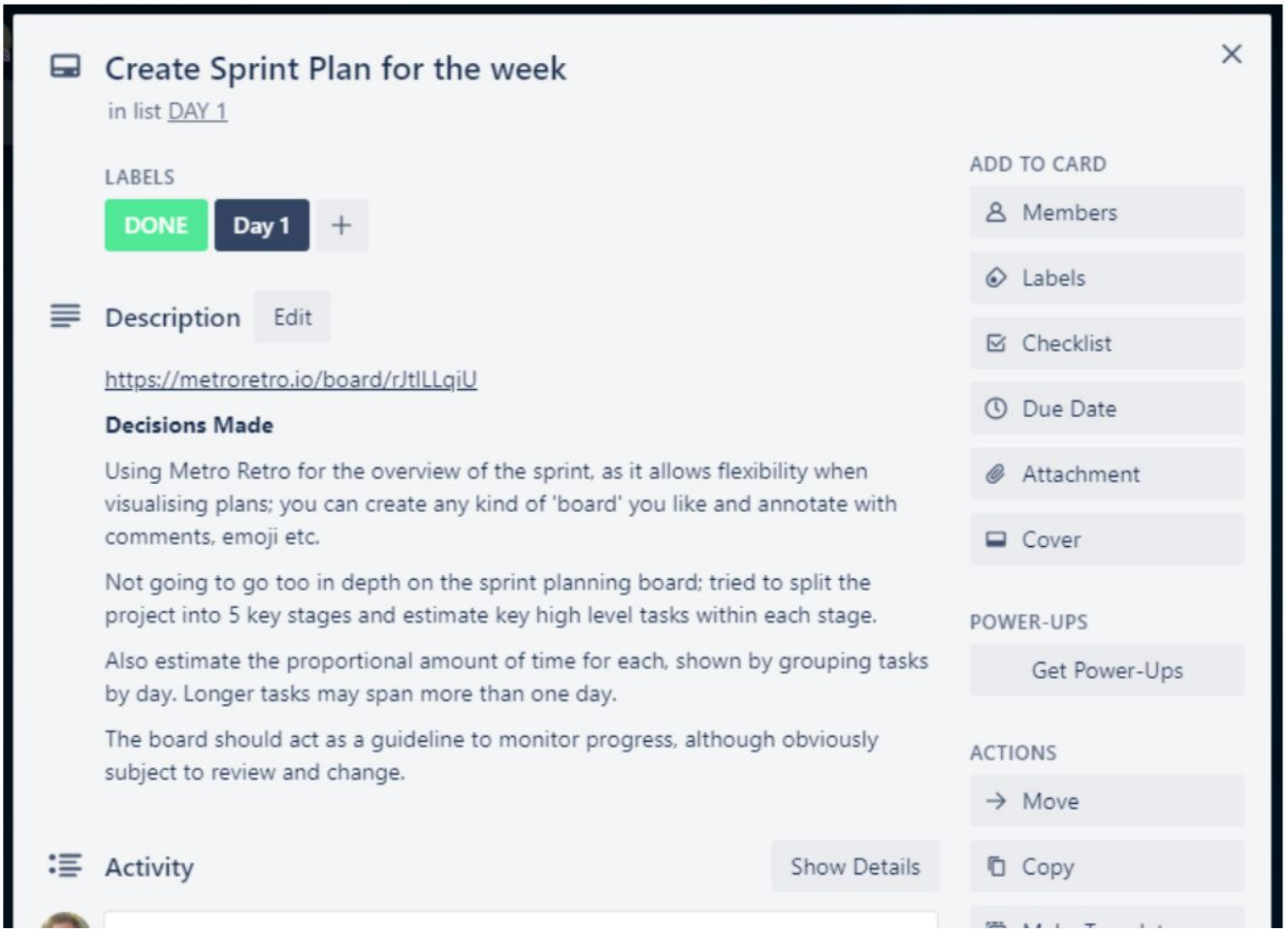
- ***Decision*** *In addition to Metro Retro, I wanted a tool which easily provided more control over tasks, and the ability to lay them out in a clear and logical manager. Trello serves that purpose, using a List/Card*

structure that can clearly map to a workflow, with cards able to contain checklists and so on, for tracking tasks in a more granular way.

To track each task as I was working on it, I created a Trello board with some basic workflow columns and created a new card for each task;



Information about the work needed for each task was added to their card;



For the admin of logging project work and documenting decisions, I thought it best to play it by ear, and hopefully manage to record things as I went. However, so as not to let it fall by the wayside, I wanted to remind myself to check for any outstanding things I should be recording, twice daily.

- ***Decision*** *To ensure I keep on top of the documentation side of the project, I'll add two daily calendar reminders in my Google Calendar. The first will be a 15 minute period before lunch, to see if there's anything from that morning I neglected to document, and the second will be a 30 minute end of day recap, in which to record a mini-retro as well as catching anything I missed in the afternoon. This should serve to not only ensure documentation is up to scratch, but give me space to check how progress is coming along and readjust my approach and/or priorities as required.*

For the application itself, I made the following decisions about the infrastructure I intended to use;

- ***Decision*** *The language I want to use is C#. I've worked almost exclusively in C# during my placement, since working with Ruby, Javascript and C# during the initial 3 month course at Makers Academy. I feel that my familiarity with the language will speed up development, and given the short length of the project, this seems crucial*

- ***Decision*** *Working with C#, I'll use the .NET Core 3.1 framework. This provides cross-platform support, and has many infrastructure platform options, not least Azure. It plays nicely in*

containerised services with tools like Docker and Kubernetes, and due to its cross-platform nature is a good choice for a distributed environment. The project brief outlined that the actual taking of quizzes will be managed by another service, so this fits well.

- ***Decision*** *For the rest of the planning, which will be more to do with the technical aspects of the application, I'll use a mix of Draw.IO and Google Suite apps. I've used both of these tools a fair amount in my placement job, and they cover most kinds of document creation, from text to detailed diagrams.*

- ***Decision*** *I'll use Visual Studio 2019 as my IDE for this project. It's what I use as my principal IDE during my placement, so it doesn't require any additional learning to use. I may also use Visual Studio code to supplement it for certain tasks, as it's more lightweight and has a very wide array of extensions that can make some tasks quicker.*

- ***Decision*** *To improve the speed of development, I'll create a standard .NET Core MVC application. There are scaffold/templating options which make it very quick to build up the bones of the main application, especially within Visual Studio*

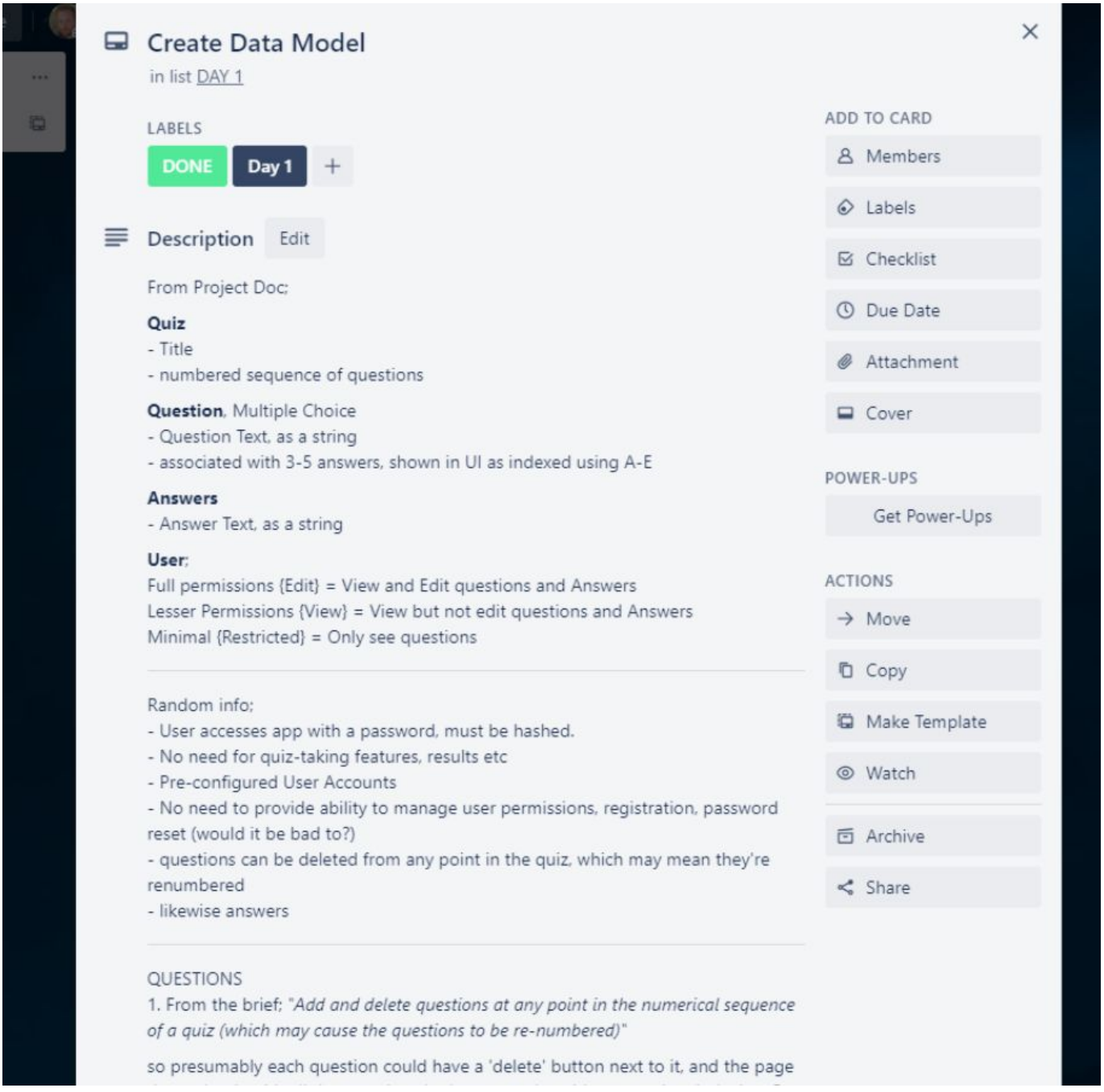
- ***Decision*** *I'll use Bootstrap for the styling of the application; it's a widely adopted, easy to use framework, and is included in scaffolded MVC applications by default*

- ***Decision*** *As the project requires that the app supports users with varying permissions, I'll include the Identity framework when creating the MVC app. This provides a fully-featured user management platform, which will actually provide more functionality than is required by the project outline, however it's possible to disable things as required while still retaining the robust framework. This means that in future if the app's user functionality were to be expanded, it could be done quickly and easily.*

- ***Decision*** *The Identity Framework makes use of Entity Framework core by default, an ORM (Object relational mapper) that provides an easy to use interface between the code and the database. By default I believe it uses a SQL Server / LocalDB, but allows connections to be made to a number of DB types and providers. For the purposes of the project, which doesn't require the app to be fully hosted, I'll try to use a LocalDB, so that whoever runs the application has a locally stored DB or DB file. This would be easily swapped out in production for a properly hosted independent database.*

Data Models

Trello card with working and plans here;



While reading through the project brief, I was unclear about the following;

1. From the brief; "Add and delete questions at any point in the numerical sequence of a quiz (which may cause the questions to be re-numbered)"

so presumably each question could have a 'delete' button next to it, and the page then reloads with all the questions in the new order with appropriate indexing. But when it says 'Add, it seems silly to have to have

an 'add here' button next to each item? Perhaps a general Add button, and the ability to re-order the list as required? If reordering is required/intended, then they'll need to be stored with order info

Or when adding, specify the index you're adding to?

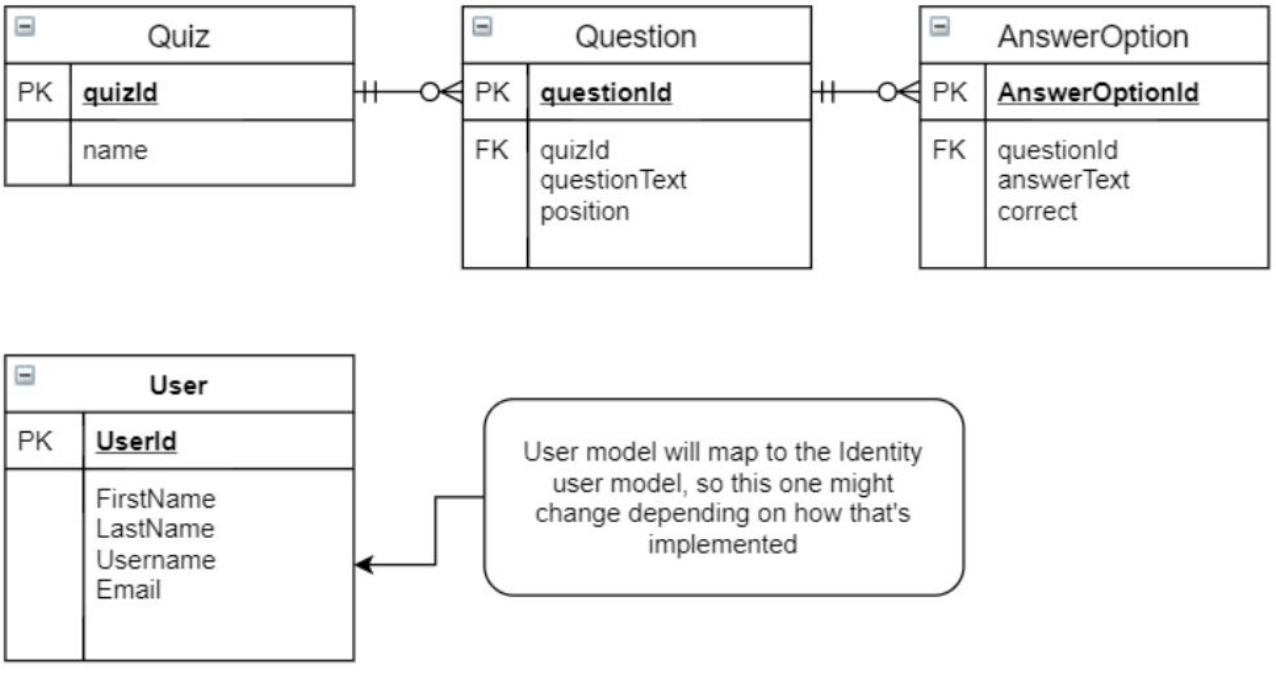
If the order is important, it'll need to be stored in the DB as a separate property, presumably... can't imagine EF does any ordering, but will check

Until implementing this feature, I'm not 100% sure how I'll approach this, but assuming I'll want to add and preserve ordering to a Quiz's questions (it doesn't stipulate the same requirement for answers), then each question will need an additional property to record this. Their IDs will be set by the DB and won't change, so can't be used for ordering purposes.

2. The brief doesn't mention anything about marking or showing answers as true or false. This is presumably useful? If so, maybe in line with the Restricted/View/Edit model? i.e. Restricted can't see the answers anyway; View can see them but can't do anything with at all (e.g see if they're true/false), whereas Edit can create/edit them, so will presumably be able to mark them as true/false and see which is currently set

Again, not 100% what I'll do about this yet, but will add a true/false property to the Answer class to facilitate it.

Intended Database Models



User Stories

1. As a user of any kind
   I want to log into the site
   So that I can view a list of quizzes

2. As a user of any kind
   I want to be able to scroll pages up and down
   So that I can see all content on every accessible page

3. As a Restricted or View user
   I want to select a quiz
   So that I can view a list of its associated questions

4. As an Edit user
   I want to select a quiz
   So that I can view and edit a list of its associated questions

5. As a View user
   I want to select a question
   So that I can view a list of its associated answers

6. As an Edit user
   I want to select a question
   So that I can view and edit a list of its associated answers

7. As an Edit user
   I want to create a quiz
   So that it is saved in the database and appears in the list of quizzes

8. As an Edit user
   I want to edit a quiz
   So that I can change its name

9. As an Edit user
   I want to delete a quiz
   So that the quiz and all associated questions and answers are removed from the application and database

10. As an Edit user
    I want to edit a quiz's list of questions
    So that I can delete or add questions to the list, and change their order

11. As an Edit user
    I want to edit a question
    So that I can change its text

12. As an Edit user
    I want to edit a question's list of answers
    So that I can delete or add answers to the list, and change which is marked as true/false

13. As an Edit user
    I want to delete a question
    So that the question and all associated answers are removed from the application and database

14. As an Edit user
    I want to edit an answer
    So that I can change its text and whether it is true or false

15. As an Edit user
    I want to delete an answer
    So that the answer is removed from the application and database

16. As a user of any kind
    I want to manually log out of the site
    So that my session is securely terminated and I am returned to the login page

17. As a user of any kind
    I want to be logged out of the site automatically after a period of inactivity
    So that my session is securely terminated and I am returned to the login page

---

Wireframes

Questions

1. Brief says Questions should have between 3 and 5 answers, but doesn't say anything about max/min questions per Quiz. Do anything about this? Think it'd make sense to let people create quizzes with no questions to come back to later, but then why not do so for Questions, too? Maybe just set an arbitrary minimum? Unsure.

2. Don't know whether clicking a question (for authorized users) should load a separate page with the Q and Answers, or perhaps open an accordion section of a card to display them? Second option would possibly be nicer UX, but would mean loading all answers along with each question, so more DB intensive. Don't know whether to worry about DB stuff given the tiny amount of data involved at this stage. Even if in widespread use, the data is simple and queries uncomplicated, so probably not a concern.

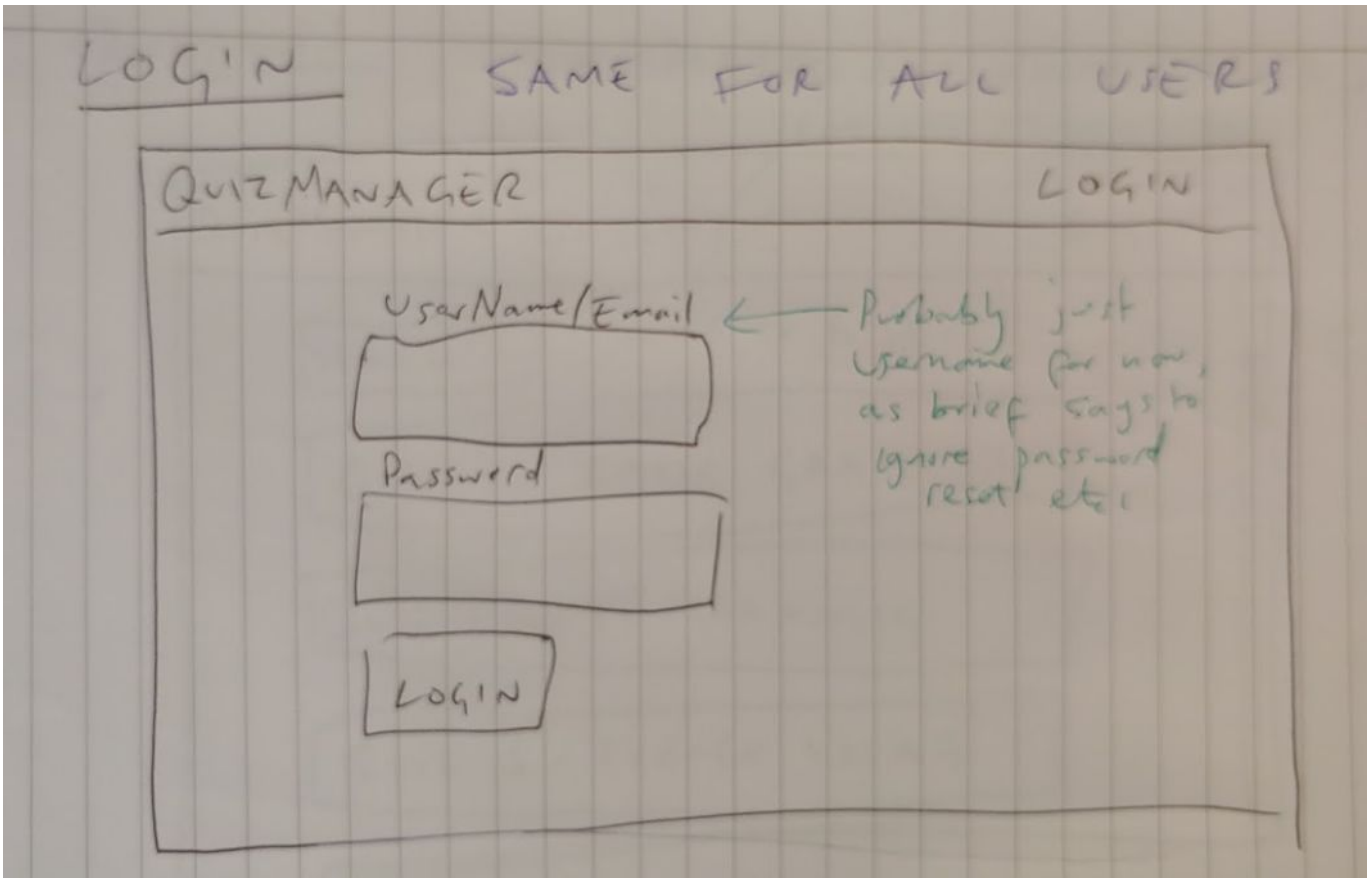
Assumptions / Ideas

1. It seems like it'd be bad UX to have people click through three layers of pages to see a quiz, then a question, then the answers. I don't know the behaviour of EF Core when using a code-first approach, which is what I'd like to try, and involves defining data models as classes and letting EF create the associated Database tables. It presumably links associated classes with Foreign Keys automatically, but I don't know if fetching an object from the DB will automatically fetch it's associated/children items. Will test, but may need to adjust the models to include ViewModels, one for the Quiz (containing a list of Question View Models), and one for the Question (containing a list of answers). This would enable me to display - potentially - all three layers on the same page, by using both view models. Concerns would be around DB access, if loading the Quiz Index could potentially request every single quiz and child object from the DB, but as per Question 2 above, I'm not sure if this is an issue. Could look to Lazy loading, in memory caching etc to mitigate this, but perhaps that'd be for a further iteration of the project. *This may all be completely irrelevant - will see what the EF behaviour is and whether it can be tweaked if necessary.*

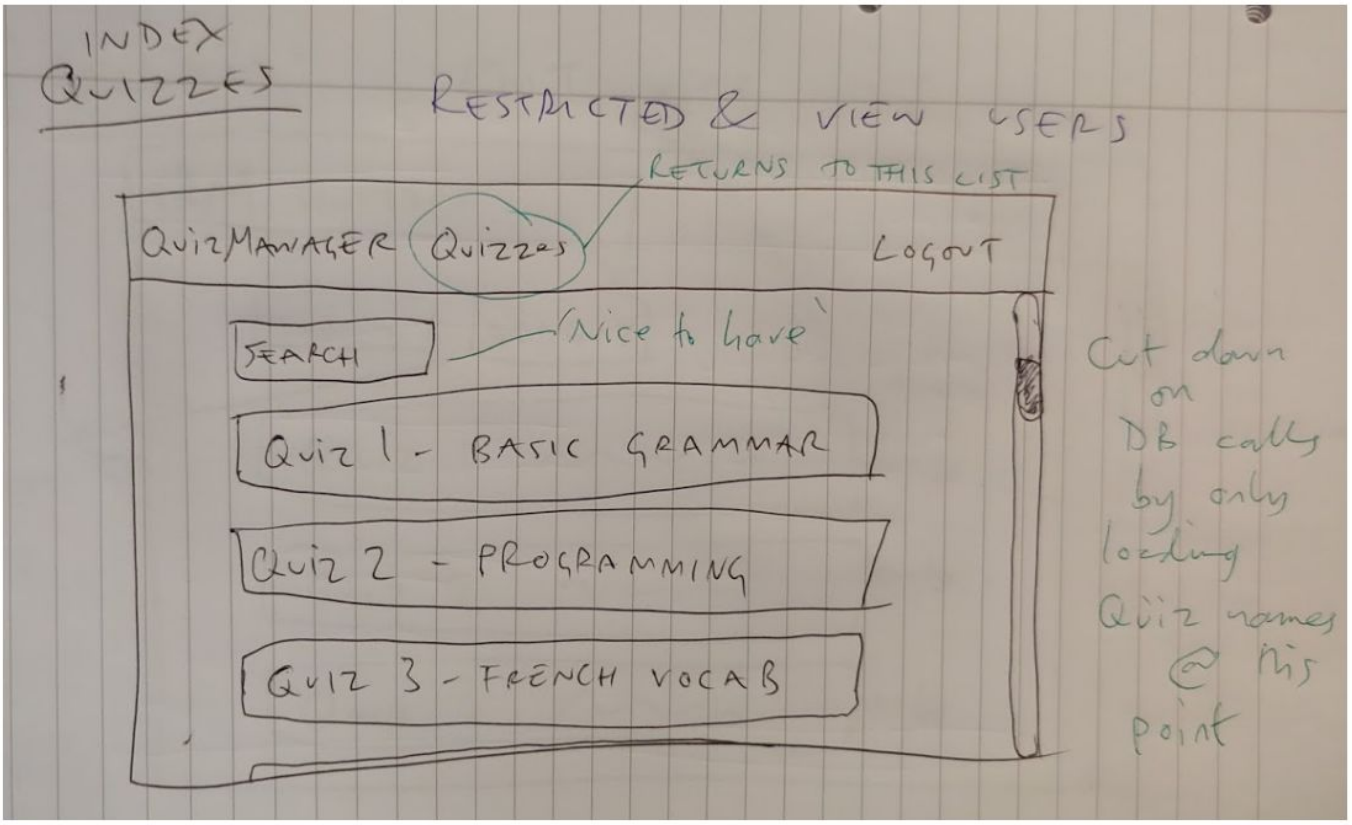
2. In line with the above, it would be preferable, however it's done, to view a *Quiz + Questions + Answers* as one item that can be edited as such, with granular control over each accessible property of the item, including its answer options.

3. Also, not that familiar with partial views, other than those used for layout etc, but maybe have partial views for e.g. question creation/editing, which then get loaded into the quiz details view as needed?

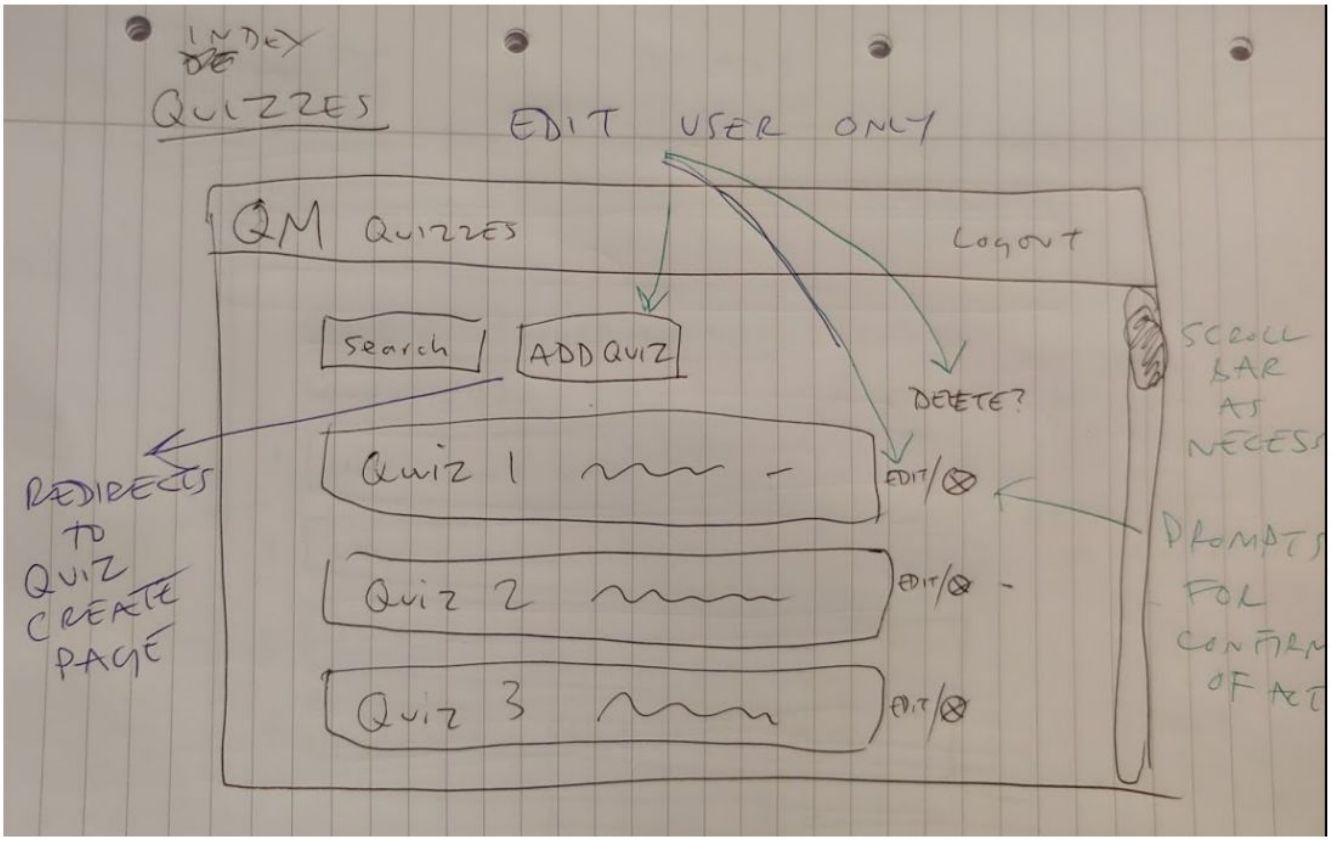
LOGIN



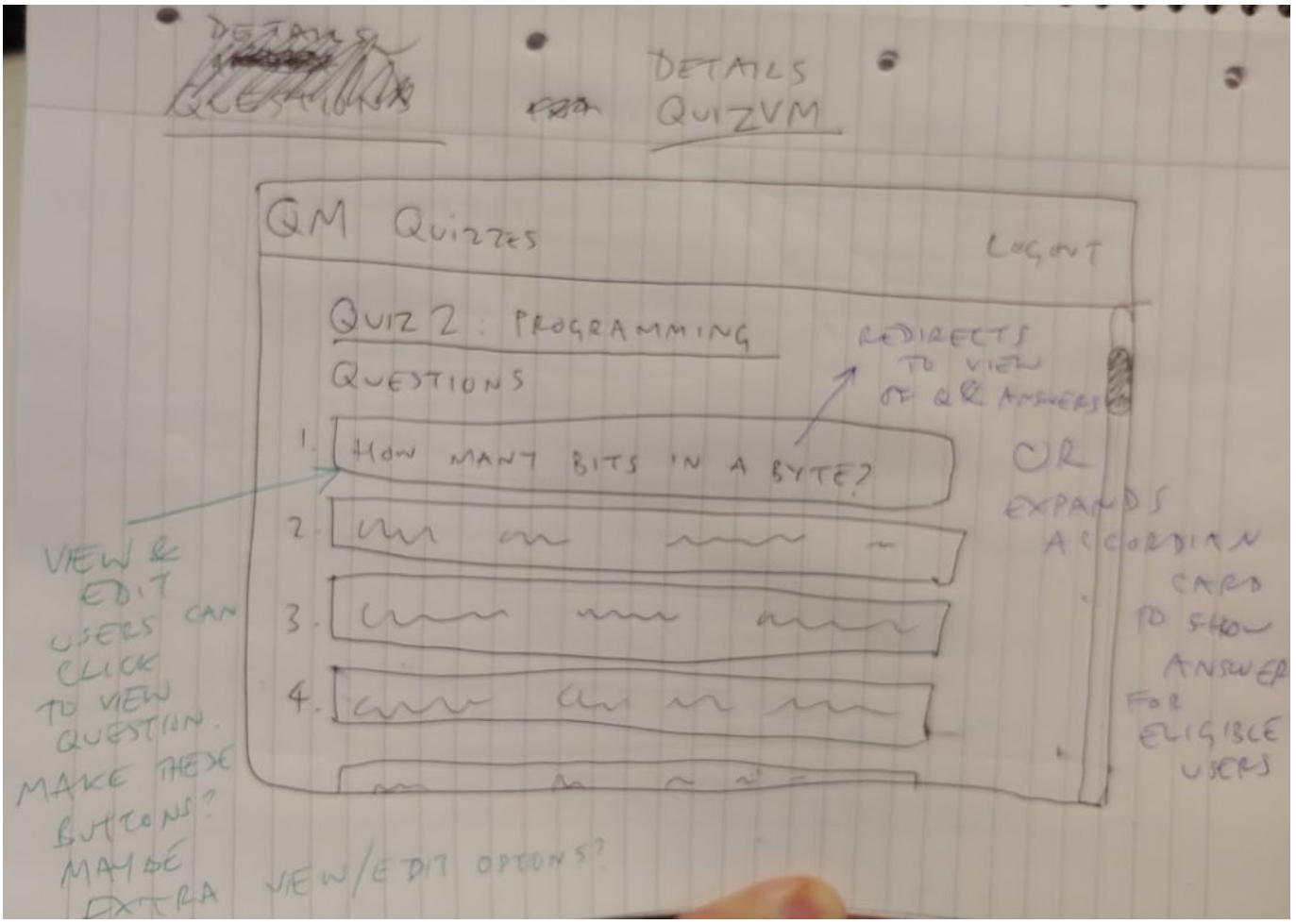
QUIZZES INDEX (Restricted and View Users)



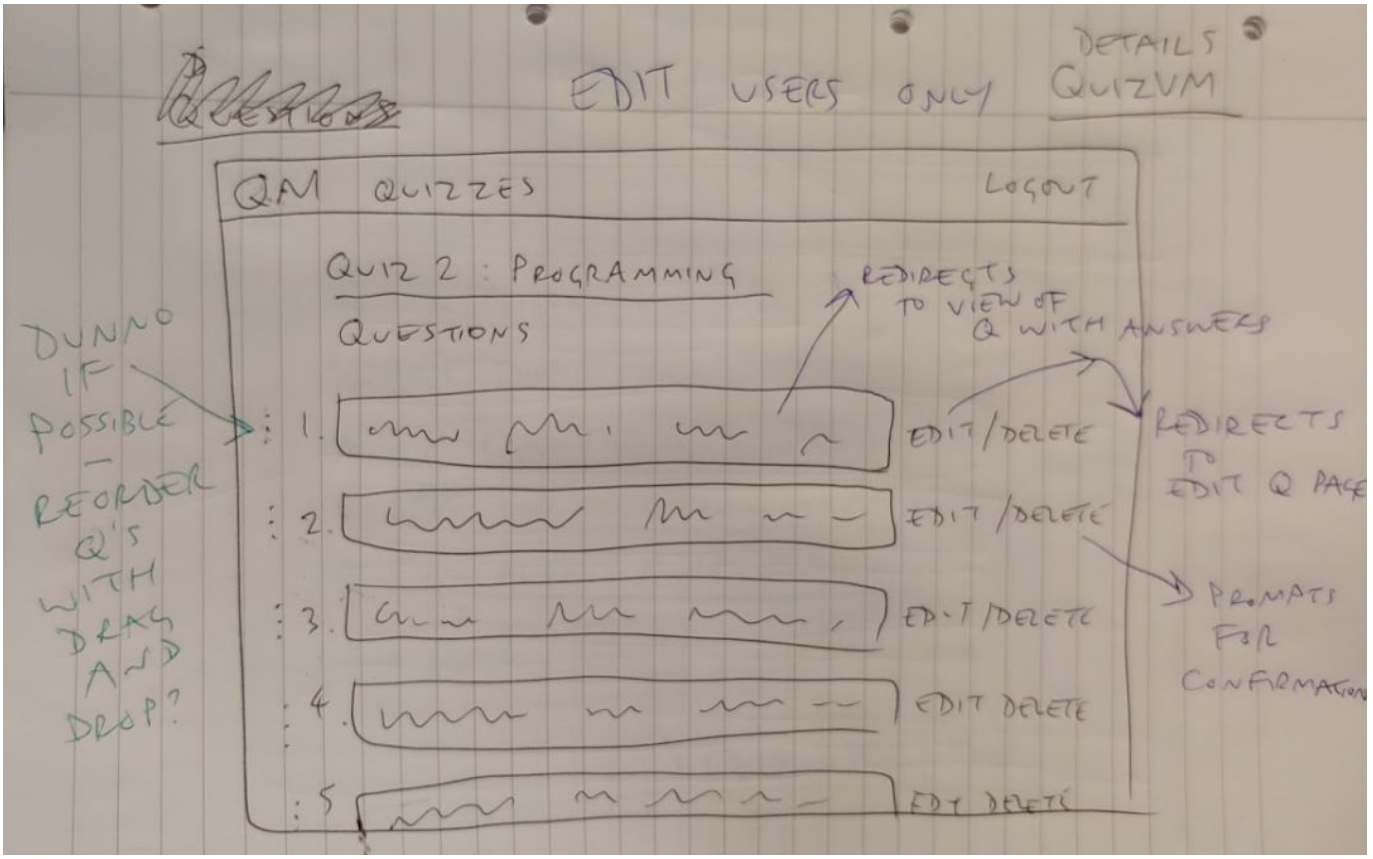
QUIZZES INDEX (Edit Users)



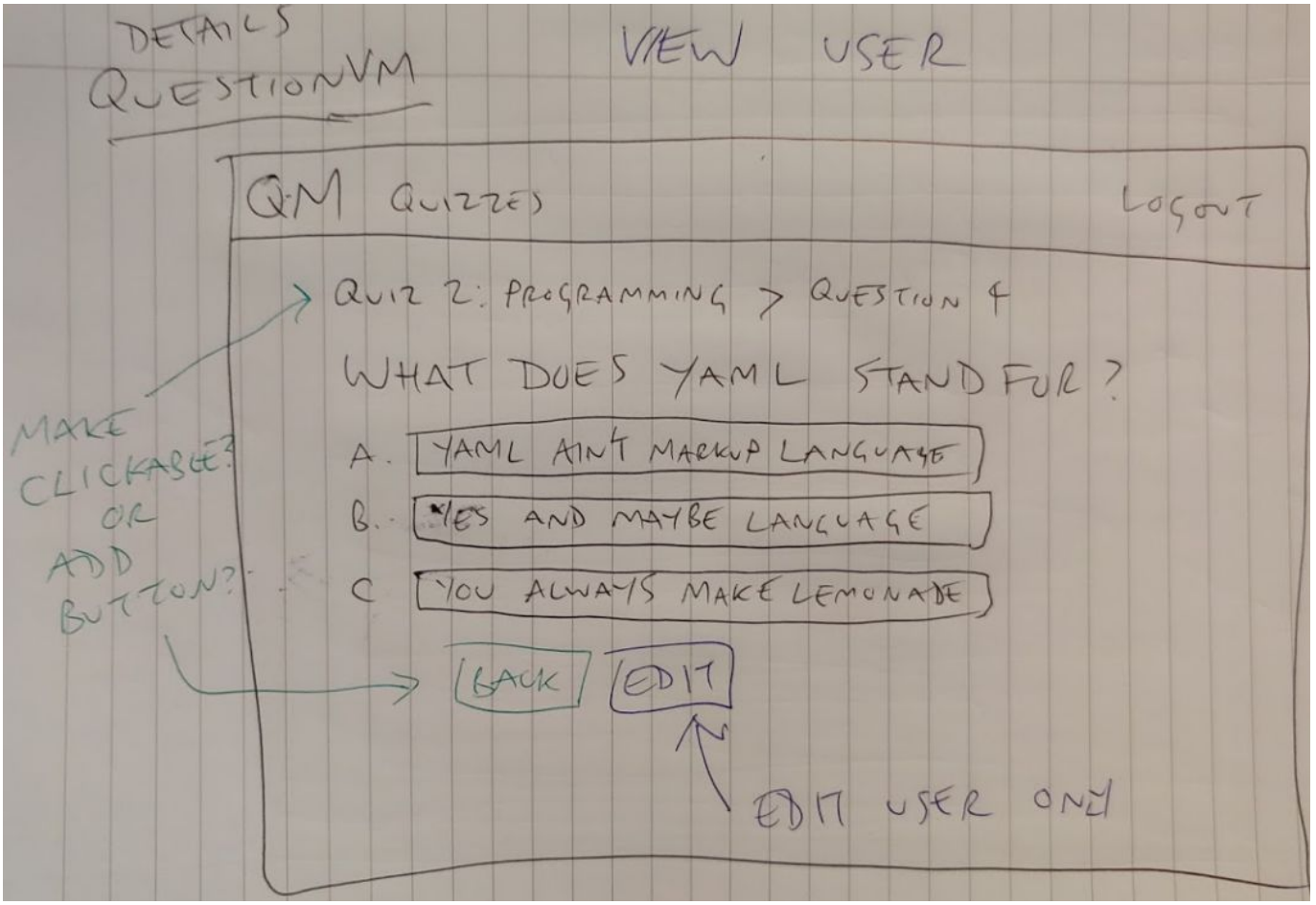
QUIZ DETAILS



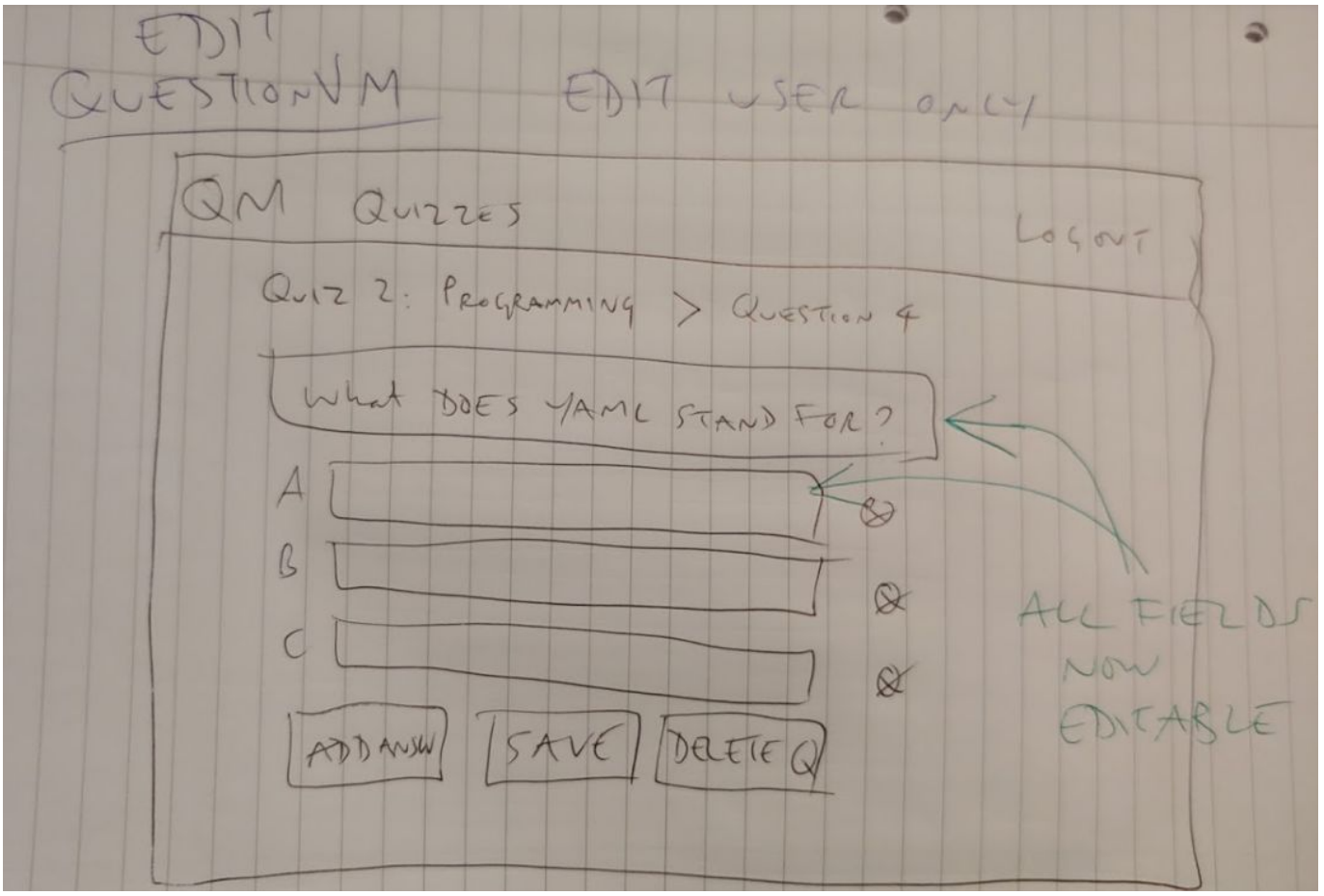
QUIZ DETAILS (Edit Users)



QUESTION DETAILS



QUESTION EDIT



Daily Retro, Day 1

Day 1

Brain Dump	Proud of
Hopefully the time spent planning will pay off, and the rest of the project will be a case of implementing things one by one without time lost deliberating. We'll see. Quite a few questions I'm unsure about. Will do some reading about UI patterns for edit/deleting items, I think. JUST REALISED THE WIREFRAMES DIDN'T INCLUDE ANSWER TRUE/FALSE :-(Also need to add EF to the decision / Tooling section	• Thoroughness of planning • Wiki for documentation • Good project outline to follow
Could do better	**Next on the plan**
• Spend less time planning! • I thought drawing the wireframes by hand would be quicker, but in hindsight it would have been clearer and easier to use e.g. Draw.IO • Manage time better in general; I forgot to take a second break, so had to finish early, so couldn't do this retro doc on the day, and had to fill in the following morning	• Very quickly finish the outstanding Day 1 tasks (Create UI mockups, create auth diagrams) • Crack on with getting the app up and running

Day 2 Wednesday 27th May

Starting the day!

I realised after sketching out some wireframes that I'd neglected to include any UI ideas for showing true/false answers. While this functionality wasn't mentioned in the brief, I think it'd be useful, but having spent a full day on prep, I feel my time would be better spent getting on with building the application, rather than revisiting the wireframes.

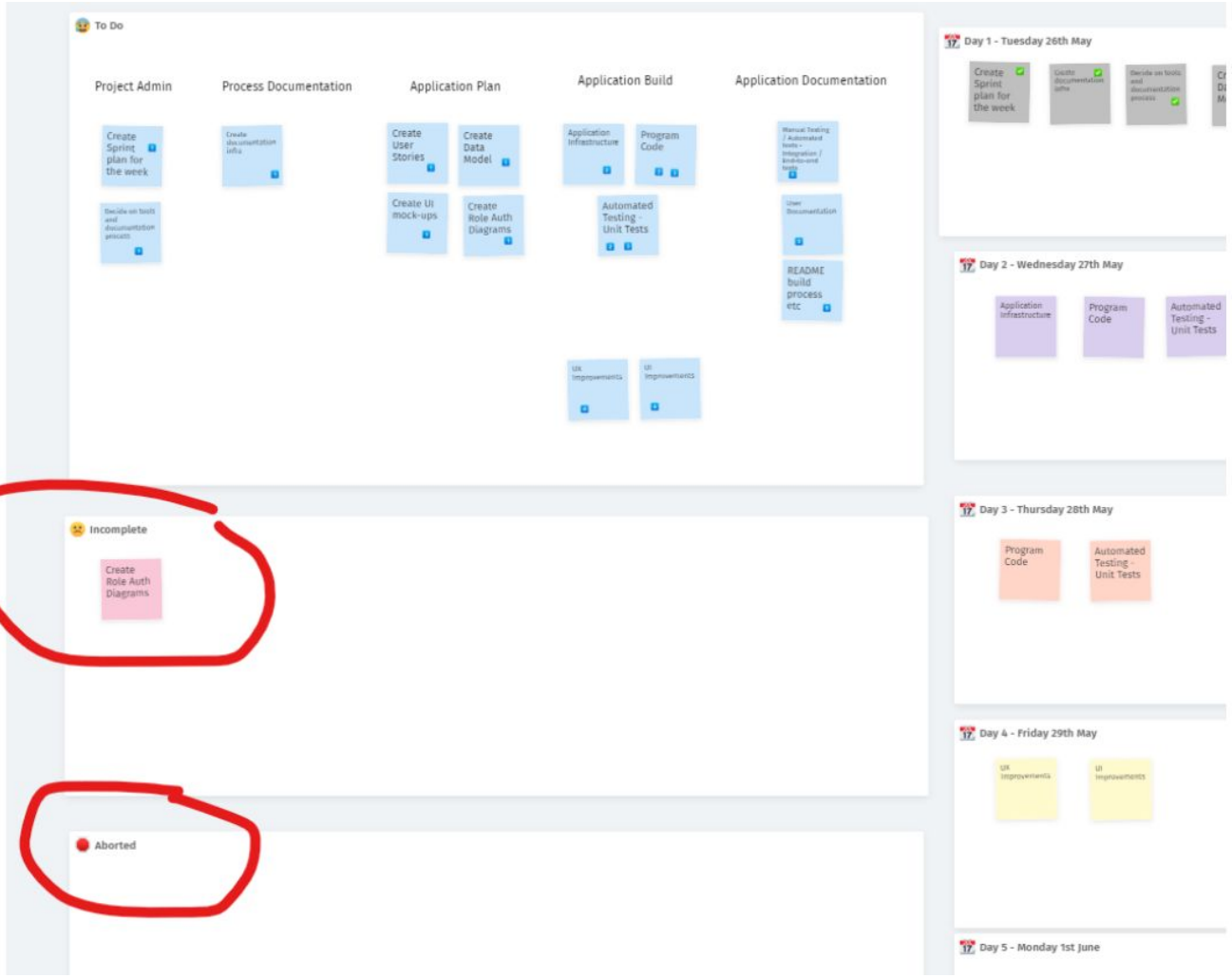
Likewise, I didn't have time to do the final task I'd planned for on Day 1 - creating a User Auth diagram to show the paths through the application. Again, I think given the short time available, I'll be better pressing on. My wireframe mockups indicate where user auth kicks in, and the brief is explicit, so I'll need to refer to those and just be very careful when adding that functionality.

I also didn't have time to complete a daily retro entry yesterday, as I mis-timed my breaks. I filled it in this morning, but that's missing the point, I guess. Need to be more disciplined about checking my tasks and guesstimating timings for them... perhaps set myself specific blocks of time for a task, and move on regardless at the end of that time unless there's anything unfinished that would block future work.

- ***Decision*** *Before starting a task, I'll estimate how much time needs to be spent on it, given the other tasks lined up for the day. At the end of that time I'll either move on, or identify blockers and timebox their completion.*

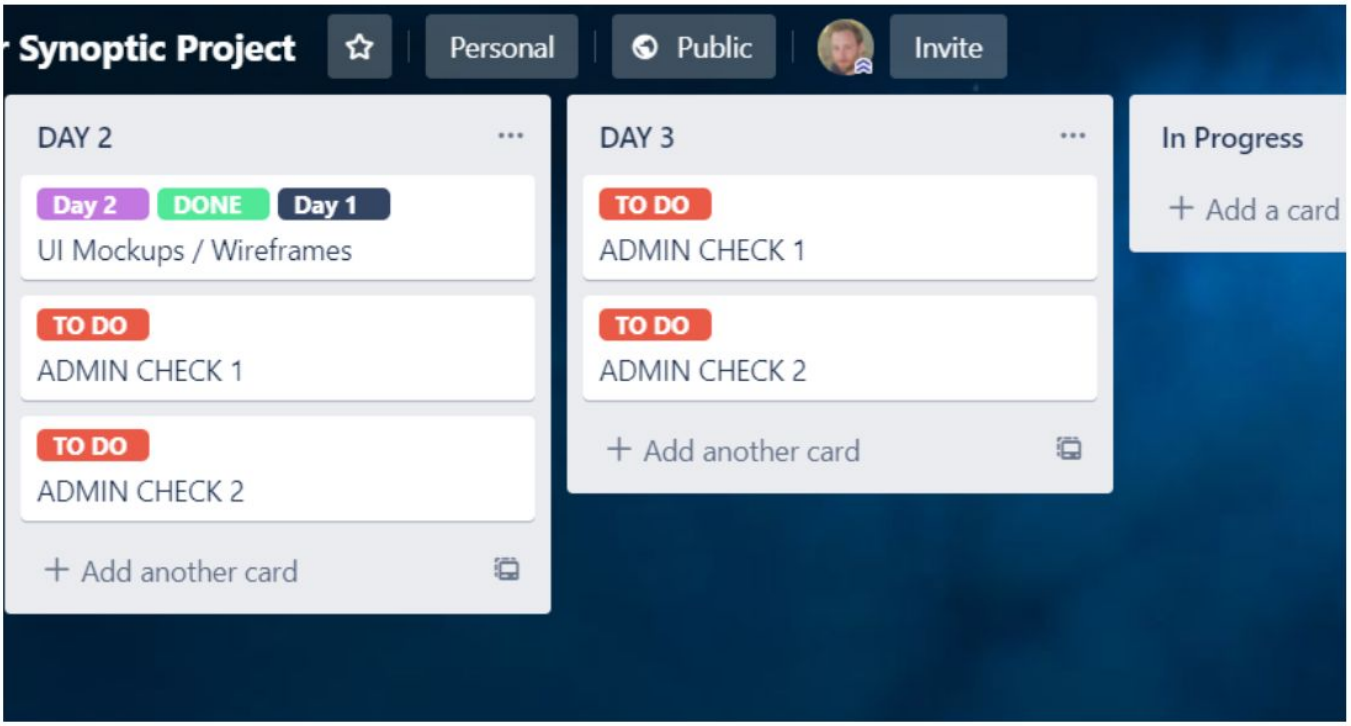
Doesn't feel great starting a day off by finishing off the previous day's work, and by leaving tasks incomplete :-(anyway, due to that I'll update my Sprint Plan board with two new areas - 'Incomplete' and 'Aborted';

https://metroretro.io/board/rJtlLLqiU



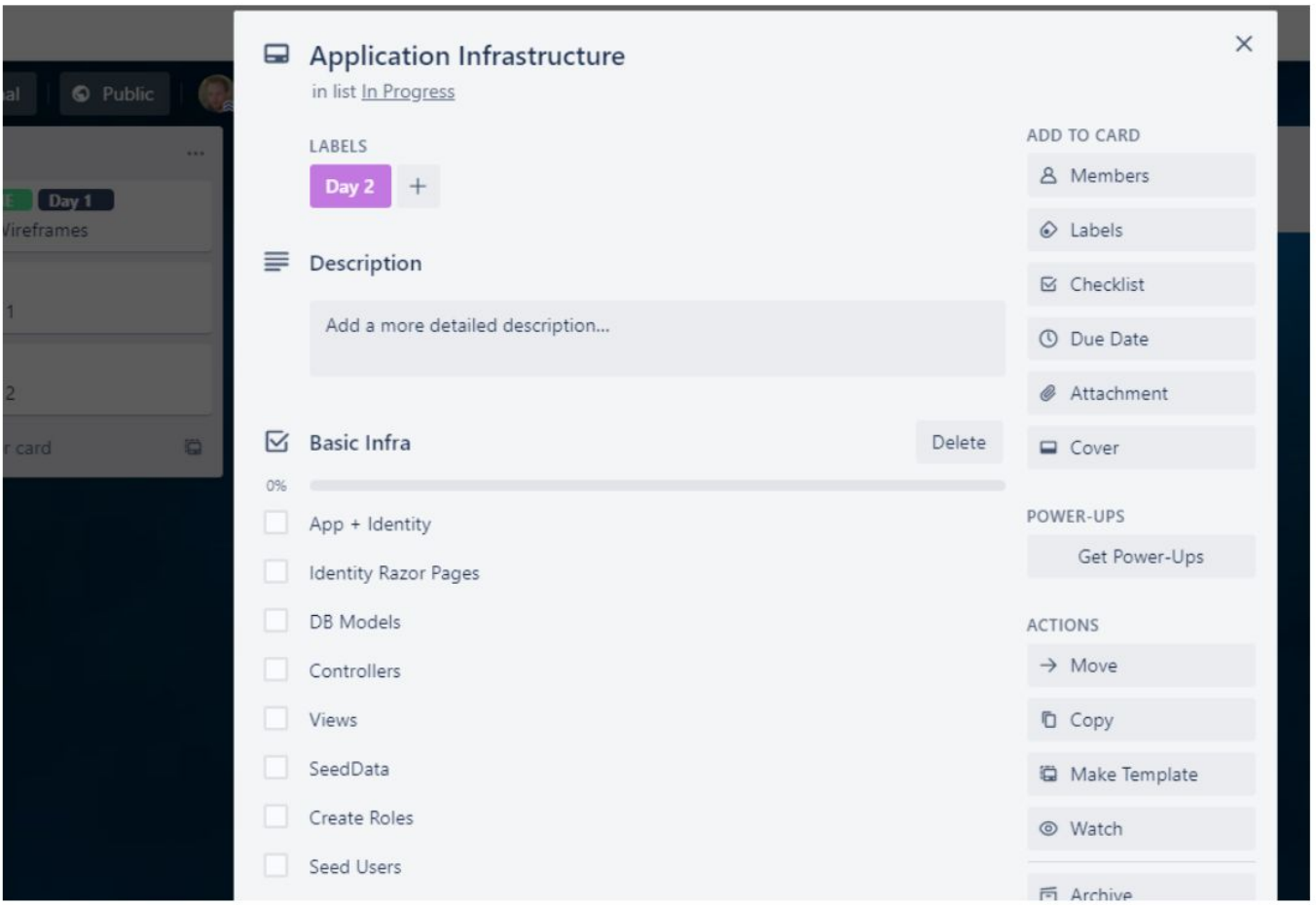
Hopefully I won't have to add too many cards to those spaces.

I've also created a list for each day of the project on Trello, and added a card per scheduled admin check-in, so that I always have a visual reminder of things I need to do before signing off for the day. These, along with my calendar reminders, should hopefully keep me out of bad habits for the remainder of the week.


Synoptic Project
Personal
Public
Invite
DAY 2
Day 2
DONE
Day 1
UI Mockups / Wireframes
TO DO
ADMIN CHECK 1
TO DO
ADMIN CHECK 2
Add another card
DAY 3
TO DO
ADMIN CHECK 1
TO DO
ADMIN CHECK 2
Add another card
In Progress
Add a card

Application Infrastructure

Approach as follows;



Create Application with Identity Framework

- Eugh. Awful start to the day. Spent two hours going back and forth trying to get the scaffolding of Controllers etc in the right order, along with Database migrations, etc etc, to get the first part of the infrastructure in place. I ended up doing some testing in a temporary repo and then completely resetting my Master Git branch, as it was a mess of reverting and nonsense, which wasn't useful for me or anyone else. I've now successfully got the following;
    - Scaffolded MVC App with Identity Framework for Individual User authentication
    - Quiz, Question and AnswerOption classes, view, controllers and associated DB tables
    - EF Migrations to build up Database as required.

This was completed by updating the database once the initial app was successfully scaffolded, to create all relevant Identity tables. After creating the Model classes, I scaffolded controllers + views for them which updated the DBcontext, allowing me to add a further migration. Applying this to the DB added more

tables for each model. Breaking for lunch now, but next on the list is to add the User table, and create some seeding data to work with. That should be it for the infrastructure setup for now.

- I will scaffold all of the Identity framework's Razor views into my project. This will allow me to edit them as required, and 'disable' any pages/actions that I don't want present in my project.

Creating Database Model classes

- When adding the DB model classes, I realised that my user class would need to inherit from the Identity User class. That class has its own `.email`, `.UserName` and `.Id` properties, so I don't need to add any to my class. The only fields not inherited from the Base class are `.FirstName` and `.LastName`, so I still implemented those on my model.

- Also realised that as the Identity user class is called User, I need to rename my model class. Convention is the name of the application followed by `User`, so going with `QuizManagerUser`.

Controllers and Views

- I've stubbed out Controllers and CRUD views for each of my 4 models, although may not end up using all of them. I thought it would be easiest to use the MVC framework to facilitate the bare bones of the app where possible, in order to save time. If anything proves particularly difficult and blocks me for any length of time, I can at least fall back on the default.

Seed Data

- Okay, I've now got a nice setup for this. There's a class with settable constants to seed the database with as many Quiz, Question per Quiz, and Answer per Question values as necessary, correctly setting the appropriate number of correct answers etc. BUT... I've spent far too long doing it, when it would have been much simpler to chuck some arbitrary data in there. I was aiming for best practice, but think I should definitely compromise on this approach going forward if I'm to finish the project. It's near the end of day 2 and I've not yet coded and application logic at all. Need to get a move on!

With that said, I felt like all of the DB calls I was making in the seeding class should have been carried out Asynchronously, to ensure items were created before creating others that depended on them. I was running into problems as the documentation I was reading online suggested adding this type of behavior to the `Program.Main()` method, which is `static void`, and I kept getting errors trying to get it to work if calling an Async task. I ended up doing all the seeding in static methods, which has seemed to work fine, and I've spent too long on it already to try to unpick whether I should have gone another route. However, I now need to seed users and roles to the database, so will try to find a way to do that asynchronously.

Create Roles

- Hmmm. I added what I thought was some straightforward config to the `startup.cs` class, to enable IdentityRoles and then call an Async Task to seed them when configuring the app. Getting all sorts of errors, which may be to do with having a mix of `IdentityUser` and `QuizManagerUser` in use. Not 100% sure the additional user class was necessary, and I may have just misunderstood how Identity is intended to work. I can certainly call the Identity `User` from code. But hey, let's see.

- Okay, sorted. I'd registered the Identity service using my `QuizManagerUser` class, but the `_LoginPartial.cshtml` view was trying to inject a SignIn and User manager that used the default `IdentityUser` class. Will probably run into this kind of thing a lot :-/

Seed Users

- Sigh. So the method works fine; my database now has 3 new users, each with the correct roles. But they're not logging in, and I'm getting default validation errors alerting me to an invalid login attempt. This is all internal to the Identity Framework. I suspect another mismatch between `IdentityUser` and `QuizManagerUser` somewhere, so will debug and see if I can spot what's going on.

- Oh, lol. The default customer creation in Identity takes prompts for an email, and then set it as the email *and* as the username. In my seeding users class, I'd given users different email and usernames, and the login form was prompting for (and validating) email addresses, even though behind the scenes it was actually looking for Username. Ha. I prefer using a distinct username for testing purposes ("Edit", "Restricted" etc), so will see if it's easy to change it to that for now. Will need to rewire the view validation and code. Long term it'd be nice to login with *either* username *or* email.

- Really showering myself with glory here. So I first needed to scaffold all of the Identity Razor pages again. I'd last done this in the test repo I started from scratch when I was having trouble getting started, and when I was ready to come back to the main repo I forgot to do it again. Just re-adding them all - hopefully won't cause any issues.

- Okay, so I did have to rename loads of Razor views and code behind files to reference my new `QuizManagerUser` class, so a bit more time spent doing that. Still thinking I could have just foregone the class entirely, but hey. I've rejigged some of the login and registration logic, but there are still some kinks. I'll leave it for now as the things that aren't working correctly (e.g. First and Last name showing up when a logged in user views their profile) isn't technically necessary for this project, and I'm a bit behind where I'd like to be so may not include them at all. BUT, all infra is now done :-)

Program Code and Testing, Day 2

It took far longer than I'd hoped to get the bones of the app up and running. The good news is that the MVC framework provides a lot of 'out of the box' functionality, so worst case scenario, I could probably have a version of the application that satisfies the working requirements fairly quickly. However, the User Experience and User Interface (UX and UI) would be based on bare bones templates, not as per the wireframes I'd like to work towards. I'll definitely need to pick up the pace tomorrow.

With just over an hour left, the final tasks I'd hope to get around to today are working on the program code and some testing. Technically there is program code already, as my views and controller do provide functionality. As such, I'm going to concentrate on getting a testing framework in place as the final task of the day. I should hopefully be able to re-use a lot of the `SeedDatabase` code to create dummy test data to play with.

- ***Decision*** *Again, due to familiarity from my placement job, I'll use NUnit and Fluent Assertions for testing my code. Fluent Assertions will allow me to write code in more easily legible syntax, while NUnit is a widely used testing framework.*

I'll split this work into 'tickets', either using Trello or Metro Retro to keep track of tasks. I'm leaning towards Trello for the ease of use, leaving Metro Retro to provide a project overview.

- ***Decision*** *The 'Program Code and Testing' section of the project will comprise the bulk of it, fingers crossed. As such, I will try to use Trello to break User Stories and Project Tasks down into their component parts, or 'Tickets', and keep track of them in separate Trello Lists. This will allow me to clearly scope out what is involved in each piece of functionality, feature or testing, and work through each identified task sequentially.*

Add Unit Test for existing Quiz Index method

The Microsoft / Entity Framework documentation suggests three approaches for testing applications that make use of a database.

- Running queries and updates against the same database system used in production.
- Running queries and updates against some other easier to manage database system.
- Using test doubles or some other mechanism to avoid using a database at all.

The encouraged pattern in my placement job is to define clear unit tests, in favour of complicated integration tests, which can be expensive to maintain and run, relative to the benefit they provide. Clearly defined unit tests with high code coverage provide a better balance, with high value and ease of maintenance. Unit tests rely on mocking out any dependencies the code under test has, which in this case would include the database and associated responses.

However, for this project, the program logic should be relatively minimal, whereas the accuracy of database queries is very important. In addition, the documentation states the following;

'We use test doubles for internal testing of EF Core. However, we never try to mock DbContext or IQueryable. Doing so is difficult, cumbersome, and fragile. Don't do it.'

As such, I decided to choose the second recommended testing option, *Running queries and updates against some other easier to manage database system*, which seemed the most appropriate of the alternatives, allowing me to leave any 'production' database (development database, in this case) untouched, so that I could arrange the test database as required, and have it running frequently without getting in the way of my development work.

- ***Decision*** *Led by the documentation, I decided to use an in-memory SQLite database. As another relational database type, it would closely mirror the functionality of SQL Server, the DB type I was using for the main application. Having it run in memory means an instance of the database can be spun up, configured and then disposed of by every test that needs it.*

The test itself was relatively straightforward to write, but running it threw up an error about there being more than one entry point for the `Main()` method of the program. Some Googling led to this article, which pointed out that unit tests are console applications, and importing a testing framework then running tests will generate another `Program` file with associated `Main()` method, leading to the overall project containing 2 instead of one. This was fixed with the following;

‘Add <GenerateProgramFile>false</GenerateProgramFile> inside a <PropertyGroup> element in your test project's .csproj file’

Daily Retro, Day 2

Day 2

Brain Dump	Proud of
I have a really bad habit of getting obsessed wit getting something ‘right’, which often means ‘jus working at all’ in this context. It’s really slowed me down today, as I should have had the infrastructure completed 2-3 hours sooner. I can’t afford to do this for the remainder of the project. Things like the overly engineered seedin class, while nice, could have been achieved in a much simpler way. While it’s good to think abou future development, I can’t lose sight of the fac this is a 40 hour project.	• Getting testing set up quickly at the end o the day • Despite it taking too long, the seed database class has some good practice in there and it’s proving useful as a template for the testing code, too. • I have a working app! It potentially has lot of ‘bonus’ functionality, with almost no extra effort, due to the frameworks chosen
Could do better	**Next on the plan**
• I decided to timebox tasks to avoid fallin behind, and then completely forgot to do it :-(• I feel like I didn’t take the time to fully understand what Identity offers, and may have wasted time unnecessarily as a result, with the inclusion of a separate User class.	• I think maybe an hour first thing breaking user stories down into tasks, and then I really need to just speed through them. • Very keen to have all requirements covered by the end of the day, even if not in the form I expected. • This will mean I’m on track and in line wit my sprint plan.

Day 3 Thursday 28th May

In hindsight, I think yesterday was pretty productive and I'm in a good place, with 3 days to go. There's a good foundation to build on, now that the data and app structure is in place. Much of the app functionality will be driven by combining info from the 3 different classes that comprise a full quiz, and EF Core uses the `DBSet<T>` for its database tables. This class implements `IEnumerable`, so I can use Linq etc to query exactly the info I need, whenever I need it.

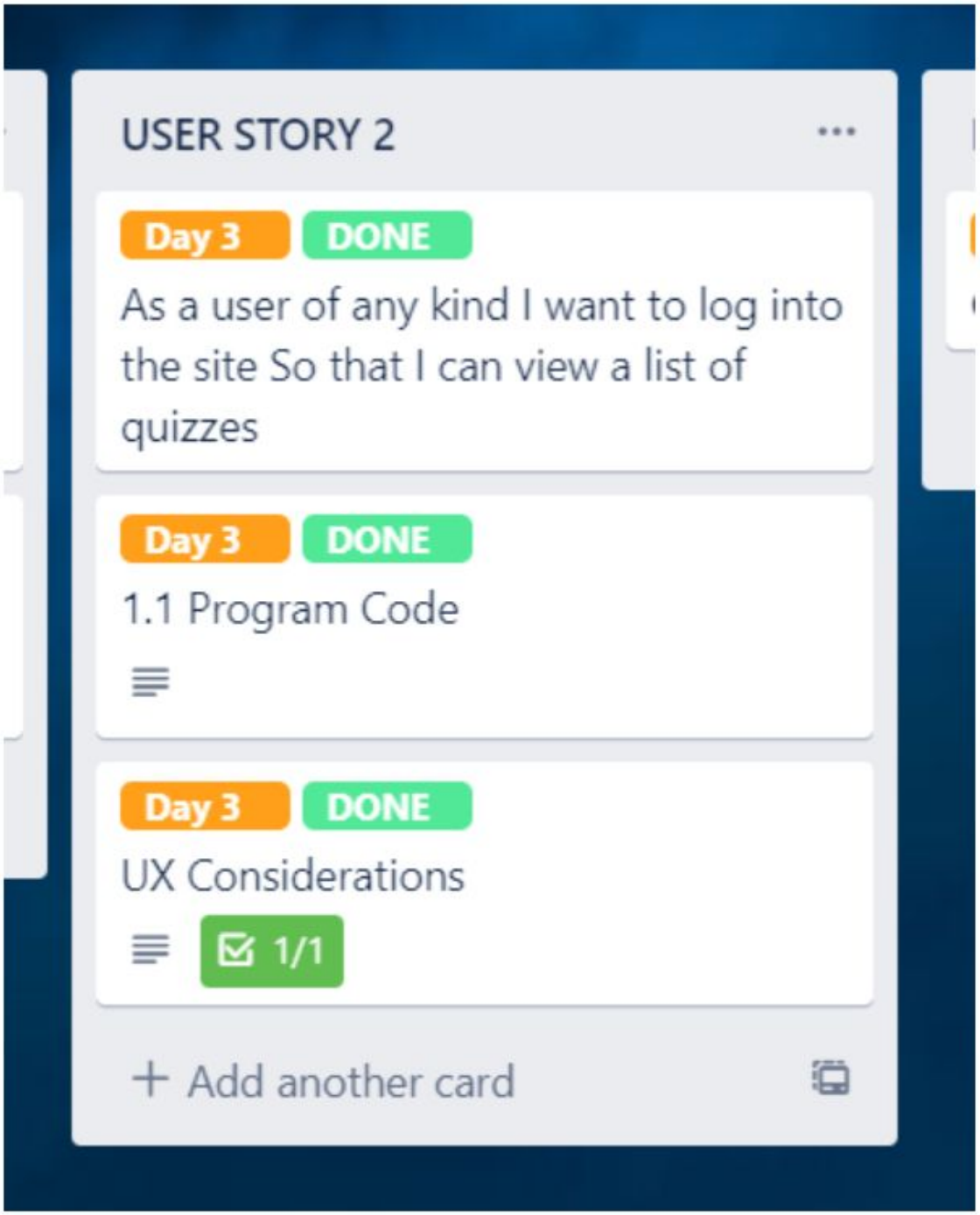
The most important thing for today is to be really clear about what I need to implement. I'll work on one User Story at a time, and break it down into discrete tasks. Each story will be a separate Trello List, with each task represented by a separate card. That should keep me on track and encourage me to get into a cycle with a really quick feedback loop. Plan, implement, test, repeat. If I've got anything wrong, or misunderstood some requirements, I should quickly realise and be able to sort it out before building on top of the mistake.

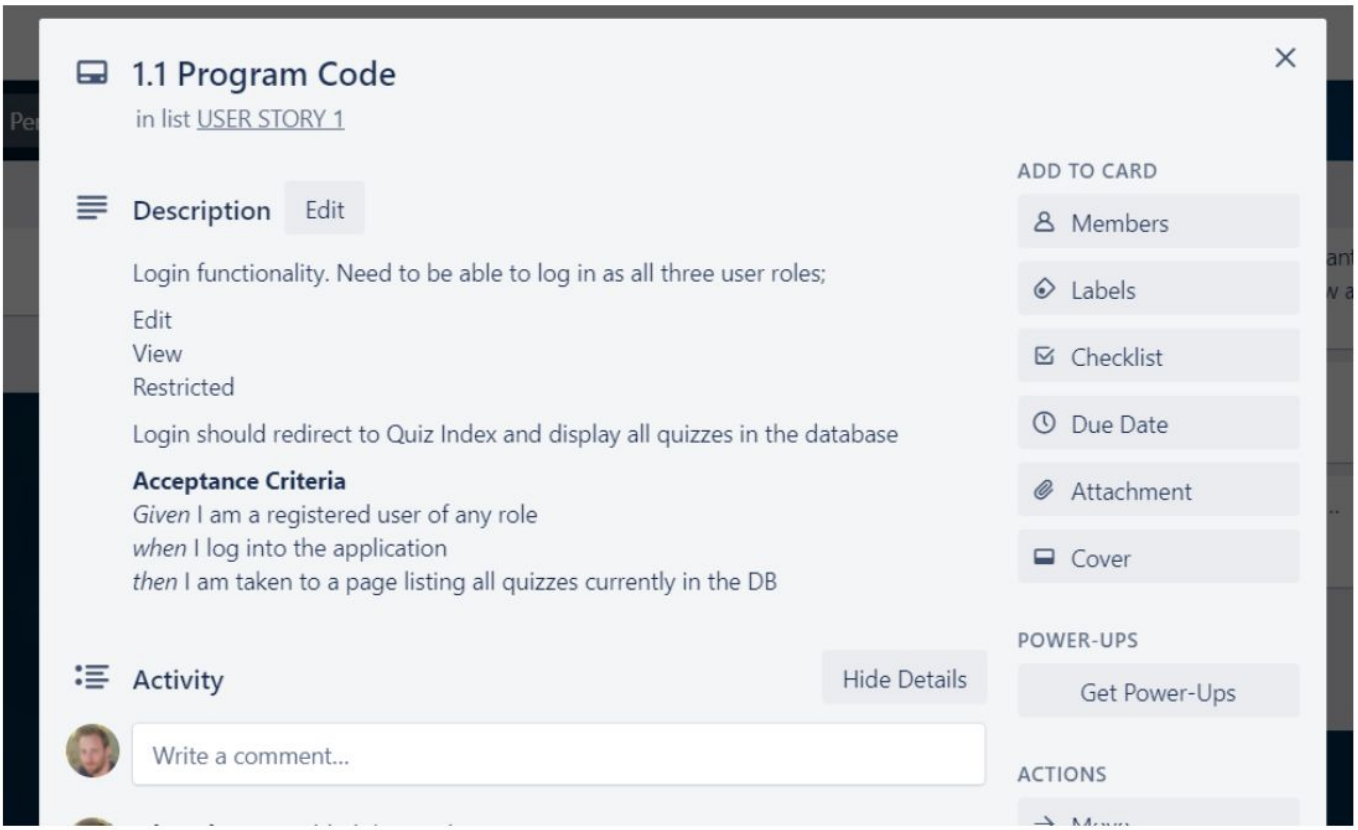
- ***Decision*** *I'll use Trello to track the coding and testing work. There'll be one list per previously defined user story, with the option to add more as required. Each list will contain one card per task, and each task will have acceptance criteria I'll define during planning.*

User Story 1 Login to view lists

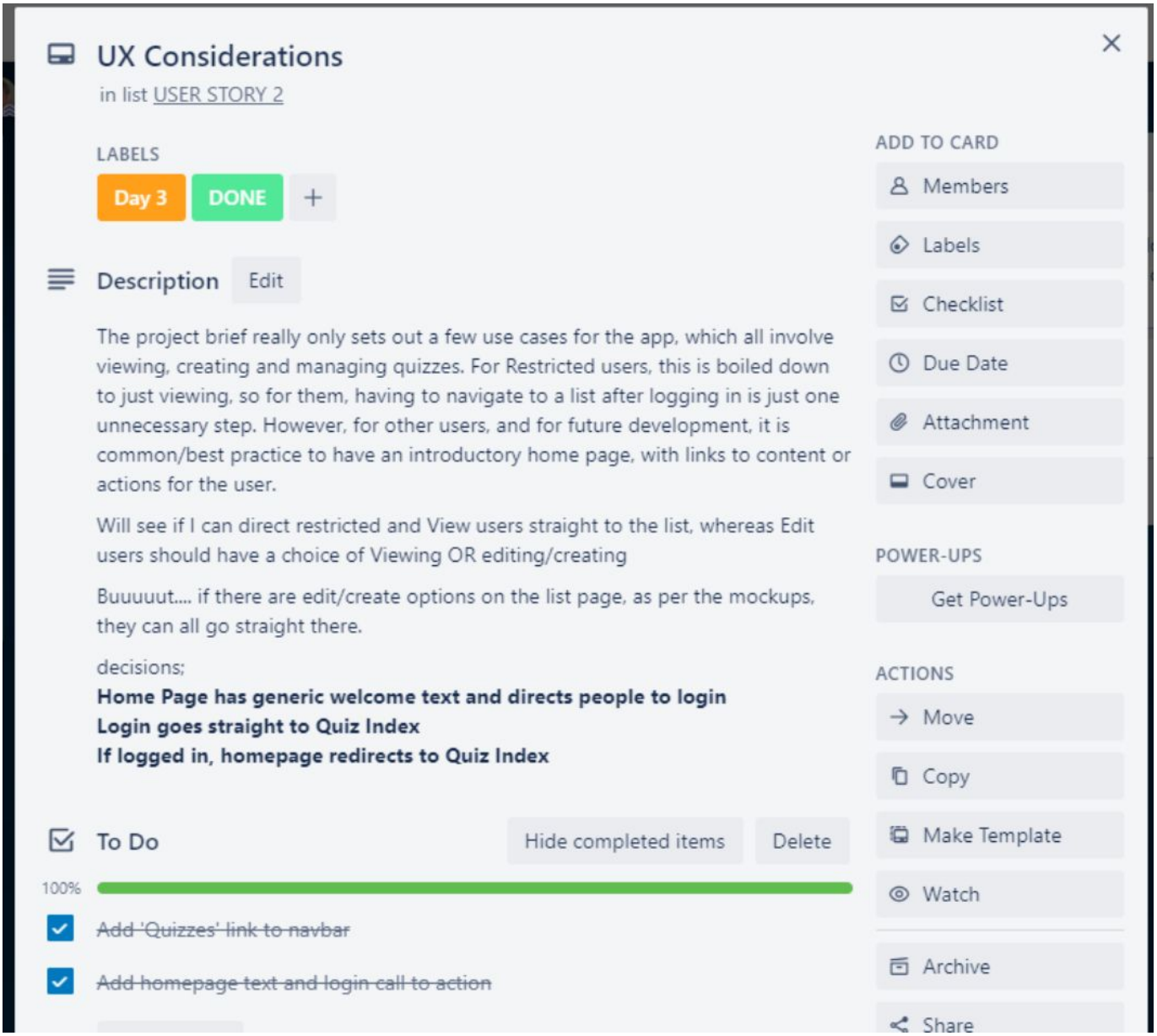
- The Identity framework is already taking care of login. I've been able to log in with each of the three users I seeded yesterday.

- The only change to the default behaviour is that it should redirect to the Quizzes index, which should display every quiz. This only involves changing a default URL variable in the code behind the Login page, so I don't think it's necessary to test. Unit tests should cover my own code, not that of a third party framework.

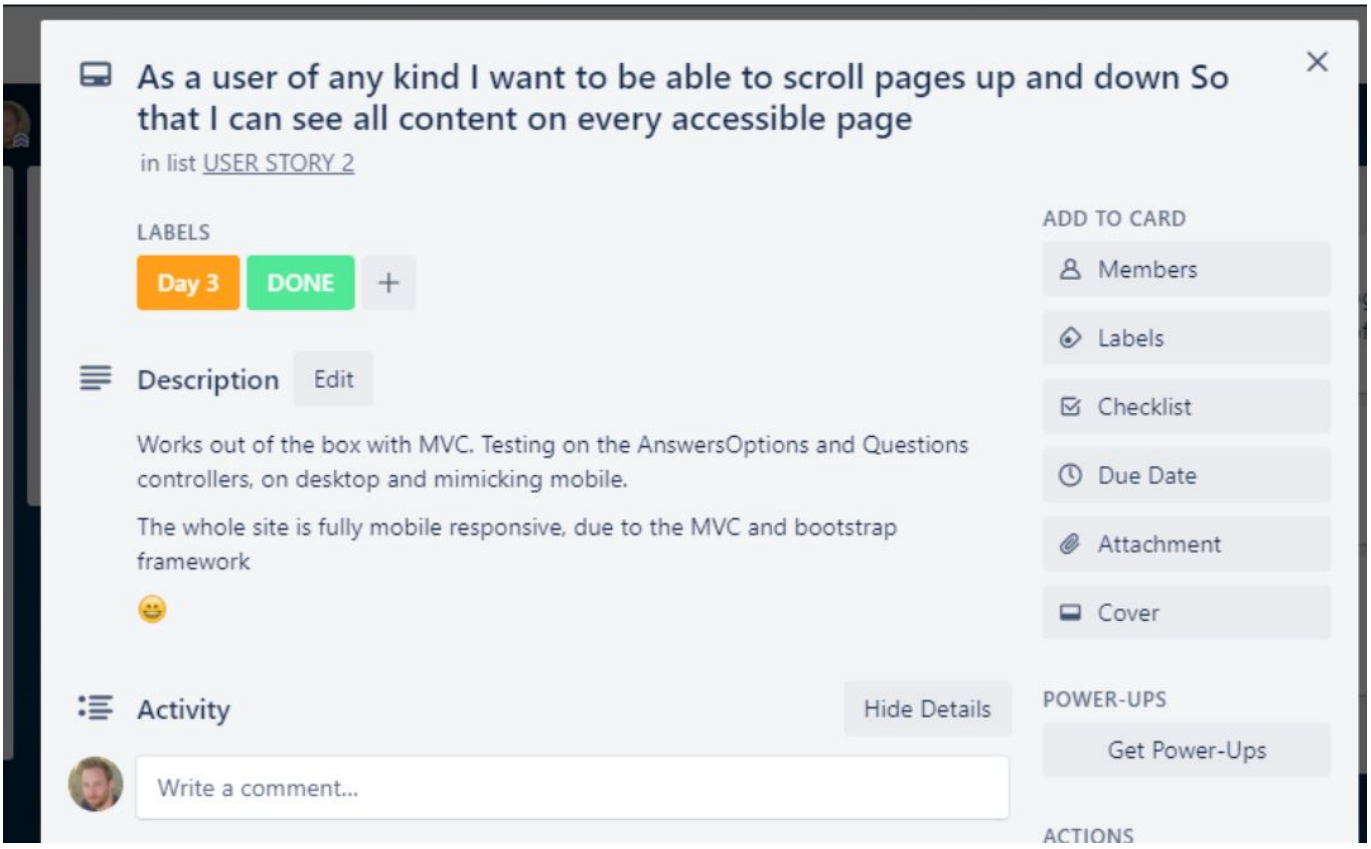




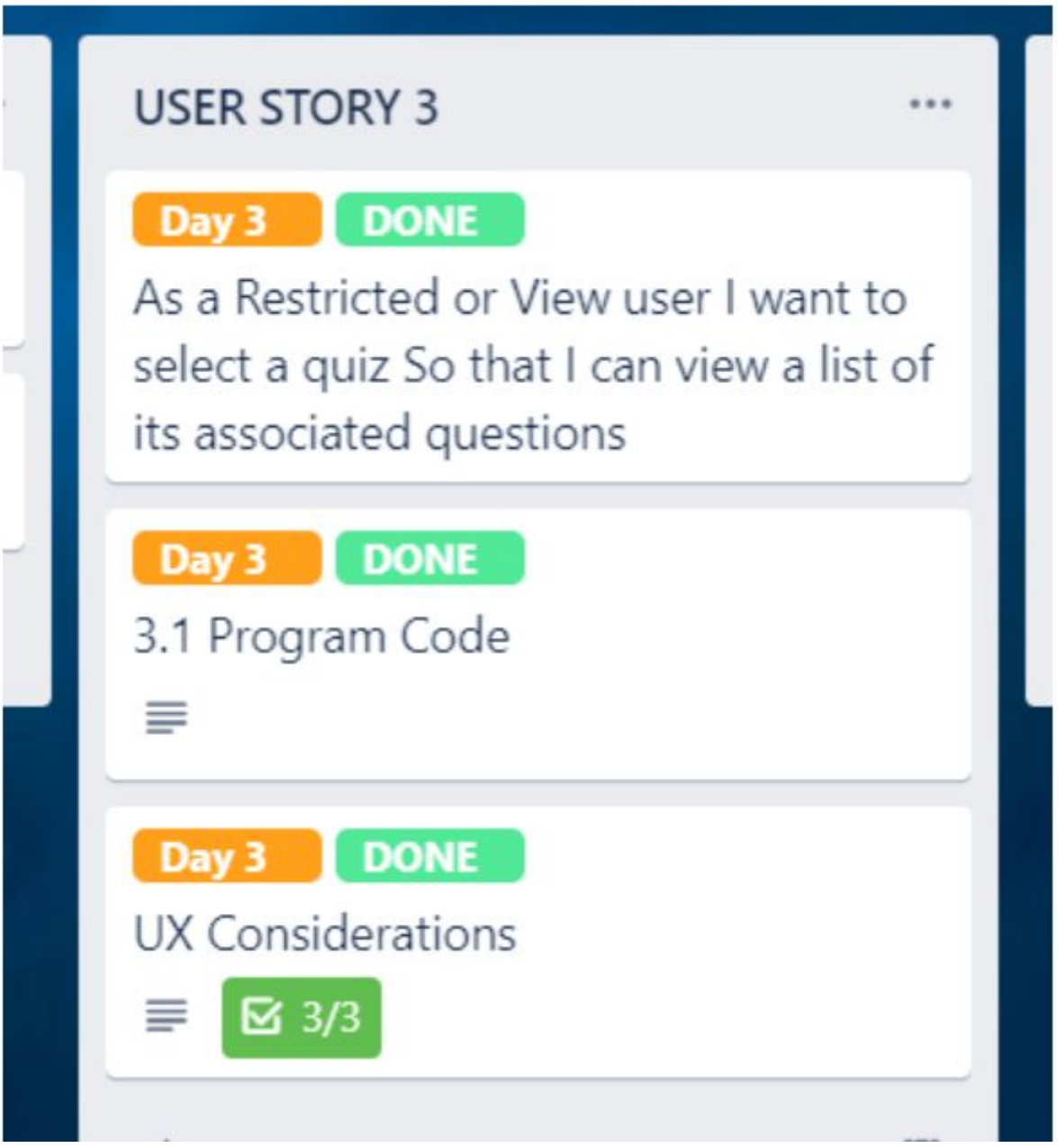
- While here, I started fleshing out the UX a little to bring the app away from the MVC default structure. I've used absolutely standard Bootstrap here, copied from their documentation examples. Can tweak as desired on UX day tomorrow if all the code is done, otherwise it'll do the job as a professional-ish looking placeholder.

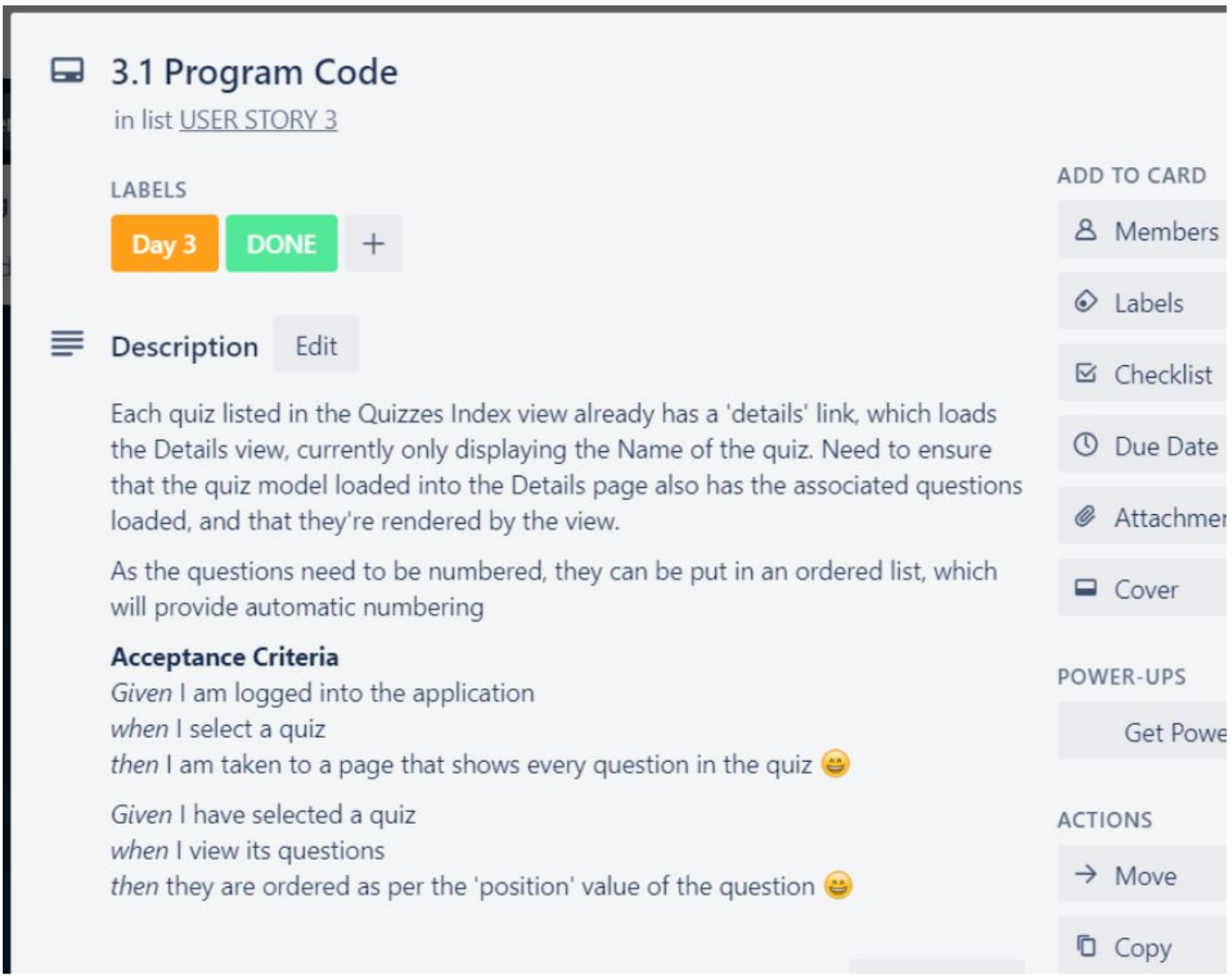

UX Considerations
in list USER STORY 2
LABELS
Day 3
DONE
Description
Edit
The project brief really only sets out a few use cases for the app, which all involve viewing, creating and managing quizzes. For Restricted users, this is boiled down to just viewing, so for them, having to navigate to a list after logging in is just one unnecessary step. However, for other users, and for future development, it is common/best practice to have an introductory home page, with links to content or actions for the user.
Will see if I can direct restricted and View users straight to the list, whereas Edit users should have a choice of Viewing OR editing/creating
Buuuuut.... if there are edit/create options on the list page, as per the mockups, they can all go straight there.
decisions;
Home Page has generic welcome text and directs people to login
Login goes straight to Quiz Index
If logged in, homepage redirects to Quiz Index
To Do
Hide completed items
Delete
100%
Add 'Quizzes' link to navbar
Add homepage text and login call to action
ADD TO CARD
Members
Labels
Checklist
Due Date
Attachment
Cover
POWER-UPS
Get Power-Ups
ACTIONS
Move
Copy
Make Template
Watch
Archive
Share

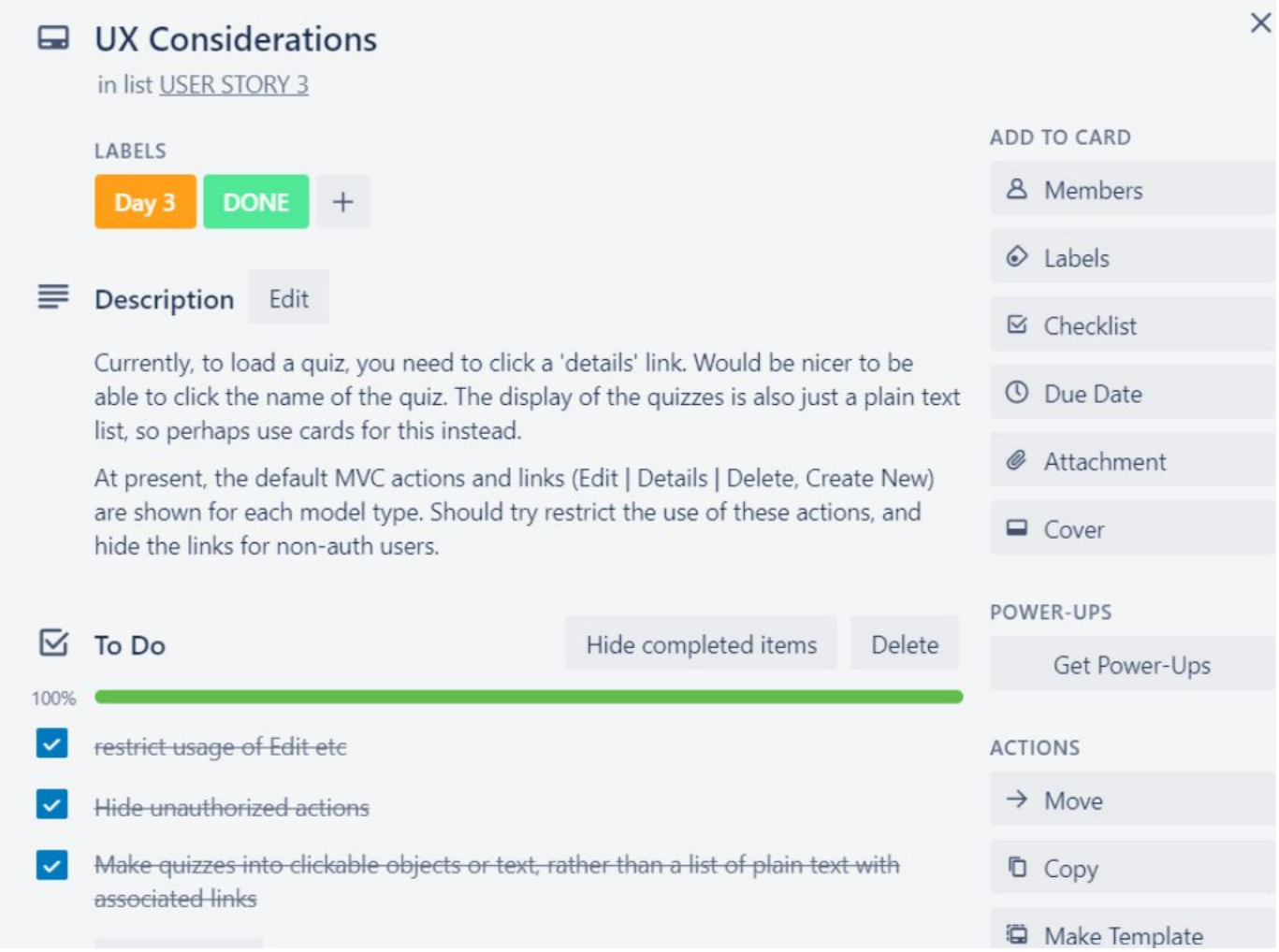
User Story 2 Scrolling Enabled



User Story 3 View Questions




3.1 Program Code
in list USER STORY 3
LABELS
Day 3
DONE
Description
Edit
Each quiz listed in the Quizzes Index view already has a 'details' link, which loads the Details view, currently only displaying the Name of the quiz. Need to ensure that the quiz model loaded into the Details page also has the associated questions loaded, and that they're rendered by the view.
As the questions need to be numbered, they can be put in an ordered list, which will provide automatic numbering
Acceptance Criteria
Given I am logged into the application
when I select a quiz
then I am taken to a page that shows every question in the quiz
Given I have selected a quiz
when I view its questions
then they are ordered as per the 'position' value of the question
ADD TO CARD
Members
Labels
Checklist
Due Date
Attachmer
Cover
POWER-UPS
Get Powe
ACTIONS
Move
Copy

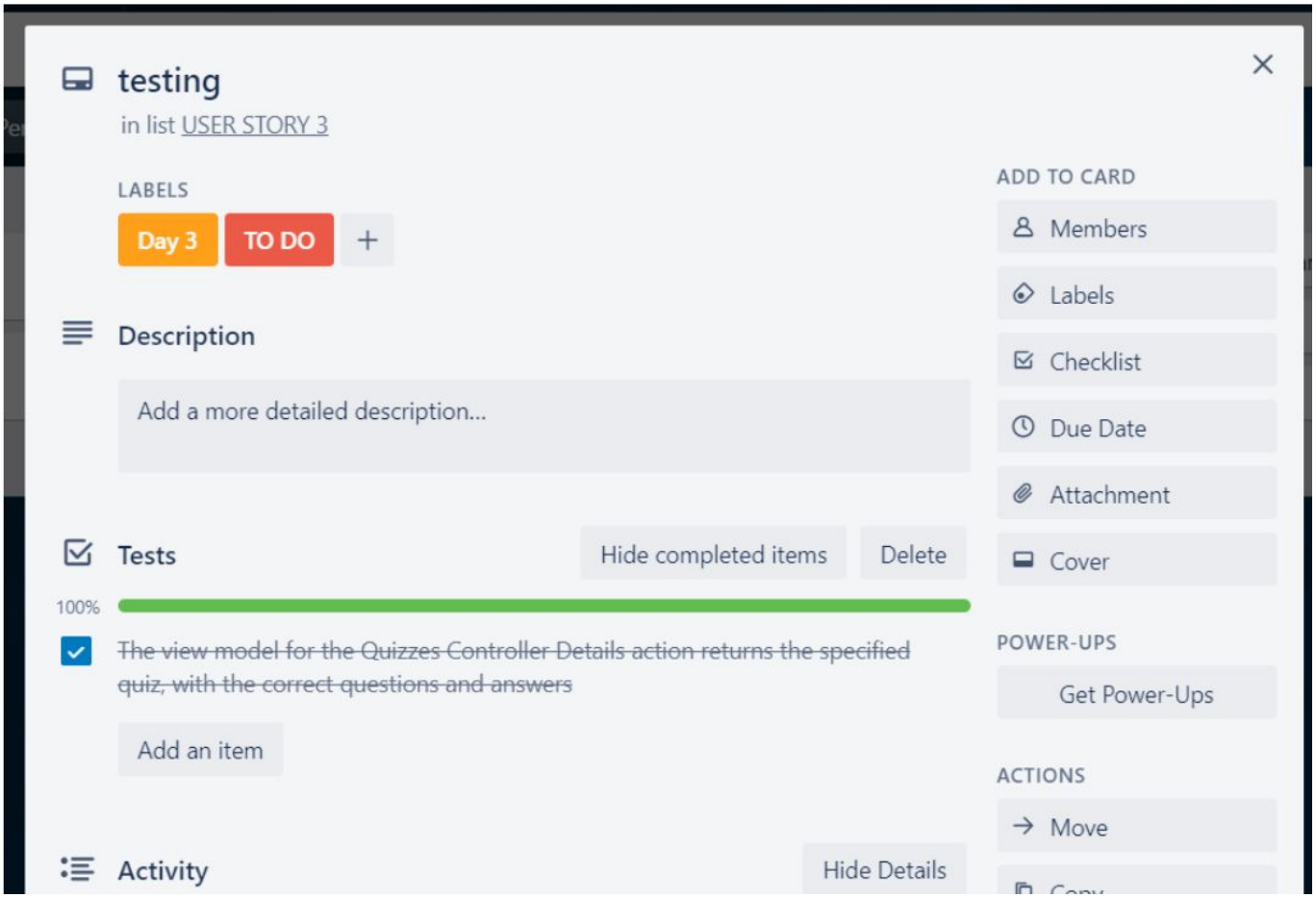


Having spent a couple of hours getting these tests sorted, I think it may have been a waste of time unfortunately. There is no specific logic in the code I've written which doesn't just leverage the Entity Framework supported Linq queries. So while these tests are working and provide coverage and documentation of the controller's intended function, I'm really just testing the EF does what it's supposed to do.

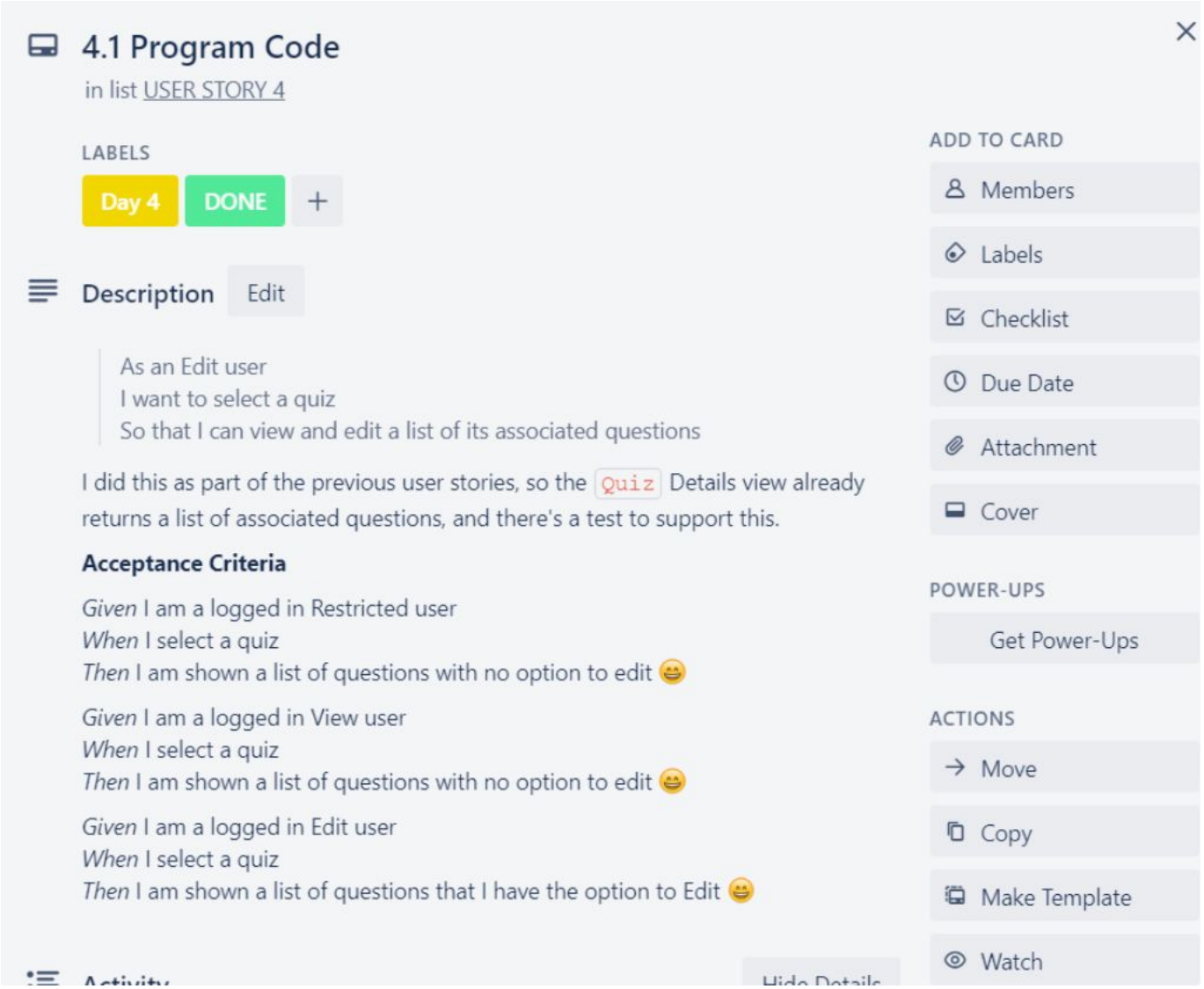
It's the same situation as the role restriction stuff - really, all I'd be doing is checking that Identity Framework is behaving as it's supposed to.

As such, due to time constraints, I'll stop writing tests unless I'm doing anything more complicated in any of my MVC action methods. My work tickets will all be written with Acceptance Criteria which will act as QA / User testing, so I'll potentially tabulate these results

- ***Decision*** *The bulk of my testing will be manual, and comprise of satisfying Acceptance Criteria on tickets, with some end to end and exploratory testing thrown in. Automatic / Code tests will only be written when the code complexity demands it.*


testing
in list USER STORY 3
LABELS
Day 3
TO DO
Description
Add a more detailed description...
Tests
Hide completed items
Delete
100%
The view model for the Quizzes Controller Details action returns the specified quiz, with the correct questions and answers
Add an item
Activity
Hide Details
ADD TO CARD
Members
Labels
Checklist
Due Date
Attachment
Cover
POWER-UPS
Get Power-Ups
ACTIONS
Move

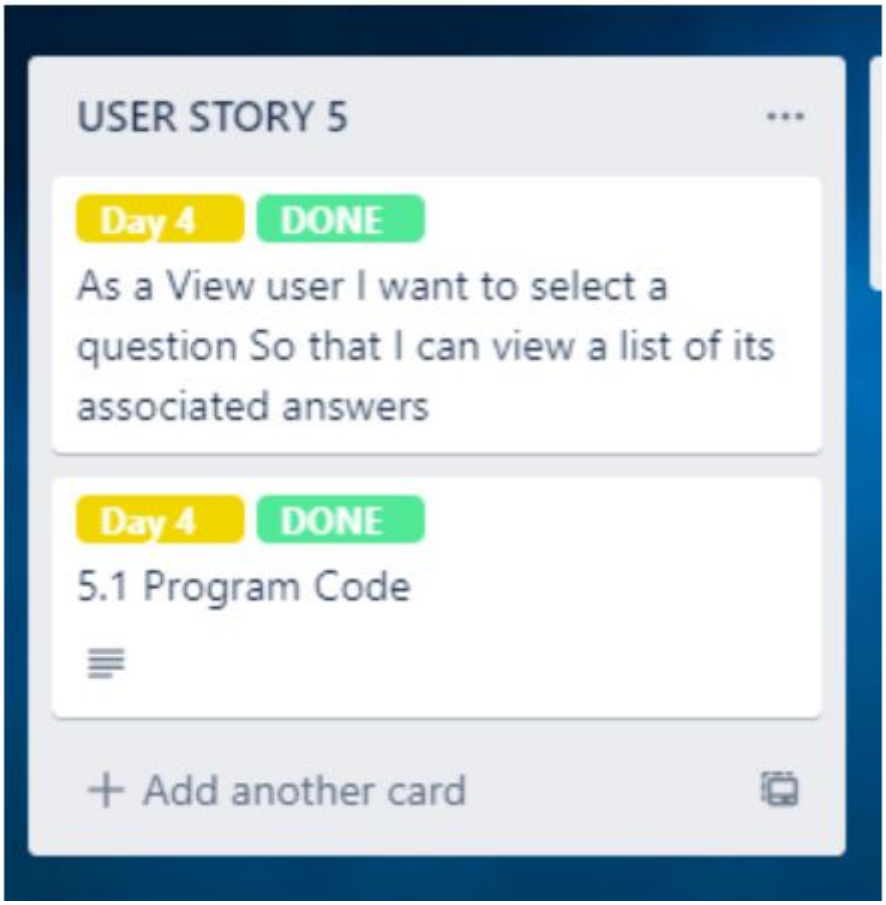
User Story 4 Edit User can edit a quiz's questions

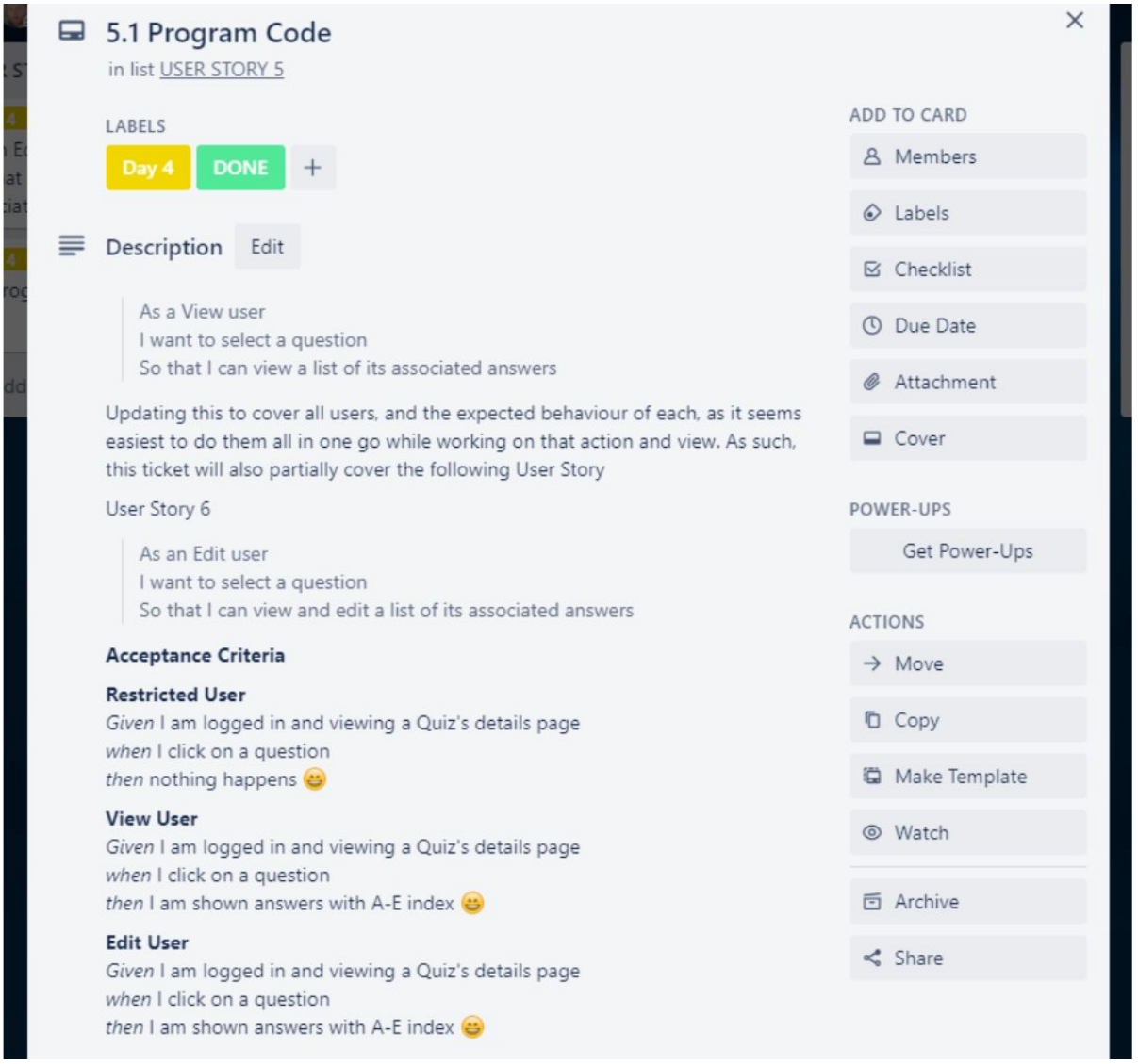


User Story 5 Select Question to View Answers

In approaching this story, and all stories which include restricting actions based on user role, I've been making sure to not only update the UI to remove any links, but also update the code to actively restrict actions based on role. This should stop people 'accidentally' doing unauthorised things by visiting a specific link/action manually, based on pattern matching or their knowledge of MVC conventions.

All UI changes are handled using server side code, rather than by hiding or disabling buttons, to ensure the final HTML served to the user doesn't have any 'clues' of functionality that they're not supposed to see.




5.1 Program Code
in list USER STORY 5
LABELS
Day 4
DONE
Description
Edit
As a View user
I want to select a question
So that I can view a list of its associated answers
Updating this to cover all users, and the expected behaviour of each, as it seems easiest to do them all in one go while working on that action and view. As such, this ticket will also partially cover the following User Story
User Story 6
As an Edit user
I want to select a question
So that I can view and edit a list of its associated answers
Acceptance Criteria
Restricted User
Given I am logged in and viewing a Quiz's details page
when I click on a question
then nothing happens
View User
Given I am logged in and viewing a Quiz's details page
when I click on a question
then I am shown answers with A-E index
Edit User
Given I am logged in and viewing a Quiz's details page
when I click on a question
then I am shown answers with A-E index
ADD TO CARD
Members
Labels
Checklist
Due Date
Attachment
Cover
POWER-UPS
Get Power-Ups
ACTIONS
Move
Copy
Make Template
Watch
Archive
Share

Daily Retro, Day 3

Day 3

Brain Dump	Proud of
Pretty good day. I didn't do what I 'should' have done, i.e. finish the program code, but feel it's okay becaus a lot of the remaining stuff is functionall there, but not with the UX I'd like, so kind of falls into tomorrow's plan anyway. Stuff like deletion of questions is there, but via a controller... I just nee to plug it in somewhere else. Also, I've been doing alot of UX and UI stuff alon the way today, so there's already overlap with tomorrow's tasks.	● The app is starting to take shape. It was worth it getting the frameworks in place as they've saved lots of time with functionality ● Accordion answers for each questio - subjective, but I like the UX of this. not sure whether I'll be able to keep it but hey ● Quicker working today - the structur of the work, feedback loop etc, was really helpful
Could do better	**Next on the plan**
● Was bad at taking breaks today, so occasionally I found myself getting frustrated with issues I was determined to crack. Would probably have done better with small, regular breaks to decompress.	● Just carry on as I have been today... think I'm on track

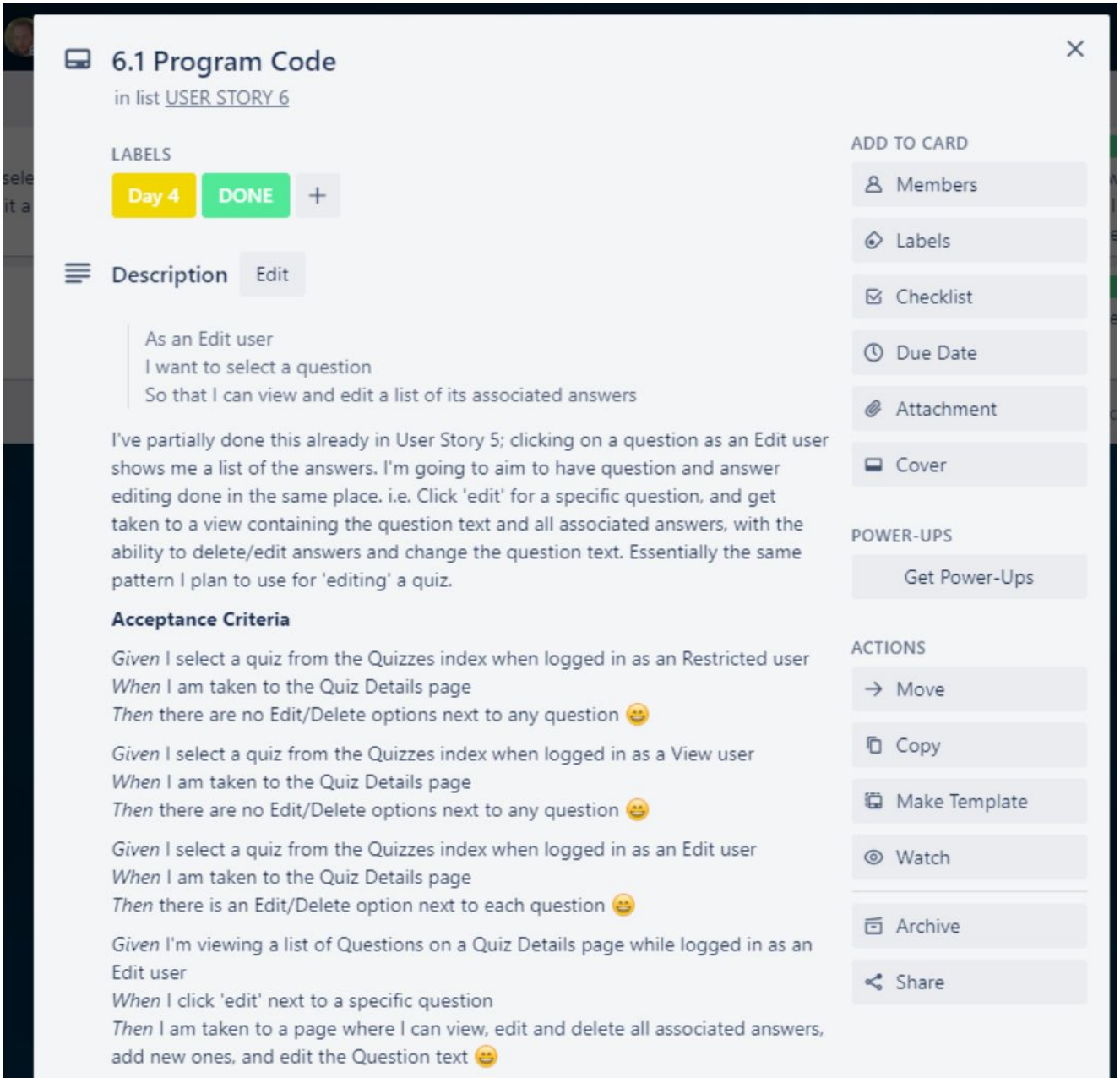
Day 4 Friday 29th May

Ready to get started again. At the end of yesterday I'd begun looking at drag and drop functionality, to enable the re-ordering of a quiz's questions. I think this is something that I could potentially spend a lot of time fiddling with, and isn't definitely required. I found the project brief a little unclear when it came to how Edit users should be able to insert extra questions.

I'll focus instead on getting all of the nuts and bolts functionality up and running first, then spend the rest of the day on UX/UI as originally planned, including a method to reorder questions, or insert new ones at a specified point.

Another task is to ensure I can migrate this wiki, currently a collection of MarkDown pages, into one document that can be downloaded along with the repo. This is due to a misunderstanding on my part, whereby I thought the assessor would have direct access to this Github repo and nothing else. They'll only have access to a download of it though.

User Story 6 View and Edit Answers



- So I've created an Questions / Edit view which shows me all current answers, along with their True/False status, and has the question and answer texts all appear as text fields you can edit. I'm trying to use a Radio button to reflect the true/false status, so that at any time, only one of the answers can be marked as true.

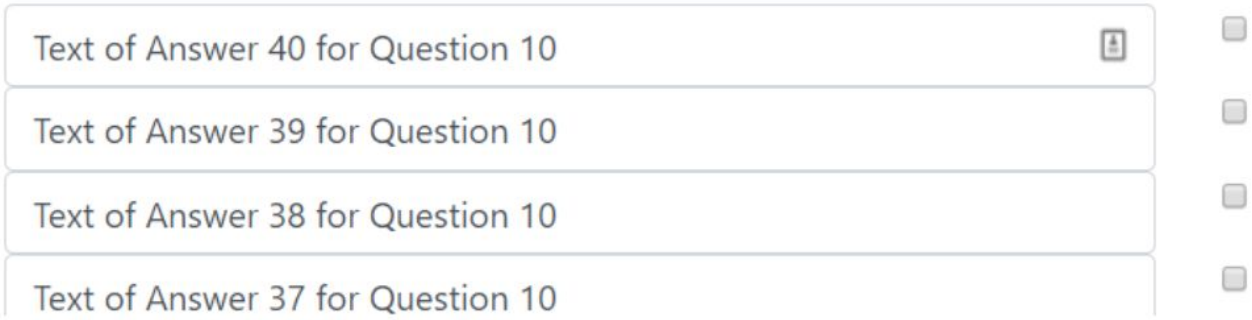
- My first attempt at changing the 'true' answer has instead created exact duplicates of each answer and added them to the question, doubling the number of answers.

As well as being wrong, this highlights that I'll need to add validation somewhere along the line so that Questions can only have 3-5 answers, which isn't currently the case.

- Okay, so debugging that Edit method showed that while I was correctly binding the answer text to the correct `AnswerOption` when sending the data back via the form, the `QuestionId` and `AnswerOptionId` were both null/0. I'd neglected to include those properties as hidden inputs in the section of the form returning the `AnswerOption` details, so the DB/EF was treating them as brand new answers and saving them against this question, alongside the 'old' ones.

- Just spent an AGE trying to get the True / False thing working, so that when viewing a question's answers, only one of them could be selected at a time *and* you could change which answer was marked as true. The natural choice for this is a radio button, but I was running into loads of issues; radio buttons can't represent booleans, so it wouldn't bind to the `Correct` property of the `AnswerOption`. Forms only return key value pairs, and there was no non-hacky way to get the ID of the selected radio button back, in order to do more code shenanigans on it to re-set the 'correct' answer. Dunno if that makes sense, I'm a bit frazzled. Anyway, I decided to use checkboxes instead, which can bind to the Model property of `Question.AnswerOptions[x].Correct` with no issues. BUT, they are multi-select by default, so I needed to use JS to only let one be ticked at a time. Off to stack overflow, and after many, many failed attempts, I realised that despite creating a working script and HTML combination probably loads of times, I'd never included JQuery, because I assumed it was included in Bootstrap. Maybe it is. But not the version I needed. Including it finally did the trick. Sheesh.

- While troubleshooting, I was getting annoyed that on a form like this;



It was tabbing from the text field to the checkbox field, before moving to the next row and tabbing into the next text field. That was counter-intuitive for me, as I like to tab through form text fields quickly when inputting or amending data, and as a user, it's more likely that I'd be changing the answer text than whether the answer was correct or not.

When looking for solutions, I saw an article making the good point that often, forms *skip* radio buttons and checkboxes by default instead, and that this is bad from an accessibility point of view, where e.g. people who struggle to use a mouse may rely on Tabbing to access all available fields in a form. As such, it seems best to leave this behaviour untouched.

- Now trying to validate the Question model so that it can only be saved with 3-5 `AnswerOptions`. The Data annotation namespace provides a few options for this, but unfortunately the only one that can limit the length of a collection is for Arrays, which don't work as navigation properties for Entity Framework model objects. I think I'll need to find a way to validate with the code and have the message returned to the user, hopefully in the same way as other validation messages are.

- Sorted. You can add validation logic and then add custom error messages to the model, making it invalid. An html helper tag in the view displays the text of the error wherever needed.

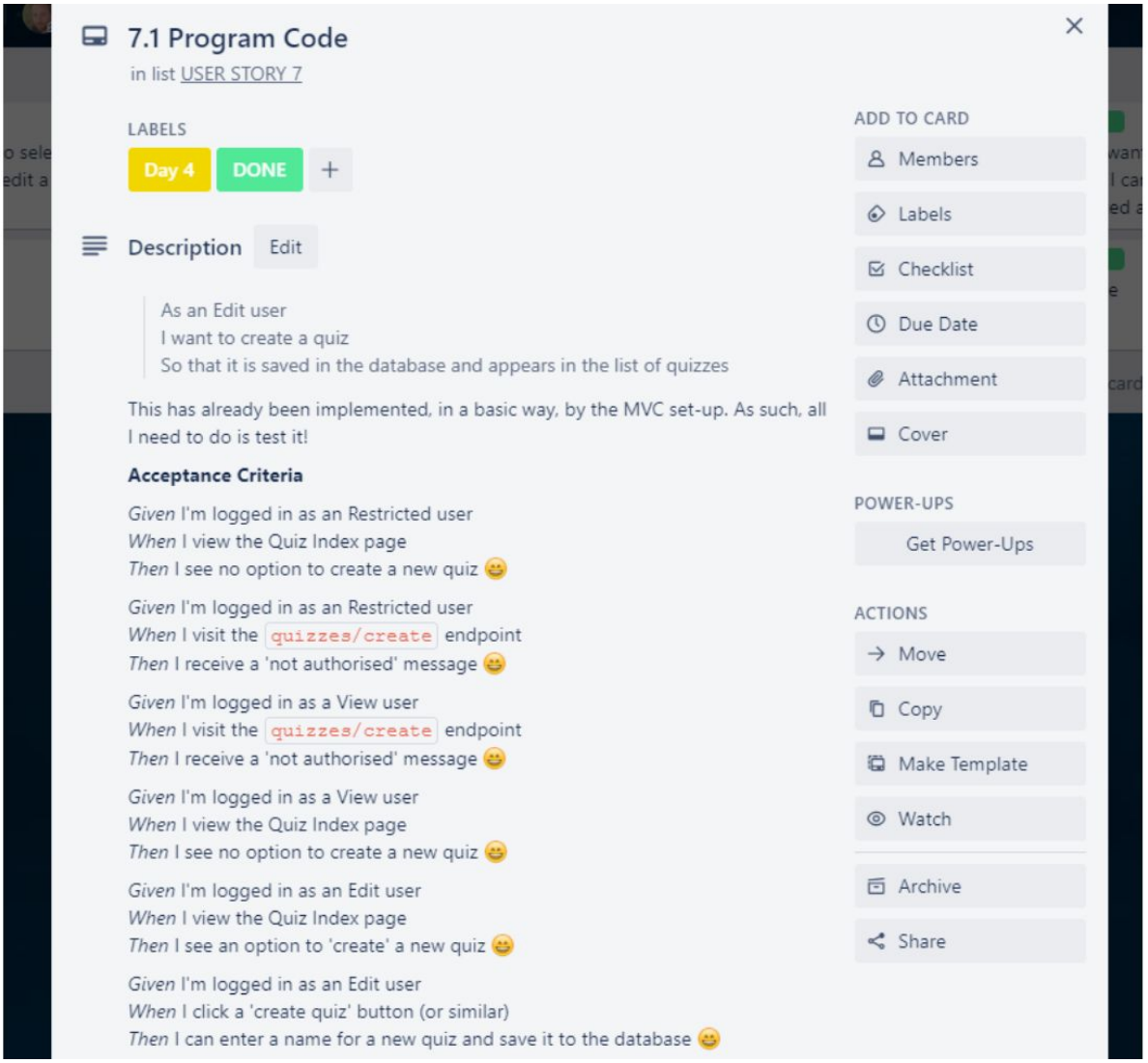
- Was concerned about security, SQL injection and the like, but apparently EF and Linq already check for that, which is nice.

- Trying to use the `ValidationMessageFor` method, which should validate properties with data annotations, and am pointing it at the `AnswerOptions` for a question being edited, but it's acting hinky and displaying by default when loading a question, even though the answer options are all present and correct. Finding it hard to debug, so will try using the other slightly hackier method of defining my own validation logic, as done above when checking how many answers are being saved.

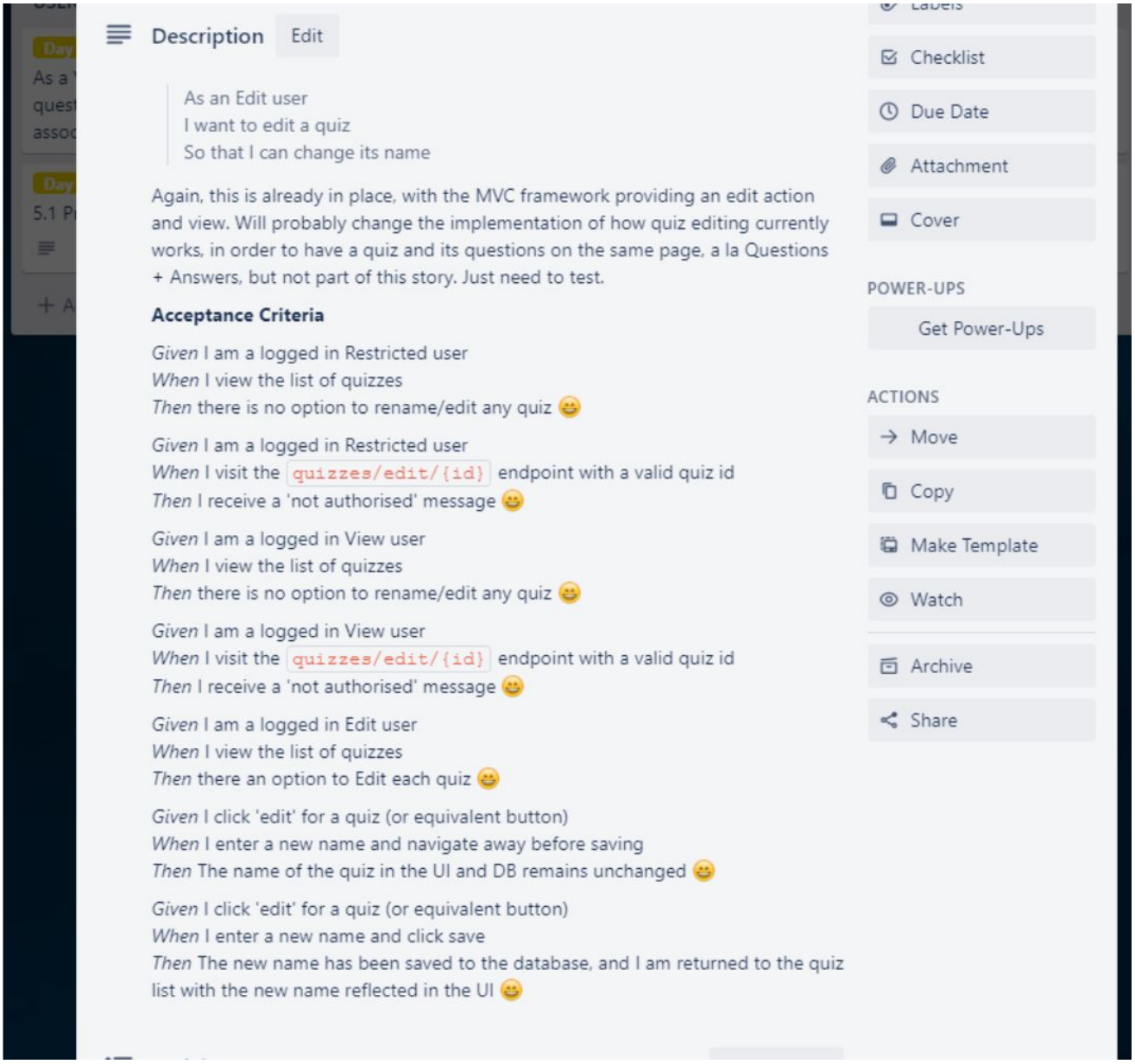
Ended up adding a separate method to validate against all answer option requirements, and using it on every controller action that leads to a `Question` with `AnswerOptions` being saved

- Ran into some trouble trying to get it to forbid adding another answer if there were already 5+ present. Have defined a standalone controller action of `AddAnswerOption`, where it will check if the `Question` already has 5+ answers, and will invalidate the model with an error if so, before redirecting back to the `Edit` view. Unfortunately the model state doesn't persist between calls, I found out after much trial and error, and using TempData to pass state wasn't working. Stack overflow to the rescue, as there were a few services and configuration settings I needed to set up in the `Startup` class to ensure I had a working in-memory cache I could use to store TempData.

User Story 7 Edit User can create quiz



User Story 8 Edit User can rename Quiz



Description Edit

> As an Edit user
> I want to edit a quiz
> So that I can change its name

Again, this is already in place, with the MVC framework providing an edit action and view. Will probably change the implementation of how quiz editing currently works, in order to have a quiz and its questions on the same page, a la Questions + Answers, but not part of this story. Just need to test.

Acceptance Criteria

Given I am a logged in Restricted user
When I view the list of quizzes
Then there is no option to rename/edit any quiz 😄

Given I am a logged in Restricted user
When I visit the `quizzes/edit/{id}` endpoint with a valid quiz id
Then I receive a 'not authorised' message 😄

Given I am a logged in View user
When I view the list of quizzes
Then there is no option to rename/edit any quiz 😄

Given I am a logged in View user
When I visit the `quizzes/edit/{id}` endpoint with a valid quiz id
Then I receive a 'not authorised' message 😄

Given I am a logged in Edit user
When I view the list of quizzes
Then there an option to Edit each quiz 😄

Given I click 'edit' for a quiz (or equivalent button)
When I enter a new name and navigate away before saving
Then The name of the quiz in the UI and DB remains unchanged 😄

Given I click 'edit' for a quiz (or equivalent button)
When I enter a new name and click save
Then The new name has been saved to the database, and I am returned to the quiz list with the new name reflected in the UI 😄

Checklist
Due Date
Attachment
Cover

POWER-UPS
Get Power-Ups

ACTIONS
Move
Copy
Make Template
Watch
Archive
Share

Daily Retro, Day 4

Day 4

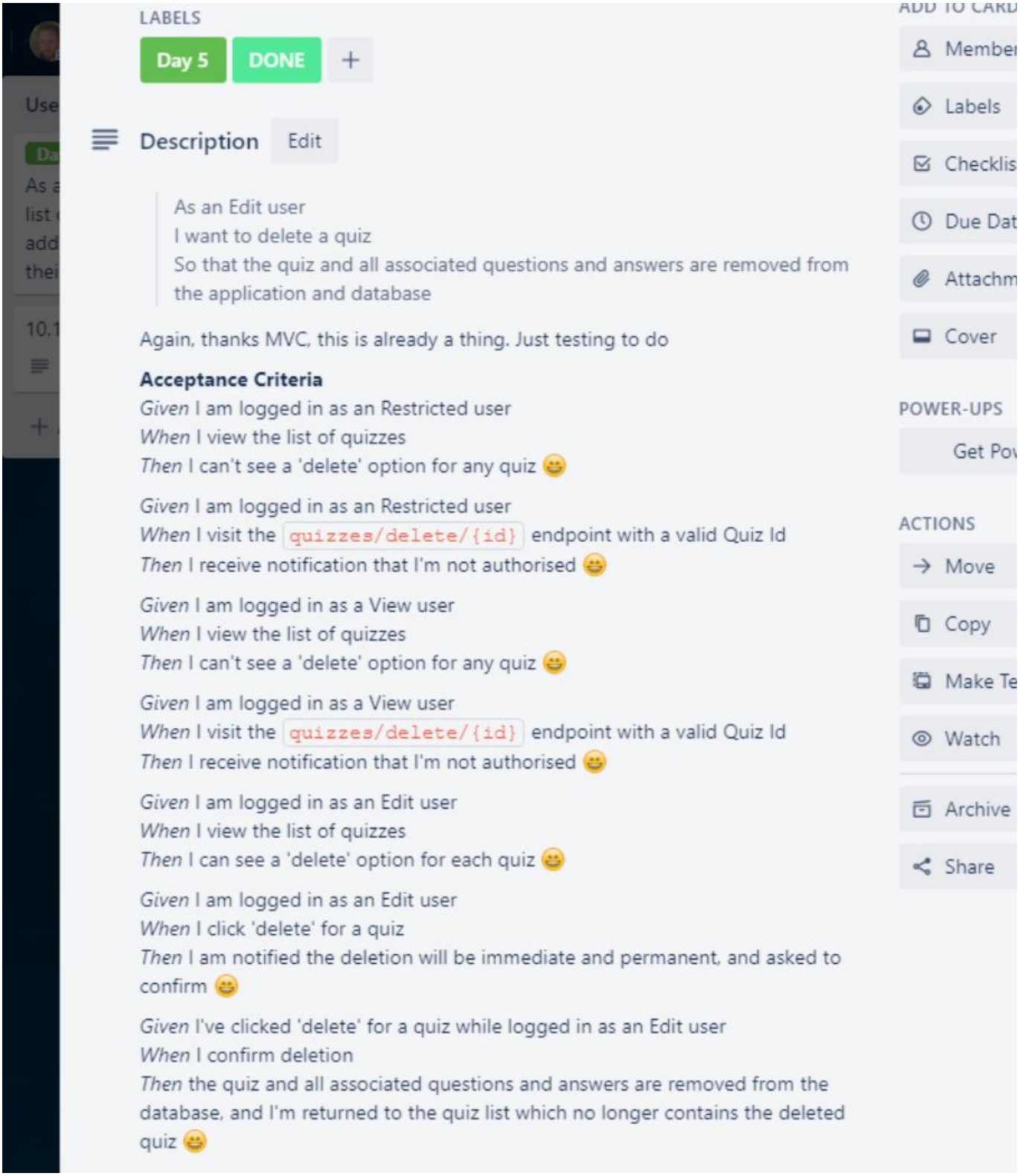
Brain Dump	Proud of
Good day today, although I didn't do as much as I would have liked. Everything is pretty much in place, though, so the remaining stories should be quick for the most part.	● Progress is good. ● Rhythm of work is good, and I'm generating documentation as I go
Could do better	**Next on the plan**
● Speed is important now, and I'm still taking a bit too long on certain things when I get intrigued about why something I'd like to add isn't working as I'd expect or hope ● Need to concentrate on just implementing the stories or tasks as they are. If there's time at the end I can improve things where possible	● First things first; Just crack on and finish it. Should be doable by the end of the morning. ● Test running a published copy on another machine, and troubleshoot as required. ● Start the afternoon with ensuring documentation is in place, and user guide. ● Any time left can be spent on tweaks and improvements, prior to a final updating of any documentation that'll need to be changed.

Day 5 Monday 1st June

Up against the clock now! I was working right until 6pm on Friday and didn't update a bunch of Wiki docs etc, so have just finished tidying up some admin which took 30 mins or so.

Priority is just to burn through the last user stories, and try not to get bogged down troubleshooting - if stuck for more than 10 mins, move on. Once done, it's time to test I can publish this to run on another machine, update the Readme accordingly, then admin tidy up, and user documentation. Tweaks at the end of the day if there's time; my Makers invigilator says I've got 1hr 40 mins of time over and above the core 9am-6pm hours, due to missed breaks or check-in times etc, and I can imagine I'll need it.

User Story 9 Edit user can cascade delete Quizzes



- This currently works using the MVC default views/actions, but I'd prefer users to be shown a pop-up asking them to confirm deletion, rather than be taken to a new page and back. I hope this should be straightforward, as Bootstrap has modals you can trigger really easily. Can't remember if they require and additional JS, but if they do, I don't think it'll be complex

- At present, the MVC model has scaffolded a `Delete` action and a `DeleteConfirmed` action for each model. First attempt at implementing the modal, which did require a little JS, rerouted to the first one. Think I can just remove one of the actions as the confirmation is now handled by the UI, and handle any necessary validation deletion etc in the remaining Delete action

- Hmm. I got rid of one of the actions and for the remaining one I set it to Post, as best practise for CRUD is that deletion shouldn't be made with a get request. Seems like my modal is sending a get. Maybe have to involve a form or something? Will have a play around.

- Don't think this as straightforward as I'd hoped. I tried using a form, but this won't work without using an AJAX query, if the form is in a modal. I could also just post directly to the endpoint with an ajax query presumably, but would then have to parse the response and use conditional logic etc to decide what happens to the modal and what's displayed to the customer. At least, that's my understanding. All of this would only serve to stop the screen view changing for the user, but it wouldn't actually save any clicks. Think I'll add this to my 'nice to have' list and stick with the default MVC behaviour for now so as not to waste time.

User Story 10 Edit user can edit quiz questions

- Running into problems trying to create a question and answers in one go. A form can't validate child properties by default, it seems, and any round trip to the controller means that related info, such as QuizID, which I'd originally passed into the view using TempDate, are no longer persisted. I feel like it'll become a recursive nightmare if I have to make that round trip each time, so will see if I can use Javascript to validate the AnswerOptions section of the form before it's submitted.

- Okay, I'm getting nervous about time here, and it looks like any javascript is complicated by the fact that I'll have a variable number of answeroption fields and so on. I don't like this, but I think I'm going to redirect the `AddQuestion` button to the `Edit` action in the `Questions` controller, and create a new question object to feed into it. This is bad for a few reasons;

  - Breaking MVC convention

  - It means creating an object that may not be saved by the user. That means I'll have to be able to delete it if the user clicks 'Cancel' instead of 'Save' when the time comes, but I'll also need to look into what happens if a user navigates away from the page or their session times out, etc.

  - Eurgh, the more I think about it the worse that option seems. Will spend a bit longer looking at the JS.

- Found an article which pointed me towards custom validation attributes, so going to give those a whirl.

- Woohoo, these are great - have removed the validation method in favour of custom validation attributes, which work great on the 'Edit' action. Now to see if they also work on Create

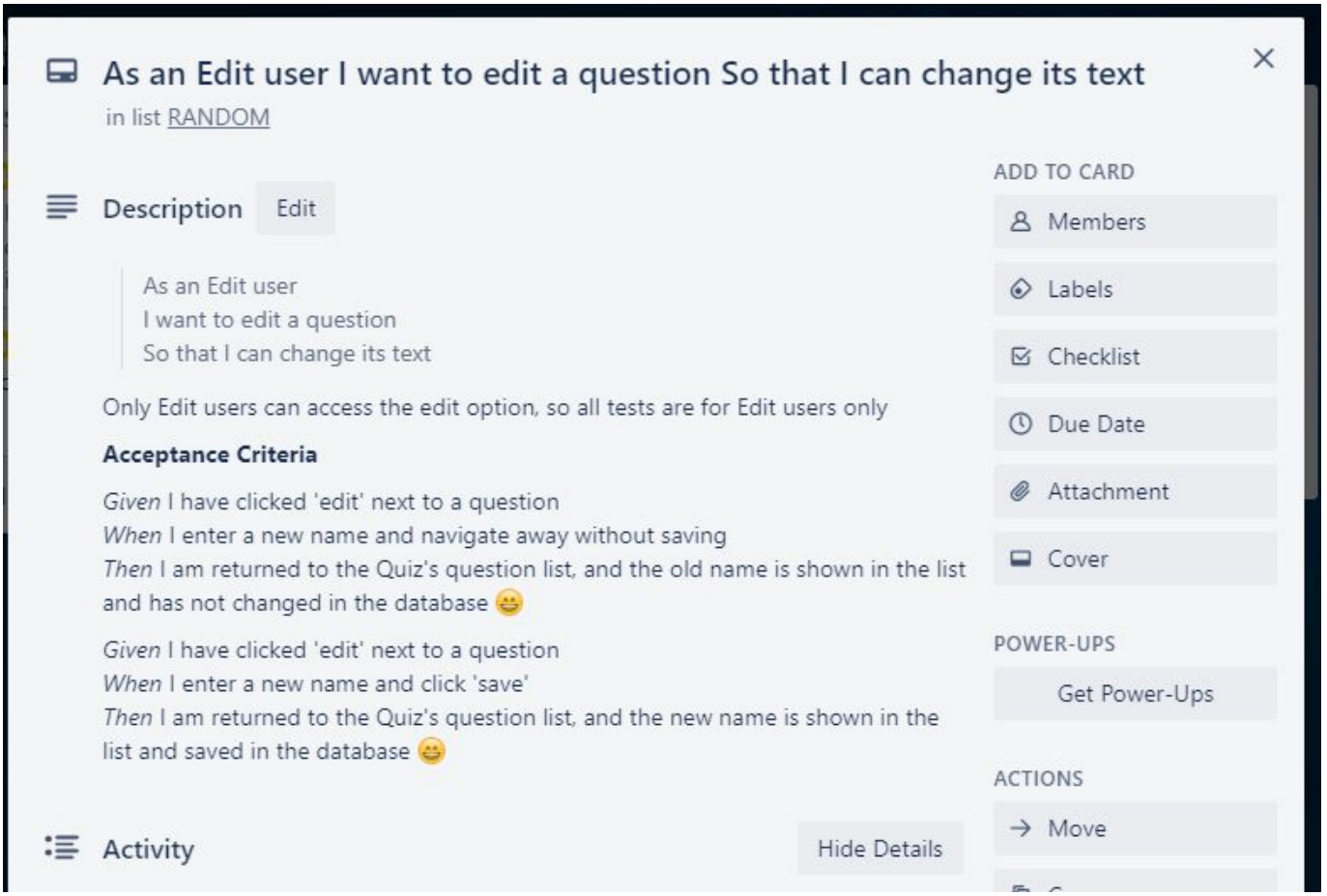
- Almost - I'm getting validation, but it's not pulling the custom messages through yet, just a generic one about fields being required

- I just wasn't binding the AnswerOptions from the form submission, but once that was in place the validation is working as intended. Happy with that! It also sorts out a few issues I was expecting to have to tackle, about what might happen if a user navigated away from a change having already saved something. Now, because all of the validation is happening before the controller gets its hands on the data, it can only get saved by a positive User action; if they navigate away, nothing will be saved.

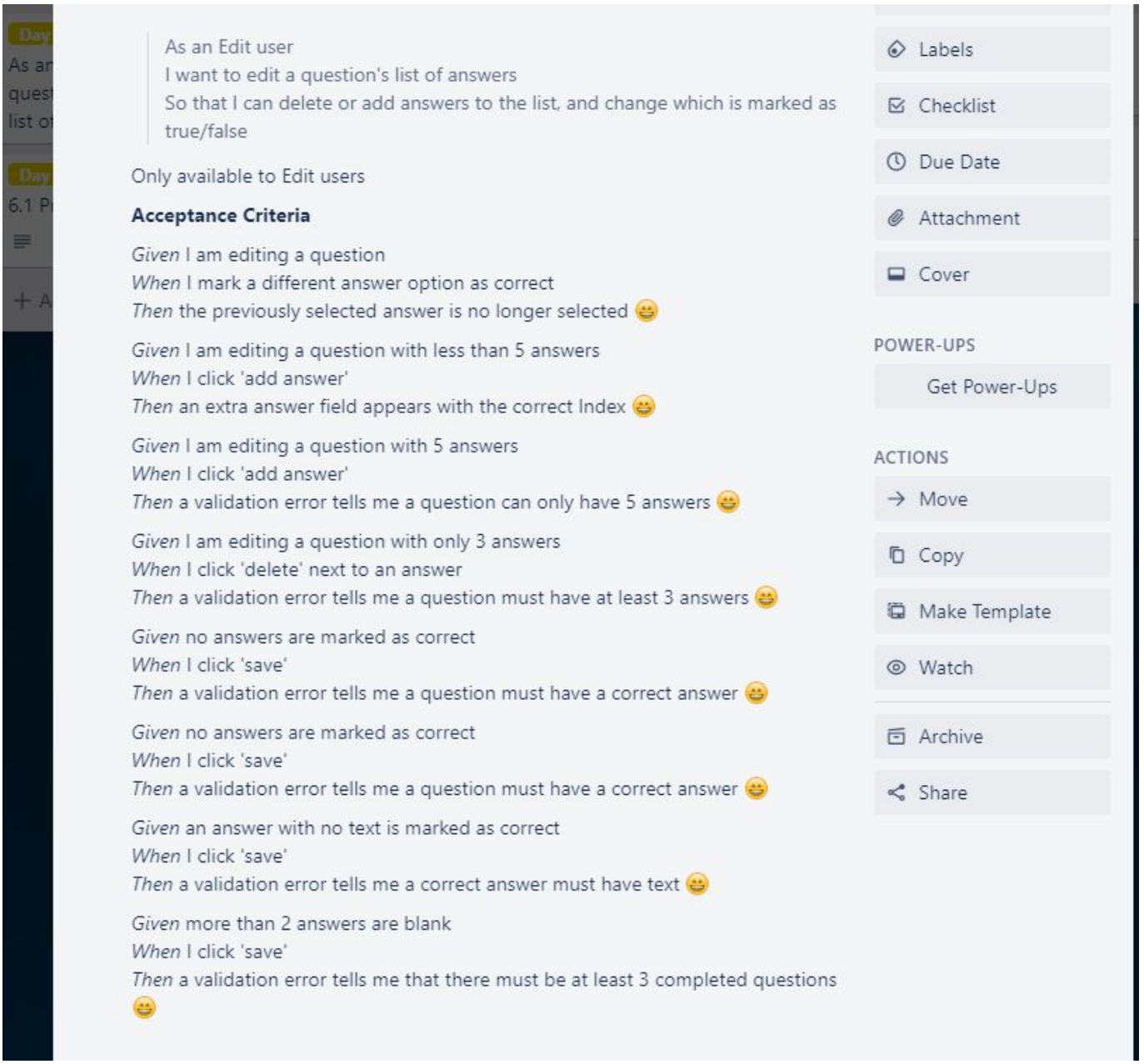
- Ran into some minor issues where the QuizId wasn't persisting, but it's because I didn't realise that TempData marked values for deletion once read. Found the `Peek` method which sorted this out.

- This became horrible, and time is getting very tight. I wasn't able to get extra answer fields added using Javascript, or at least I was, but they don't persist between refreshes. I was going down the rabbit hole, so instead I've started the Create view off with 5 blank answers, and changed the validation rules to require at least 3 completed answers. Any blank answers will be deleted on save. Same behaviour is true for editing existing ones, too. Bodgy and hacky, but I need to move on!

User Story 11 - Edit a Question to change its text



User Story 12 - Edit a question’s answers



> As an Edit user
> I want to edit a question's list of answers
> So that I can delete or add answers to the list, and change which is marked as true/false

Only available to Edit users

Acceptance Criteria

Given I am editing a question
When I mark a different answer option as correct
Then the previously selected answer is no longer selected 😄

Given I am editing a question with less than 5 answers
When I click 'add answer'
Then an extra answer field appears with the correct Index 😄

Given I am editing a question with 5 answers
When I click 'add answer'
Then a validation error tells me a question can only have 5 answers 😄

Given I am editing a question with only 3 answers
When I click 'delete' next to an answer
Then a validation error tells me a question must have at least 3 answers 😄

Given no answers are marked as correct
When I click 'save'
Then a validation error tells me a question must have a correct answer 😄

Given no answers are marked as correct
When I click 'save'
Then a validation error tells me a question must have a correct answer 😄

Given an answer with no text is marked as correct
When I click 'save'
Then a validation error tells me a correct answer must have text 😄

Given more than 2 answers are blank
When I click 'save'
Then a validation error tells me that there must be at least 3 completed questions 😄

Labels
Checklist
Due Date
Attachment
Cover

POWER-UPS

Get Power-Ups

ACTIONS

Move
Copy
Make Template
Watch
Archive
Share

Final Tests - Done in a hurry!

User Story 13:

As an Edit user

I want to delete a question

So that the question and all associated answers are removed from the application and database

- **Given** I have clicked ‘delete’ next to a question
- **When** I confirm my choice
- **Then** I am returned to the Quiz details page; the question is no longer in the list, and it has been removed from the database, along with all associated answer options. **DONE**

User Story 14

As an Edit user

I want to edit an answer

So that I can change its text and whether it is true or false

- **Given I** have changed the text and Correct? Value of an answer option
- **When** I save the Question
- **Then** I am returned to the Quiz details page; the new answer option is reflected in the UI, and in the database. **DONE**

User Story 15

As an Edit user

I want to delete an answer

So that the answer is removed from the application and database

- **Given** I have clicked ‘delete’ next to an answer option
- **When** I confirm my choice

- **Then** I am returned to the Question edit page; the answer option is no longer in the list, and it has been removed from the database. **DONE**