**Testing and Evidence Log**

|  |  |  |
| --- | --- | --- |
| Component 1 – Setting up Entry Frame | | |
| **DATE** | **WHAT I DID** | **EVIDENCE** |
| 20/5/21 | Test plan for this component:  All the aspects of the job should print into the shell when submit is pressed.  Other than that it should behave (on expected input only) as the final program, so the final test plan (specifically everything but the next and back display testing).  These screenshots show the testing plan, but with the boundary and exceptional tests crossed out because they are not relevant at this point. |  |
| 20/5/21 | For the very first version of my program (just the entry frame) I created all the widgets that I would need for this frame and gridded them where they will be placed. I have not added any variables or functionality to any of them, and there still is a lot of formatting to do, but this will be the base for my program. |  |
| 20/5/21 | Next, I got the customer name and distance travelled to print to the shell. These were grouped together because they are quite easy to get working. I added variables for them, linked the variables to the widgets, then set a command to the submit button that prints the value of these variables when pressed. |  |
| 20/5/21 | Next, I got the check buttons to also work and be printed when submit is pressed. When the submit button is pressed, if the variable for their respective variable is set to the on value, it will print out that service. When only virus was ticked it only printed that, vice versa for WOF, when neither were ticked it did not print anything, and when they were both ticked it printed both. |  |
| 20/5/21 | Finally, the last thing I did in this version was make it so that the minutes entry printed. I added a command to the virus checkbox so that you can only type in the entry box when that checkbox has been ticked. If the checkbox is unticked but there is still a value in the entry box, it will clear the entry box, so it does not print anyway. At the moment for it to work with the entry box the minutes variable is a string var. This will need to be changed when it is time to do calculations with the minutes value. |  |
| 20/5/21 | In this version I was saving the inputs into an object and calculating the charge. As I was testing that it was working I came across an error message. I had both WOF and virus ticked, but when I unticked WOF so only virus was selected this error popped up. I traced the error back to its line and realised it was because I had put the wrong variable (the virus one, when it should be the WOF one) in the else statement. I fixed it and the error went away.  Here is the code I changed and a screenshot showing a few of the tests I did. One of the charge numbers has a lot of decimal places but python does that sometimes and it won’t affect it when it is rounded to 2dp (because it is money). I added some constants to do with the charge amounts, a Job class, a method for calculating the cost, and changed the submit method so that it creates a job object and prints the variables directly from that object, so I know it has saved properly.  Next I need to get the job number to iterate. |  |
| 20/5/21 | I added code to the end of the submit method so that the job number would iterate. Shown are screenshots of some testing examples with the job number iterating, and also one of the GUI showing that it also updates on the label. |  |
| 20/5/21 | Finally, before testing, I made some minor formatting changes to the frame. I spaced everything out a little bit, so it is easier for the user to read and see which label corresponds to which widget. I also made it so that the minutes spent description label greys out whenever virus protection isn’t ticked so the user knows they can’t enter values into it. |  |
| 20/5/21 | I tested that everything in this component was working before I moved on to the next component. I tried a variety of distances, minutes, and combinations of WOF/virus, and nothing broke. This means that I can continue and start on the next component. | Evidence is in the testing videos folder called “testing component 1” . |
| 20/5/21 | With that, component 1 was finished! I was now running a couple days ahead of schedule. Here are screenshots of my trello board / calendar that show my progress so far. |  |
| Component 2 – Setting up Display Frame | | |
| 21/5/21 | Test plan for this component:  For this component I want a display frame to work on one hard-coded example job. | Test plan:   * Title should say 1/1 * Job number should be 1 * Customer name will be Olivia * Distance will be 20, virus yes, minutes 38, and WOF yes, so **charge will display as $147.90** * Logo image should appear at the bottom * Buttons don’t need to be working yet |
| 21/5/21 | I set up the GUI for the display frame and put in the hard-coded details. There were no problems with the actual creation of the GUI or the displaying of the details on the labels, however there was a problem with the image of the logo. The logo is much wider than I want it to be and is stretching out the window for the program. My next step is to figure out how to resize that logo. |  |
| 21/5/21 | First, I tried to change the width in the label directly. However, this ended up cutting off the image.  After some research online, the only easy way to resize an image inside of python was to import another plug-in. So, I instead opted to resizing the actual file to a smaller image. This also meant the file size was not as big. |  |
| 23/5/21 | Before I tested this component, I wanted to do some formatting to make the output more readable, and also just to make things nice and balanced and more like what I envisioned when I drew the layout diagram. I added padding and changed the “sticky” alignment of a few widgets. |  |
| 23/5/21 | For testing, there was no need to take a video because it is just one lot of output that does not change (yet), and the buttons do not do anything (yet). I went through the test plan and ticked off all the things I had said I wanted to display. Everything was as expected, meaning I can now move on to the next component!  The next thing to work on will be making the display frame show multiple jobs and scrolling through them with back and next. As seen on the calendar, I am still running ahead of schedule. |  |
| Component 3 – Display Frame to show Multiple Jobs | | |
| 24/5/21 | Test plan for this component:  I will add three more example jobs into the program and get the next and back buttons to cycle through them.  This test plan demonstrates how I want it to work – a next and back button that change what is displayed on the labels, and they disable when you cannot go any further. | * Press next button   + Display labels should change   + Title label should say 2/4   + Description labels, image, buttons, and general layout should stay consistent * Press back button   + Should go back to the first and be identical to how it was before * Press the next button until you reach the end of the list   + Next button should disable until you press the back button   + Each time the labels should update and show the unique job * Press the back button until you reach the start of the list   + Back button should disable until you press the next button   + Going back over a job you have scrolled through, nothing should have changed on an individual job. |
| 24/5/21 | First, I added the other hard coded jobs. Then, I set up the functionality for the next button. The next button itself now has the next method as a command. When it is pressed the position, counter is added to and the labels updates. However, I have not yet made it so that it disables when you reach the end of the list. When I pressed next after reaching 4/4, I got error messages, and the job display counter kept on going up without the number of jobs changing (e.g. 8/4). |  |
| 24/5/21 | Next, I added a method that would check the position variable and disable/enable the next or back button depending on where you are in the list. I also added back button functionality, which was the same as the next button, but the position subtracts. It worked, but I found a few problems with this. Firstly, the back button only disables when you press the next button and then go back. When you first start the program, you can just keep on going back. I need to make the button disable from the start. Also, there is a lot of repeated code (back method is almost identical to next method. I will fix this by adding the repeated code in the next and back button methods to the check position method, which is already called in both of them. |  |
| 24/5/21 | I fixed all the issues outlined in the previous log entry. I set the back button to be disabled by default and moved the repeated code to the check position method (and renamed it to include the updating labels part). Now the buttons were working as I wanted them to (as shown in the screenshots). |  |
| 24/5/21 | I went through and tested everything that was in the test plan on my program. Everything worked as I wanted it to.  With that, I was finished with this component. The next component is to combine the entry and display frames into one program (right now they are on separate python files). According to the calendar, I am still ahead of schedule. | Testing evidence is in the testing videos folder, called “testing component 3”. |
| Component 4 – Combining Entry and Display Frames | | |
| 27/5/21 | Test plan for this component:  The plan component is quite simple, as not much is changing. It is just the combination of two already working parts. I just need to make sure that you can switch between the frames easily, and that nothing that was working before is broken. | * Pressing new job on the display frame   + Will hide the display frame and show the entry frame. * Pressing cancel on the entry frame   + Will clear all the entry fields, hide the entry frame, and show the display frame. * Pressing both buttons consecutively and repetitively should take you back and forth without any problems. * Refer to previous test plans to make sure nothing that was previously working is broken. |
| 27/5/21 | I copied the code from the display and entry frames into one python file, then created a method that would switch from the display frame to the entry frame when the New Job button was pressed. As you can see, the next job ID is 5 because there are already 4 example jobs. |  |
| 27/5/21 | I added a function that would act similarly to the new\_job one, hiding the entry frame and showing the display frame. However, I also wanted it to clear all the entry fields back to the default if the user had typed in them. This worked, as shown in the screenshots to the right, but the minutes spent entry field was not disabled despite minutes spent not being ticked. This will need to be fixed. |  |
| 27/5/21 | Since when the cancel button is pressed it’s always going to set virus protection to its off value, I just added a line to cancel\_entry that disables the minutes entry because there should be no situation where it doesn’t need to be disabled when that button is pressed. |  |
| 27/5/21 | I tested this component, going back and forth between the frames and testing the entry and the next and back buttons on the display. However, when I was testing, I noticed something that was a problem. When I entered a new job and returned to the display frame at first it said 1/4, but when I pressed next and the labels updated it said 2/5. This means I need to update the labels when I return to the display frame. | Testing evidence is in the testing videos folder, called “testing component 4”. |
| 27/5/21 | To fix the problem that I found while testing, I called the check position and update method in the cancel entry method. This updates all the labels, so it shows how many jobs there are when you return to the display frame. I also added a .title to the code that creates the job object so that names always display with a capital letter at the start of each word. |  |
| 27/5/21 | I tested again now that I had made these changes, and everything worked as I wanted it to. I was now done with this component and could move on.  According to the trello board, next up was to add proper functionality to the display frame (i.e. no example jobs), and the calendar showed I was still ahead of schedule. | Testing evidence is in the testing videos folder, called “testing component 4 (version 2)”. |
| Component 5 – Display frame functionality (no hard coding jobs) | | |
| 28/5/21 | Test plan for this component: | * Program starts, there are no jobs that have been entered so there should be a label that says this on the display frame, and the display labels should be hidden. * Once a job has been entered the “no jobs entered” label should be hidden and the display labels should show and everything should work the same as it did when the jobs were hard coded. [from this point on the test plan from component 3 can be used to make sure buttons are still working as expected] |
| 28/5/21 | I removed the code that made the display labels display jobs upon creation (e.g. charge text used to be ${:.2f}, but now it is only set to that in check\_pos\_update, and is just set to “” upon creation). I called the check\_pos\_update method in the init, because I added some code to it that means if there are no jobs in the job list it will hide the job display labels and show the “there are no jobs entered” label, and if not it will hide that label and show the others. I put this code in that function inside of the if/else statement so that if I want to add the option to delete jobs later, that can display if all jobs in the system have been deleted. Next I need to adjust what happens when you press submit on the entry frame. | (first starting up the program) |
| 29/5/21 | I added a label to the bottom of the entry frame that displays when a job has been entered so that the user knows their submission has gone through. This is configured when the submit button is pressed. Pressing the submit button also clears all the entry fields and unticks the checkboxes. Because both the cancel and submit buttons need to do this, and because I wanted to avoid repeating code, I made a method dedicated solely to clearing the fields and unticking the checkboxes. |  |
| 29/5/21 | I tested the changes that I had made in this component, and everything was still working properly.  According to my trello board, the next thing to work on was error handling. I was over a week ahead on the calendar. So, I decided to move the due date for the error handling to a couple of days earlier. I want to use the extra time that I have to make the program as good as possible, and a major part of that is working on user feedback. I think that I do not need more than a week on error handling, and that I can put that time into acting on user feedback – which often picks up errors that I don’t even know about anyway. | Testing evidence is in the testing videos folder, called “testing component 5”. |
| Component 6 – Error Handling | | |
| 1/6/21 | After this component I will be able to use the general test plan written before I started coding, so I do not need to write a test plan, however now that I know what my program looks and functions like I would like to create a more specific plan as to what errors I will be handling. | * Make sure customer name isn’t empty * Don’t let distance slider go to 0 km * Either virus or wof check button must be ticked * Minutes spent must be a numerical value ABOVE 0 (if virus has been ticked) |
| 1/6/21 | First, I added the more simple error handling – the customer name and the distance slider. For the slider, I just had to specify the value that I wanted it to start from, so it went from 1 instead of the default 0. This will stop users from entering an invalid distance, as they can’t go below 1. The second thing I did was add an if/else statement to the submit method that means if the submit button is pressed but customer name is empty it does not submit the job, it instead prompts the user to enter the name. If they enter something into the entry it submits as normal. |  |
| 2/6/21 | There are three class periods left to work on this internal and I am halfway through my error handling (I changed the checklist on the card as shown to the right to reflect my actual progress with the error handling). I think I am on track to finish by the hand-in date. |  |
| 2/6/21 | There were quite a few error messages I had to fix here, so I have put some notes in the evidence column to help make it clear what the screenshots are showing. I added some if/else statements to make sure at least either WOF or Virus Protection was ticked, and also call a function when virus protection is selected to make sure the minutes entered is an int. However, I ran into some problems as the code that I had made would not submit the job if only WOF had been selected, because the submit code was under an if branch for the minute’s validity, and this could only be worked around by repeating code. So, I moved the code that actually submits (creates a job object and prints for testing) to another method and I called it after the minutes were valid or if only WOF was ticked. However, this also caused error messages because there were some local variables in the submit code. This was a quick fix, as I just set the local variables required as parameters for that method. After this I ran into one more error – when I ticked only minutes and entered a valid number an error message appeared. This is just because I forgot to set min\_number to the actual minutes value if virus is ticked. I fixed this and it worked, but then I noticed that I do not actually need the min\_number variable. I removed all mentions of the min\_number variable from the “submit” method, however it was still needed in “actualsubmit” (I will rename these methods to better things when I am done) because that is where the int version of the minutes is saved. I tested different valid and invalid entries after this and did not run into any error messages or bugs. | This does not submit the job if only WOF has been selected.    This code does not work because the submission code which has been moved to another method works on local variables.    This code works as expected at first, but then….  I got an error message after entering a valid minutes entryThis code finally worked as I wanted.  This is the check\_int method.  My reasoning showing why I don’t need the min\_number variable.    I took out min\_number from this submit method  But I still needed it here because I needed to convert the minutes into an integer to calculate cost  And everything still worked well (here are successfully submitted jobs with all valid combinations of WOF and Virus Prot. – basically anything except for False False). |
| 2/6/21 | I added the error handling that would make sure that the integer the user entered is over 0. I updated the error message as well, to reflect this change. So now, in addition to the error message showing up when strings and decimals are entered, anything below 1 will also trigger the warning message. This meant I was now done with all the planned aspects of the error handling. |  |
| 4/6/21 | I tested the error handling that I had done using the test plan I wrote before the coding of the program (this can be found in the planning and processes document). I tested missing out the name, different distance options, missing out checkboxes, different minute inputs, and a variety of distance/minutes/WOF combinations to check that the charge was calculating as it needed to. Everything worked as the testing plan stated, so I removed the printing code from the submit method so it is ready for user trialling. With this, the erorr handling component was finished and it was time to move on to final testing. I had already done the first run through of the test plan and there were no changes needed, so now I have the next 4 days to get end user feedback and change/test based on that. | Testing evidence is in the testing videos file, there is the full “testing error handled component 6” or the sped up “SPED UP component 6 testing”, as this test took around 4 minutes. |
| 6/6/21 | I asked some users to trial my program and give any feedback on what worked and what didn’t, and what they think should be changed. I took note of the changes that they suggested and ended up with 5 things that I needed to change. Both users wanted there to be a screen where you could see all the jobs that you had entered, and user 2 also wanted this to display some totals – total number of jobs, total money charged, total distance travelled. User 1 pointed out some changes that should be made to the checkbox error message and the cancel button, and User 2 wanted to go back and edit jobs as well as a type-in entry box linked to the distance slider to get an exact value easily.­­­ | I added a card with a checklist to keep track of the changes that I was making. |
| 6/6/21 | First, I made the changes to do with the entry frame, because I considered these the easiest/fastest to change. The first two changes were really simple – just changing the wording on the confirmation label configuration and the cancel button’s text to make it less confusing for the user. Then, for the next change, I had to add another entry widget. I sketched up a quick layout diagram to figure out the new column grid numbers, then added the entry box. Then I did some formatting to remove the number on top of the scale, make the entry smaller, and set the entry’s textvariable to the same variable as the slider so they were linked (when I change the number in the slider it changes in the entry and vice versa). I expected to have to do some extra error handling at this point, however the entry actually handled the IntVar very well. I put a temporary print statement in the submit method to see what it would do when I entered exceptional values. When I entered a negative it set it back to one, when I typed in letters it just stayed at the last valid number, when I went over the 100 limit it set it to 100, and when I put in decimals it automatically rounded them. |  |
| 6/6/21 | The next change to be made was to let the user edit jobs after submitting them, for example if a customer’s details changed or there was a mistake made inputting. I added two buttons to the display frame – an edit button, and a view all button. The view all button will have functionality added in the next change and is only there for layout right now. When the edit button is pressed it calls a method that brings up the entry frame like normal but sets it to an edit mode with a different submission button, and instead of the entry fields being cleared they are filled in with the details of that job. Like when submitting a new job, it calls the check\_job method, which now calls either submit or update methods depending on what “mode” the program is in, so that the user can’t make errors and input invalid things when they are editing. The submission should still work as it did before – editing should not affect the functionality or output of the new job functionality. Through my testing this seemed to be working mostly as expected, except for the fact that on the entry frame when pressing new job after editing a job the job number label showed as the number of the job you were just editing instead of the next id. This does not affect the actual new job submission, it just doesn’t update until you press submit job. I’ll have to fix this by adding a line of code that updates the job number label when new job is pressed. | Modified code in the check\_job method  Modified code in the new\_job method  Two new methods  GUI code    Making and editing a job  The latest job is 1But the entry label shows the next job as 1 |
| 6/6/21 | Since the number label is updated to the current job after calling new\_job if the user is editing, and otherwise being left the same, I added a line to the new\_job method that configures the label to display the next\_id (what it usually displays). This came up with an error message. This was a really easy fix, however, as I realised that I just forgot the “text =” part of the configure statement. After this everything seemed to be working as I wanted – as shown in the evidence I edited job number two when there were three jobs in total and after pressing new job the number that came up was 4. After this, the last change to do was to add the screen that showed all of the jobs. |  |
| 7/6/21 | I sketched the GUI layout for a summary frame that will show all of the jobs and allow the user to scroll through them, and also display some potentially useful totals below. I wrote the GUI set-up code on a separate python file to make sure that everything was laid out how I wanted it to be before putting it into the main program and adding the code to make it function. |  |
| 7/6/21 | I put the summary frame code into the main program and added methods that would switch between the summary and display frame using buttons, and also go through the list of jobs and display them all in the scrolledtext. I am yet to add the functionality to the total labels, so that is what I will do next. |  |
| 7/6/21 | I added the code that would update the labels on the summary frame to show the totals. To calculate the totals I just made variables that were set to 0, then for each job in the list after I put them in the display I would add the relevant number from that job to the variable so that at the end it had the totals from all the jobs, then I configured the labels to show these variables with any necessary formatting (e.g. charge with a dollar sign and 2dp). |  |
| 8/6/21 | Now that I had made all of these changes I ran through the test plan again to make sure that everything was working as I wanted and that the changes I made hadn’t caused any bugs. Because a lot of changes have been made to the program and the testing plan doesn’t really cover every aspect of the program anymore, this video has a voice over of me explaining what I am testing at each point. I did not find any more bugs or changes that needed to be made, so I was now done with the development of my program. I now had the rest of the week to complete the write-up and submit. For the rest of the write-up, refer back to the Planning and Processes document. | Video is in the testing videos folder, called “FINAL TESTING.mov” |