Realised the initial plan I made was closer to a waterfall approach. Did a reworked plan using an excel template that is split down into releases with each release having 4 sprints.

04/11/2019

LOGIN

Created initial single page setup

I used jquery mobile and jquery to create a single page setup. I had a number of issues getting jquery mobile to work with new version of jquery I have reverted to an older version for now. Will concider updgrading if I come across functionality I need. Page candy can be done using jquery mobile.

Made the login authenticator redirect to a login page instead of a popup as the popup UX on mobile is not very good

Notifications

Created a set of custom elements to allow notification data relating to the user to render. This is current unordered and will show every notification ever.

Intially had an issue with getting connection call-back to fire, this was a problem because I had misnamed the element registered to the custom registry. Found out I can use attribute change call-backs to modify displayed data on elements. This is particularly useful when dealing with asynchronous calls, like the promised used by firebase.

Created team invite accept mechanic to add new teams to the users own team collection. This will potentially be an issue when doing permissions. The user needs access to team data in order to add a reference in there own collection, permission may be achieved by adding a user reference in the team data someone to check against.

05/11/19

Meeting with Jarod confirmed I am on the right track with planning. Raised concerns of implementation without proper design. After I have proven that some of the higher risk elements of this release are possible I will review and enrich wireframes and use cases previously done.

I had to add these additional tasks to the current sprint as its imperative this is done early to reduce the change of UI redesign. Fortunately contingency and task over-estimation should extend the amount of work this sprint beyond workable limits.

Also raised concerns with GDPR particularly where sending personally identifiable information like names. This has made me rethink the invite team process as the ability to search for users by name or synonym will violate some GDPR principles.

06/11/19

Enriched some of the wireframes initial created as part of this process I created a new set of use cases and created flow diagrams to think about screens and process flow. This was inspired by a conversation with Jared about the different types of users considering company structure. I created a use case for a more vertical type of company where creation of tasks and team invites is reserved for management roles. Created flow diagrams to help inform the design of screens and application flow.

Refactored the notification class extracting query control and snapshot CRUD esc functionality into a separate class. This should be enough to create any updatable list from. This was helpful and help clean up the notification class. I did this in order to make a team custom element list.

The team list is another example of a query-controlled snapshot list. This query creates a list of divs from an accounts team data. This displays any teams that the user is subscribed to by having it in the collection under an accounts document. This may have potential issues with permissions, I believe access to the team document can be controlled in this way.

Created functionality to allow testing of notification invite system. This is just a form on the team row that “sends an invite” to any player. This is done by creating a team invite notification for the desired account, on accepting the account will then have a record of the current team. This allows permission control for both R/W access of a team as well as simplifying the query to access all team data.

08/11/2019

Created flow diagrams from use cases this was to aid thinking about the user’s actions involved a process. This helped iron out the differences between a vertical and horizontally operating organisation and tie them to system functionality. This has a profound influence on the design as I realised, I will need to accommodate for these differences, rather than simply relying on a team’s internal processes. An example of this is on completing a task, I need to provide the ability for review of completion to take place.

Creating flow diagrams has highlighted that when creating a new task or team the process will be very similar. These objects will both require forms to enter data and a client-side validation process to make sure the minimal information is entered to perform application operations. In creating display and modifying any of this data the processes may be able to use the same custom elements and code. I am glad to see such a clear benefit OOD already.

As part of the GDPR issues I have decided searching for users may be a potential violation as well as being a source of extensive data reads. By making a user present a QR code or user id to someone wanting to invite them we can circumvent issues with storing names/emails on the system. Using a QR code generation and reader should streamline UX issues with entering complex codes. I have decided to move this functionality into this sprint as I am ahead of time in terms of my estimates and have not utilised my contingency.

11/11/19

Created QR code generation opted to use ”<https://github.com/davidshimjs/qrcodejs>” this was relatively easy now that there is access to the unique ID of the authenticated user. I then investigated scanners again having decided that computer based inviting is still possible. I tried a variety of different APIs to access the camera and decode QR data, after extensive research I found several cross-compatibility issues using a web browser to do this. I tested the ability to scan with the default functionality of the phone, this is considerably worse UX then using a Vue based scanner but does work and has the advantage of innate cross compatibility.

The plan I have composed has already been useful in focusing my efforts on specific function groups, I have decided that the base invite functionality will work in its current form. UI improvements should help alleviate some of the UX issues. I am moving onto implementing some of the data structure suggested in flow planning.

DOWNTIME

Created structures that can execute the creation of database objects. Created first example which is a form custom element that creates editable fields from a json. This can be used to apply default fields and as a basis for validation. Need to father investigate database time as we may not need to explicitly save time objects. Struggle with some of the object referencing methods to make access correct values. Use some solutions suggested here <https://stackoverflow.com/questions/12622744/how-to-loop-through-the-attributes-of-a-json-obj>. Pre planning has help make this element polymorphic as it will be able to modify and create task data when development moves onto that. I am increasingly concerned with some of the repeated code inside implementations of the query list.

Discovered Vue.js as a potential alternative to JQM with custom elements. Seems to have a significant amount of compatibility with the project I will investigate this as an alternative.

Split up the team creation task into more atomic parts, there has been significant under estimation of the time it took to do this. I believe this is because the task was too broad and involved the POC of creating a team, modifying a team and displaying all teams a user can see, actually making the team and a significant amount of generalisation to make the component reusable.

12/11/19

Developed the form mechanic further to allow for modification of document records using the same set of custom elements. Adding to the data properties and switch statements will allow any object to be modified by these elements. My placement help me significantly here, having programmatically generated a number of form elements using server side languages I understand some of the underlying properties and how to interact with them with JavaScript.

Found an issue with the basic query list generation class. It will work perfectly where you are accessing records directly in the query. The issue comes when you don’t want the query direction but a reference inside of it. The notification page implementation of this has an issue where it is applying listeners to the whole result. The solution to both of these issues Is the same, I will make a new query list that applies listeners in a more manual way. This will give me control in implementing classes allowing me to apply listeners to references or not at all in the case of notifications.

Made changes to the how a user sees team data, because there will be more properties on a team in the future I think it no longer makes sense to have a separate local nickname on the user. This also lead to UX issues where a user may think they are changing the whole group name when in fact it is only there local reference.

Conversation with Jarod increased my concern with the UX of the QR scanning process. This may become a reason for a user to not use the application and will be revisited in later sprints.

13/11

There was an issue where updates to team names where not applying. This was because the listener for the document was being added to the users users-team collection. There are two main ways round this, adding a new version of query list that listens to documents referenced in query child documents or rearrange documents so the query returns teams as top level results. I decided to change the data. The advantage of the later is that less database trips are required to get the list of teams. The disadvantage over the first are significant data changes are required and preserving existing data will be time consuming and more changes are required in the code to do so as the invite system will also need to change. I opted to change the data structure as there is no userbase so, at this point, I can simply bin the data as and this will limit the variety of implementations of query list listeners.

After making this change to the database and clientside code to accommodate, the implementations of team and notification list listeners became very similar. This means it was worth refactoring the query list custom elements to include a static and active elements. The active element gives a class for the similar functions of team and notification lists to inherit. After completing this refactor I noticed a bug: the list would add cards of documents that are added to the query after snapshotting this meant when removing the new document from query results the list would remove the first element in the list. This would have been present before refactoring. I fixed it by updating the array of elements by splicing the new element in instead of pushing and using insert before rather than append child to update the dom.

Discovered the method :

firebase.firestore.FieldValue.arrayUnion(code)

as a way of updating arrays, this has the advantage of not having to explicitly pull the array down modify it and save the new array, I believe this is executed as a server side operation.

As an alternative to the previous security modle I believe I can use the pending-invites property to give permission for a user to read the team document.

I have gone significantly over my estimate for this period of time due to unforeseen refactoring needed, I should have done UML diagrams earlier and made more satalight tests as I may not have gone so far writing the way a user queries there team.

14/11/2019

Create new users and tested the development made this sprint. User login may need some looking into as it may be advantages to testing to be able to make test accounts on my application rather then creating them as google accounts.

Creating new user  
The user page generates a new user when they are authenticated for the first time. This process has no errors so far, an issue maybe that we are using the useruid but this seems to be a fairly well trodden path.

Creating new team

Team is created with the correct name, description and type. I’m not completely comfortable with the process of working out and displaying select options. The way it is currently done has minimal database overhead as we are only storing a small database value storing, as an int over the name of the property we show the user but it isn’t clear looking at documents what these values mean for auditing purposes. There is a small bug where form generation is performed in the opposite way as form generation which may cause UX and styling issues.

Invite user  
 Notification is created correctly on the database with the id typed in. Team data is correctly modified when inviting a new user and ignored when re-inviting

I need to do better checking of this value as the id a user can have is specific. Currently you can create invite notifications of any 28length string, this should be a major issue but is potentially abusable.

Accept Invite

Notification creation correctly causes an update on target user, this mechanism will be very handy to alerting users to team milestones promoting a sense of community.

Accepting invite updates members list correctly updates team data. This triggers updates to users team list correctly. Not sure if the user modifying team data is the best way to go about this but using the firebases internal method reduces error potentials with dual/multi writes on the same document.

Modifying team

Data on team that is modified is modified correctly – there is currently no check to see if the document is changed although this shouldn’t cause a bug it does create an unnecessary round trip.

Any changes visible to user’s correctly cause DOM updates. Any user can change the team data and the changes are visible to all users. I haven’t testing dual writes on the same document but this will cause updates for both users so the last write to execute will persist. Checking whether this document has changed could be included in the functionality of change form.

18/11/2019

As part of sprint planning properly breaking down the tasks. Last sprint a couple of bigger tasks overrun as the amount of work to do in them was hard to estimate. As a result during this sprint planning I broke any task of significant length down into smaller ones.

Created basic bones of functionality to load tasks, this highlighted some of the advantages to class based custom elements. I was able to use the generic list builder to product a list of tasks under a hard coded team. This demonstrates it will work as I intend.

Created basic navigation to tasks screen, this was particularly tricky as the button for navigation requires an action external to the element in that the task page needs to know what task is being viewed. Decided that a global variable is fine as we want to manage and update a cookie at this strange so that the user can navigate away and still have the same team open as they did before. I need to investigate potentially filtering tasks, this may be a simple change to allow multiple teams to be viewed at ones.

Fixed bug of form input removal caused by the DOM array resizing between loops. In order to change the fields shown in a form I am removing children form the parent when a new type/document is loaded. This prevents an excessive modification to dom elements. There was a bug where the child element collection was resizing during deletion of elements, this causes an out of bounds exception where an attempt is made to access an element beyond the resized array. To fix this I uses a while loop to remove the form rows, this has the advantage of being able to delete any content within the fieldset.

19/11/19

I had a meeting with Jarod today and that highlighted a couple of points that are worth noting:

Discussed arranging meeting with second readers, following this I emailed the second reader and arranged a meeting.

Jarod reminded my of a unit of work done in the second year using Jquery as a method of reading QR codes. I will add this to the current sprint as the poor UX may be more of an issue than I realised.

I discussed something I like about the client driven application. Making the client powerful in terms of knowing where to store records allows for a self forming database, I can simply delete the data and, although current data is gone, the application can continue creating documents and collections.

Although I like this capability there are Security issues with allowing collection creation. Applications that other people create can generate an unintended ecosystem of documents and collections that wont be subject to constraints from database rules.

I was also introduced to the idea of tunning this kind of reflection against individual tasks. This is something I can definite do and keep track of.

Discussing login testing custom login vs google UX. Creating my own login method would have advantages in being able to create and demonstrate multiple users. Currently I am using multiple gmail accounts to create them.

20/11/19

Created a test of the suggested Jquery based QR reader suggested by jarod. This test worked very well and also enables cross compatibility with computers as an image can be uploaded. This test worked really well and I went about adding it to the main test application. To do this I needed to refactor some of the document form methods into a simpler class, I then extended this class to form a simpler form that just contains the data nessisary to read QR codes. This also removed a lot of code out of team card class which makes a lot of sense as that class should only really be concerned with DOM updates.

Major styling issues around where the form complete and cancel buttons are. I believe the user will expect these to always be in the same place regardless of the size of the form fields. Discovered a method of doing this using flex boxes.

<https://stackoverflow.com/questions/5817233/align-button-at-the-bottom-of-div-using-css>

I prefer this method over using tables as it is more reactive and requires significantly less html and dom changes. I fixed these with css in the body of the index for now, I will be moving this outside when the main styling task is being done.

Looking at the documentation it may make more sense to change application buttons to jqm based anchors. I am investigating bootstrap and vue.js as an alternative to jqm so will hold out on that descition.

Buttons that hide show modals needs its own class as the methods are very similar this will also allow leverage of the custom element collection to create links to the 3 main modal popups.

Whilst developing task card displays, I ran into the issue of complex data type reflection in the DOM. I would prefer to change card attributes over making the card do another round trip to observe and create data less round trips and data reads. I know that I can parse and unparse json in an HTML attribute but I came across the following suggestion.

<https://stackoverflow.com/questions/16223786/store-and-retrieve-javascript-arrays-into-and-from-html5-data-attributes>

The solution of creation a string representation of this data is a good idea. Less character modification on the dom objects and the experience rewards is a relatively simple data type so can be represented perfectly this way.

20/11/19

21/11/19

VIVA

Document Monday

Include sequence diagrams

25/11/19

Added additional functionality to my document-form elements this was to allow the creation and modification of task data. This presented a few problems: in order to requirement data I needed a way for a user to add values to a list, this list comprises of the requirement in terms of skill type and the actual level required in that skill. This takes the form of a key:value map on the database. In order to represent this to the user and allow them to modify this data the form needed a way of the user adding and removing rows as well as the pairing of the key and value. I created a new custom element for this that contains the two within a div. I then modified the form parsing to only go through the top level elements. This allows the two to be wrapped in a div with a specific type and the parser to respect the values as a pair parsing them in a different way when it sees a particular element type.

Created a way for tasks to have metrics on them: these may change but are currently importance, impact and urgency to simplify task experience allocation and give a more understood and controlled way of assigning tasks experience. Currently the task experience values are a factor of these three things and the requirements on a task I aim to allow a requirement of 1 to be done by anyone but provide the minimal amount of experience.

Had to research some properties of the slider element. I aim on implementing a css library to make forms look better.

<https://stackoverflow.com/questions/36399209/html-input-range-showing-intervals>

I investigated using the library select 2 to add additional functionality to select boxes and improve their UX and appearance. This worked but created additional DOM modifications when creating select fields. This is actually very not good as the only select fields being created at the moment are relatively small so the ability to search through them Is actually no big plus.

<https://select2.org/>

Re found a create reference on custom elements that taught me how to control reflection via attributes.

<https://developers.google.com/web/fundamentals/web-components/customelements>

When creating the task creation process I discovered that mozilla doesn’t not understand timedate form inputs. I will investigate a way of a achieving this through either a library or my own custom form element.

27/11/19

Created class diagram of all classes made so fair, this is in preparation to discuss some things with Ali, I will be revisiting these next sprint to create some sequence diagrams and smarten it up for discussion.

Going through this process helped highlight some of the successes with the hierarchy. The query list element implementation are now really clean with the only methods on more extended classes are the ones particular to that functionality. These classes can now be used to render a list of documents in any web base firebase project, written in vanilla JavaScript means they have no framework requirements.

Highlighted several issues with the hierarchy the main ones being: card objects need a base class to stop generic HTMLElements being the only type available to base factories, QR code display probable wants to be modal based and so may need looking into with respect to the modal functionality of forms. Potentially extend a base modal popup to have the functionality for both the forms and user QR display.

Found program UML star to create UML diagrams. Has the bonus of creating a web documentation of the diagrams. This is going to be handy to document structure and will save time in the long run when it comes to formal documentation.

I severely underestimated the time it would take to document the application. This came down to having to revisit UML techniques to understand how to document relationships.