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.

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

Refactored query lists remove significant amounts of redundant and repeated code

Rearranged data in query invite process requirement of values to validated authorative changes moved to member / pending invite objects/

Significant issues with array modification

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.