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.

\*\*\*\*\*\*\*\*\*\*\* - TaskWireFrames // flow\*\*\*\*\*\*\*\*\*

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.

\*\*\*\*aforcanace based css\*\*\*\*

21/11/19

VIVA\*\*\*\*\*\*\*\*\*\*\*\*

Document Monday.com\*\*\*\*\*\*\*\*

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. Whilst doing this I created client-side validation for text fields, the simplest way to do this is allow forms to define there own validation, this may need to be revisited to check after forming the json.

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.

29/11/19

\*\*\*\*\*\*\*\*\*Unit Tests\*\*\*\*\*\*\*\*\*

team test

create

edit

task test

Invite test

02/12/19

Adopted task complete which was one of the main tasks from the next release to allow time to play question-air. This is something that was suggested in the VIVA. A questionair will allow for a way to view the success of the application and evaluate any major UX concerns. This has promoted some of the hardening and UX evaluation tasks forward to allow for a more representative evaluation of the final product.

Due to previous success the task complete development was broken down further into, respect requirements, UI Control, and experience awarding. This allowed task rewards user account display and task/leveling fan fair to remain in release 2, as a result there is an acceptable amount of time in both this sprint and next release.

# 04/12/19

Task Completion(TC) - change status

Changing the status manually on the database has no effect on the tasks. There are 3 things that should potentially happen: trigger fanfair for relevant users , task needs to award experience to the user that completes it and tasks need to hide from the screen as there needs to be as little visual clutter to maintain mobile friendliness.

TC - give experience rewards

Giving experience rewards introduced a whole new way of manipulating firebase data. Previously when adding two maps I havn’t wanted any repeated keys and would be fine over-righting any value data. In this situation the two maps need to be an addition of each other, so any value in both is added and any value present in one should be present in the other. In forms the data is already present and easily accessible, when completing tasks the list card does not have the document loaded. This means the data needs to be fetched from the players current xp added to the task reward and then added back to the players experience. This will be at least 1 read more than we need but is quicker to write concidering my current hirachy. The task list may be better suited to have direct access to the snapshot list to prevent any superfluous round trips.

Experience is now acquired from completing a task, in order to add additional personalisation, I had the idea of adding skill configuration to teams. Changes where required to the way a document form gets the select options but creates a cool interaction. Teams can now create personalised skills that are available to add as requirements to tasks, this creates a whole set of customisability to a teams experience of the application.

TC - UI control

Added a button on task card to trigger the task completion process. This initial performed a simple update on the document it is representing. Initially this just changes the status to complete, previous changes to the list query allowed these to automatically hide.

Client-side validation teams

There are some issues with the hierarchy of the document form. This was using a case statement to parse object definitions in order to make the form. This was very clumsy and didn’t leave much room for configuring validations on an object and field bases and according to my plan I am ahead of schedule. For these reasons a significant rewrite was preferred. The document form now parses form configuration objects in order to create the fields. This removed the requirement of routing with a case statement as the configuration objects can define their own creation. This meant there was a substantial over run of task length but the result is an object that can now interact in a much more dynamic way with data which will help with modifications I the long run. A few tweaks are still required for compound form row validation for value maps and custom string collections. I am incredibly happy with the changes and have tested all the functionality is now the same.

Found that JSON.parse(JSON.stringify(JSONObject)) has major issues with deep copying. This was causing a formatting error with the new form configuration issues. Found the following JQuery implementation to do this.

<https://stackoverflow.com/questions/5364650/cloning-an-object-in-javascript>

Discovered that this is infact a better way of doing it as less risk of JSON interpretation breaking the data inside it.

09/12/19

Tasks to respect requirements

Created a couple of functions to convert experience values to and from levels. These functions are in tern ran against an accounts experience values to calculate whether a user has the required levels to complete a task.

Added check to task completing inside the map combining iteration, this prevents additional loops and only performs the checks when required. Prevents evaluation relying on client-side values, need to investigate how well firebase rules can check that this is doable.

Display Experience rewards \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Created a bar based display from custom elements. These use the functions to convert experience to and from levels to work out what level they are in and therefore how far the user is into it, displaying it as a width of the bar. Need a way of triggering a fan-fair when a user has levelled up, this needs to be generic enough so that notification list can also trigger it, we may want to trigger fanfairs whenever a user has completed a new task. Currently these are not actively updated, update of these may need to be done in a similar way to task experience. Consider making a generic class to handle the similarities in functionality

10/12/2019

Meeting

Discussed making surveys, this is something that was discussed during the viva and is something I am going to incorporate into the project. This will give me the ability to evaluate the UX of the application as a form of measuring success. This has been scheduled for release 2 this is important as there is a good chunk of functionality already in the application, with enough time to reprioritise work based on the feedback of these reviews.

Creating a list of potential features there is a growing list of functionality and changes that may want to be implemented as part of version 1. I am going to create a list as part of the review at the end of this release and create an evaluation of priorities. These will be discussed with Jarod and as pointed out by him there priority may change due to feedback from the survey.

Discussed issues with hours spent in the project. I have spent a substantial amount of hours on the project so far, somewhere in the region of 180 hrs. This is significantly longer then I expect for the timeline of the project. The majority of this is due to setting more risky and time consuming tasks at the beginning of the project; the point of this is that there is more spare time for tasks to overrun and less other assignments out then there will be for release 2. Jarod helped alleviate some of my concern as there is a substantial amount of things I am learning, which makes evaluation duration difficult and individual projects often end up will a lul in the middle, presumably because of other deadlines.

Discussed implementing custom page based elements over JQM. The library is huge loading screens are required to make the process smoother and improve UX. Jarod suggested doing some research into loading times and how users perceive them in order to evaluate the necessity to develop my own SPA system.

11/12/2019

Created the initial sequence diagrams for team / task creation. Using star UML this also provides HTML based documentation which is really useful to review the design of the application flow. Had to do significant research into both using star UML and creating sequence diagrams in general.

As JavaScript is not a strictly typed language this is quite hard as the classes are not as well defined. I have used classes that are both the view and controller, this has the advantage of linking data changes directly to user display but becomes difficult to document and may have extensibility consequences. Due to some of the abstractions afforded by the firebase API some of the more complicated update logic is handled in a clean way.

Had to cut short this for today as there are pressing concerns with other projects, will finish later in the week.

12/12/2019

Created the final essential sequence diagrams of active list updating and the invite process. I am significantly more confidant with star UML so had to think less about how to use the tool and had more headspace to consider the design.

The invite process diagrams could use some cleaning up as this involves multiple actors, so the diagram is significantly harder to interpret. The process required multiple cators to maintain exclusive read/write/update permission of teams to team members. The other alternative would be to have the user query the database to add themselves to a team, this would mean a normal user would need to have write access, I consider this to be significantly less secure.

Documented active list update process, this process has been significantly refined during development. I really like the asynchronous nature of listening to snapshot changes. This allows a streamlining of DOM changes where change call-backs are causing only specific sections of the DOM to be updated. This should have a positive impact on loading times and hardware usage.

14/12/2019

Unit tests are now In a separate document. This process has highlighted a problem with the design, unit testing with code is very difficult as data is created and committed in one process. This will need modification to the underlying code to allow for proper unit testing. Tasks are being renamed to Testing as the actual nature of the tests are somewhere between UAT and integration testing.

16/12/2019

Sprint has been extended to three weeks. This allows me to take a week off. I am suffering with mild burnout and need to time to visit family etc over Christmas.

The purpose of this sprint is to create a questionnaire and prepare any major UX/ functionality floors that will prevent the questionnaire being effective.

UX Empty Team / task list

Added a change to the title of the tasks page so the user knows they don’t have a team selected and need to do so to see their team list.

17/12/2019

New User skill-list empty

There was a major bug uncovered by testing that was run last sprint. When a new user is created, they do not have a skill-levels collection. This then causes an error when the application checks requirements for a task when the user goes to complete it. There are two main ways of fixing this: making the task complete execution involve creating the map when modifying a user’s skill levels or adding an empty collection under the user when making this object. As skills may need to become a collection in release two (Task Rewards User Display) the later is a more reasonable solution. This is the quickest way to remove the blocker from surveying via the questionaire.

Task over ran significantly. This was a poor estimate on my part in the time it would take to fix, surveying possible solutions at the time of testing may have improved the speed of this task.

The UX evaluation task became informed by the questionnaire and urgent development tasks arose during the process of testing the survey.

30/01/2020

Initial testing of the first questionnaire draft uncovered some major UX issues that need fixing before a survey is viable:

**UX Form Input Labels**

Are presenting database keys instead of normal English, the form configuration objects are capable of storing this information but was not integrated into the display scripts.

This required a quick modification to the create form row methods to show the name of the form row as defined by the configuration object. This allows for example “personalised-skills” to be presented as “personalised skills”.

**Form help text**

The purpose of some of the form inputs are not clear enough when tested. The form configuration objects where created with the capacity to add help-text, including help text was not something I was initially planning for the first release. Testing the questionnaire showed these where needed as it was unclear what adding requirements was doing to the task experiences.

**Unmet level requirements**

Added a notify handler to show when completing a task has failed. This proved to be an effective enough way to communicate that the user did not have the required level. This will require further UX evaluation through the means of the survey.

**Skill reward Icons**

Added reward type text to get around issue of no icons. Will be adding icons later as It is required for an effective UX evaluation.

**UX slider issues**

One of the people testing the initial survey with me made the mistake of sliding urgency etc values to 0 instead of one. This level of failure would result in a succession of failures for the rest of the tasks on the survey. This was fixed by adding value change listeners and making them display the value the slider represents.

03/01/2020

Doing reflections as I finish each section of development has been very effective. I have been able to see a clear progression in my ability to manager the project as well as developing my coding skills particularly ES6 based JavaScript, custom elements, jQuery mobile and using firebase.

Completed a second test and discussion with someone about the survey. They suggested including some evaluative sections at the end of it to try and gauge feelings on styling and general feel of the application.

06/01/2020

**Sprint planning\*\*\*\*\***

**Userpage experience update**

The user needs a way of viewing experience changes as they game them. The experience bar custom elements are effective when updated manually. Using document change listeners we can update experience bars as a map. This way any changes will be mirrored on the UI, this means experience can be awarded externally to a user’s actions and the changes will be perceived. This is required to make the questionnaire effective as there are UX questions regarding experience and levels.

**Fanfair mechanic introduction**

Application needs some way of triggering fanfare, this initially is required for the survey as a section required the user to level up. This mechanic needs a way of triggering application wide with a variety of appearances.  
 Some research into notify.js allowed me to create the non modal popup that is now used to indicate a levelup, this method is currently pretty simple and will be extended further. This took some back a forth as non official documentation is using a bootstrap version of notify.js. This highlights another good reason to be using bootstrap, due to its lack of compatibility with JQM we can no longer switch as I would need to move to a vue.js page setup. This is a big shame.

07/01/2020

Meeting

Discussed anxiety with so much work to do, jarod said he is happy with the work I have done so far which helped significantly.

Collecting demografic information

13/01/2020

**User level-up trigger**

Created styling for the level-up message the method will also trigger audio and animations. This looks significantly better and actually seems to add to the reward of completing tasks. There is currently an issue where if levelling a skill not currently owned by a player it will not trigger the notify.

**Jquery Inconsistant button rendering**

I managed to work out what was causing an inconsistency with jquery mobile styling. I have made the mistake of creating buttons as HTML button elements, including extending the button class to create the edit button. For some reason that needs investigation JQM has consistency issue with styling buttons that are dynamically created. Instead anchors with the class UI-btn can be used, this has a significant consistency improvement. This fix was not scheduled as I had a suspicion I would come across a fix while doing development, it only took 20 minutes so is not being included in the plan.

**Information Architecture**

A number of issues with conducting the survey have arisen from poor information architecture. Spacing and segmenting task data made little sense after applying the Jquery consistency fix and skill icons. Some custom elements base HTML needed changing to allow divs to wrap related information together these where: task, team and notification cards, form elements, experience bars and reward cards.

14/01/2020

**Conduct survey**

Managed to conduct survey on a couple of participants this day. I was surprised with how successful the affordance of the application was. Both participants managed to complete all the tasks successfully. This is a good sign of the success of the application especially considering that both participants where seeing the application for the first time. There are potential biases arising from the actual survey in that user actions are.

Some issues that where raised during the process:

* Iphone / safari has major issues with displaying modal forms. This is either a click issue related to the button event or a style issue to do with the modal itself.
* Both users had an issue when adding personalised skills: the form does not indicated a content change and the user cannot see the new additional form row. This means they cannot see the entree for the personalised skill which is very confusing.
* Tasks requirements are not shown on task cards. For the purpose of the survey this was fine as participants are indicated, via notify.js, they can’t complete a task due to not meeting a level requirement. It was pointed out that this will be a major issue, if a user sees the task on the application and goes and completes it in real life (being unaware they don’t have the required level) they can then no longer mark the task as complete. As a result of this concern requirements display are going to be added.
* Deadline display
* Extra features

**Wireframes**

Want to do surveys fully before doing this as there its hints of UX changes required

20/01/2020

**Sprint Planning**

Due to issue with arranging time to conduct the remaining surveys with participants they survey period needed to be extended. This has implications for development as I don’t want to make application changes during the survey period as this will make results less comparable.

Database access rules introduced some way of getting around some very tricky navigation of error handling from firebase. When running access rules against a users request if access is denied the message handed back to the client is hard to identify over a transaction that has failed for a different reason. I know from reading firebase documentation that queries launched that include results that a user doesn’t have access to see will fail completely, I have form document structure in such a way that Discussions with Jarod regarding this made suggestion at using TDD for this process so I can confirm the rules are working as intended, this is particularly important when integrating with the application.

For the reasons stated above I am including test construction in this sprint to allow for TDD of the database rules that restrict a user’s access to data. At the moment there is a fundamental security issues as the application can be modified client-side, allowing a hacker to have access to any team/task/user objects. They can also create collections without restriction, this is something that is useful for development as the collections can be deleted and reformed creating a clean database with ease but is a potential avenue for abuse.

Using google authentication allows a rogue app to be created that can use database configuration (which is supplied client-side) to access the database without going through an internal validation process. This increases the propensity for abuse through the two above.

21/01/2020

Started the first tdd tests creating a separate application to check access to documents. Creating a display for the result of each tests and a series of buttons to perform them. The test application logs into google in the same way the normal one as any database access already requires users to log in.

Tests are published to the same domain as the application just prevent CORS errors from database code access.

**Write access Permissions tests- users**

**Test account:** JordanHarrisonTester@gamil.com

Created the first test as a user access test to get an idea of what methods will be used to display all the test results. The access tests have both a fail and success to make sure the application continues to function as normal with access to data that an authenticated user should have. The test that is expected to have successful database access, is the test application attempting to access the currently authenticated user document. The test that is expected to fail is attempting to access a hardcoded user that is not the test account.

The collection in quest here, ‘users’ is going to be pretty restrictive. The application has been written deliberately so that there is no reason for a user to have direct access to another users document. Access to a user document would have security risks associated, particularly if further development requires personal information to be stored against users.

22/01/2020

Tests written today took substantially less time then estimated, the writing of the majority generic testing methods was done when creating user tests.

**Write access Permissions tests- teams**

Created teams’ tests using the same application as the users. This has a fail and 2 success test. When the user attempts to access a team that they are neither a member, owner or pending invite they should get denied.

When a user attempts to access a team that they are either inside the members or pending invites array or they should be able to update and read the team. If they own the team out-right they should have RUD access to the team.

**Write access Permissions tests- tasks**

Access to tasks is done through a team document. This means tasks should be securable through there relationship to team data. If rules are created against a team document, these should apply to any task document unless deliberately overwritten. Tasks access contains test with the same conditions as the teams.

**Requirements Display**

Added requirement tiles, this is something that the survey has indicated will be needed, these can be written without promoting outside of the ‘development’ environment. I don’t want to make an application changes to the live site to make sure there is a consistency with different participants. Whilst adding requirement tiles I notices there is a significant noticed a similarity between the reward and requirement tiles. These are therefore a good candidate for a custom element.

24/01/2020

**Date display**

Added the deadline date to a display on the task cards. This is an essential part of the display as users will want to know when a task needs to be finished by. It may also be a good idea to sort tasks be deadline date, to make tasks that need to be finished soon to be shown at the top of the list. The code for displaying this will want to be in before the new wireframes are drawn. To simplify workflow when implementing new design.

**Team Types Access - Edit button visibility**

Control over whether the edit button is done at the stage of creating the card. The DOM property is added to cards which causes the edit button to be created when the connection call-back is made. This allows a calculation against: the team, user and team to decide whether documents should be editable. This won’t be protected by database rules as it would make the rules significantly more difficult to test and developed. This is being flagged as a potential improvement to the application in the future, the test application could also be extended to allow for these rules to be TDDd.

26/01/2020

Survey with S3

28/01/2020

Refactored card lists into there own custom element. Removed repeated code and put card display calculations where they belong

02/01/2020

Survey with S1