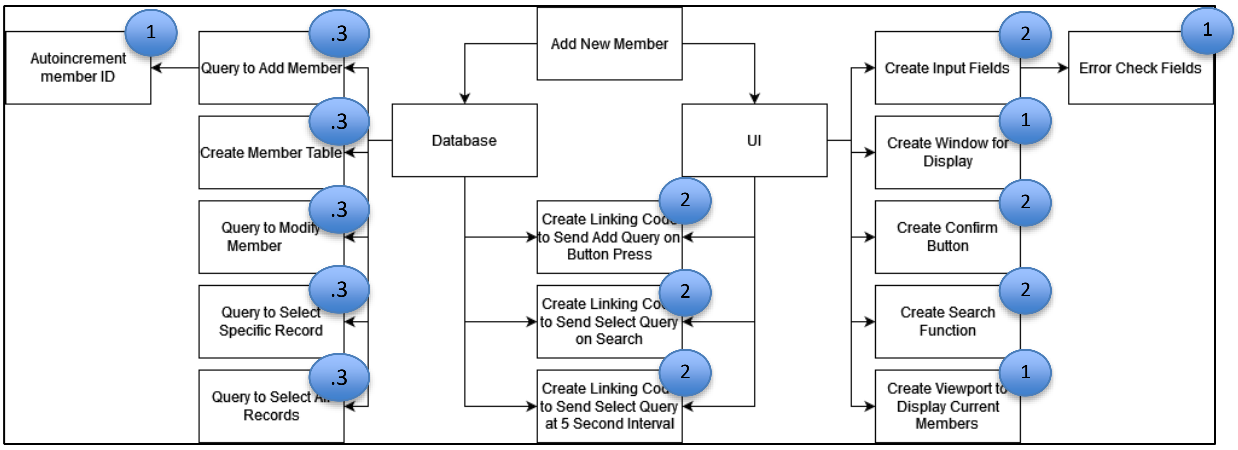
Estimation Accuracy: GotoGro-MRM

Estimation by The Numbers

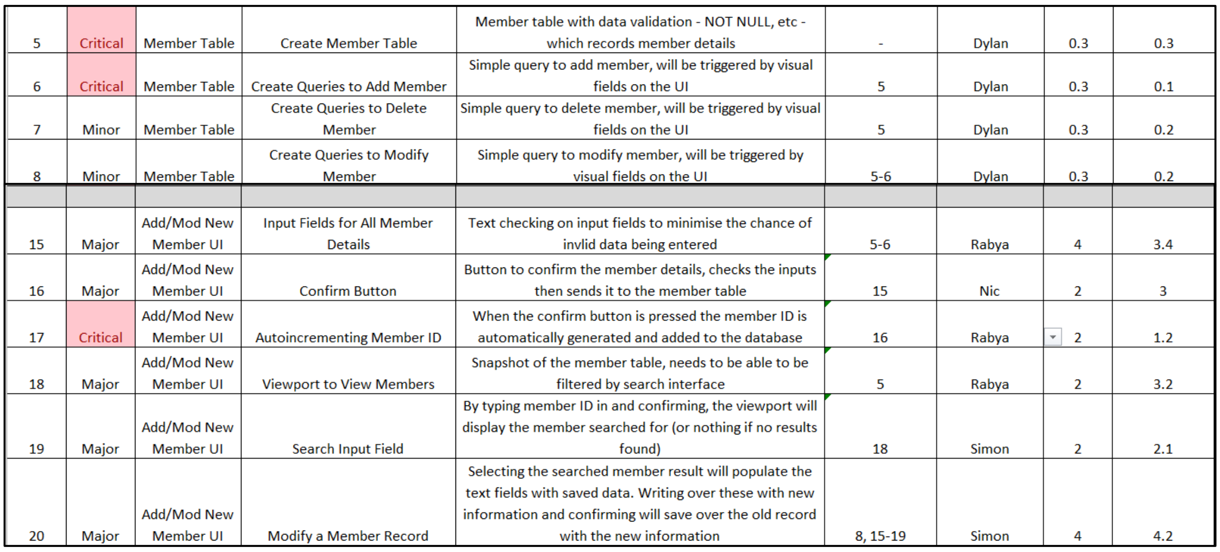
The sprint backlog item selected for the estimation task 61C was the “Add new Member Interface”. To refresh, the proposed timing breakdown is shown in **Figure 1**:



Add New Member
Database
UI
Autoincrement member ID
1
Query to Add Member
.3
Create Member Table
.3
Query to Modify Member
.3
Query to Select Specific Record
.3
Query to Select All Records
.3
Create Linking Code to Send Add Query on Button Press
2
Create Linking Code to Send Select Query on Search
2
Create Linking Code to Send Select Query at 5 Second Interval
2
Create Input Fields
2
Error Check Fields
1
Create Window for Display
1
Create Confirm Button
2
Create Search Function
2
Create Viewport to Display Current Members
1

***Figure 1:** Estimated time breakdown for the Add Member Interface.*

This totals to 17.5 hours of work. During the sprint, the time taken to actually do these tasks was captured by the team so an easy comparison can be made using **Figure 2**:



5	Critical	Member Table	Create Member Table	Member table with data validation - NOT NULL, etc - which records member details	-	Dylan	0.3	0.3
6	Critical	Member Table	Create Queries to Add Member	Simple query to add member, will be triggered by visual fields on the UI	5	Dylan	0.3	0.1
7	Minor	Member Table	Create Queries to Delete Member	Simple query to delete member, will be triggered by visual fields on the UI	5	Dylan	0.3	0.2
8	Minor	Member Table	Create Queries to Modify Member	Simple query to modify member, will be triggered by visual fields on the UI	5-6	Dylan	0.3	0.2
15	Major	Add/Mod New Member UI	Input Fields for All Member Details	Text checking on input fields to minimise the chance of invlid data being entered	5-6	Rabya	4	3.4
16	Major	Add/Mod New Member UI	Confirm Button	Button to confirm the member details, checks the inputs then sends it to the member table	15	Nic	2	3
17	Critical	Add/Mod New Member UI	Autoincrementing Member ID	When the confirm button is pressed the member ID is automatically generated and added to the database	16	Rabya	2	1.2
18	Major	Add/Mod New Member UI	Viewport to View Members	Snapshot of the member table, needs to be able to be filtered by search interface	5	Rabya	2	3.2
19	Major	Add/Mod New Member UI	Search Input Field	By typing member ID in and confirming, the viewport will display the member searched for (or nothing if no results found)	18	Simon	2	2.1
20	Major	Add/Mod New Member UI	Modify a Member Record	Selecting the searched member result will populate the text fields with saved data. Writing over these with new information and confirming will save over the old record with the new information	8, 15-19	Simon	4	4.2

***Figure 2:** Actual time taken for each product of the Add Member Interface.*

The sum of the actual time spent is 17.9 hours, which is slightly greater than the predicted time, but still extremely close to the estimate. Close enough to consider the estimate accurate by all definitions, especially in software which can fluctuate wildly.

Discussion

As mentioned, software engineering is notorious for inconsistent time estimations, mainly due to issues in problem definition, scope creep and obviously unforeseen debugging problems which can eat up large amounts of time. Fortunately, the Add Member Interface is only a single backlog item, meaning we are already working with reduced complexity. With that in mind, there are a few reasons that this particular prediction was so successful.

First off, the software implementation we are working on is fundamentally easy with most team members being very comfortable that the challenges that will arise will be within their understanding. This keeps the time down as it reduces time spent researching or searching for working examples on the internet.

The other most successful factor comes from the planning phase. For this project, the planning stage was extensive, allowing plenty of time for the team to get their heads around the vision for each page. This is essential in keeping the estimations accurate as it directly counteracts things like scope creep and problem definition. By clearly defining the end goals of the product with very little room for expansion, it is much easier to quantify the steps needed to reach the end. This goes hand in hand with breaking the product down into as many small pieces as possible. The increase in specificity significantly helps as it allows each member to easily visualise what the task entails and use their prior experience to make a judgement.

In an example, instead of leaving a single item called SQL queries, every SQL query was specified so that the team member making them could accurately see how many had to be made and generically the type they were (SELECT, INSERT, etc.)

In the end, all of the SQL queries ended up being faster than anticipated, which leads into a discussion about the inaccuracies with the estimation: Though overall the time tallied to the same number, individually there are discrepancies in the items.

The SQL components are simply explained by erring on the side of caution. The team was confident they would be less than 10-minute jobs, but they were rounded to 20 mins, as it's always better to overestimate than underestimate.

The largest of the discrepancies came from the Confirm Button the Input Field items, which took than an hour longer and 40 minutes shorter than budgeted time respectively. The reason I bring up both of these is because they are directly correlated. Here, the input field was expected to involve coding input sanitisation and error checking, however as the project progressed, limitations in knowledge and the software meant that it was much easier to implement all actionable code on a single button press, the natural choice being the Confirm Button. This left the Confirm Button with an extra section of code that was not planned for in the estimation and left the Input Fields as very simple UI components leading to one taking much less time and one taking much more.

This was a good case for agile development as time was basically rerouted from one task to another, which means that overall, the time estimate was sound, however, to keep an eye out for this in future the code itself could be itemised as a backlog item. This would mean that regardless of where it ends up being attached to, the time taken to write it is still accounted for and clearly shown for the relevant team member.

That last discrepancy to discuss is a bit more classic. The Viewport item took more than an hour overbudget but did not have the same fortunate correlation as the items above. Simply, this item was more complicated than the team thought, and extra time was needed to make it work. This was due to the program being used for the UI component (Ignition) and the unfamiliarity of the relevant team member with it. Though the SQL was working, there were unforeseen syntactical issues in trying to make the UI interface with the code and actually display the table snippet.

Ultimately, the overall time budgeting was extremely accurate (within 5%), however a range of reasons contributed to minor discrepancies within each product item.