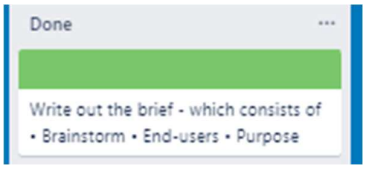
Brief:

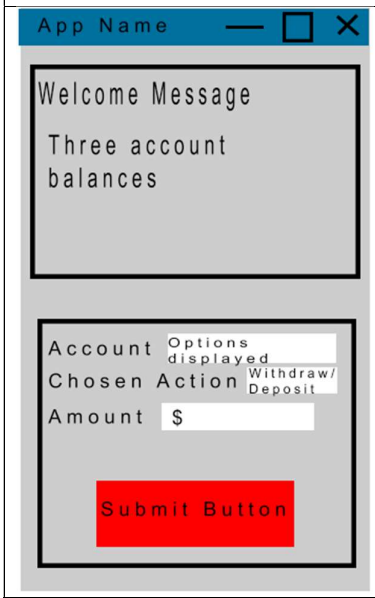
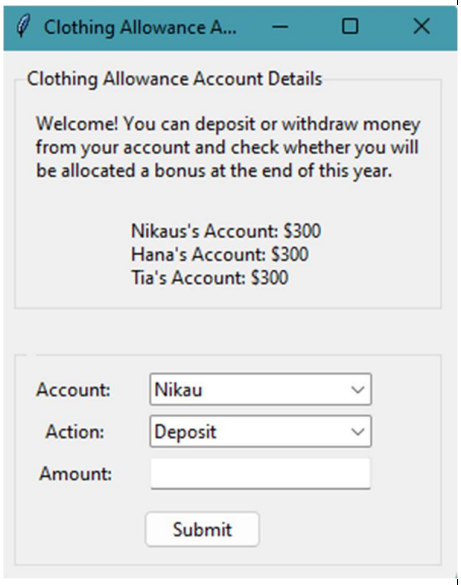
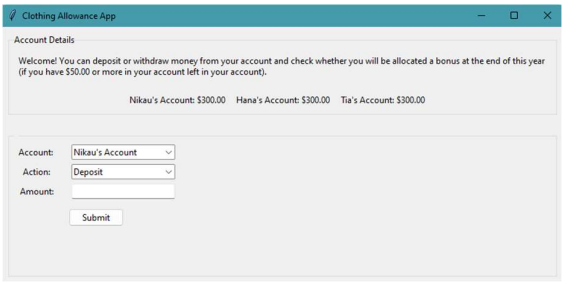
The task here is to create a GUI interface that will allow me to display the users' account names balance, transaction, and welcome messages as well as other relevant inputs/information. It was made for the purpose of the parents allowing them to see their children's account balances and provide them with bonuses.

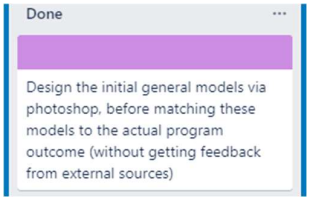
This specific task is to create a more user-friendly display that is appealing to the users, is set up in an aesthetically pleasing way, whilst being laid out in a sensible and logical manner. As I have been given basic specifications which I have ensured to include in the program design, the other components are largely left up to me to include. To make sure that the further specifications I include are well received by the users, I will trial and test these designs thoroughly with my peers, teacher, clients/stakeholders as well as other people with no programming understanding to ensure it can be easily used by the general population.

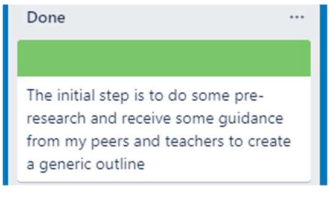


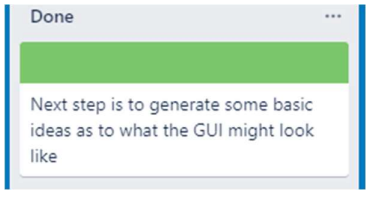
- This task is completed from the Trello Board

Initial Ideas Brainstormed:

Drawing Version	***First actual unmodified (without feedback versions)***	***Final actual unmodified (without feedback version)***
		 Upon further feedback the final version will be made based upon user reviews and data collected.
In the initial drawn up version, the app was made to be very narrow, and fit all the message in the parameters, given, the colours were to be neutral and not clashing as well as aesthetically pleasing.	In the actual version, the layout was more centred, and there was more width to fit in the welcome message as well as the additional heading details which weren't in the drawn-up version. The colour scheme remains relatively similar as this was the design before seeking external review from stakeholders, peers, family and friends, users, etc.	In the final actual version, the only changes were made to increase the width of the app screen, this was done in order to prevent the words from being too close together, as well as showing too many lines worth, as it looked unorganised and made it harder to read. The increase between each individual section was increased to make it easier to read. The submit button was not made read as it may be more difficult for those who are colour-blind to read it.





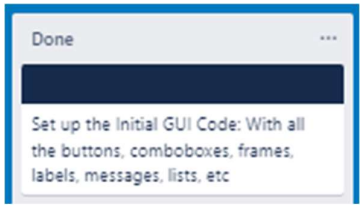


- These tasks are completed from the Trello Board

Setting Up of Initial GUI Code:

The image below is a screenshot of the initial GUI part of the code before carrying out the decomposition and gathering feedback. This is the generic model which contains all the comboboxes, labels, frames and messages. From this version, as the code is developed based on the feedback generated from the forms the code will be adapted and changed.

```
##****************************************** GUI CODE ***************************************************
root = Tk()
root.title("Clothing Allowance App")

#Create the top frame
top_frame = ttk.LabelFrame(root, text="Account Details")
top_frame.grid(row=0, column=0, padx=10, pady=10, sticky="NSEW")

#Create and set the message text variable
message_text = StringVar()
message_text.set("Welcome! You can deposit or withdraw money from your account and check whether you will be allocated a bonus at the end of this year \n(if you have $50.00 or more in your account left in your account).")

#Create and pack the message label
message_label = ttk.Label(top_frame, textvariable=message_text, wraplength=1000)
message_label.grid(row=0, column=0, columnspan=2, padx=10, pady=10)

#Create and set the account details variable
account_details = StringVar()

#Create the details label and pack it into the GUI
details_label = ttk.Label(top_frame, textvariable=account_details)
details_label.grid(row=2, column=0, columnspan=2, padx=10, pady=10)

#Create the botton frame
botton_frame = ttk.LabelFrame(root)
botton_frame.grid(row=1, column=0, padx=10, pady=10, sticky="NSEW")

#Create a label for the account combobox
account_label = ttk.Label(botton_frame, text="Account: ")
account_label.grid(row=3, column=0, padx=10, pady=3)

#Set up a variable and option list for the account Combobox
chosen_account = StringVar()
chosen_account.set(account_names[0])

#Create a Combobox to select the account
account_box = ttk.Combobox(botton_frame, textvariable=chosen_account, state="readonly")
account_box['values'] = account_names
account_box.grid(row=3, column=1, padx=10, pady=3, sticky="WE")

#Create a label for the action Combobox
action_label = ttk.Label(botton_frame, text="Action:")
action_label.grid(row=4, column=0)

#Set up a variable and option list for the action Combobox
action_list = ["Deposit", "Withdraw"]
chosen_action = StringVar()
chosen_action.set(action_list[0])

#Create the Combobox to select the action
action_box = ttk.Combobox(botton_frame, textvariable=chosen_action, state="readonly")
action_box['values'] = action_list
action_box.grid(row=4, column=1, padx=10, pady=3, sticky="WE")

#Create a label for the amount field and pack it into the GUI
amount_label = ttk.Label(botton_frame, text="Amount:")
amount_label.grid(row=5, column=0, padx=10, pady=3)

#Create a variable to store the amount
amount = DoubleVar()
amount.set("")

#Create an entry to type in amount
amount_entry = ttk.Entry(botton_frame, textvariable=amount)
amount_entry.grid(row=5, column=1, padx=10, pady=3, sticky="WE")

#Create a submit button
submit_button = ttk.Button(botton_frame, text="Submit", command=manage_action)
submit_button.grid(row=6, column=0, columnspan=2, padx=10, pady=10)
root.bind('<Return>', lambda event:manage_action())

#Create an action feedback label
action_feedback = StringVar()

action_feedback_label = ttk.Label(botton_frame, textvariable=action_feedback)
action_feedback_label.grid(row=7, column=0, columnspan=2)

#Create an allocated bonus message when accounts have more than $50 left in it
bonus_message = StringVar()

#Create a savings message when accounts have less than $50 left in it
savings_message = StringVar()

#Create and pack the bonus label
bonus_label = ttk.Label(botton_frame, textvariable=bonus_message, wraplength=1000)
bonus_label.grid(row=10, column=0, columnspan=4, padx=100, pady=10)

##*********************************************************************************************************
#Run the mainloop
update_balance()
root.mainloop()
##*********************************************************************************************************
```



- This task is completed from the Trello Board

GUI Trello Board:

Initial To-Do List (Before Anything is Done):

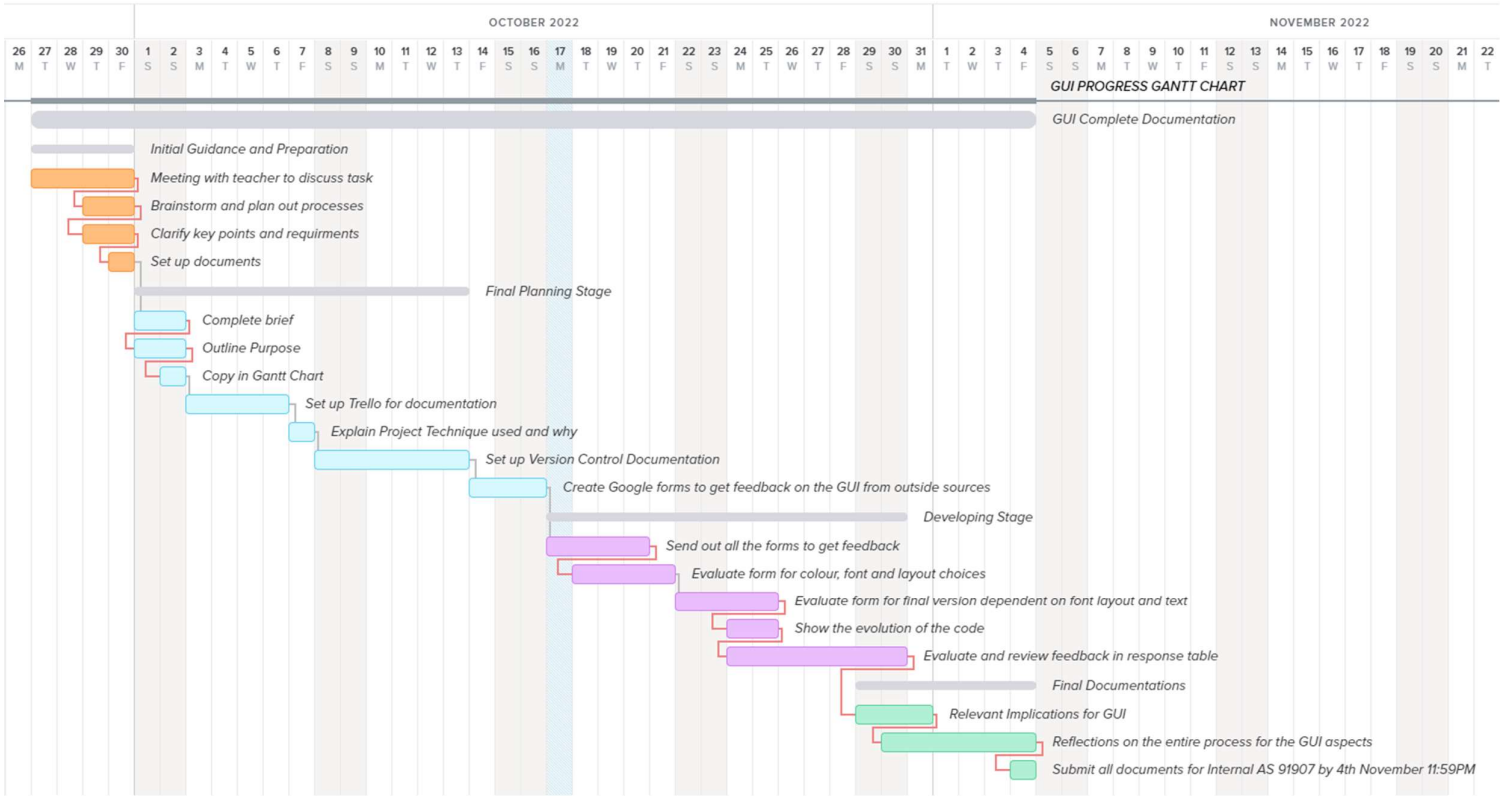
The Trello board below splits all the tasks left to be done to finish creating the final version of the GUI is split into four basic sections each of which have been colour coded, Planning (Green), Developing and Research (Orange), Design (Purple) and Writing the code (Black). This section comes under creating the generic outline and the initial step within the planning (alongside the Gantt charts shown below).

Throughout the document, as tasks are continued to be completed, they will move through sections from To do – Doing – Done, currently all of the tasks are in the To do section

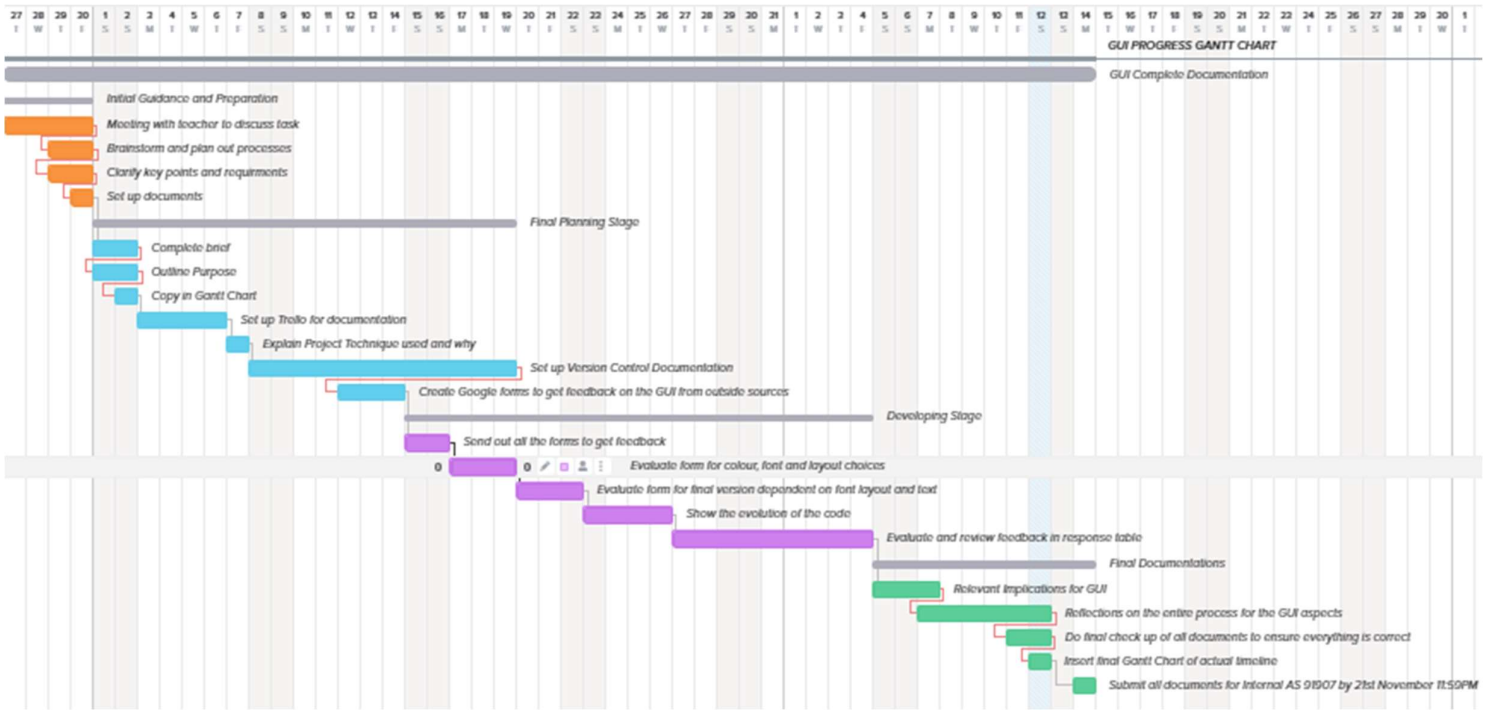
I will be using the Trello Board throughout my project to help me with appropriately planning out the tasks required for the project. The use of the Trello Board will thus make it easier to ensure every aspect of the project is completed as I must manually move specific tasks (essentially ticking off activities).

GUI Gantt Charts:

Initial GUI Gantt Chart:



Final GUI Gannt Chart:



Project Technique:

The project technique I've opted for is 'Waterfall'. The reason I've used the waterfall technique as opposed to agile or other ones, is due to sequential, linear process of this project management technique. It makes it easier to concentrate on one specific aspect of the project and work until it's completed before moving onto the next aspect. Whilst some tasks can be done at relatively similar times everything has to be done until completion in order. I feel like this is a better system as for most of the tasks the prior tasks must be completed before carrying on to the next part, for example with gathering feedback from stakeholders, peers, friends and my teacher this task cannot be completed before the forms itself have been created; and planning out the entire process must be done before I can start any task.

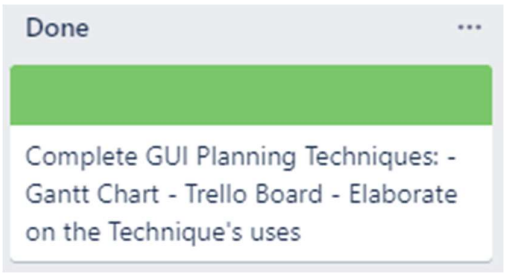
Due to the rigidness of the waterfall technique, a solid basis of planning must be completed well in advance to starting. I've done this by having multiple discussions with my teachers and peers regarding what the order of systems should look like and what they believe will be the best fit. I've created a generic document of notes on my planning, as well as the basic gist of what the structure should look like – based on this I've created the Gantt chart and started the GUI Interface project. As I work on completing the project, I will change the Gantt Chart (for the final version) to show how the project ended up running.

I've broken down the project technique into four distinct sections; Initial Guidance and Preparation, Final Planning Stage, Developing Stage and Final Documentation. Allowing the workload to split into specific group depending on relevance and thus create documentation for each specific section without it clashing.

The reason I selected this project management technique is because I believe this technique is simple and easy to use as well as understand and doesn't require any sort of programming related understanding to comprehend the information it contains. It sets up the project outlines and what needs to be done by a certain time more easily, as well as this if things do end up overlapping or taking longer, then it allows you to readjust the tasks left so that you still end up finishing at the right time.

I will work to complete each model using the waterfall technique and trying to set to the time frame, but will ultimately work until completion, and work around the extended time so that I get it done before the due date.

For document on Functions Definitions and the Outlines to the Program Code:
Refer to the other Brief and Planning Document attached



- These tasks are completed from the Trello Board

Resources: For this process, I will use the following computer/laptop, LucidChart, Python 3.7. Wing IDE with specific imports OS and Tkinter (on both my personal laptop and school computer), Tekura for more specific feedback and help, my time, my family, peers, and client (to make suggestions, etc), I will also use word documents to present the testing, relevant implications, user guide, feedback/reflections, and planning/writeups.

The tools I will use to carry out the planning aspect of this project include a Gantt chart, Trello board, google forms and flow charts.

I will use the Gantt chart as a timetable sort of system, referring to it on a daily basis and evaluating my position weekly to make adjustments as to when I actual complete tasks, and look to see how this affects the remaining tasks I have. I will use the Gantt chart to get an understanding of how far I'm coming along and whether I need to speed things up or adjust the Gantt chart according to my progress, and if so how much time does this give me for other tasks. I have created an initial Gantt chart which is how I expected my project to carry out before actually getting a start on it, and I will add in the final Gantt chart which will show the actual time frame it took for tasks to be completed within my project. (Gantt Charts are shown above).

As mentioned earlier, I will use the Trello board as a to do list, in which I'll be able to electronically tick things off as they're done or move them between the different stages (Doing and Done). This will allow me to better understand what tasks I have left as a whole and whilst using the Gantt chart simultaneously will

give me a better understanding of how much time will be required for the remaining tasks in the project. The Trello board will also be accessed regularly (every time a task is started and finished), and with a reference to the Gantt chart will show when tasks have been completed on this document. (Initial Trello Board shown above, with stickers shown after each task is completed)

The flow chart I've created (in the other brief and planning document) was used to break up the code into smaller sections (through various definitions as well as separated sections with the imports, definitions, classes and GUI aspect to the code. By creating this flow chart, it made it a lot easier to see what parts of the code I had to create and in what order, and which parts were connected to which. It also allowed me to plan ahead and see what I need to add to a specific definition or section of the code to make it easier to code the next section. It also makes writing the code seem less daunting and overwhelming and more approachable. The flow chart was one of the first things created in the planning phase of writing the code.

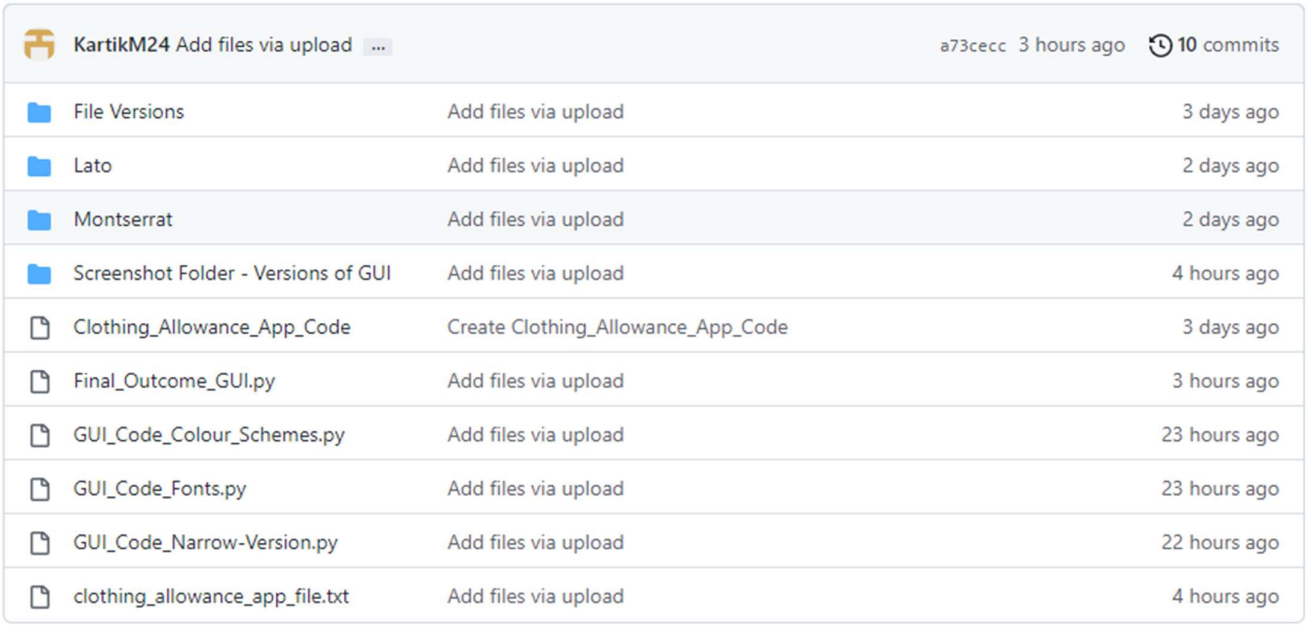
I've also used google forms to gather feedback on the GUI I created to find suggestions on ways to better the GUI, thus creating different versions of GUI to gather more feedback on what the best design is and what more should be changed to perfect the GUI. This is further discussed under collaboration tools.

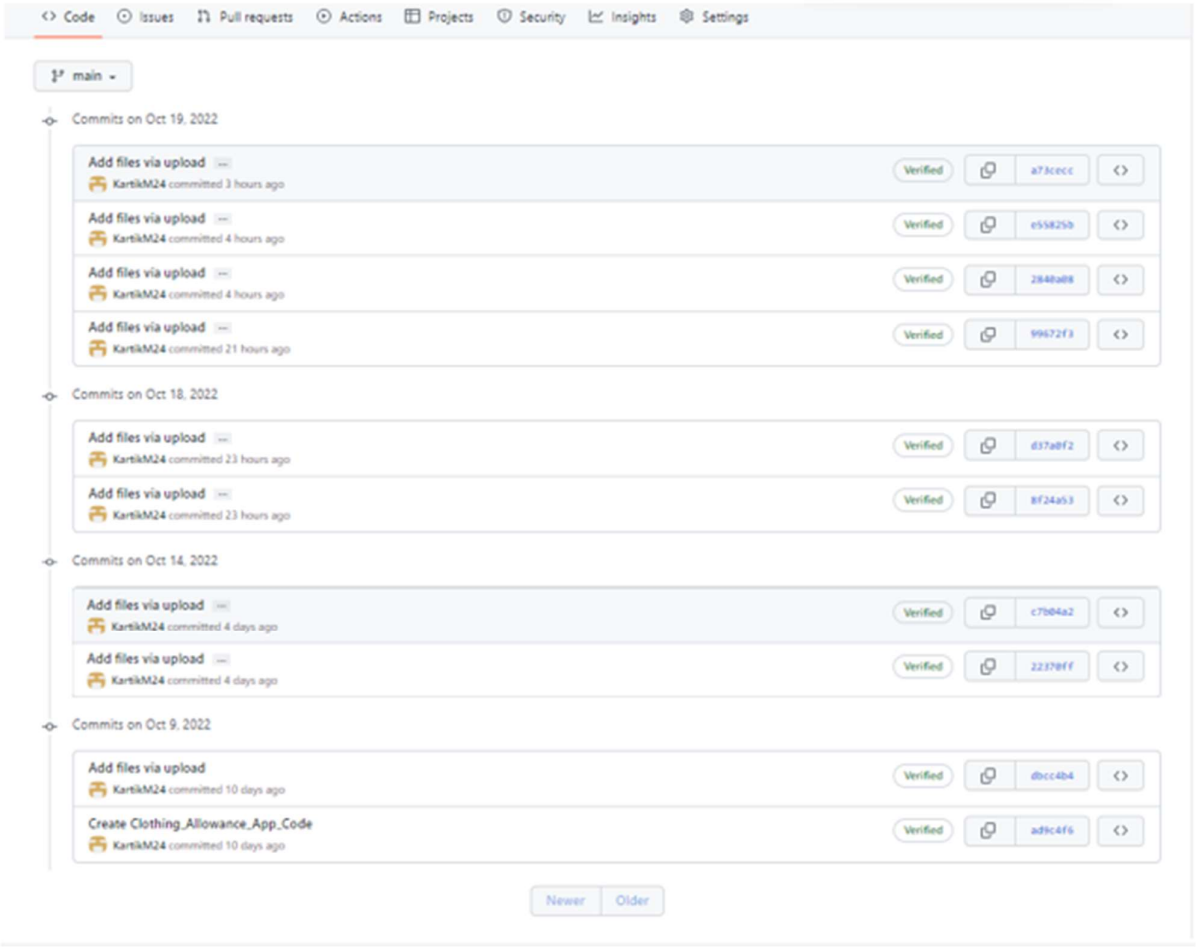
All of the tools mentioned above (aside from the Flow chart) are updated either daily or weekly to stay up to date with my real-time progress. This allows me to make further adjustments in the rest of the project or other sections, as well as plan ahead, and change/add or adapt specific sections as required.

Version Control for GUI:

For the version control of the code, I have used Github to track my progress as well as store all the designated files related to the GUI project in one place. It is a great way to store all my file history as well as commit and comment/provide a description of the code/specific file.

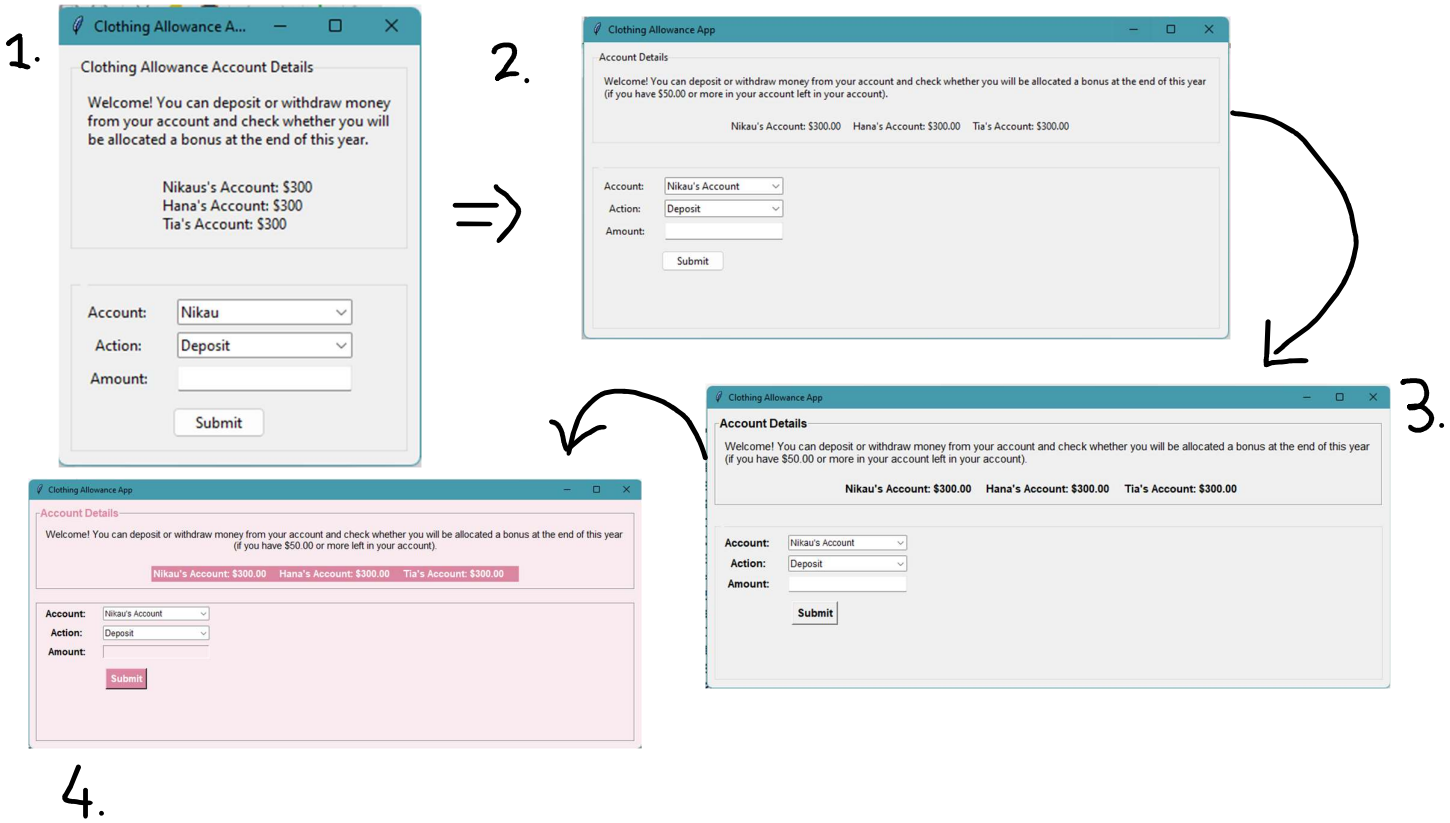
I will largely be using Github alongside my collaboration tool of google forms. As I have completed the base code for the GUI, without any of the implemented changes suggested by the respondents of the form. Github will be used to track my version history and comments. Allowing me to experiment with different solutions (by branching off) before coming to a final version that works (committing this branch back to the main file). It also means if I add new things to the code and make a mistake, I can look back at older versions to see where I went wrong or what I need to remove. This will be very helpful when working to implement different layouts, colour themes and font styles. Also, will be great to show the evolution of the code from the start (initial template) to the final model of the pre-GUI adapted code, through all the changes to the GUI aspects of the code and finally the last actual outcome of the GUI code.



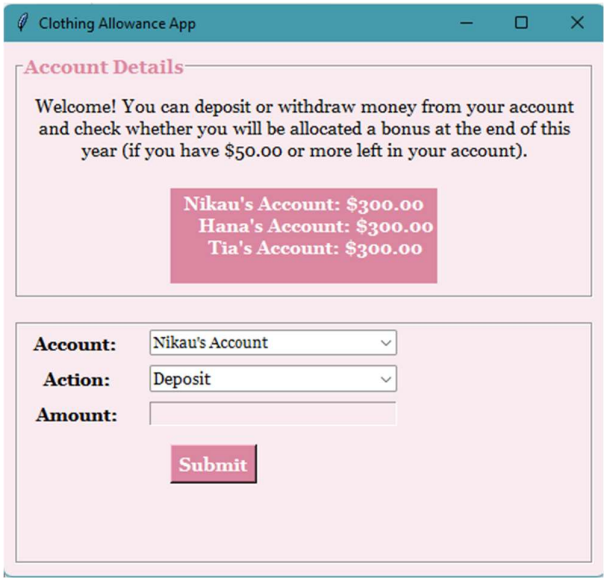


All of the files above have been attached alongside this document, and shows the evolution of the code overtime. It starts with the initial basic design of the definitions and functions and updates these definitions and functions to make a fully functional app. It then moves on to develop the GUI aspect of the code working on changing the layout, font styles and colour of the code before producing the final design.

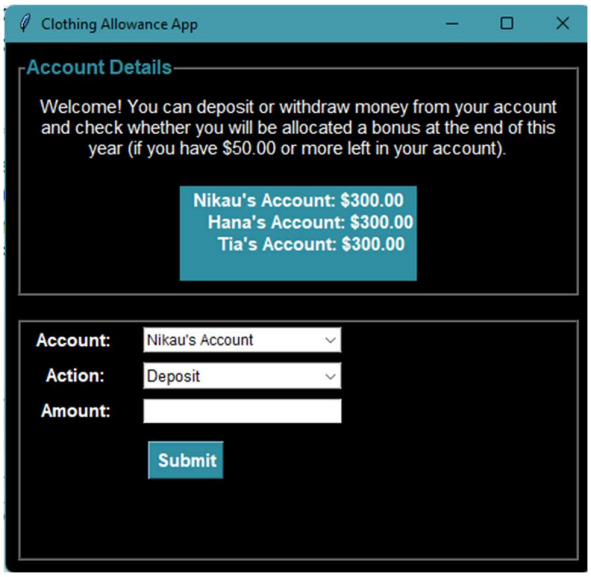
The evolution of the GUI outlook is shown below:



5.

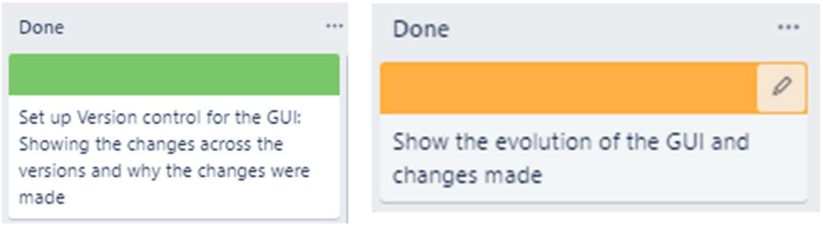


6.



This is the final design just above

The first design made was the generic version before any bonus messages, files or input verification was added. The second design is the final version of the non-GUI aspect of the code with all the functions tested and appropriately working. The third design was the changed fonts, with differing sizes, boldness (to emphasise and highlight certain things) and the font (with the font being set to Lato). The fourth design was a change in colour and font colours, to make the GUI more aesthetically pleasing, and further highlight certain features (such as the submit button and the amount of money in each account). The fifth version of the code changed the layout to a more narrow design similar to the first version to have a more efficient use of the space, and to make it so that the user doesn't have to look around to much to find what they're after. The final version of the app and the one that is the official version, is design 6, in this design the colour scheme is changed to a dark mode with blue highlights and white font colour. This was done as dark mode was seen as a more aesthetically pleasing and of better relation to the clothing allowance app. The usage of the blue colour brought a bit of light into the app, emphasised the key parts and made the app look more aesthetically pleasing. Giving the overall app a modern, easy to read and use and clean style, fit for its purpose.



- These tasks are completed from the Trello Board

I will decompose the GUI code into separate sections based on the font (size and layout), layout (positioning of all the components within the app) and colour scheme of the app. I will trial and test different versions of these aspects to create different versions of the app. Using these different versions I will then send out forms to gather feedback on what other people believe to be the best features or what improvements they suggest I add/make. As amendments are made or different versions are created I will update the changes on Github showing the different variations of the code and its evolution.

Collaboration Tools:

Research Done Prior to setting up the Forms:

Before setting up the initial forms to gather feedback on the best designs and features to implement to the GUI Clothing Allowance app, I had to conduct some research regarding the factors. The factors I investigated were the best possible colour schemes, font styles and layout types that would work best for the Clothing allowance app; whilst meeting the specifications (appearing aesthetically pleasing, efficient use of the different styles, being trendy/most compatible with current users (which is more so the target audience of the app) and being easy to use). Based on the research I've done I created a form containing the most promising potential features that I can implement into the GUI app. I will then share these forms to generate data and what features are most liked.

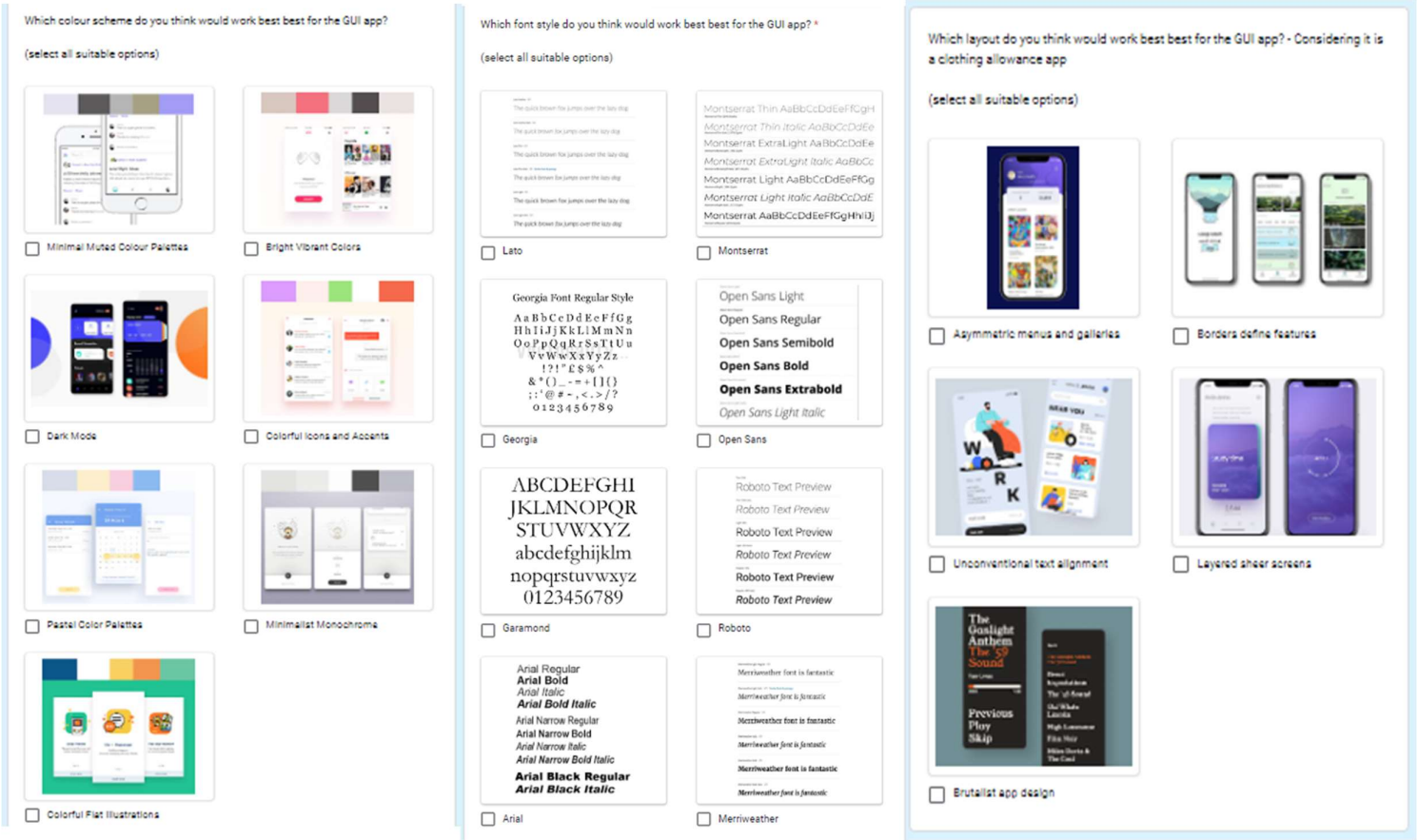
The best potential colour schemes are:
- Minimal Muted Colour Palettes - Bright Vibrant Colours - Dark Mode - Colourful Icons and Accents - Pastel Colour Palettes - Minimalist Monochrome - Colourful Flat Illustrations

The best potential fonts are:
- Open Sans – Montserrat – Roboto – Garamond – Lato – Merriweather – Georgia – Arial

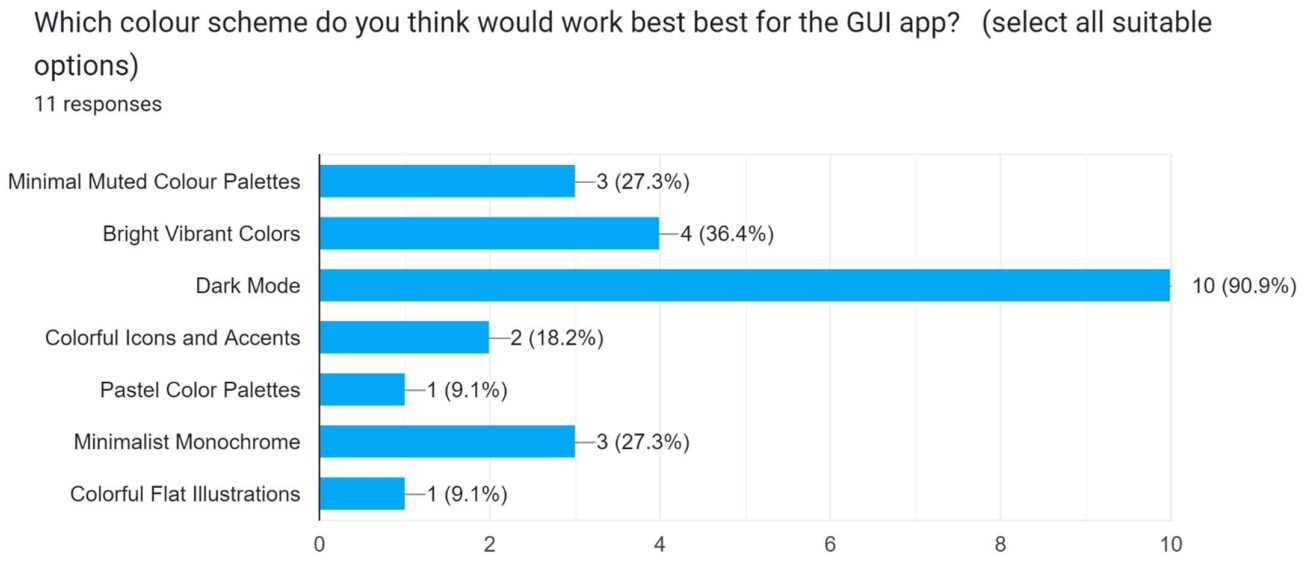
The best potential layout styles are:
- Asymmetric menus and galleries - Borders define features - Unconventional text alignment - Layered sheer screens – Brutalist app design

I've set up two sets of forms using Google drive to gather feedback from my peers, potential users, friends, teachers and stakeholders. In the initial form I set up the questions to gather feedback on what the users believed to be the most suitable colour palette, font and layout format style that best suits the GUI Clothing Allowance App. The form as depicted below displays some suitable options for possible colour palettes, font styles and layout (based off the research I've done for what's most suitable and trending for the given specifications – clothing allowance app). The form asks the users to select what they believe to be best suited based off the options provided (and their corresponding images). It then gathers this feedback and generates graphs to more easily show what the most preferred options were. The first of the two forms is depicted below.

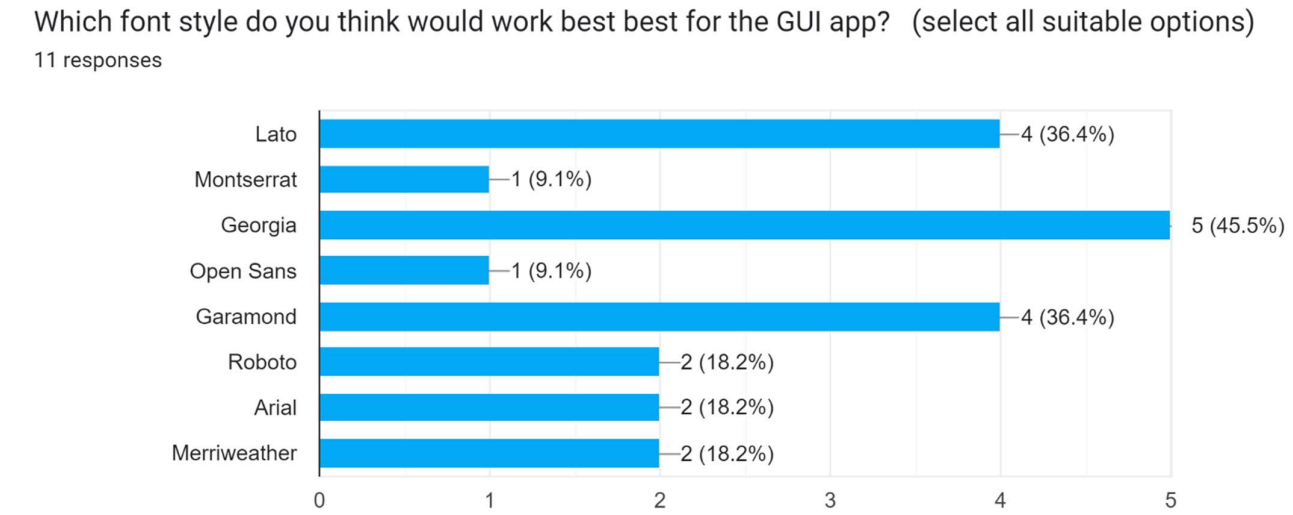
Initial Gathering Feedback Form for GUI



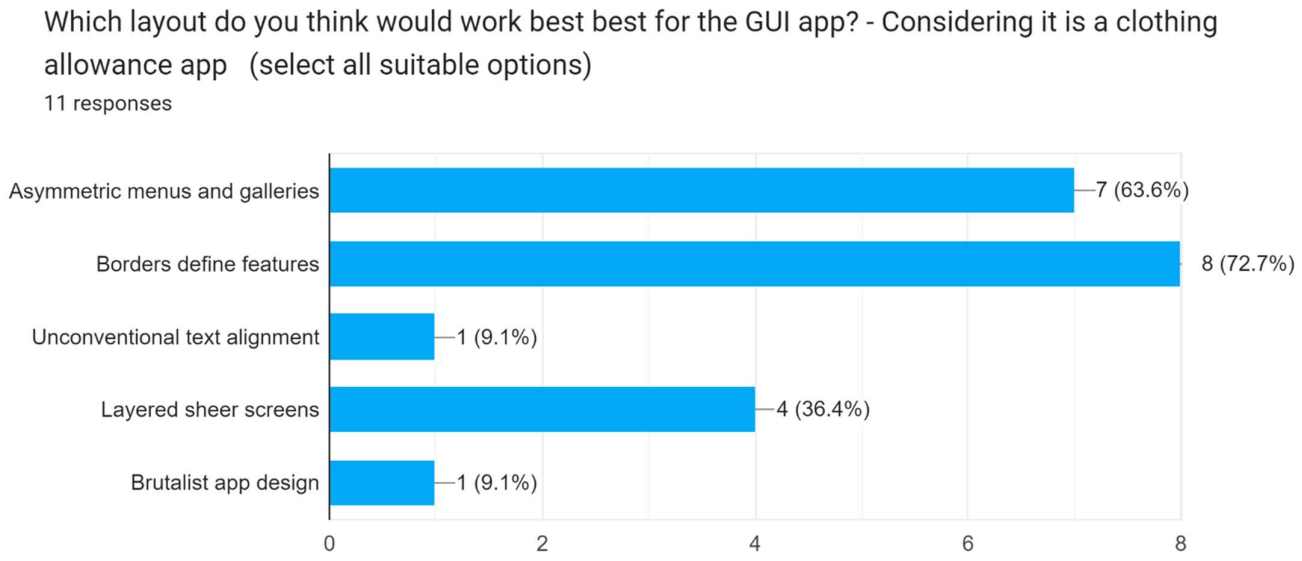
The following are the results/data of what the most popular options were to implement into the GUI app (from the form):



For the colour schemes, the most popular option was the Dark mode which got the majority of votes from the respondents (with 10 votes). This was clearly the most well-liked option, with the second most liked options having fewer votes for the colour scheme (4 votes) – Bright Vibrant Colours. As these are the colour schemes that seem to be most favourable by the respondents when decomposing the GUI code and creating the different possible versions, I'll base them on these colour schemes to find the most preferred options by the respondents.



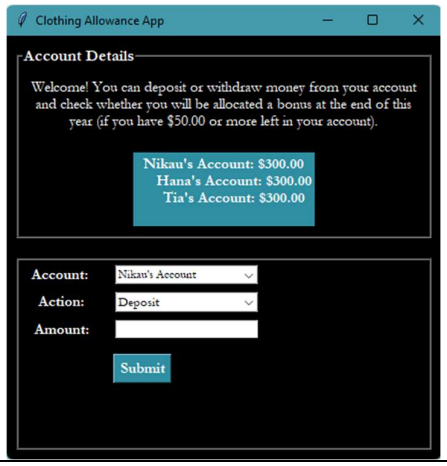
For the font style, the most popular font was Georgia which got the most votes from the respondents (with 5 votes). This was the most well-liked option, with the joint second most liked options having only one less vote for the font style (4 votes) – Garamond and Lato. As these are the font styles that seem to be most favourable by the respondents when decomposing the GUI code and creating the different possible versions, I'll base them on these font styles to find the most preferred options by the respondents.



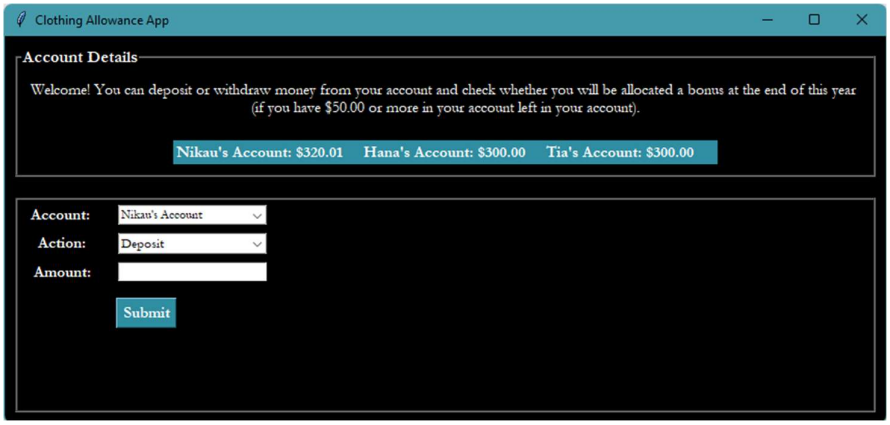
For the layout, the most popular option was Borders define features which got the most votes from the respondents (with 8 votes). This was the most well-liked option, with the second most liked options having only one less vote for the font style (7 votes) being Asymmetric menus and galleries. As these are the layouts that seem to be most favourable by the respondents when decomposing the GUI code and creating the different possible versions, I'll base them on these font styles to find the most preferred options by the respondents. This should help give me a better idea as to which layout style works best with the colour scheme and font, based on App and it's purpose.

After analysing the initial forms and getting the best possible options for all three criteria, I then implemented the suggested changes in a mixture of possible options, i.e. mixing the most highly rated fonts with each of the most highly rated colour palettes and the highly rated layouts mentioned. Providing the respondents with a range of possible options and what they prefer the most. Then based off their option(s) I asked if there are still any minor adjustments they would like to make to their selected option and why they selected the options they did. The final options created are shown below:

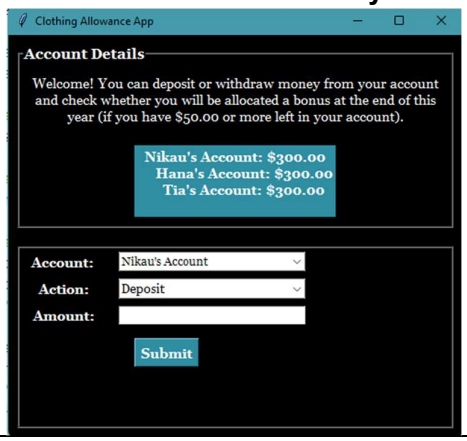
Dark Mode - Garamond - Narrow Layout



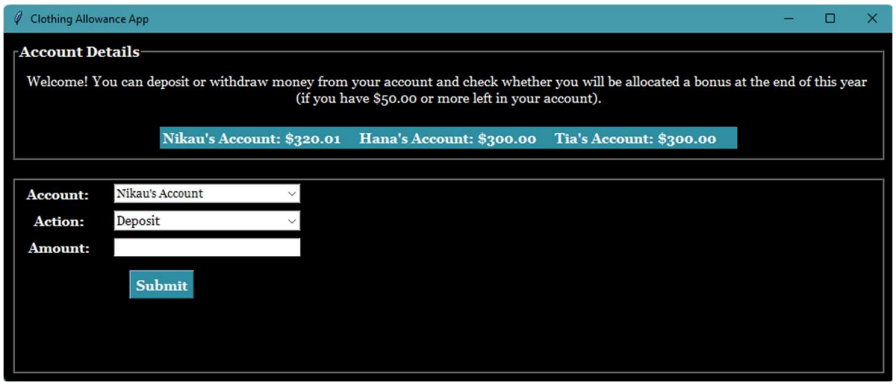
Dark Mode - Garamond - Wide Layout



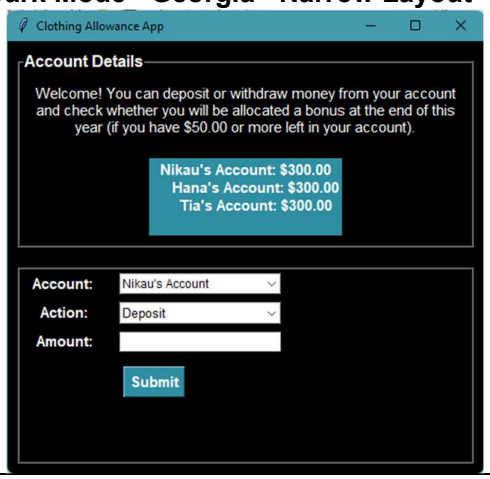
Dark Mode - Lato - Narrow Layout



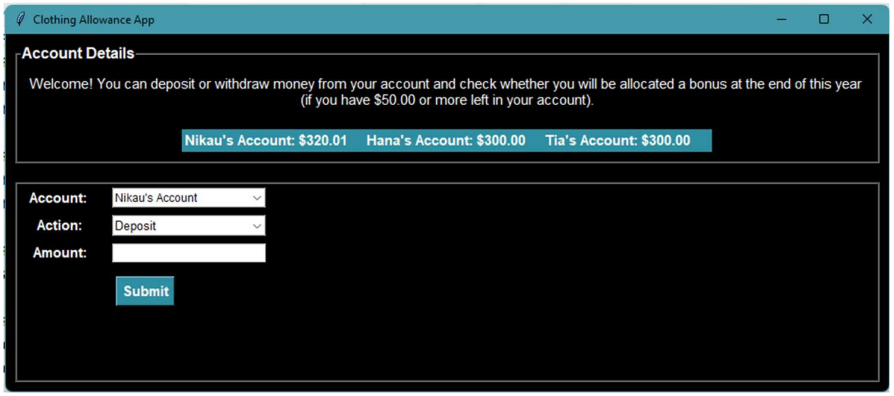
Dark Mode - Lato - Wide Layout



Dark Mode - Georgia - Narrow Layout

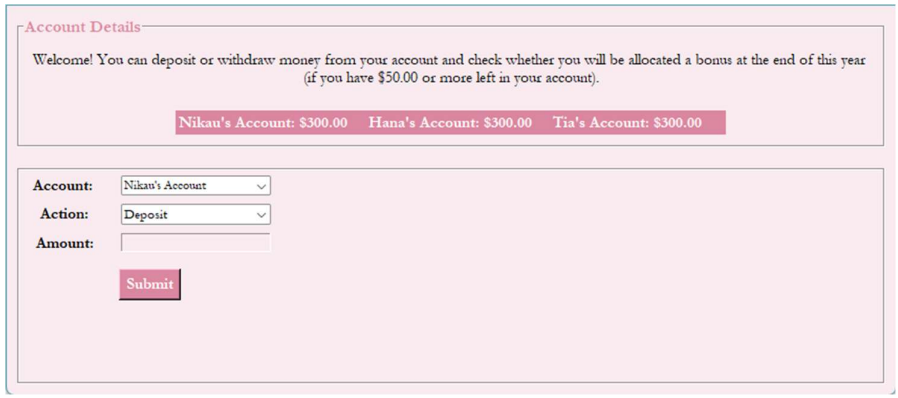


Dark Mode - Georgia - Wide Layout

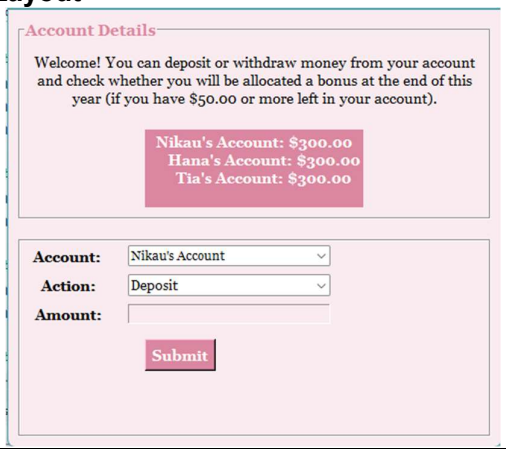


Bright Vibrant Colours - Garamond - Narrow Layout

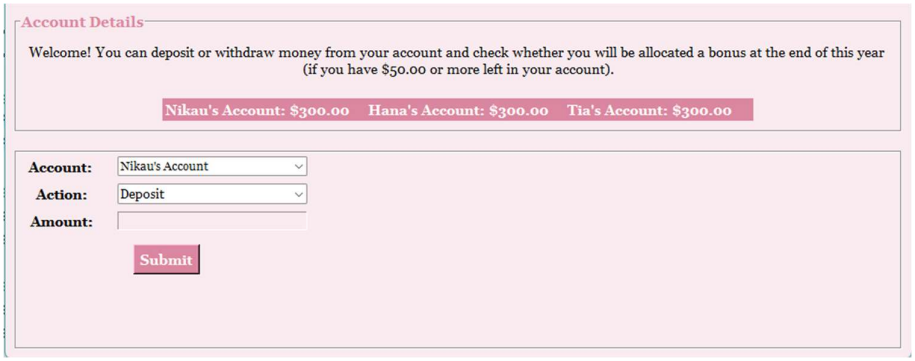
Bright Vibrant Colours - Garamond - Wide Layout



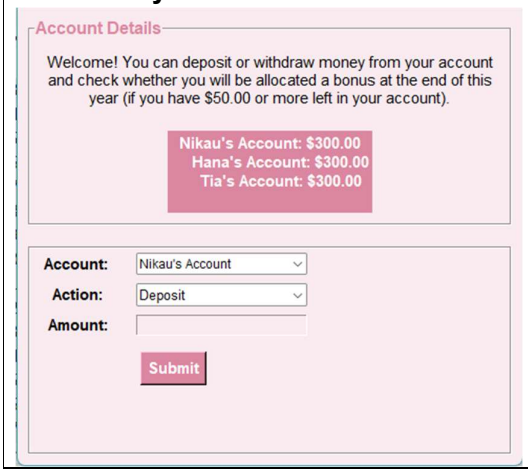
Bright Vibrant Colours - Lato - Narrow Layout



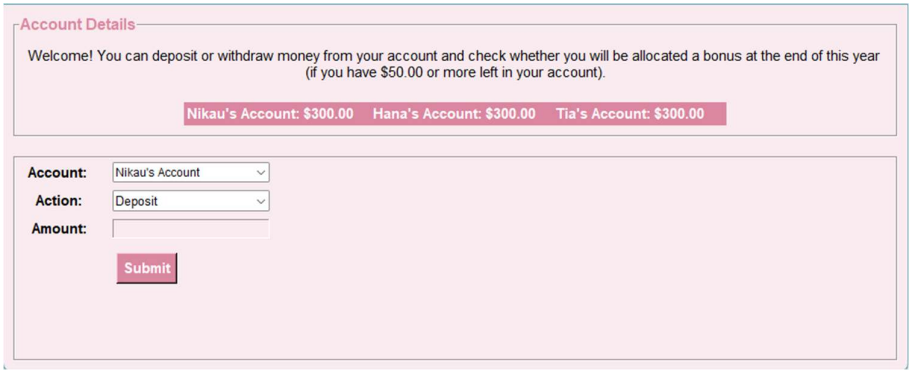
Bright Vibrant Colours - Lato - Wide Layout

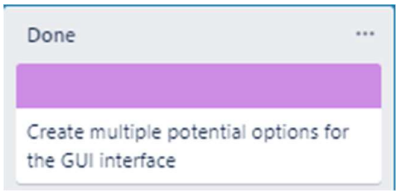


Bright Vibrant Colours - Georgia - Narrow Layout

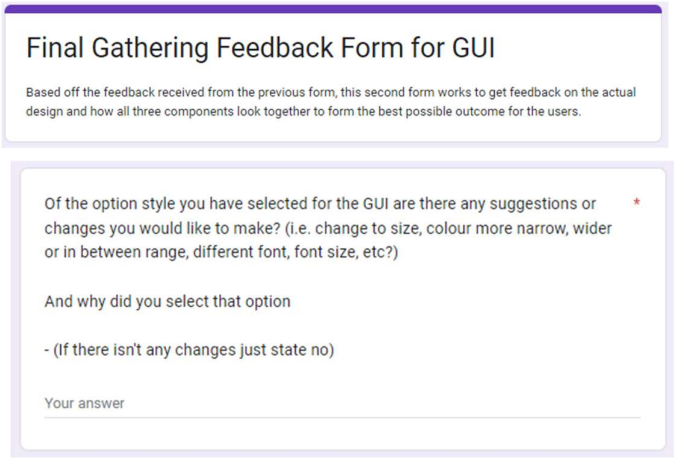


Bright Vibrant Colours - Georgia - Wide Layout





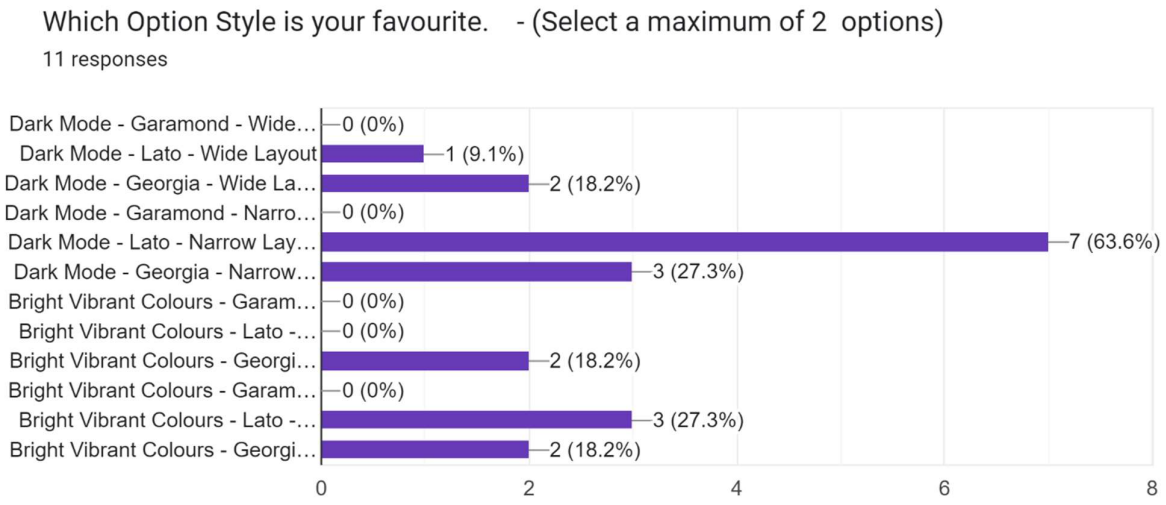
- This task is completed from the Trello Board



The final feedback form was created, displaying the above versions of GUI, and asking the respondents to select the option they believe is most suitable for the Clothing Allowance app. It then asks them whether they would make any changes to the option they've selected and why they selected the option they've chosen.

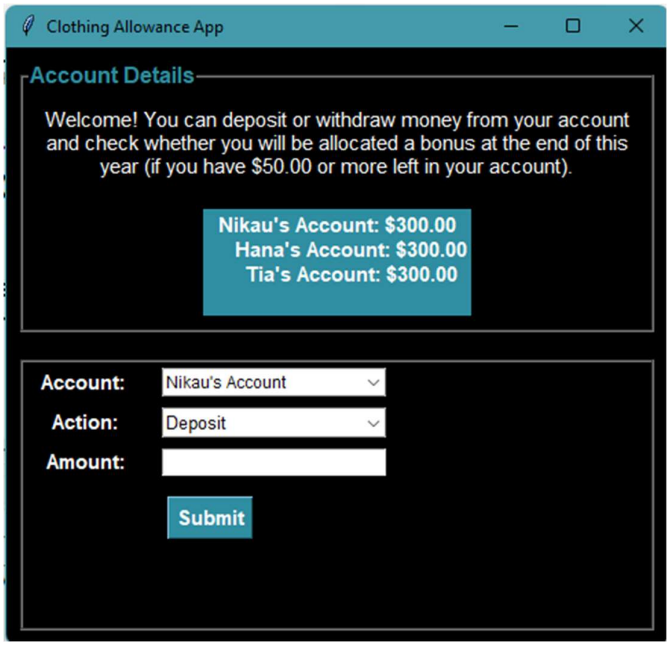
From this form, the feedback received overwhelmingly showed that the dark mode version and font styles of either Georgia or Lato were preferred with varying responses with the layout (but a slightly greater preference to the narrow layout).

The graph below shows the results for the most preferred options:

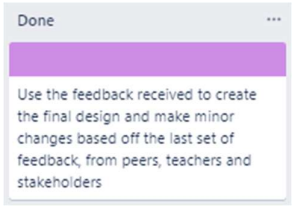


The graph above clearly indicates one design style was preferred more than others (Dark Mode – Lato - Narrow Layout), with 7 votes. The next closes options were relatively similar in that they both had a Narrow Layout, but different fonts (Georgia and Lato) and different colour schemes. But from the second option, it is clear to see that the narrow design is the most highly rated. Another trend that's visible in the graph is that the dark mode colour scheme is more highly rated with a total of 13 votes for the darker colour scheme as compared to 7 for the lighter colour scheme. Another obvious trend is none of the respondents chose options with the Garamond font. So as can be deducted from the graph the best option is a design that has a narrow layout, a darker colour scheme option as well as a more clear-cut font such as Lato or Georgia. So using these statistics and incorporating some of the feedback received in the form the final design will be created.

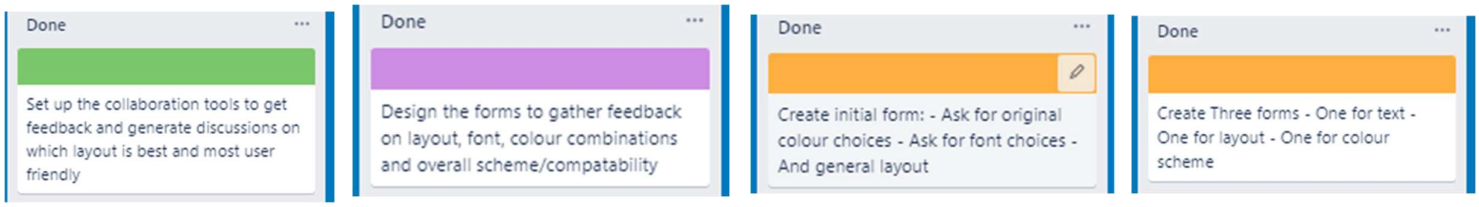
After compiling all the feedback and analysing the data and graphs I created the end version, which is the most efficient, aesthetically pleasing and with the best usability for all possible users. This is the final and most well-liked overall GUI for the Clothing Allowance app was the following:



Colour Scheme: Dark Mode
Font Style: Lato
Layout Structure: Narrow Layout

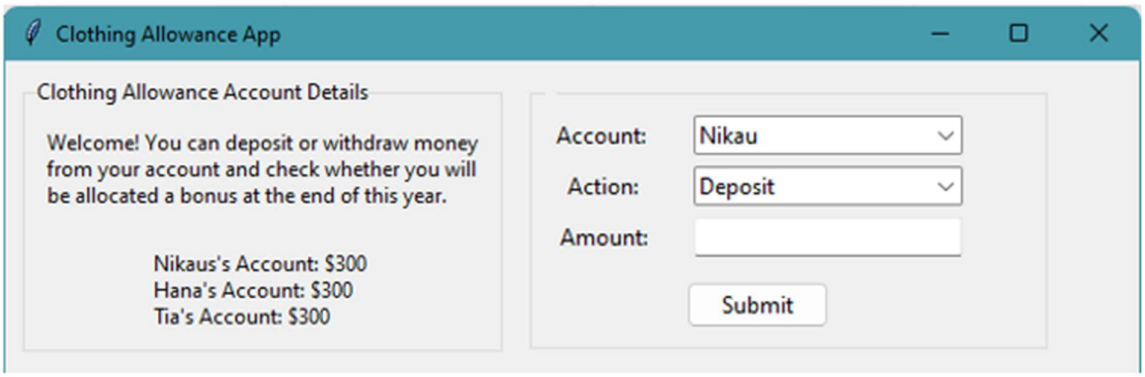
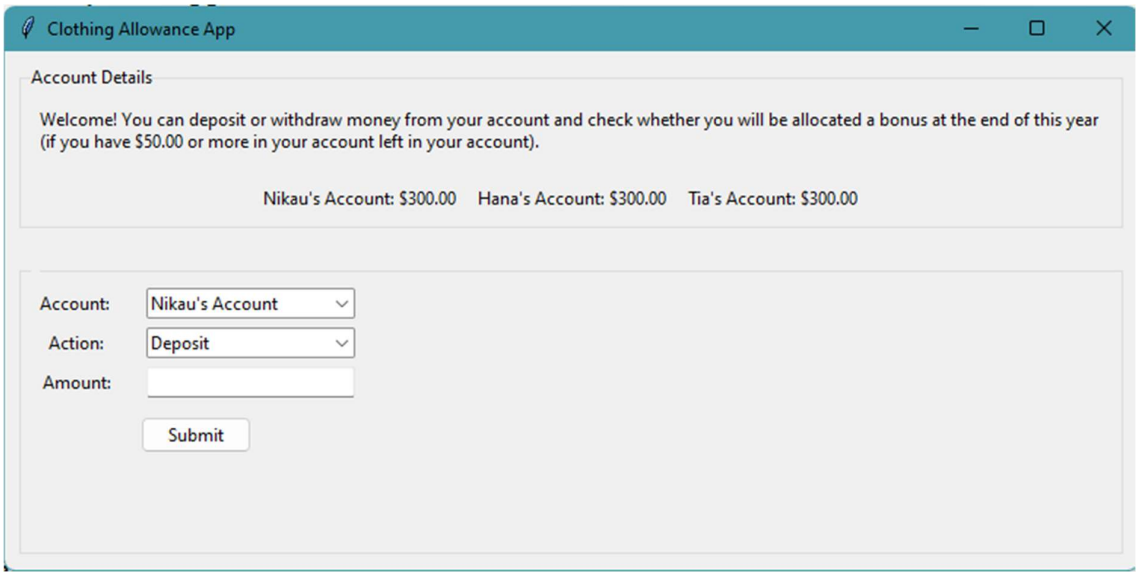


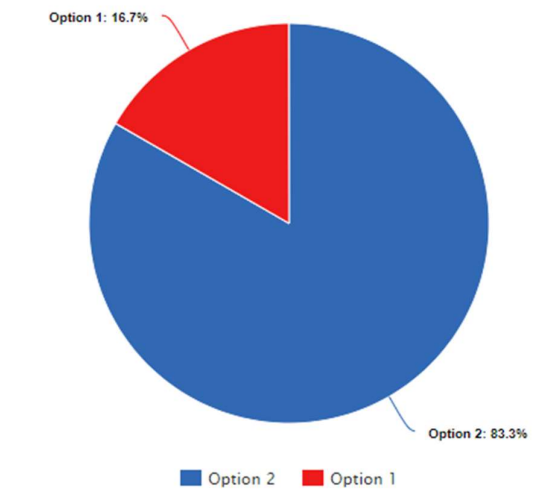
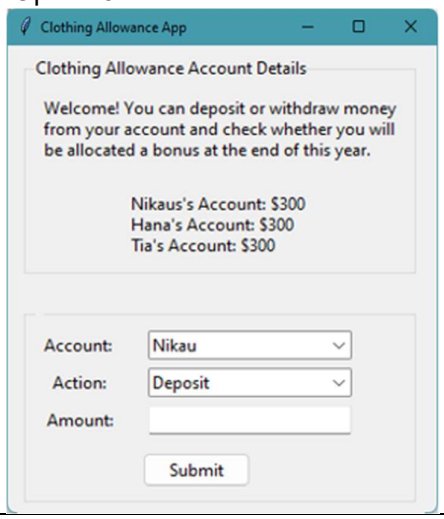
- This task is completed from the Trello Board

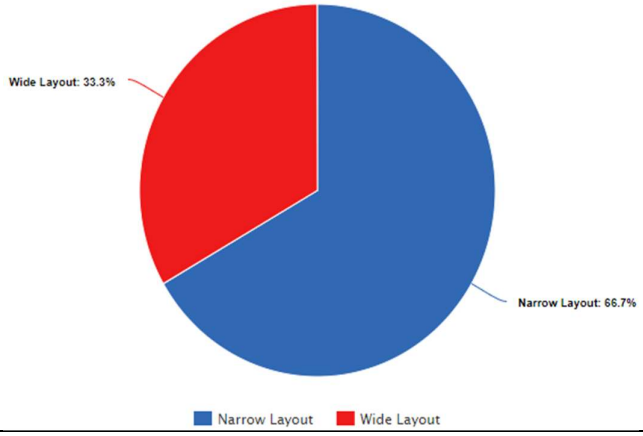


- These tasks are completed from the Trello Board

Decomposition of the GUI Code:

Component 1: Layout	
Trail 1:	For this trial I will be using a wide layout design for the home page, I'll compare whether a linear layout or a vertical layout works better for the app.
Initial Response/ Feedback	After sending the form with the layout design options to the respondents the layout option most users preferred was Borders define features which got the most votes from the respondents (with 8 votes). The second most liked option with only one less vote for the font style (7 votes) is Asymmetric menus and galleries. So based on these suggestions and the purpose of the app, these designs were made similar to the initial pre-designed templates.
Testing	Option 1:  My personal thought is though the app is efficient in it's usage of space its layout is kind of off-putting and weird looking, considering most apps of a similar purpose layout. Also this layout wouldn't work too well for devices that are a bit more narrow, and for devices like phones and tablets would not look appealing when held vertically. So the functionality and usability of the app are not well considered with this layout considering that the users might use a range of different devices (not just a laptop). Option 2:  My personal thought is that the app isn't as efficient with its usage of space. However, the layout is less off-putting and there is more space for error messages or feedback messages. However, the design could be more compact.

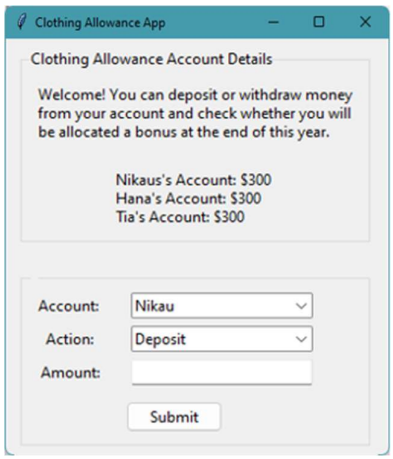
<table>
<tr><td>Response/Feedback</td><td> 
</td><td>Based upon the research I conducted most people also felt that despite the better usage of space in the first option, the second option had a better aesthetic look to it. However, the biggest suggestion made was to make the design slimmer and make better usage of space, as this would not only make the app look better but also make it easier to read what was written.</td></tr>
<tr><td>Trial 2</td><td colspan="2">For the second trial I will compare the narrow design with the wide layout to see which layout style is most preferred.</td></tr>
<tr><td>Testing</td><td colspan="2">Option 3:
 

The third option I trialled was similar to Option 2, except this design was more narrow had a better usage of space, and the two boxes and the contents within the boxes were larger to make it easier to read. This design works well across all devices as it fits on a laptop screen and works perfectly for smaller devices too, increasing the app functionality and usability.</td></tr>
<tr><td>Response/Feedback</td><td> 
</td><td>Based on the research I conducted on what the responders thought was the best option style for the GUI considering the purpose of the app. The feedback gathered shows that most people preferred the narrow layout compared to the wide layout.

This was because they believed the GUI to have better use of the layout, the words to be easier to read and the app to be able to be used across multiple devices more easily.</td></tr>
<tr><td colspan="3">Component 2: Colour scheme</td></tr>
<tr><td>Trial 1:</td><td colspan="2">For this trial, I will be using a neutral design for the home page of the GUI app. I'll gather feedback to see what colour scheme option works best for the app in terms of its aesthetics and formality of the app.</td></tr>
<tr><td>Initial Response/ Feedback</td><td colspan="2">After sending the form with the colour scheme options to the respondents the colour scheme most users preferred was the Dark mode which got most votes from the respondents (with 10 votes). The second most liked option having fewer votes for the colour scheme (4 votes) – Bright Vibrant Colours. So based on these suggestions and the purpose of the app, the following design options were generated.</td></tr>
</table>

Testing:

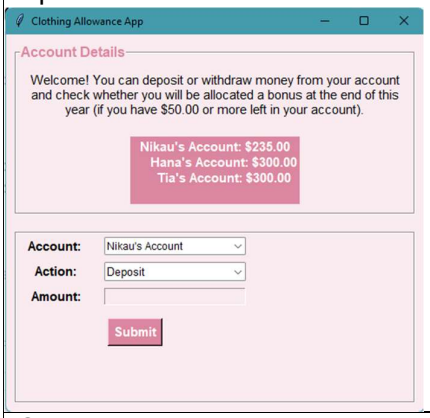
Option 1:



The first option was well-liked by most respondents. They believed it to look more formal, neutral and aesthetically pleasing. The colour scheme does suit the app but the appearance is somewhat bland and could use a bit of colour to brighten things up and make the GUI look even better. So, suggestions were made to add colour to the app.

Based on the initial form I sent out to gather feedback on what users believe to be the most suitable colour scheme for the app I created two alternative versions.

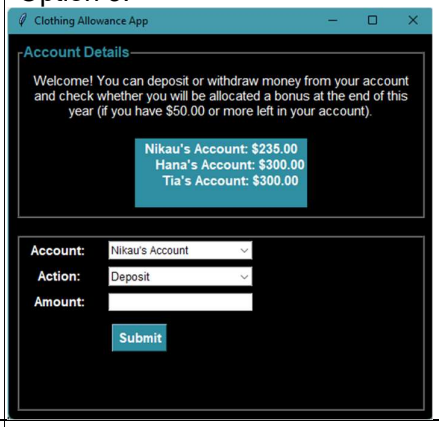
Option 2:



The second option is based on the “Bright Vibrant Colours” colour scheme that was liked by the respondents. I chose soft vibrant colours that still allow the features and the words of the app to be easy to read whilst adding some flair to the GUI.

However, this option was not as well-liked as compared to the other two options. Some of the feedback was that they “liked the bright colour option, but I’d look at using an alternative colour potentially.” Whilst others said that “bright colours don’t suit the text and make the app appear to be less formal/more trivial.

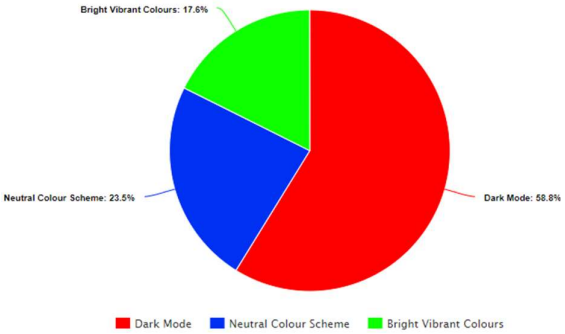
Option 3:



The third option is based on the “Dark mode” colour scheme which was the most liked option by the respondents. I chose black and white colours as it makes the app sleeker and the additional blue colours adds a bit of colour to the app and increases the aesthetic look of the app. Whilst ensuring the features, messages and texts are all still easy to read.

This colour scheme got the most positive feedback, with all of the respondents liking the look of the app.

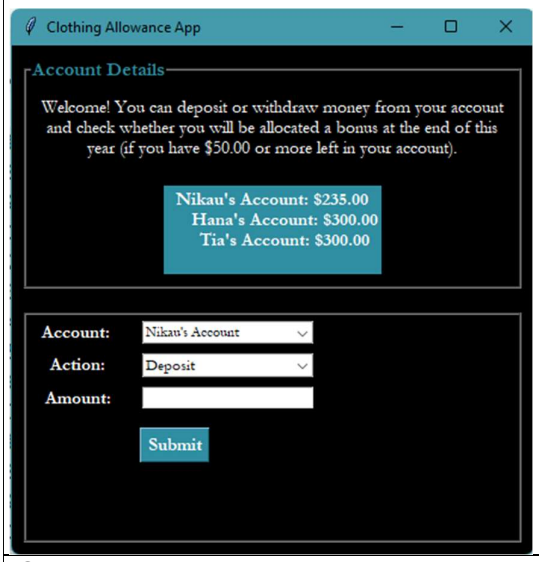
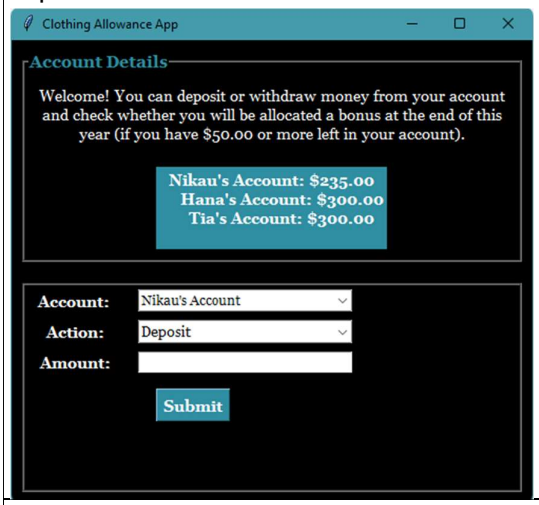
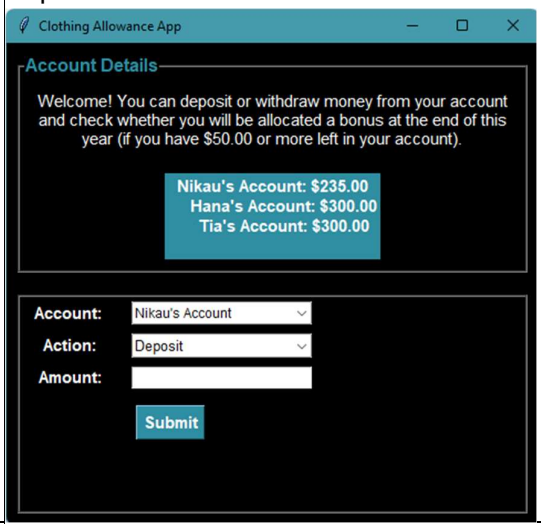
Response/Feedback

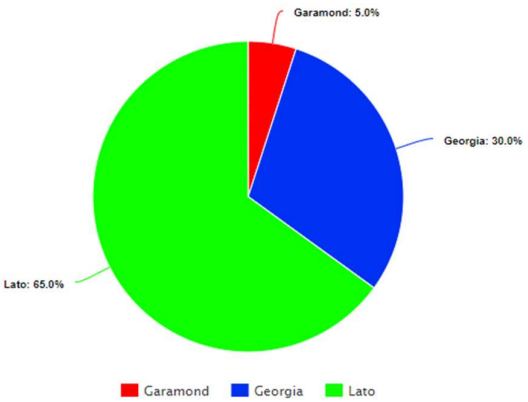


Based on the research I conducted on what the responders thought was the best option style for the GUI considering the purpose of the app. The feedback gathered shows that most people preferred colour scheme was the Dark Mode colour scheme.

This was because they felt that the dark mode colour scheme was a better fit for the app. It suited the app’s purpose making it look modern, classy, and aesthetically pleasing whilst emphasising certain aspects of the GUI. Everything was easy to see and understand.

Component 3: Font Styles

Trial 1:	For this trial I will be researching what font works best for an app and specifically a 'Clothing Allowance app' I will then gather feedback from other peers and people to see what they believe works best for the app in terms of the purpose of the app and it's aesthetics.	
Initial Response/ Feedback	After sending the form with the font style options to the respondents the font style option most users preferred was Georgia which got the most votes from the respondents (with 5 votes). The joint second most liked options having only one less vote for the font style (4 votes) – Garamond and Lato. So based on these suggestions and the purpose of the app, the following design options were generated.	
Testing		Option 1: The first option wasn't as well-liked as the other fonts, though this font was rated highly as a potential font style to use in the app, once it was incorporated the biggest issue the respondents had with the GUI was that the font was a bit hard to read as it was too curly and the words looked too small and bunched up. The font style is pretty but based on the app's functionality and for the sake of the users, it's not an ideal font to use.
	Option 2: 	The second option was well-liked, as this was an easier-to-read font style like the Garamond font but less bunched up and thus more appropriate for the app. Most of the respondents did like this font, but felt for the headings the font might not be bold enough and it looked more suitable for just the feedback and welcome messages.
	Option 3: 	The third option was the most well-liked font-style for the GUI. The respondents felt that this was the most perfect font style for the app. This was because the font was easy to read, clear, not too small nor too bulky, made the text stand out more and suited the headings as well as the general text. It was the most ideal font style for the app, as it improve the aesthetic look of the app but also the functionality and usability of the app was taken into consideration by using this font. As users would have no issues with reading based on the font.

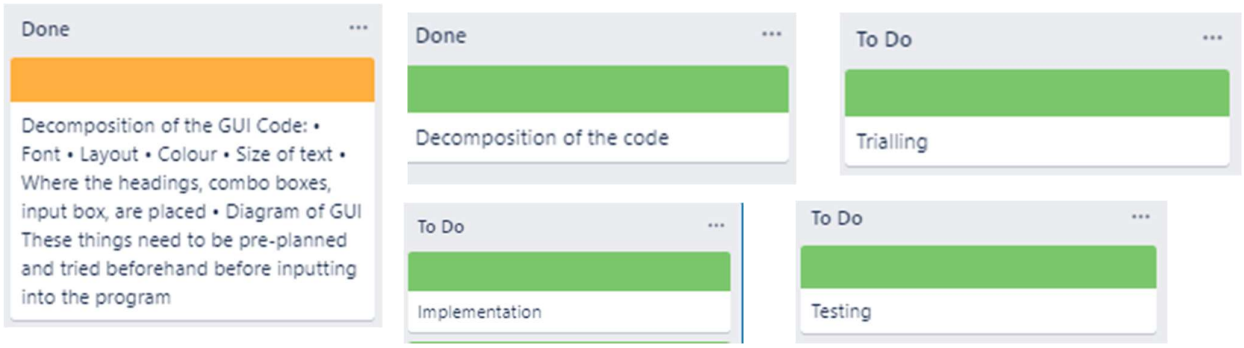
Response/Feedback		Based on the research I conducted on what the responders thought was the best option style for the GUI considering the purpose of the app. The feedback gathered shows that most people preferred the font style Lato. This was because they felt that this font was most suitable for the app, based not only on the aesthetics but the purpose of the app and the font's functionality within the app.

Overall Feedback on the Most Ideal GUI Design from Respondents:

After sending out the form to gather feedback on what was the best option for the GUI design, the following feedback was received.

- "I love the Dark Mode, black background, and blue accent. I think it gives you the opportunity to play around with neon colour buttons/ accents. Regarding the layout, I think the first option works best as there is less wasted space."
- "I chose the Dark mode colour as I believe that this colour scheme best suits the app and its purpose whilst highlighting certain things to make them stand out more, it makes the app look aesthetically pleasing and more formal. I've also selected the fonts Lato and Georgia, as opposed to Garamond, this was because I feel that the Garamond font was a bit too hard to read at times though it does look nice, but just based on the functionality the other two fonts are better and look nice too. Finally, I selected the narrow version because I feel it looks sleeker and I feel the usage of space is better and thus the app is more efficient with its layout."
- "I like the bright ones as they're more colourful, but I'd look at using another colour potentially."
- "Bright colours don't suit the text, darker colours are more suited to the text. The narrow layout makes it look a little bunched so the wider ones are better, but something in between might be the best choice. The curlier fonts also don't suit the text as much so Lato is probably the best one in my opinion."
- "No changes like the narrow look in both styles, I prefer the Lato font because it is easier on the eye."
- "This one is most space efficient, easy-to-read font, and best-looking style aesthetically, I think Dark Mode - Lato - Narrow Layout is the best design."
- "Dark and narrow is easier to read and your eyes don't have to move across the screen as much."
- I like the dark mode narrow layout in the Lato font because the colour aesthetic is nice and the font is clear and formal. Maybe you could make the text size a bit bigger."

After receiving this feedback and taking the votes into consideration. It was fairly evident that the Dark mode colour style, narrow layout and Lato font style are the best combinations of features for the Clothing Allowance App. Taking some of the feedback from the respondents, I did increase the size of the font and the width of the screen slightly to make it less narrow and thus avoid the words getting too bunched up, to produce the final design.



- These tasks are completed from the Trello Board

Final Outcome Testing:

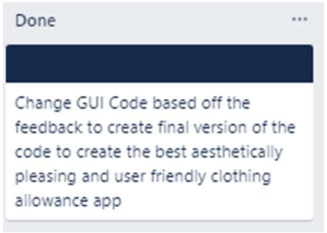
Testing of input/output messages has separate documentation 'Testing' (submitted alongside this document), refer to the Testing document for further information on those aspects.

Final Code After All Changes are Made:

The image below is a screenshot of the final version of the code after all the changes (based on the feedback) have been made to make the most efficient, functional, aesthetically pleasing and easy-to-use Clothing allowance app.

```
##****************************************** GUI CODE ***************************************************
root = Tk()
root.title("Clothing Allowance App")
root.configure(bg='#000000')

#Create the top frame
top_frame = tkinter.LabelFrame(root, text="Account Details", font=("Lato", 12, "bold"), bg='#000000', fg='#2F8EA1') #Georgia, Garamond, Lato
top_frame.grid(row=0, column=0, padx=10, pady=10, sticky="NSEW")

#Create and set the message text variable
message_text = StringVar()
message_text.set("Welcome! You can deposit or withdraw money from your account\n and check whether you will be allocated a bonus at the end of this \nyear (if you have $50.00 or more left in your account).")

#Create and pack the message label
message_label = tkinter.Label(top_frame, textvariable=message_text, wraplength=1000, font=("Lato", 11), bg='#000000', fg='#FFFFFF')
message_label.grid(row=0, column=0, columnspan=2, padx=10, pady=10)

#Create and set the account details variable
account_details = StringVar()

#Create the details label and pack it into the GUI
details_label = tkinter.Label(top_frame, textvariable=account_details, font=("Lato", 11, "bold"), bg='#2F8EA1', fg='#FFFFFF')
details_label.grid(row=2, column=0, columnspan=2, padx=10, pady=10)

#Create the bottom frame
bottom_frame = tkinter.LabelFrame(root, bg='#000000', fg='#FFFFFF')
bottom_frame.grid(row=1, column=0, padx=10, pady=10, sticky="NSEW")

#Create a label for the account combobox
account_label = tkinter.Label(bottom_frame, text="Account: ", font=("Lato", 11, "bold"), bg='#000000', fg='#FFFFFF')
account_label.grid(row=3, column=0, padx=10, pady=3)

#Set up a variable and option list for the account Combobox
chosen_account = StringVar()
chosen_account.set(account_names[0])

#Create a Combobox to select the account
account_box = ttk.Combobox(bottom_frame, textvariable=chosen_account, state="readonly", font=("Lato", 9))
account_box['values'] = account_names
account_box.grid(row=3, column=1, padx=10, pady=3, sticky="WE")

#Create a label for the action Combobox
action_label = tkinter.Label(bottom_frame, text="Action:", font=("Lato", 11, "bold"), bg='#000000', fg='#FFFFFF')
action_label.grid(row=4, column=0)

#Set up a variable and option list for the action Combobox
action_list = ["Deposit", "Withdraw"]
chosen_action = StringVar()
chosen_action.set(action_list[0])

#Create the Combobox to select the action
action_box = ttk.Combobox(bottom_frame, textvariable=chosen_action, state="readonly", font=("Lato", 10))
action_box['values'] = action_list
action_box.grid(row=4, column=1, padx=10, pady=3, sticky="WE")

#Create a label for the amount field and pack it into the GUI
amount_label = tkinter.Label(bottom_frame, text="Amount:", font=("Lato", 11, "bold"), bg='#000000', fg='#FFFFFF')
amount_label.grid(row=5, column=0, padx=10, pady=3)

#Create a variable to store the amount
amount = DoubleVar()
amount.set("")

#Create an entry to type in amount
amount_entry = tkinter.Entry(bottom_frame, textvariable=amount, font=("Lato", 10), bg='#FFFFFF', fg='#000000')
amount_entry.grid(row=5, column=1, padx=10, pady=3, sticky="WE")

#Create a submit button
submit_button = tkinter.Button(bottom_frame, text="Submit", command=manage_action, font=("Lato", 11, "bold"), bg='#2F8EA1', fg='#FFFFFF')
submit_button.grid(row=6, column=0, columnspan=2, padx=10, pady=10)
root.bind('<Return>', lambda event:manage_action())

#Create an action feedback label
action_feedback = StringVar()

action_feedback_label = tkinter.Label(bottom_frame, textvariable=action_feedback, font=("Lato", 12), bg='#000000', fg='#FFFFFF')
action_feedback_label.grid(row=7, column=0, columnspan=2, padx=20, pady=1)

#Create an allocated bonus message when accounts have more than $50 left in it
bonus_message = StringVar()

#Create a savings message when accounts have less than $50 left in it
savings_message = StringVar()

#Create and pack the bonus label
bonus_label = tkinter.Label(bottom_frame, textvariable=bonus_message, wraplength=1000, font=("Lato", 12), bg='#000000', fg='#FFFFFF')
bonus_label.grid(row=10, column=0, columnspan=4, padx=20, pady=2)
##*****************************************************************************************************
#Run the mainloop
update_balance()
root.mainloop()
##*****************************************************************************************************
#Inputs from file to copy for trials
#Nikau's Account, 300
#Hana's Account, 300
#Tia's Account, 300
```

This task is completed from the Trello Board -

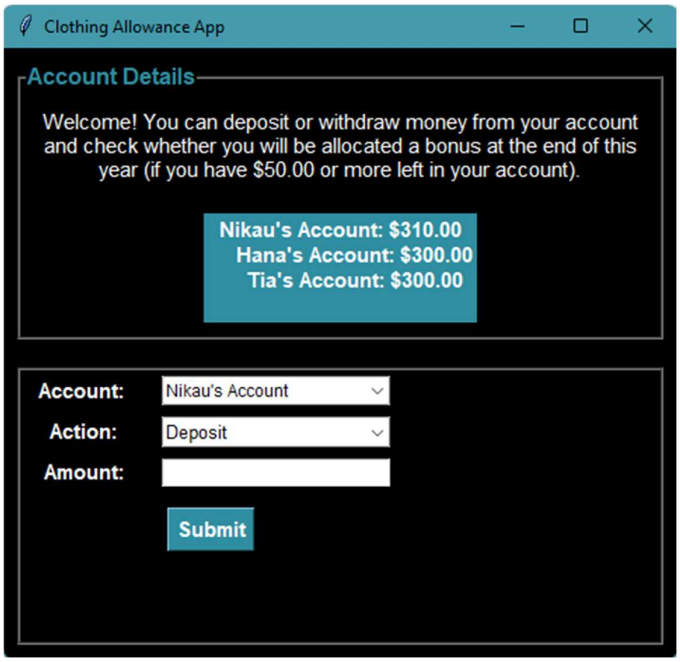


Relevant Implications:

- *End User Considerations:*

As this app was designed to be used by Nikau Hana and Tia as requested by their parents. The app contains certain features relevant to what their parents requested, and the file has been customised as such. The accounts have been named after the three children and each contains $300.00 for the starting account balance. There is also a bonus message system, set in place to remind the users to keep at least $50.00 in their accounts at the end of the year to get a bonus sum of money from their parents. Because of these requests, the app is designed slightly differently than it might have been had it been used for the broad public (though by changing the file and bonus message system this can easily be set up). Considering the users, the app had been made in a modern, family-friendly way with lots of feedback messages to let the user know what's happening as they do it, confirm transactions and report errors or invalid inputs.

- *Aesthetics:*



This program is easy to use, read and understand, the font used (Lato) is clear, and the size and word gap of the output is organized and appropriate depending on whether it's a title heading or general text. As demonstrated in the screenshot to the right; he writing is easy to read and understand without straining your eyes. The GUI is set up with two frames (bottom and top frame), with the top frame containing the account details (each account's balance) and a welcome message regarding how to use the app, and some information on the bonus message. Whilst the bottom frame contains the list of accounts to choose from, the action the user can carry out on the account, and the transaction amount, as well as displaying transaction messages and bonus messages. The layout is clear and aesthetically pleasing, the background colours are all neutral (as is the font) so the app looks formal, and the colour scheme (dark mode) is appropriate for those who are colour-blind. The app name is clearly outlined at the top of the app in the banner and transaction and bonus messages are all spaced out appropriately and appear at the right times. The font style, colour schemes and layout structure has all been trialled and tested by gathering feedback from various sources on what they believe is the best possible mix considering the app's purpose and ensuring maximum efficiency as well as good aesthetic design to the app. If I hadn't conducted the research to gather feedback and create the best possible app considering the suggestions made, then it is likely that the users of the app would have found the app less appealing and might have been less interested in using a dysfunctional hard to understand and use app, or maybe have considered the app to be not fit for it's purpose if the design wasn't considered sleek or formal enough.

- *Usability:*

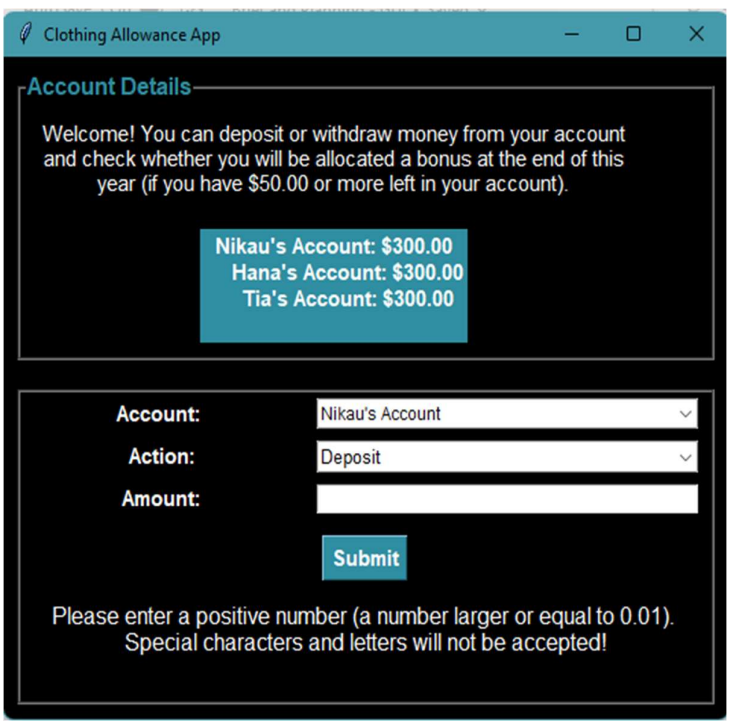
The program is very simple and straightforward to use, with directions and instructions with every message and error message being printed to the screen, and due to the appropriate aesthetics being used it is also easily readable. This program does not require any previous programming understanding and requires minimal effort and information input by the user to use. The instructions make it a lot easier for the user to use the program. The app was created for the three children to use (Nikau, Hana, and Tia), so the intention of the app is to make it so that children can easily use the app without having any hassle.

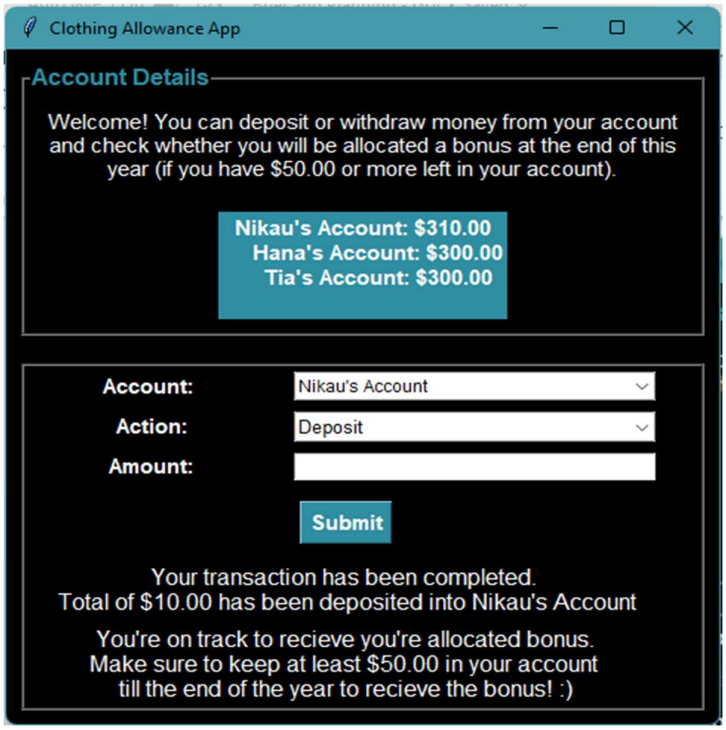
The user is constantly supplied with information as they carry out their actions and provided feedback on whether their inputs are valid (and what transaction was carried out), etc. The program is made so that no invalid inputs/impossible inputs can ever be accepted, and if the accounts file hasn't been set up for them the program has it pre-set up (refer to the 'Testing' document for further evidence of this). The final design was completed after gathering lots of feedback, creating different designs and incorporating the best aspects to make the final design, this was done so that it would be easy for the user to use the program. Thorough testing of inputs was also completed to make sure there are no possible impossible inputs that can be used to crash the program or accidentally accept these responses, and countermeasures are set in

place so the user doesn't lose their progress (with the account options they've selected or the files crashing thus losing their accounts) and understands why the input was invalid or not accepted. If this testing and these measures weren't put into place the user would have found the program cumbersome to use and error-ridden.

- *Functionality:*

The program works as it is supposed to, showing the user their balance, and allowing the user to carry out certain actions and enter a certain amount. As shown in the screenshots below.





It also displays the transaction messages explaining the transaction that occurred (as well as error messages in the transaction) and bonus messages regarding whether the user will receive their allocated bonus (for more thorough testing of the app refer to the testing document).

There are no errors in the programme, and this is because it had undergone intensive testing, trialling and improvements through the feedback gathered from peers, friends, stakeholders and teachers. Had I not carried out these improvements and ensured the app was fully functional, the users of the app would have encountered many issues and would have found the app hard to use, inefficient and frustrating.

- *Accessibility:*

This program is available to anyone wishing to use the program as long as they have a computer able to install and run python. As this program is free it should be easily accessible for almost all users. To use the program, they simply have to have access to the app (via downloading the file) before running it on python, no other files or apps are required as the App itself generates any files required by it. If the app did not do this itself, or if it ran on a programming language that required the user to download other files or required money to use, the users would be far less likely to use the app. This is because they might not want to go through all the hassle to use it or wouldn't want to pay money to run the app.

- *Health and Safety:*

This app contains no health and safety threats and is safe to use for children. As it is a clothing allowance app it doesn't incite any bad behaviour, have any threatening images, etc. Ensuring safe usage of any of the users.

- *Intellectual Property:*

All of the code made for the app as well as the features have been written by me. There is no breach of copyright, as all the colours, fonts and words used are not copyrighted, and were taken from free asset libraries (google fonts, HTML colour codes). The app also doesn't contain any images, songs, music/sound

effects, so there's no breach of copyright on those aspects. If I had not used fonts, colour styles and words that weren't copyrighted free this could have led to legal actions being taken against the app, leading to further ramifications against me, for allowing countless other users to use it.

- *Privacy:*

This program does not exploit user data, such as their names or account details. All statements are saved in one file accessible by those who use the app containing the specific file (i.e. the parents and their kids) therefore there is no breach of privacy. Though the file does contain private/sensitive information as it is not accessible to the public and only remotely through obtaining the file. If the file wasn't private and was easily accessible for multiple users using the same app then users would stop using the app and no longer trust it as it breaches their privacy.

- *Ethical Issues:*

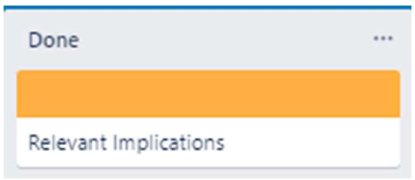
The ethical issues in this program are that the children could take deposits out of their sibling's accounts without them knowing, as there is no segregation between the accounts, no logins or passwords, and no privacy between the three accounts, which is another issue (the lack of privacy regarding the children's financial status). As well as this, other ethical issues include the parents potentially putting restrictions on their children (even when they are older and earn their own money), as the account was created by the parents to monitor their kids' funds. However, as these are the specifications set up by the parents for the app it has been designed as such, and as we don't know the age of the kids or the whether the accounts are set up as a joint account between parents and children, we cannot make any changes to the app.

- *Social:*

This app can be used by people of all races, religions, genders and ages as long as they understand English. However, as its purpose is as a clothing allowance app, the app is more directed at families, i.e either for parents to provide for their kids or for kids to use themselves. As the target audience of this app is more so for families, the app is made to have a nice aesthetic look, which is neutral and works for any particular user. The app is made to be user-friendly and open to people of all sorts of natures in order to attract all sorts of users.

- *Cultural:*

This game is intended for use by people of all backgrounds and cultures (regardless of religious, ethnic or race-based beliefs). However, as all the messages and numbers used are in English, therefore if users plan to use this program do not understand English/cannot write in English or read English they will have difficulty using the program. As the intended use of this app is specifically for Nikau, Hana and Tia this app should work for their purpose. If a user does decide to use the app without knowing English, they would have to alter the messages displayed by the code to the language of their preference. This app is a clothing allowance app, so it is a fairly neutral and formal app not containing any symbols or images. Because of this, no user should find anything used in the program as culturally offensive. This app is designed in such a way as to not off-put users from using the app.



- This task has been completed from the Trello Board

Reflections:

- *Project Management Tools and Techniques:*

Through the usage of a wide range of project management tools including the Trello Board, Gantt Chart, google forms and Flowchart for the code I was able to develop a quality program for a Clothing Allowance App, that doesn't have any sort of bugs, issues and errors within it. This has been made possible through the tools, testing and trialling, planning and feedback I gathered whilst carrying out this project.

The first tool the Gantt chart was very helpful in giving me a gauge of where about I should be in order to complete the project on time and ensuring that each section is completed properly allowing me to carry on to the next part (which in most cases needed the prior aspect to be completed beforehand). The Gantt chart was an effective method to evaluate how the work was being managed, and whether I needed to spend more time to get certain tasks done or if I had to update the chart because certain tasks required more time than expected. This was also helpful because it showed how the timing of one task can affect the other tasks following suit.

The Trello board was also very helpful because it made sure I didn't forget any tasks or the order in which the tasks needed to be done, and increased the efficiency at which I completed tasks. This meant that whilst referring to the Gantt Chart for general information about the tasks needing to be completed in the week, I could refer to the Trello Board to get more specific details about each task and how they related to each other. It also meant that I could move tasks from 'To-do' to 'Doing' and then finally 'Done', this was helpful because it not only showed what I had left to do but also added a sense of accomplishment as I could figuratively tick each task off. This method of planning allowed me to carry out each task chronologically and after finishing the task also reminded me to update the Gantt chart as to when the task was finished, notifying me as to whether I needed to change the time frame of the tasks ahead, if I was on track, or if I needed to work faster on the following tasks to get back on track.

The 'waterfall' project management technique, is a technique where you work on each task in sequential order until each specific task is completed. You can't carry on to the next task before completing the previous task and in doing so it ensures that each task has no errors and all the following tasks which rely on the previous tasks are able to be completed seamlessly. It also reduces time required in revisiting tasks, or fixing issues in previous tasks as everything has already been completed. This method is efficient in that it prevents multiple tasks from being started without being completed and prevents the workload from becoming convoluted. It worked well in developing the GUI code and managing the project as it meant that I couldn't progress without completing each task. This was useful as it forced me to complete tasks to the best standard ensuring there were no bugs, issues or errors before I carried on to the next task. For instance, with the decomposition of the code and managing the collaboration tools, there were times when coming up with the code for a certain component I wanted to trial was difficult and cumbersome as I kept coming up with errors. But rather than skipping on to the next, I stuck it out until competition which meant that everything was done in an orderly fashion without mucking up the sequence (and thus making it more difficult to plan the project); also some of the errors I ended up having in the following iterations of the code were solvable through the trials and techniques I used to fix previous issues in the code. However, this did make certain tasks last longer than expected and increased the timeline of the project, as rather than solving other tasks further ahead in the meanwhile I was stuck on the task at hand. But overall, the benefit of not repeatedly getting stuck on random sections of the project and only one section at a time (which is more manageable) was easier to handle and more efficient.

The flowchart I created when writing the initial base code was helpful in breaking down the entire project into more easy-to-manage pieces. It allowed me to look at the functions, classes and definitions I had to individually create before piecing them together. This was a much easier process through the flow chart as I was simply able to look at the chart to see what function I needed to write the code for, and upon finishing just had to see where this function tied to (with relations to other functions). This also meant that further ahead in the code if I created a function that connected to previous parts of the code I just had to refer to the flowchart and look where it connected before adding the code to the specific sections. It allowed me to keep track of what I was doing at all stages of writing the code, and more easily break down the versions of the code depending on when big changes were made.

The usage of collaboration tools (google forms) to gather feedback was one of the most helpful tools in terms of developing the best possible version of the app. Through the usage of google forms, I was able to gather feedback from peers, stakeholders, teachers, friends and family. It is through their feedback on various aspects of the GUI that I was able to improve, change and redesign the app. The feedback regarding font, font size, layout, structure, and colour theme was immensely helpful in perfecting the app. The usage of google Forms not only allowed me to gather feedback but also statistical evidence regarding the percentages and charts of how well-liked certain designs were making it easier to understand visually. This was all done online so I was able to share various possible designs or ideas, and then demonstrate them and gather feedback online. This made it easier for people to answer in their own time and not have to be physically around or in the class to answer/see the designs.

Using GitHub for version control was also very helpful, as it allowed me to keep all coding files, files relevant to the code and additional files/folders in one easy-to-access place, whilst demonstrating when things had been done, the changes made over time and the branches of the code which shows the trialling that had to be done in order to come to the final error-free version of the code. It also made it easier to get past errors in code, as after trialling one option I didn't have to revert all the changes and see where I went wrong before retrying another method.

Through the use of all these tools and techniques throughout the project, I was able to create a very high GUI and app fit for its purpose and designed to the best of its abilities considering the feedback/input provided by peers, stakeholders, friends, teachers and family. An app that has taken all of the stakeholders' inputs and satisfies them; and carries out all the tasks required to a high standard and without any errors. These tools helped me gather feedback, stay on track and complete each task in an orderly fashion until the task was perfected. Allowing me to carry out the project seamlessly and make it a lot easier to manage. Through gathering feedback, I was able to effectively trial and test out key components of the GUI to improve upon the functionality, usability and aesthetics of the app, to create a modern, sleek and efficient design.

Through the extensive planning done before starting the project, I was able to set a strong base point to start the project. Through multiple class sessions with the teacher and people from Te Kura I got a strong understanding of the tasks needed to be completed in this project and the basic gist of what each task entailed. By using exemplars and documents explaining the project I was also able to gauge a basic frame for each task and set up my own pre-planning schedule. Using this pre-planning schedule I started the outlines of the planning, with the brief, Gantt chart, Trello board, setting up of Github, collaboration tools and decomposition outlines. As I had pre-planned all the tasks required in the project, I had everything already set up before starting and was able to confirm with my teacher if I was on the right path. In doing so also made it quicker and easier to start the initial tasks such as completing the brief, Trello Board, Gantt Chart, discussing the project techniques I'd be using, etc because I already had a good understanding of what I wanted to happen in my project and what I envisioned the end outcomes for the task and the entire project to look like. Without carrying out the strong planning beforehand I would have been severely underprepared when approaching each task scrambling to find out what I need, referring to my teacher and peers constantly and encountering a lot more issues. This would have made carrying out the entire project more time-consuming and annoying. Instead by doing all the planning in one go and asking the teacher what was required in one sitting, I had a better understanding of what was needed and by transposing this into my own planning schedule was less likely to forget what I needed.

This planning also helped me foretell what sort of issues and errors I might encounter and thus prepare for them, by either changing the task or preparing more for it. For example, had I not realised that the code had to be decomposed so that I can see the versions of the code as I make changes to it and can use this to refer to in collaboration tools. I would have had to re-write codes when showing all the different designs I came up with before coming up with the final concept I wanted to test and trial. This would have wasted a lot of time and been repetitive. The planning also meant I had set all the tools required to carry out the project, without those tools I would have been very lost as to what needed to be done, in what order and how it related to other tasks, but by using the Gantt chart, Trello Board and Flow chart all these things were managed in an organised and timely fashion.

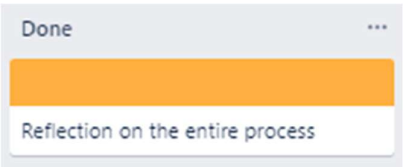
- *Trialling, Testing and Feedback:*

Carrying out extensive trialling and testing of the App was a very important part of this project due to the project management technique being used 'Waterfall'. This technique meant that every task had to be perfect before carrying on to the next task, and by trialling and testing each task I was able to make sure I

was carrying out the ‘Waterfall’ methodology properly. So by carrying out these trials and tests not only was I able to weed out the errors and issues in the code, but also ensure my design had been evaluated from all aspects to form the best design possible. Gathering feedback was also crucial in perfecting the design of the app. As it was through gathering feedback from multiple sources that I was able to gather differing perspectives from my own to as to what other people deemed was a good fit for the app. Allowing me to experiment with new ideas; different font styles, colour schemes, and layouts and trial different codes to get a better design. Thus, increasing the usability and functionality of the app as it is deemed better across a broader range of users. It also provided me with multiple sources to further examine the app for any faults or issues that I had missed with their fresh eyes. So overall the feedback was crucial to perfect the app so that the users of the app would be very satisfied and wouldn’t be the ones who ended up using it only to find heaps of issues and complain about these. For example whilst gathering feedback I received a lot of comments saying that the font size in the initial design at times was hard to read and also lacked any emphasis in the headings or the bank balances. So the user had to search through the entire app to look for something that should be more prominent and vividly shown. Myself having looked at the app so many times it had become fairly obvious to me, but for the respondents having only looked at it once it was glaringly obvious to them that those features weren’t highlighted to drag their attention to the more important details. Thus, by gathering feedback I was able to make these changes to further improve the app. Trialling and testing also meant I was able to work on different ways of solving errors in my code or find more efficient and flexible ways to write the code so it is more adaptable to change. Using GitHub was very helpful for this, as I was able to branch out from the main code to these trials before finding my final solution which I believe to be the most efficient and robust option which was added back to the main code (across the version history).

Through trialling, testing and gathering feedback I was able to make a final design that was far better than the initial designs created beforehand, as these incorporated the thoughts and opinions of the stakeholders, users, peers, teachers and friends not all of whom had programming experience. Meaning it was tested to work with people who had little or no understanding of programming so from more likely users of the app. It incorporated features that made it easier for their experience and brought new ideas which I had not thought about in designing the app. Making the app overall more modern and efficient in its use. By using the collaboration tools to gather feedback I was also able to get a wide range of answers from a range of sources and statistically analyse the responses, so more designs and choices were provided to the user. Without this feedback, the app I would have created would not have been up to the standard it is now, as it would be missing many features and designs which vastly improve the app’s functionality and usability. It’s also likely that the app would have been as well-liked and easy to use by the users, and might have even contained a lot of errors or design mishaps that were overlooked. So the high-quality outcome of a Clothing Allowance App that has been produced now would not have been produced.

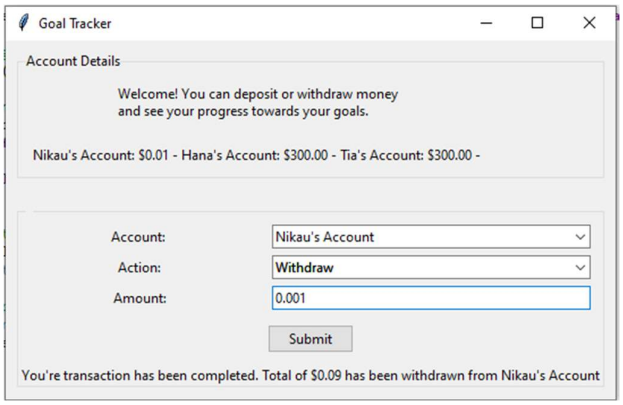
In carrying out this reflection/evaluation of the tools, planning, testing, trialling, and feedback-gathering processes used in carrying out this project I’ve come to learn how important all these things are in creating a successful app. The planning provides a stable base to start on with and helps with managing time, whilst the trialling, testing and gathering of feedback creates different designs, shows different perspectives and provides evidence to show what features are best for the app. Without all these concepts the app would not have been refined and the project would have been a lot harder to carry out.

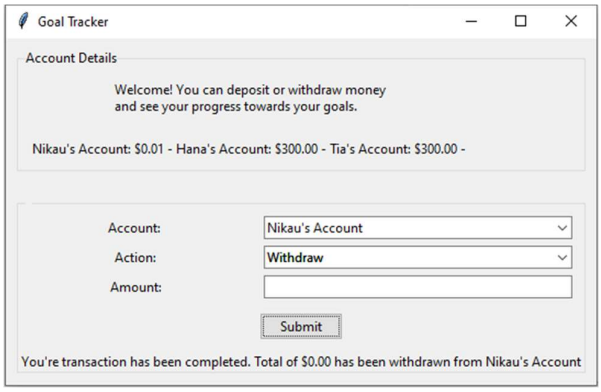


- This task has been completed from the Trello Board

Issues I had writing up the code and how I fixed them:

One of the issues I had whilst writing the earlier version of my code was when the bank account has 1 cent left in it, and a withdrawal of less than a cent, rather than rounding it to the nearest cent, that value is removed from the account leaving the account to look as if it has $0.00 left in it when it has $0.009. So instead, I wanted to change the code, so it rounds the transaction amount to the nearest cent. So, I need to change the code so that only inputs by the user of the smallest value of 2 cents (i.e., all values rounded to 2dp). The screenshot below shows how the transaction input was accepted, and how it carries out the wrong actions.



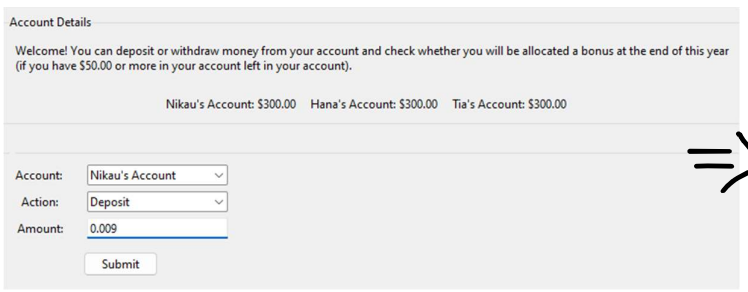


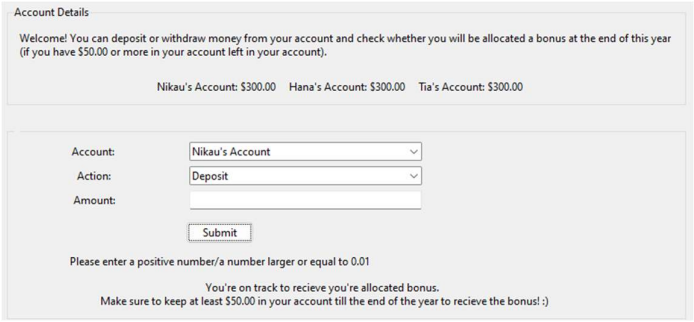

```
# Deposit method adds money to balance
def deposit(self, amount):
    if amount > 0:
        self.balance += amount
        return True
    else:
        return False
```

```
# Deposit method adds money to balance
def deposit(self, amount):
    if amount > 0.1:
        self.balance += amount
        return True
    else:
        return False
```

Feedback I had some issues regarding the withdrawal and received some feedback from my teacher and peers and then I trialled changing the amount values to see whether the inputs would be accepted or rounded

Now any value less than 0.01 cannot be withdrawn or deposited






```
##****************************************  FUNCTIONS AND SETUP ****************************************
# Create a function to read data from the file
def get_data():
    if os.path.exists("clothing_allowance_app_fle.txt"):
        if os.path.isfile("clothing_allowance_app_fle.txt"):
            account_file = open("clothing_allowance_app_fle.txt","r")
            line_list = account_file.readlines()
            for line in line_list:
                account_data = line.strip().split(",")
                Account(*account_data)

            account_file.close()
    else:
        print("File is not present new account file is being set up.")
        account_file = open("clothing_allowance_app_fle.txt","w")
        account_file.write("Nikau's Account,300.0\nHana's Account,300.0\nTia's Account,300.0")
        account_file.close()
```

The issue I had here was that when the new file was created, the definition “get_data” wouldn’t re-kick in so the file’s line’s weren’t read and therefore the account_data was empty. Because of this names_list, account_list and account_names were also empty so when line 141 was run there was nothing in the variable account_names so the code would crash

```
141 chosen_account.set(account_names[0])
142
143 # Create a Combobox to select the account
```

Debug I/O | Python Shell | Exceptions | Debug Console | Modules | Bookmarks | Messages | OS Commands

[pid 24212] trialthing.py (exception) — Ignore this exception location

File "C:\Users\karti\OneDrive - Lynfield College\Documents\13 PAD\Level 3 AS Internal\trialthing.py", line 141, in <module>
chosen_account.set(account_names[0])

builtins.IndexError: list index out of range

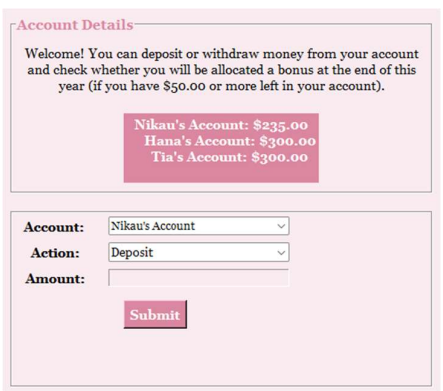
To fix this issue I simply reran the get_data() function by adding it to the end of the else statement, so after the new file is created the get_data() function is manually started. This prevented any further errors.

```
##************************************  FUNCTIONS AND SETUP ************************************
# Create a function to read data from the file
def get_data():
    if os.path.exists("clothing_allowance_app_fle.txt"):
        if os.path.isfile("clothing_allowance_app_fle.txt"):
            account_file = open("clothing_allowance_app_fle.txt","r")
            line_list = account_file.readlines()
            for line in line_list:
                account_data = line.strip().split(",")
                Account(*account_data)

            account_file.close()
    else:
        print("File is not present new account file is being set up.")
        account_file = open("clothing_allowance_app_fle.txt","w")
        account_file.write("Nikau's Account,300.0\nHana's Account,300.0\nTia's Account,300.0")
        account_file.close()
        get_data()
```

Another issue I had with the GUI aspect of the code was when trying to add colour to the GUI, or changing the font style/size, despite the numerous methods I tried from importing fonts, downloading font styles and opening the files within the code, manually setting the colour of each aspect, there was an error with the imports, the version of the python or some other sort of issue. After discussing the issue with some peers I found out that they were also having similar issues and to resolve the issue we searched google for ways to work around it. After a while of searching I found out that by using ttk for all the buttons, frames, comboboxes, labels, etc the colour and font could not be set, and instead I had to change the code so that it used tkinter rather than ttk. After changing the code I was able to set the background colours, fonts and font sizes without any errors in the code.

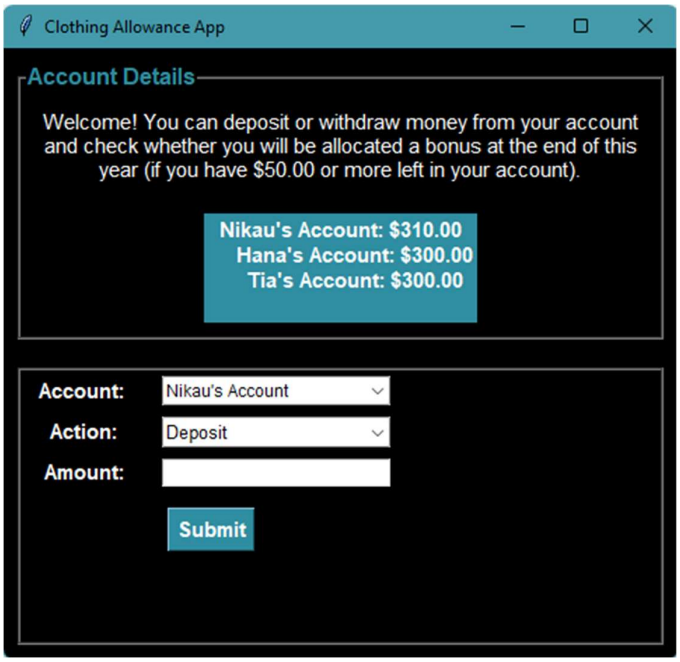
```
root = Tk()
root.title("Clothing Allowance App")

#Create the top frame
top_frame = ttk.LabelFrame(root, text="Account Details")
top_frame.grid(row=0, column=0, padx=10, pady=10, sticky="NSEW")

#Create and set the message text variable
message_text = StringVar()
message_text.set("Welcome! You can deposit or withdraw money from your account ar

#Create and pack the message label
message_label = ttk.Label(top_frame, textvariable=message_text, wraplength=1000)
message_label.grid(row=0, column=0, columnspan=2, padx=10, pady=10)
```

```
##************************************ GUI CODE ************************************************

root = Tk()
root.title("Clothing Allowance App")
root.configure(bg='#000000')

#Create the top frame
top_frame = tkinter.LabelFrame(root, text="Account Details", font=("Lato", 12, "bold"), bg='#000000', f
top_frame.grid(row=0, column=0, padx=10, pady=10, sticky="NSEW")

#Create and set the message text variable
message_text = StringVar()
message_text.set("Welcome! You can deposit or withdraw money from your account\n and check whether you

#Create and pack the message label
message_label = tkinter.Label(top_frame, textvariable=message_text, wraplength=1000, font=("Lato", 11),
message_label.grid(row=0, column=0, columnspan=2, padx=10, pady=10)
```

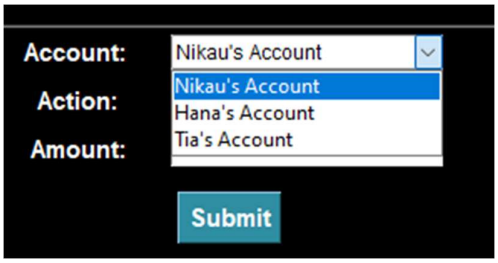


User Guide:

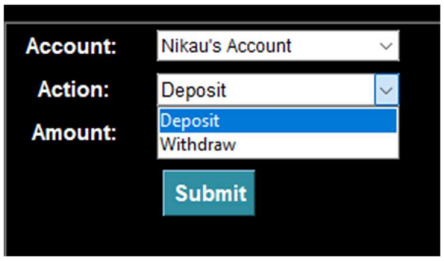
To use this app you start by running the file on python. If the file corresponding with the app doesn't exist the app will create the file before launching. After doing so it will load up the app's home screen. As shown below:



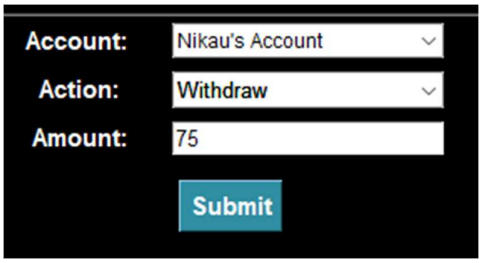
From here the user can read the welcome message, check their account balance, and decide what they want to do. If the user wants to make a withdrawal, they must simply select the account, select an action and input a value for however much the user wants to withdraw. The user must ensure to input a valid number greater than 0.01 and make sure that it's no more than the balance of the account they wish to carry out the transaction from. If the user makes a mistake or types an invalid input they will receive an error message letting them know and the transaction will not go through. However, if the user makes a valid transaction, they will be shown a transaction statement letting them know what action they carried out (deposit/withdrawal) from which account and the amount. Before finally displaying a message letting them know whether they're still on track to receive their allocated bonus at the end of the year. The user is then free to make any number of transactions after that or exit the app. The app will save all transactions that have gone through to the file, updating the bank balances.



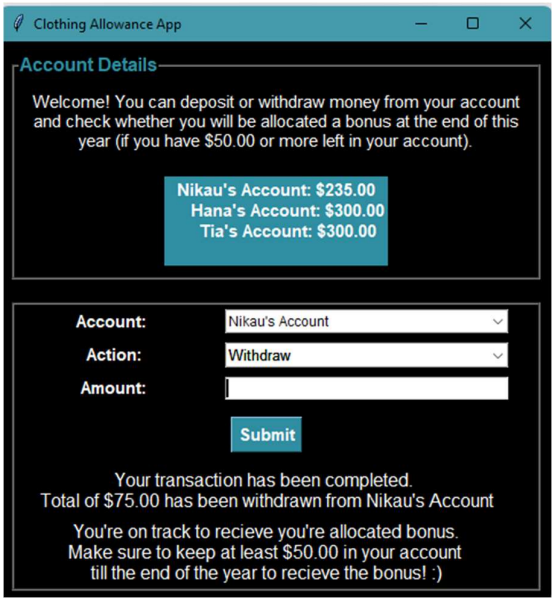
Select the account



Select the action



Input valid amount

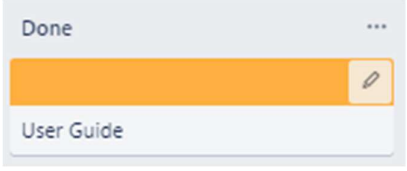


Press the blue submit button for the transaction to go through.

The transaction message will be displayed as well as the bonus message.

The account balance will have also changed.

The user can then exit the app by pressing the x button on the top right corner or carry on with another transaction by repeating the above process.



- This task is completed from the Trello Board