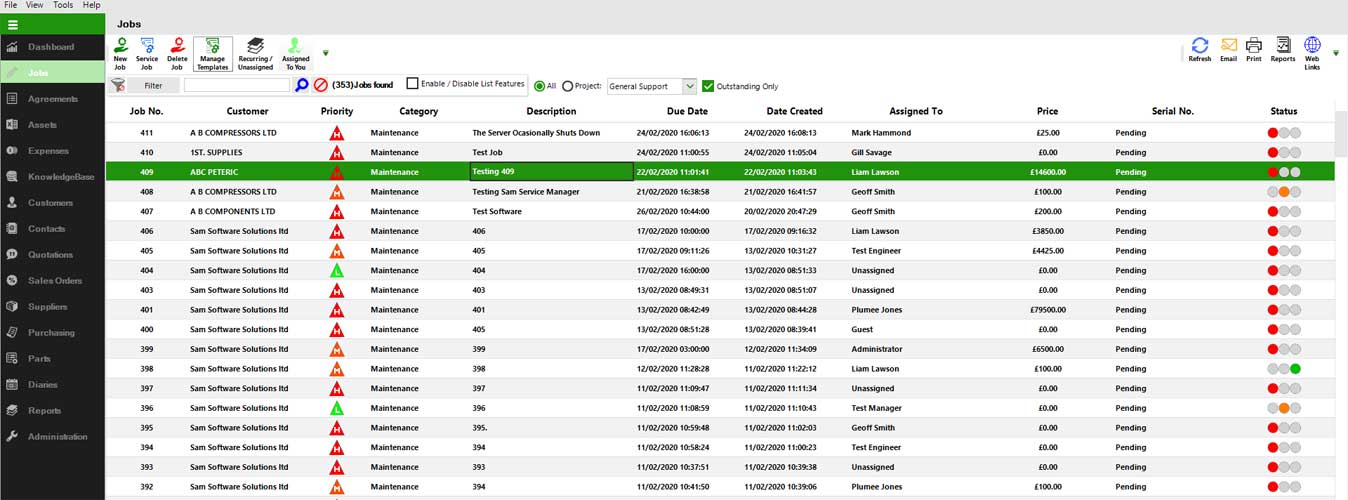
**1 – RESEARCH**

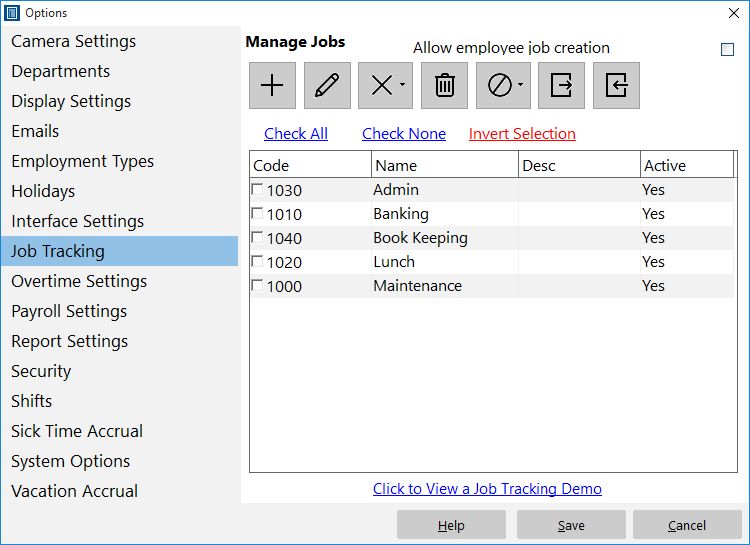
Before even planning my program, I needed to research similar programs to get an idea of how they function / look. The type of program that I will be creating is a job management program for a mobile tech repair/support company. The way the program should function is outlined in the brief, however there is no suggestion as to how it should look. A lot of similar programs require you to register for a trial, involving entering credit card information, so I instead searched for images of interfaces because I already know how I want the program to function, I just need to get an idea of how the interface should look.

**Interface 1:**



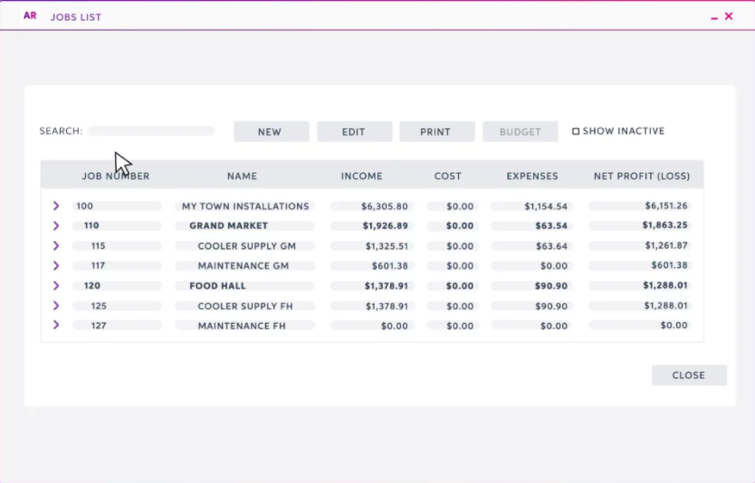
This interface shows a list of jobs that a user has entered. It shows the job number, who the job is for, its priority, description, date it is due, when the job was entered, who in the company will be doing the job, how much it costs, progress, etc. This makes the interface quite cramped and complicated. Because this business is only run by one person and jobs won’t be “in progress” – Suzy goes and completes the job in one visit – there is not a need for all this. I want to keep my interface as simple and easy to use as possible. There are buttons up the top to add/edit/delete jobs, but they are very small and hard for the user to find amongst all the other things.

**Interface 2:**



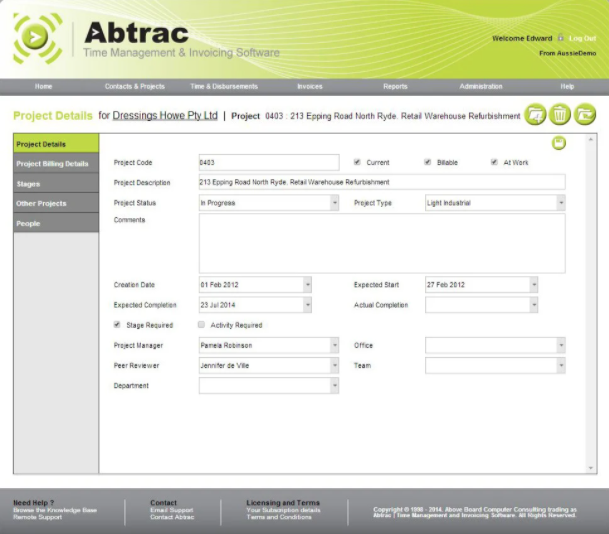
This interface is a lot simpler, and likely a bit closer to what mine will look like. It is not too crowded and the icons/buttons etc. fit in with the computer’s interface. The sidebar allows the user to switch to different screens, which is something I could possibly implement in my program. In this interface the user can edit jobs while on the job list screen, which I’m not sure if I want in my program.

**Interface 3:**



This interface is really good because it is simple and clear to the user what they can do. There is a list of jobs, they can search, create a new one, edit the jobs, and print. Not all of the functionality will be relevant to me – you cannot print from python tkinter – but this is a good interface to reference.

**Interface 4:**



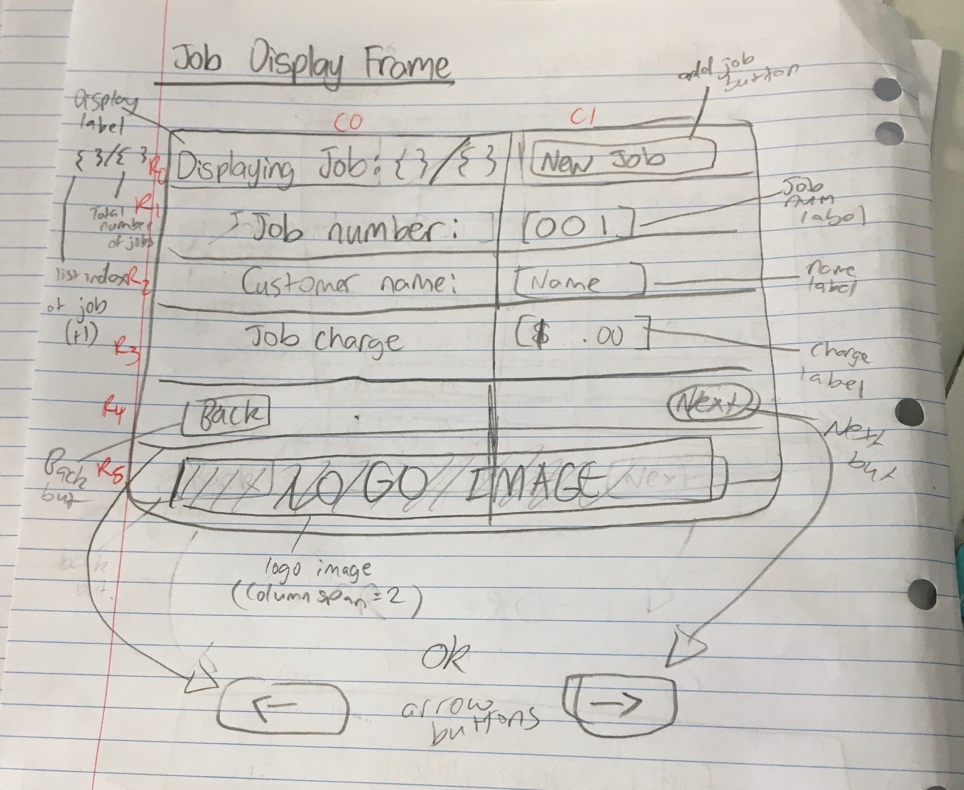
This interface shows the job entry part rather than the displaying jobs part of the program. There are lots of entry boxes, but also some checkboxes and drop-downs to limit user entry error. In my program I should try and use things like this as much as possible (without making the interface too clogged and confusing for the user) to prevent errors. ­­ However not all of these entries will be relevant for my program. I will only include the things outlined in the brief. So, for example, there can be a checkbox for “WOF and tune” rather than the user having to put that in an entry box.

**2 – PLANNING**

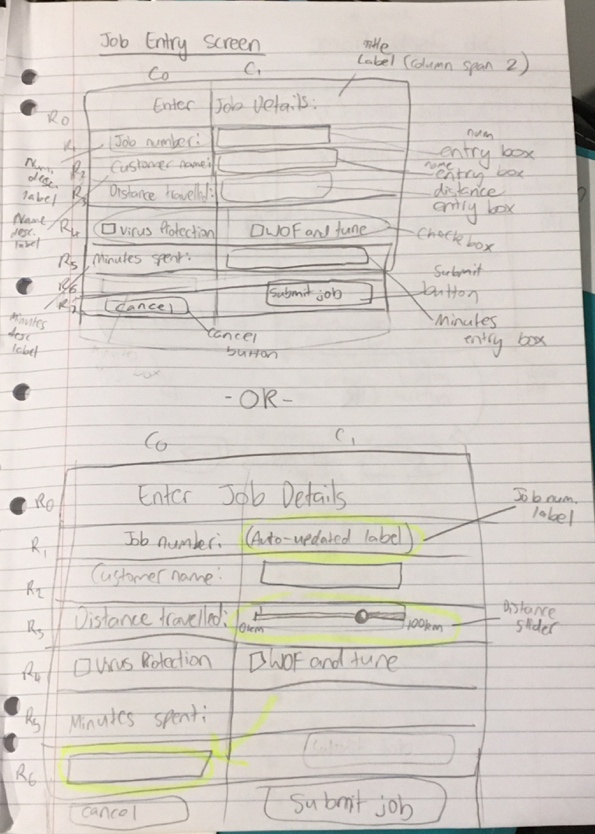
Now that I had conducted some research, it was time to start planning my program.

**Layout Diagrams**

First, I planned how I wanted my program to be laid out based on the specifications from the brief using a layout diagram.



This rough diagram shows what I want the job display frame to look like, and where I want different labels and buttons to be positioned on the screen.

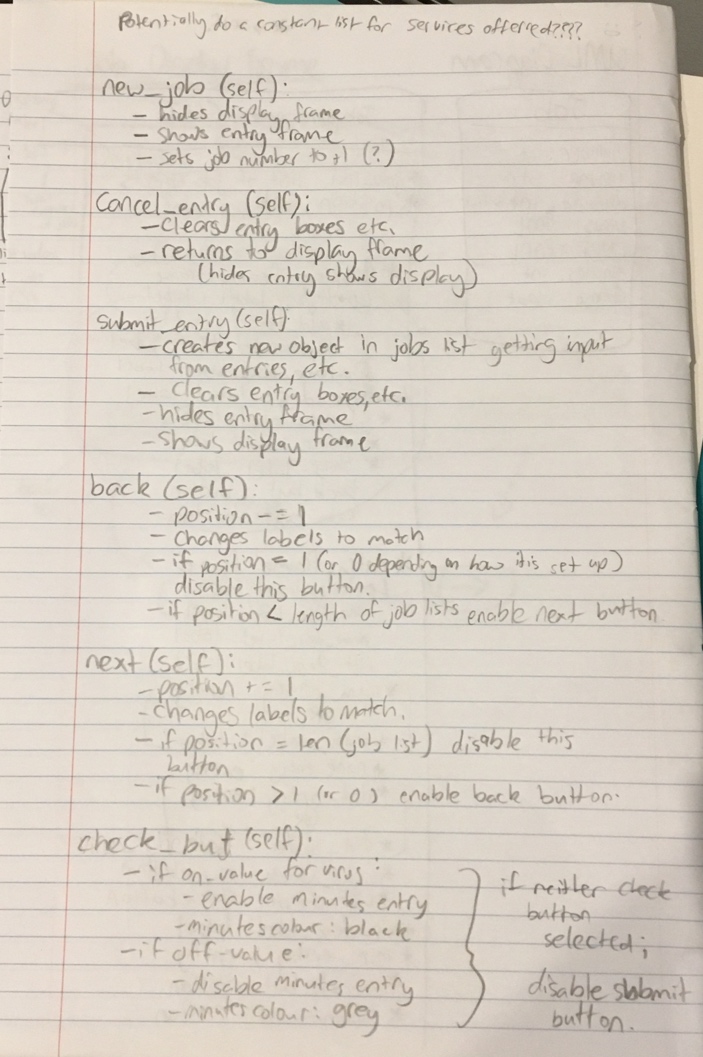
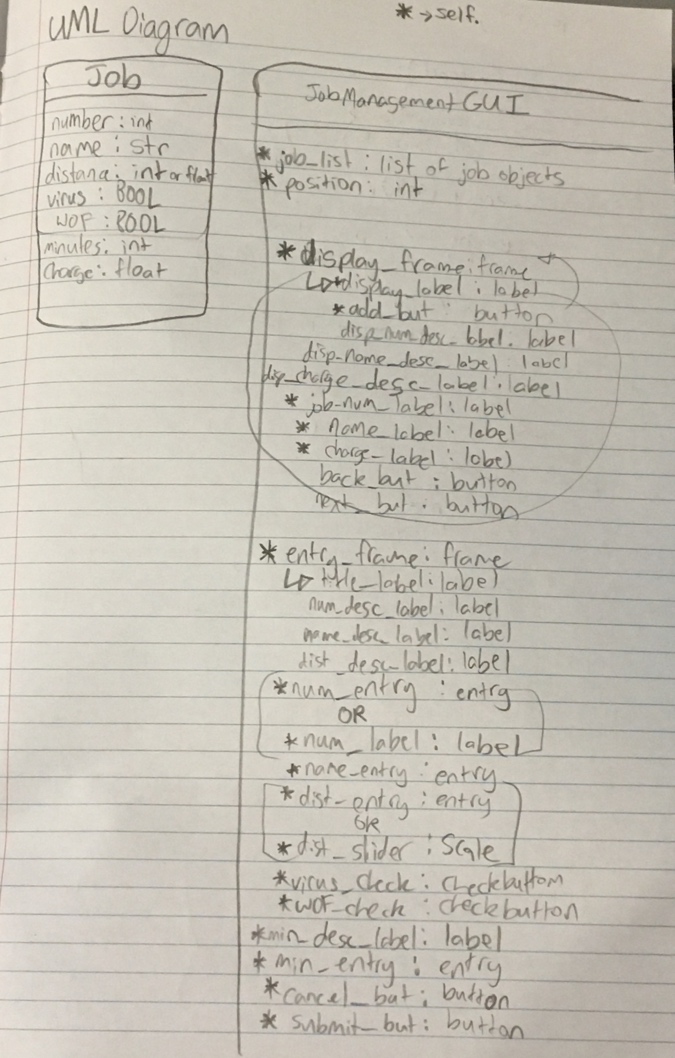
For the entry frame, I made an alternate version of the diagram with a few changes – an auto updating label instead of an entry box for the job number, a slider scale instead of an entry box for the distance travelled, and the positioning of the minutes spent entry box.

I then asked which version testers preferred, so that I could get an idea of what people wanted from my program. The second version of the diagram with the auto updating label, slider, and moved entry box was the preferred option – it was said to be more user friendly and clear.

Now I know, when I code my program, to go off of the second layout diagram.

**UML Diagram**

This diagram will help me create the classes that I need for my program. In this diagram I outlined the two classes that I plan to have – the GUI class and a jobs support class – and their instance variables and methods.

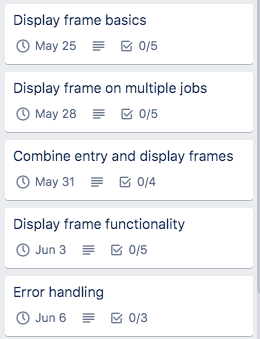


**Decomposing the Program**

Because this is quite a complex program, I need to break it down into smaller components so I can focus on programming one thing at a time and getting each part to work before combining everything into one final working program.

1. Entry frame basics  
   - Get the entry frame working on expected input by printing jobs into the shell.
   1. Set up layout
   2. Get customer name and distance travelled working
   3. Get checkboxes working
   4. Get minutes entry working
   5. Get auto updating job number working
   6. Get charge calculation working
2. Display frame basics  
   - Get the display frame working on ONE hard-coded example (buttons should not be working at this point)
   1. Set up layout
   2. Add in logo image
   3. Get display labels to work
3. Display frame on multiple jobs  
   - Get the display frame to cycle through MULTIPLE hard-coded example jobs
   1. Add next button functionality
   2. Add back button functionality
   3. Ensure display label (e.g. job ¾) is updating correctly
4. Combine entry and display frames  
   - Lets the user switch between the two frames by pressing a button
   1. Add functionality to the cancel and submit buttons – hides entry frame and shows display frame
   2. Add functionality to the new job button – hides display frame and shows entry frame
5. Display frame functionality  
   - No more hard-coded examples and printing to the shell. Jobs entered in the entry frame should appear in the display frame.
   1. Remove hard coded jobs and printing code
   2. Save entered objects to a list when submit is pressed
   3. Get display labels to update based on what is in the list
6. Error handling  
   - Making sure that the user cannot break the program
   1. Boundary error handling
   2. Exceptional error handling

After planning the decomposition, I put these into cards on trello.



I put due dates on each of these to help with my time management. Here is a screenshot of part of the calendar showing when these different components are due:

**Considering Implications**

There are some implications that I need to address throughout the creation of my program:

* Functionality
  + It is important that when you have a program it does what it is meant to do. No matter how good something looks, it is useless if it does not function as intended. I need to make sure that my program functions as outlined by the brief I was given so Suzy can use it for her business.
* Usability
  + Users should be able to navigate the program and use it without asking for help. The program should be designed in such a way that makes sense to users and prevents them from making errors as much as possible.
* Accessibility
  + I need to ensure that Suzy is able to access the program. I will use language that is user friendly and relevant to her business and find a way to allow her to use the program without having to download python.

**Test Plan**

|  |  |
| --- | --- |
| **Test** | **Expected Results** |
| Name Entry | |
| [Any name] | It should go through no problems – it is a string input so anything you enter aside from leaving it blank should be accepted |
| Leave it blank | It should not allow you to submit the job |
| Distance Entry | |
| 1km | All of these are valid entries, the slider should not allow any values that are not valid. |
| 56km (or any random middle amount) |
| 99km |
| 100km |
| Checkboxes | |
| Ticking only virus | Should enable the minutes spent entry box. |
| Ticking only WOF | No visible change, but saves that it is a WOF job |
| Ticking both | Should enable the minutes spent entry box. |
| Ticking neither | Should not allow the job to be submitted. |
| Minutes Spent | |
| 0 | Should not allow the job to be submitted. |
| 1 | Will be a valid option and will be saved to the job |
| 37 | Will be a valid option and will be saved to the job |
| Any text e.g. “one” | Should not allow the job to be submitted. |
| Leaving it blank | Should not allow the job to be submitted. |
| 3.7 (or any float value) | Should not allow the job to be submitted. |
| -56 | Should not allow the job to be submitted. |
| Cancel and Submit | |
| Press the cancel button | Will clear all of the fields and take the user back to the display frame |
| Press the submit button | Will do the same, but the job entered should now show up on the display frame |
| Charge amounts | |
| Just WOF and tune  Distances: |  |
| 1km  3km (or any in between value)  5km | 110.00 |
| 6km | 110.50 |
| 46km | 130.50 |
| 99km | 157.00 |
| 100km | 157.50 |
| Virus only (Distance set at 1km)  Minutes: |  |
| 1 | 10.80 |
| 7 | 15.60 |
| 50 | 50.00 |
| 129 | 113.20 |
| (For both WOF and virus, just add 100 to these charges) | |
| Distance – 20km  Virus – 38 minutes  WOF – yes | 147.90 |
| Next and Back buttons | |
| Press the next button until you reach the end of the list | Next button should disable until you press the back button |
| Press the back button until you reach the start of the list | Back button should disable until you press the next button |

**3 – Developing the Components**

With the planning done it was now time to start developing the program. The development process will be documented in the Testing and Evidence Log document.