

Readme!

Please feel free to edit this template by adding / editing and removing slides. I've tried to cover most of what you might want to include but there are many ways of providing evidence so don't be afraid to delete slides that you don't need.

Slides with a green background should be used to show evidence of planning / project management. The slide with a purple background should be duplicated and used to show 'relevant implications' evidence.

The instructions in the grey boxes should be deleted once they have been read. Copies of the instructions are underneath the slides in the speaker notes (just in case they are needed at a later stage).

91906 & 91907 Complex Programming

Please add links to your work on this slide. Then remove this instruction. Note that most of the instructions have been supplied as speaker notes for later reference :) **Ensure your Github repositories and Trello board are public.**

Trello board / Project Management:

<https://trello.com/invite/b/xHlvpQnb/ATTI882ea659b518d107bf733b3e79d332a6B5491A3F/digital-storage-unit-convertor>

The project on Github:

https://github.com/BingheLi/2024_Digital_Storage_Unit_Convertor

Final Project Filename: <ie: Completed_v3>

Addressing Relevant Implications

Use slides with this colour background to show that you have addressed the relevant implications. To do this easily, remove this instruction and then duplicate this slide.

Implications text goes here

Addressing Relevant Implications-Functionality

Functionality is crucial to my program because it ensures that the program operates as expected, with each component performing its intended tasks. Proper functionality is essential for providing a positive user experience, as any malfunction would prevent users from utilizing the program effectively. Each component should be able to handle unexpected or boundary inputs from users. To address this, I continuously tested each component to ensure it works as intended, and clearly display error message when results are unexpected, this ensure the user can continue using the program without the unexpected outcome breaking the program. I also addressed functionality by the implementation of the dropdown boxes to select unit. Users can easily select the units to convert from and to. This simplifies the selection process, as instead of having to type in all the units they want to convert, which could causes issues if units are entered incorrectly or in a format the program does not recognise. Having this dropdown menu allow user to easily and quickly to complete the conversion process. I have also addressed functionality with my history import/export component. The two components are two way communicated. Meaning that the exported file from history export component can be imported back with the history import component. This allow user to work with the conversion history in a easier way, as user don't have to manually repeat all the previous conversion process.

Addressing Relevant Implications-Usability

Another important aspect of my program is usability, which determines to how easily users can navigate and interact with the program. A clear and user-friendly interface is essential to guide users through the program and keep them engaged. Good usability enhances user-friendliness and makes the overall experience more enjoyable, as users won't have to spend time figuring out how to use the program, therefore avoiding confusion. I addressed this by using descriptive labels like 'From:' and 'To:' that can clearly communicate with the user and guide them through the conversion process. I also use large bold text with contrasting colours to its background in my program so that it can clearly communicate with the user, this allow users with vision problems such as colour-blindness or unclear vision to easily pick up on the content of my program, improving their user experience. I also addressed this by implementing clear error messages, for example if user try to convert with the empty entry grid or invalid value(non-numerical), a error message will show up and tell them what they did wrong and what to enter. I also addressed usability by implementing a instruction gui, where the user can click on the instruction button to bring up a new window with instruction explain the functionality of each button and how to use the program.

Addressing Relevant Implications-Aesthetics

Another important consideration is aesthetics, which is about the design and appearance of the program. This is crucial because it affects user experience and willingness to use the program. If the design is dull or uninspired, users may be less engaged, while an overly vibrant design can also be off-putting. I addressed this by maintaining a balance, keeping the program simple yet visually appealing. My approach to this is using high contrast colour theme but also not too overly colourful. I used Light cyan(#EAF6FF) and light black(#232528), the two colours are based of high contrasting white and black which suits the seriousness of a conversion tool, but both have a shade of blue in them, creating a contrasting harmony. This approach ensures the program isn't just filled with text, but has enough color and imagery to enhance the user experience. Layout of button are all centralised and symmetric, with buttons all aligned. This make the program more visually appealing and easier to navigate. I also use large sized bold text with contrasting colour to its background, this ensures the user can read the text properly and also makes all the components and conversion results stand out.

Graphical User Interface Design...

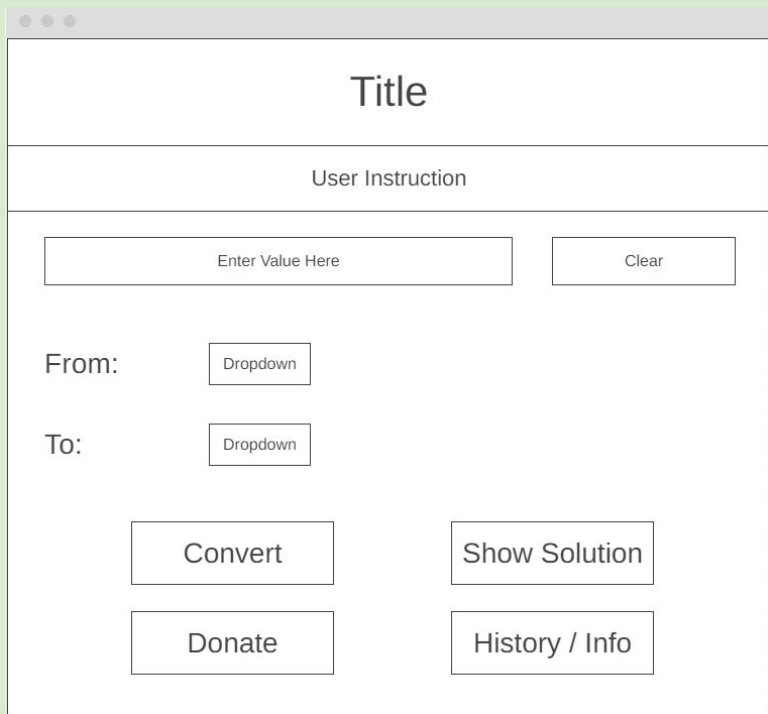
Main UI (scratch)

The scratch UI design for the Digital Storage Unit Converter features a window with a title bar and a close button. The title is "Digital Storage Unit Converter". Below the title, there are two input fields labeled "From :" and "To :", each with a dropdown arrow. To the right of these fields is a vertical double-headed arrow icon. Below the input fields is a text input field labeled "Values to Convert :". At the bottom, there are four buttons: "Clear", "Answer", "Help/Info", and "History/Export".

The polished UI design for the Digital Storage Unit Converter features a window with a title bar and standard window controls (minimize, maximize, close). The title is "Digital Storage Unit Converter". Below the title, there are two input fields labeled "From :" and "To :", each with a dropdown arrow. To the right of these fields is a vertical double-headed arrow icon. Below the input fields is a text input field labeled "Values to Convert :". At the bottom, there are four buttons: "Clear" (red), "Solution" (yellow), "Answer" (green), and "History/Export" (yellow-green). The "Solution" button is larger and more prominent than the others.

Graphical User Interface Design...

Main UI v2



Title

User Instruction

Enter Value Here

Clear

From: Dropdown

To: Dropdown

Convert

Show Solution

Donate

History / Info

#FFE4E1

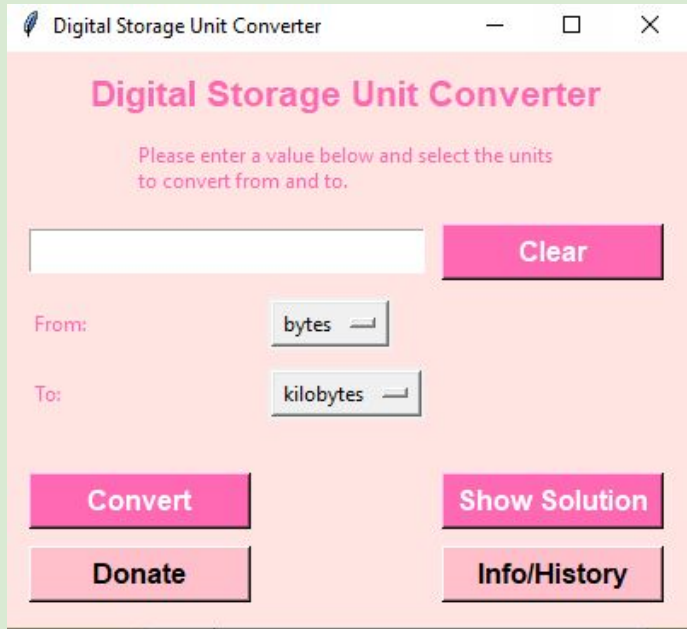


#FF69B4



Graphical User Interface Design...

Main UI v2 proof



Digital Storage Unit Converter

Please enter a value below and select the units to convert from and to.

From:

To:

#FFE4E1



#FF69B4

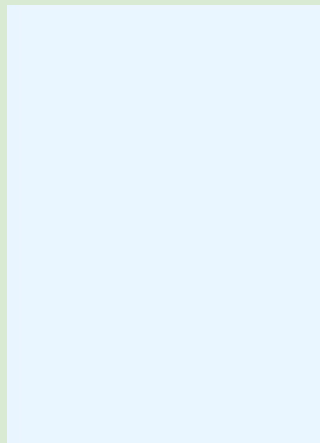


Graphical User Interface Design...

Final user interface design

The wireframe shows a window with a title bar containing three dots and the word "Title". The main content area is divided into two sections. The top section contains a "Value:" label, a text input field, a "Clear" button, a "From:" label, a text input field, and a "To:" label, also with a text input field. The bottom section contains five buttons: "Instructions" and "Show Solution" in the top row, "Convert" in the center, and "History / Import" and "History / Export" in the bottom row.

#eaf6ff



#232528



Graphical User Interface Design...

Final UI

Digital Storage Unit Converter

Value:

Clear

From:

bytes

To:

kilobytes

Instructions

Show Solution

Convert

History / Import

History / Export

#eaf6ff

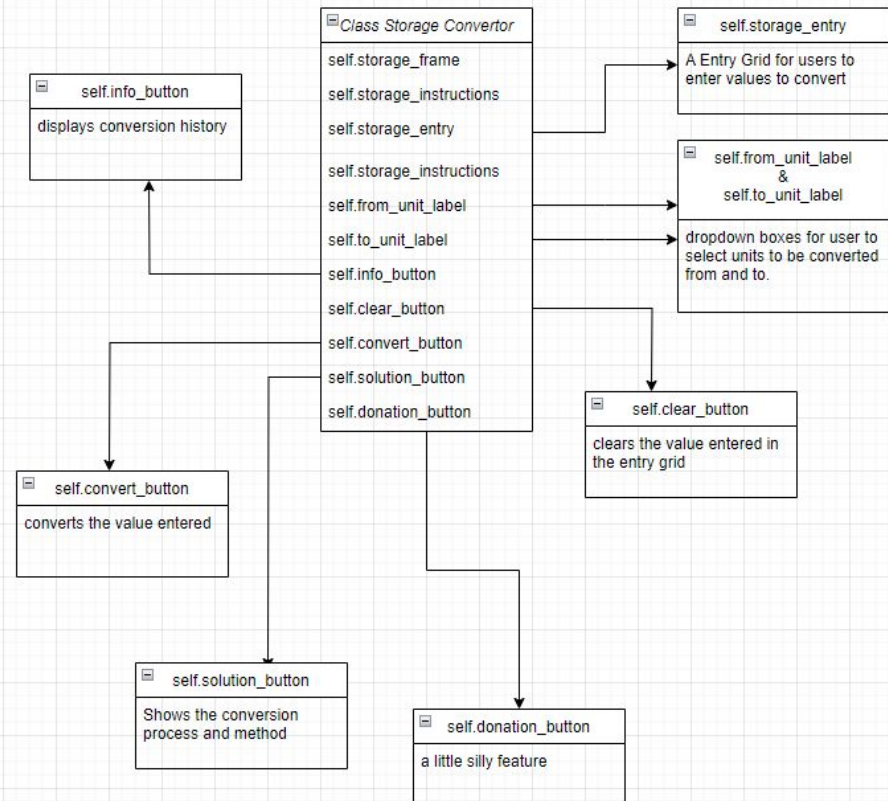


#232528



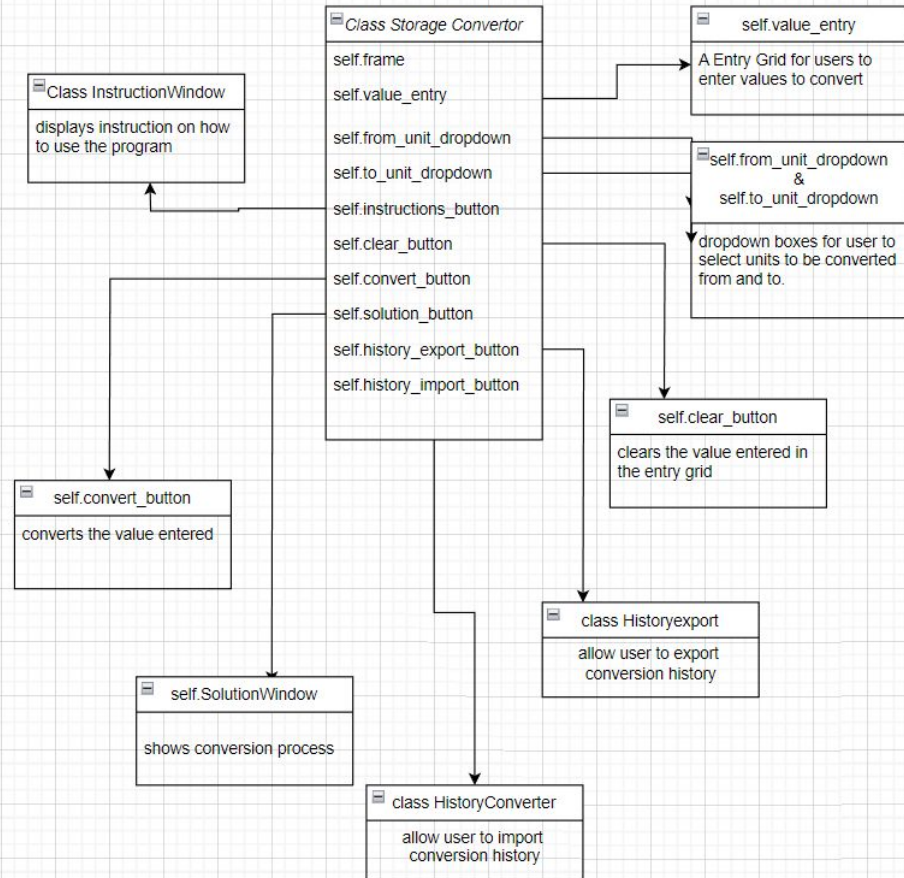
Program Structure

Program Structure Plan

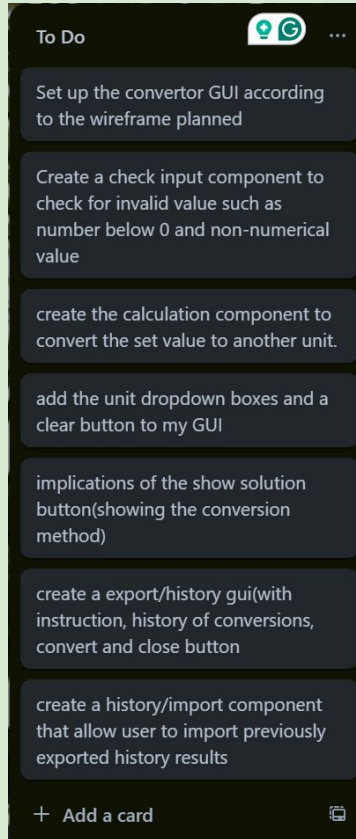


Program Structure

Program Structure -
finished

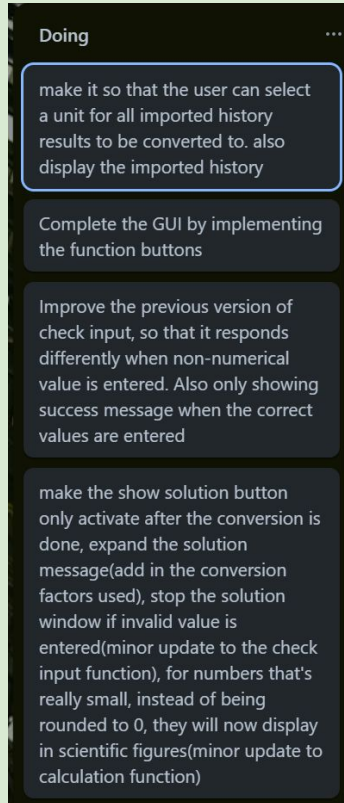


Problem Decomposition



Initial decomposition of the program with all the planned components

Problem Decomposition



update the calculation component so it converts the ENTERED value to another unit.

implicating the write_to_file (exporting) function to history/export window

make a instruction component where it explains the functionality of each buttons and how to use the program

Improving and finishing up all my component to improve usability and functionality

Develop your Components!!

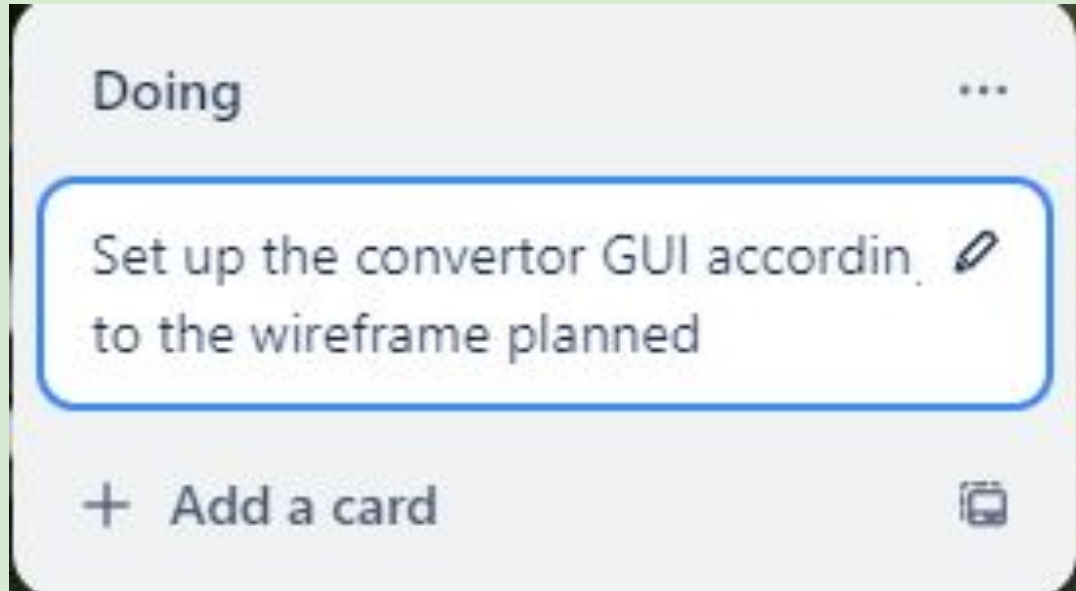
Make as many copies of the next three slides as you need. For each component you should have...

- A trello screenshot / evidence of ongoing use of your project management tools
- A test plan for the component that has been created BEFORE you start coding. Your plan should allow you to test all logical pathways for this component. It should also include test cases for relevant boundary and unexpected values.
- Screenshot evidence showing that you have worked through the test plan you developed.

You should also provide evidence of trialling multiple components & techniques. Please make slides as needed for that evidence.

Component Planning...

Convertor GUI v1



Component Test Plan <Edit Me>

Test Case	Expected Outcome
Title of the GUI	Display on the top of the GUI
Instruction	Display below the Title
Entry Grid	A Entry Grid for user to enter a value

Component Trialling

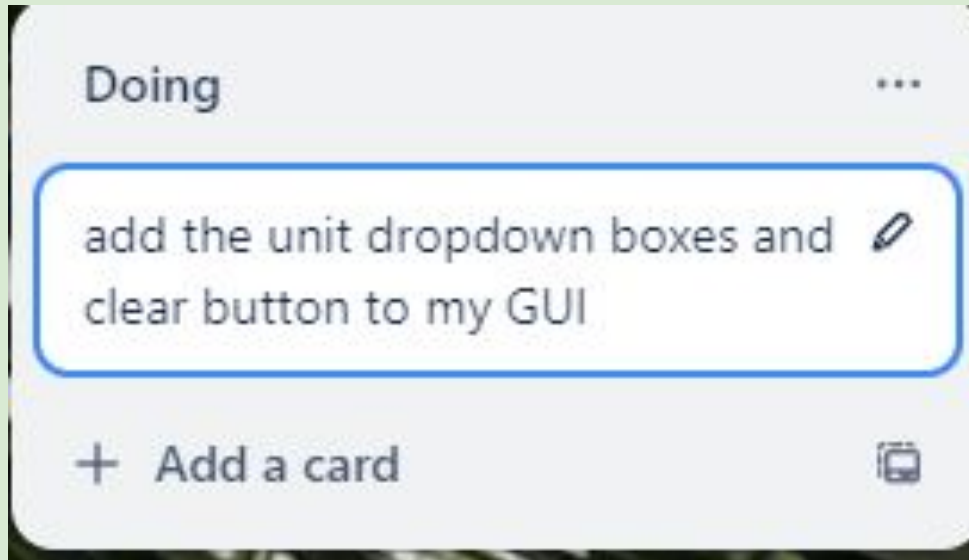
The trialling results meets my expected outcome and programs runs according to plan.



The image shows a screenshot of a web application window titled "Digital Storage Unit Converter". The window has a light pink background. At the top, there is a title bar with the text "Digital Storage Unit Converter" and standard window control buttons (minimize, maximize, close). Below the title bar, the text "Digital Storage Unit Converter" is displayed in a large, bold, pink font. Underneath this, a smaller pink text prompt reads: "Please enter a value below and select the units to convert from and to,". At the bottom of the visible area, there is a white rectangular input field for entering a value.

Component Planning...

Convertor GUI v2

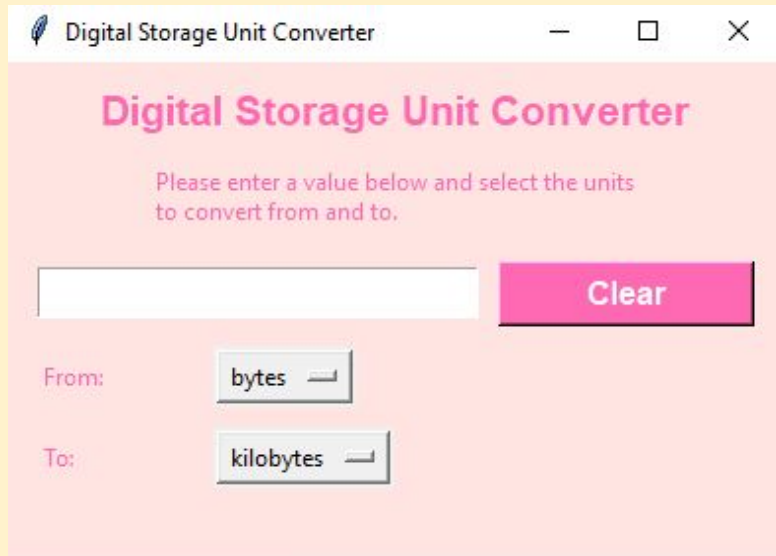


Component Test Plan <Edit Me>

Test Case	Expected Outcome
clear button	A clickable clear button display next to the entry grid
From Unit dropdown box	dropdown box that shows the units when clicked
To unit dropdown box	dropdown box that shows the units when clicked

Component Trialling

The trailing results meets my expected outcome and programs runs according to plan.



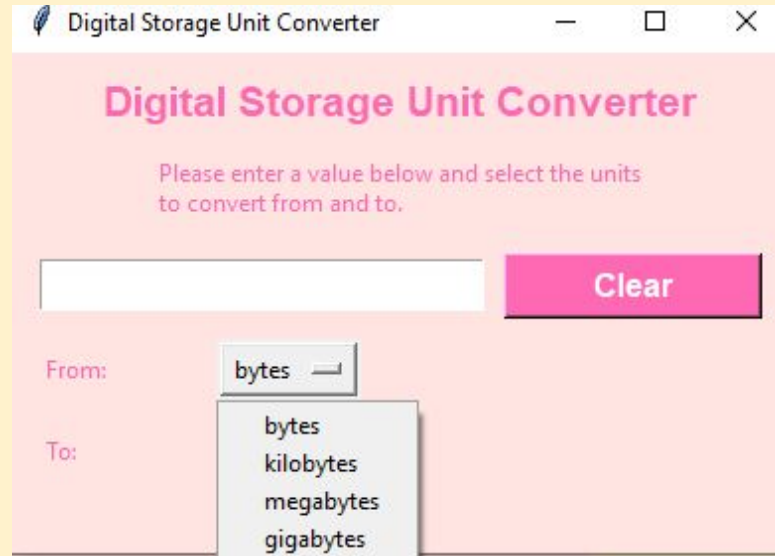
Digital Storage Unit Converter

Please enter a value below and select the units to convert from and to.

Clear

From: bytes

To: kilobytes



Digital Storage Unit Converter

Please enter a value below and select the units to convert from and to.

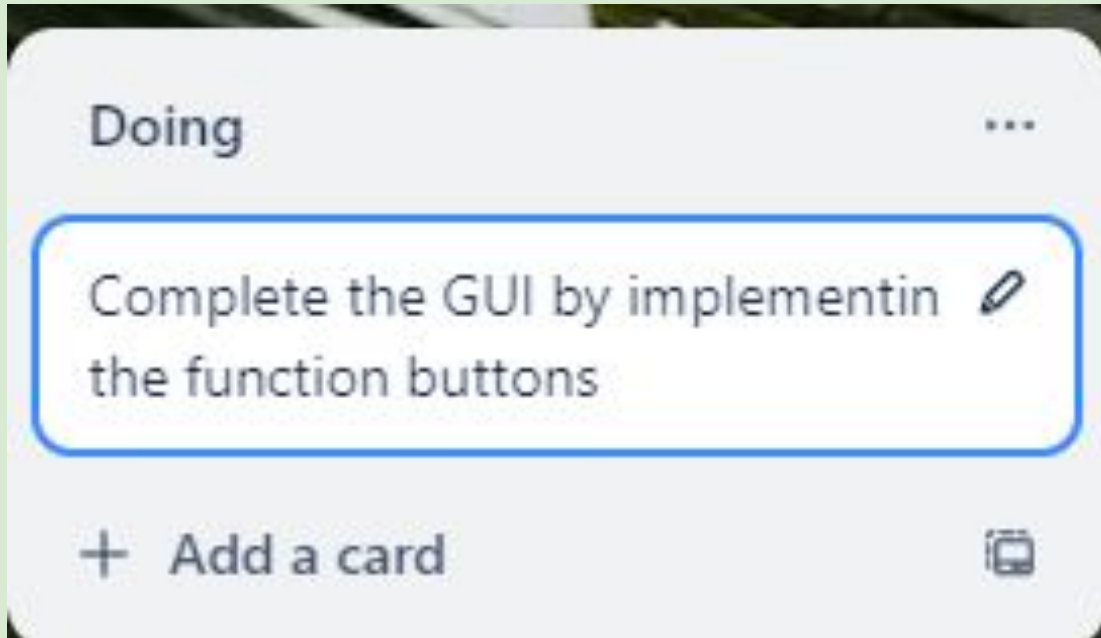
Clear

From: bytes

To: bytes
kilobytes
megabytes
gigabytes

Component Planning...

Convertor GUI v3

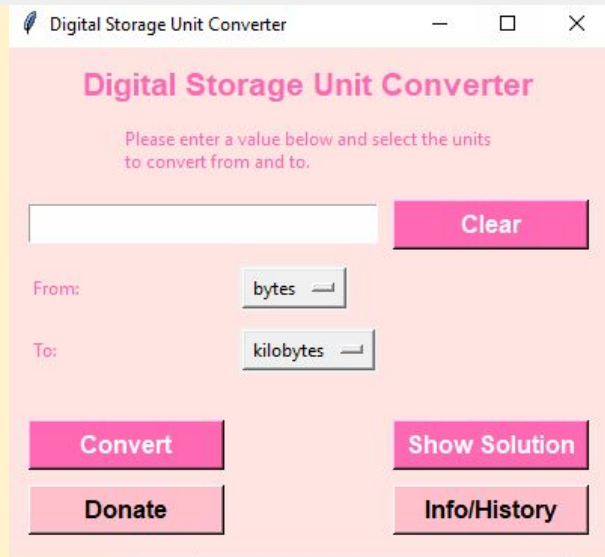


Component Test Plan <Edit Me>

Test Case	Expected Outcome
Convert	A clickable button
Show Solution	A clickable button
Donate	A clickable button
Info/History	A clickable button

Component Trialling

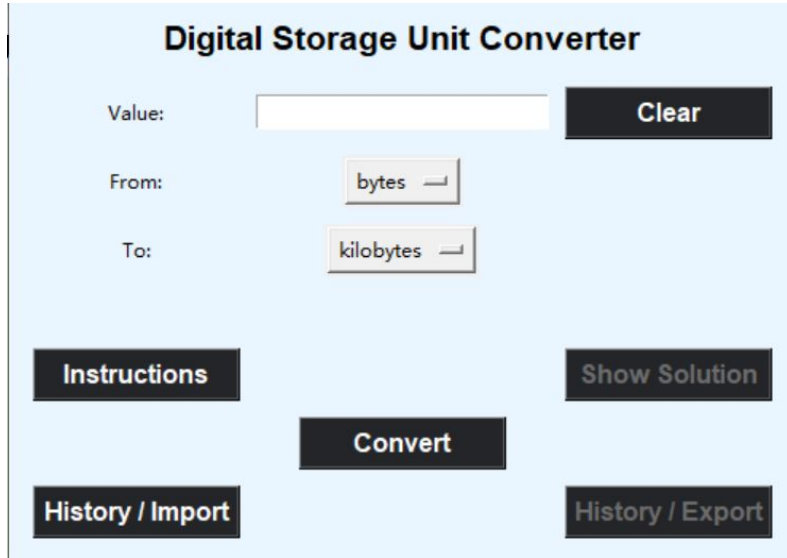
The trailing results meets my expected outcome and programs runs according to plan.



The screenshot shows a web application titled "Digital Storage Unit Converter" in a browser window. The application has a pink background and contains the following elements:

- Title Bar:** "Digital Storage Unit Converter" with standard window controls (minimize, maximize, close).
- Header:** "Digital Storage Unit Converter" in bold pink text.
- Instructions:** "Please enter a value below and select the units to convert from and to."
- Input Field:** A white text box for entering a value.
- Clear Button:** A pink button labeled "Clear" next to the input field.
- From Unit:** A label "From:" followed by a dropdown menu currently showing "bytes".
- To Unit:** A label "To:" followed by a dropdown menu currently showing "kilobytes".
- Convert Button:** A pink button labeled "Convert".
- Show Solution Button:** A pink button labeled "Show Solution".
- Donate Button:** A pink button labeled "Donate".
- Info/History Button:** A pink button labeled "Info/History".

Final Converter GUI



The image shows a screenshot of a web-based application titled "Digital Storage Unit Converter". The interface is set against a light blue background. At the top, the title "Digital Storage Unit Converter" is displayed in a bold, black font. Below the title, there is a form with three main input areas: "Value:", "From:", and "To:". The "Value:" field is a text input box, and next to it is a "Clear" button. The "From:" field is a dropdown menu currently showing "bytes". The "To:" field is a dropdown menu currently showing "kilobytes". Below these input fields, there are four buttons arranged in a grid: "Instructions" (top left), "Show Solution" (top right), "Convert" (center), and "History / Export" (bottom right). Additionally, there is a "History / Import" button located at the bottom left. All buttons have a dark grey background with white text.

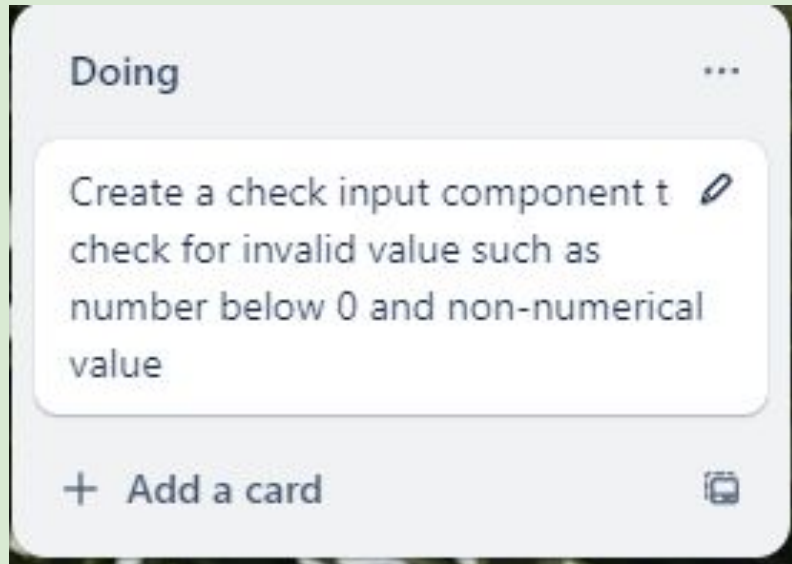
For my final complete program, I made some minor changes to improve the Aesthetics and Usability of the program.

Aesthetics: layout of the program is now more centralised, colour theme will no longer be overly colourful pinky colour.

Usability: got rid of the brief instruction and replaced it with a separated and more detailed instruction, can be accessed by clicking the instruction button

Component Planning...

Check Input V1



Component Test Plan <Edit Me>

Test Case	Expected Outcome
123	Success
qweqw	Please enter a number that is more than 0
-1233	Please enter a number that is more than 0
;:;[Please enter a number that is more than 0

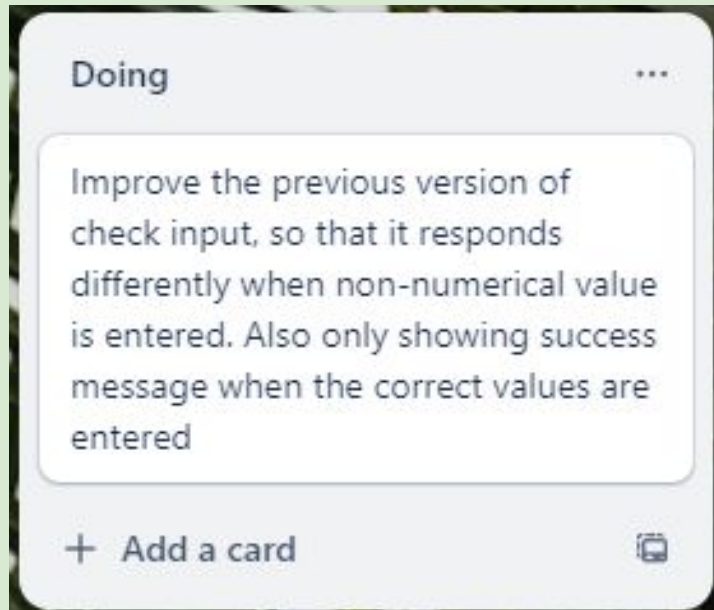
Component Trialling

The trailing results are slightly different to my expected outcome, I wasn't expecting the 'success message to show after every test case. But it will be fixed in the next version.

```
Choose a number: 123
Success
Choose a number: qweqw
Please enter a number that is more than 0
Success
Choose a number: -1233
Please enter a number that is more than 0
Success
Choose a number: ;];][
Please enter a number that is more than 0
Success
```

Component Planning...

Check Input V2



Component Test Plan <Edit Me>

Test Case	Expected Outcome
123	Success
qweqw	Error: Please enter a numerical value.
-1233	Please enter a number that is more than 0
;];[Error: Please enter a numerical value.

Component Trialling

The trailing results expected.

```
Choose a number: aweqw
Error: Please enter a numerical value.
Choose a number: -1233
Please enter a number that is more than 0
Choose a number: ;];][
Error: Please enter a numerical value.
Choose a number: 123
Success
```

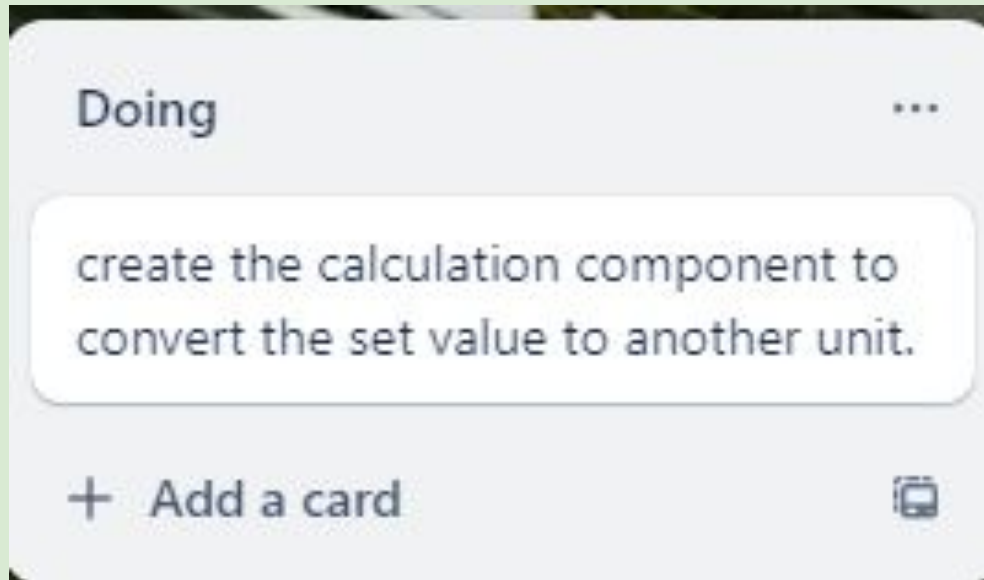

Check Input

The present check input component has been abandoned from the final finished program. It has been integrated into the calculation component.

Details on slide 41

Component Planning...

Calculation GUI v1

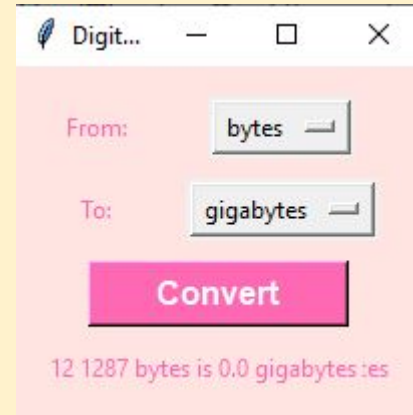
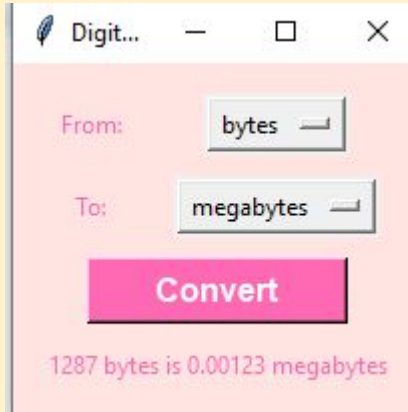
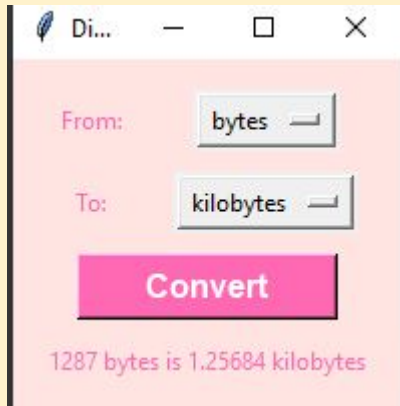


Component Test Plan <Edit Me>

Test Case	Expected Outcome
1287 bytes to kilobytes	1.25684 kilobytes
1287 bytes to megabytes	0.00123 megabytes
1287 bytes to Gigabytes	1.287E-6 Gigabytes

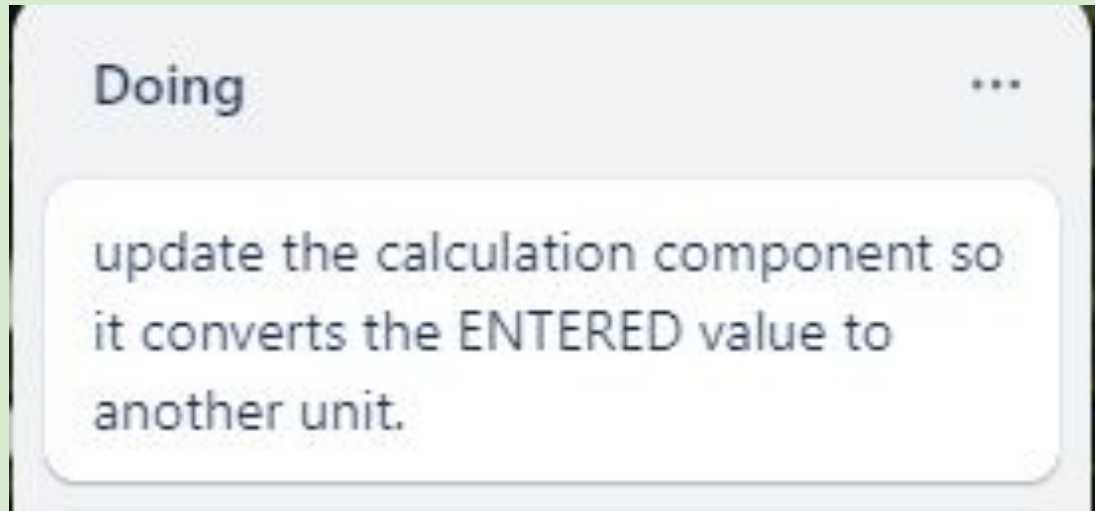
Component Trialling

The trailing results are slightly different to my expected outcome, I wasn't expecting the '0.0 gigabytes' result, However it makes sense as converting from bytes to gigabyte is too much a difference. And my program is set to round to 5th decimal places. So it rounds the result to 0.



Component Planning...

Calculation GUI v2

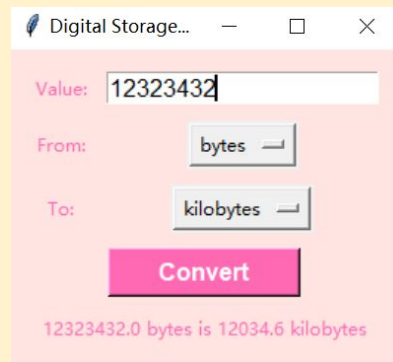


Component Test Plan <Edit Me>

Test Case	Expected Outcome
12323432 bytes to kilobytes	12034.6 kilobytes
5123345 bytes to megabytes	4.89 megabytes
74234325 bytes to Gigabytes	0.7 Gigabytes
1234 kilobytes to bytes	1263616 kilobytes

Component Trialling

The trailing results are the same as my expected outcome.



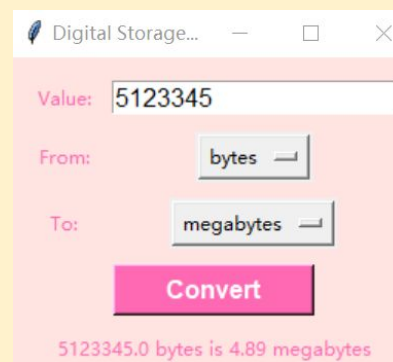
Digital Storage... — □ ×

Value:

From:

To:

12323432.0 bytes is 12034.6 kilobytes



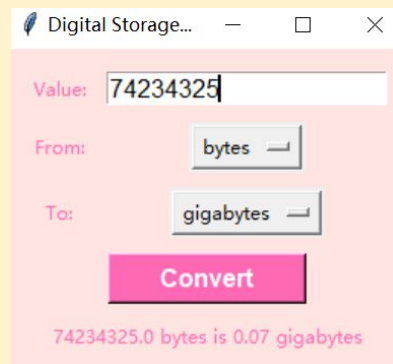
Digital Storage... — □ ×

Value:

From:

To:

5123345.0 bytes is 4.89 megabytes



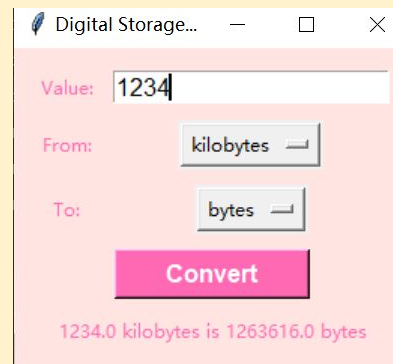
Digital Storage... — □ ×

Value:

From:

To:

74234325.0 bytes is 0.07 gigabytes



Digital Storage... — □ ×

Value:

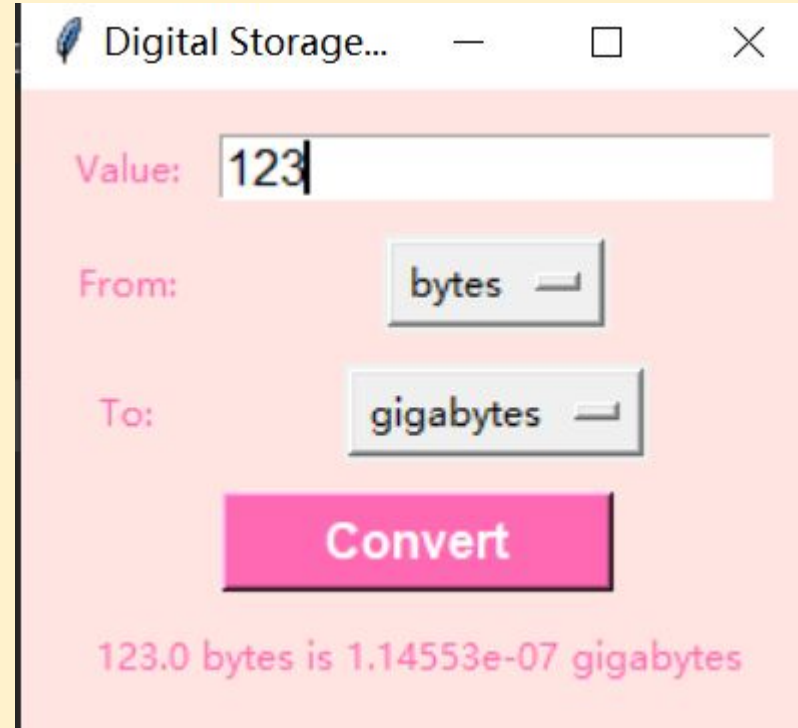
From:

To:

1234.0 kilobytes is 1263616.0 bytes

Extra minor update(from show solution function)

Values that are smaller than 5 decimal places will no longer be rounded to 0, instead they will display in scientific figures



A screenshot of a software window titled "Digital Storage...". The window has a light pink background and a white title bar with standard window controls. It contains a text input field labeled "Value:" with the number "123" entered. Below this are two dropdown menus: "From:" set to "bytes" and "To:" set to "gigabytes". A large pink button labeled "Convert" is positioned below the dropdowns. At the bottom of the window, a line of text displays the conversion result: "123.0 bytes is 1.14553e-07 gigabytes".

Digital Storage...

Value: 123

From: bytes

To: gigabytes

Convert

123.0 bytes is 1.14553e-07 gigabytes

Integration with check_input

Test Case	Expected Outcome
Empty entry grid	Input field is empty. Please enter a number.
Negative number	Invalid input. Please enter a positive number.
Non-numerical value	Invalid input. Please enter a valid number.

Digital Storage Uni... — □ ×

Value:

From:

To:

Input field is empty. Please enter a number.

Digital Storage Unit ... — □ ×

Value:

From:

To:

Invalid input. Please enter a positive number.

Digital Storage U... — □ ×



Value:

From:

To:

Invalid input. Please enter a valid number.

Debugging of calculation GUI

```
▼  conversion_factors = {dict: 4} {'bytes': 1, 'kilobytes': 1024, 'megabytes': 1048576, 'gigabytes': 1073741824}
  01 'bytes' = {int} 1
  01 'kilobytes' = {int} 1024
  01 'megabytes' = {int} 1048576
  01 'gigabytes' = {int} 1073741824
  01 '__len__' = {int} 4
> ? Protected Attributes
  01 converted_value = {float} 0.1201171875
  01 formatted_value = {str} '0.12012'
  01 from_unit = {str} 'bytes'
>  self = {CalculationGUI} <__main__.CalculationGUI object at 0x000001C305B0D910>
  01 to_unit = {str} 'kilobytes'
  01 value_to_convert = {float} 123.0
```

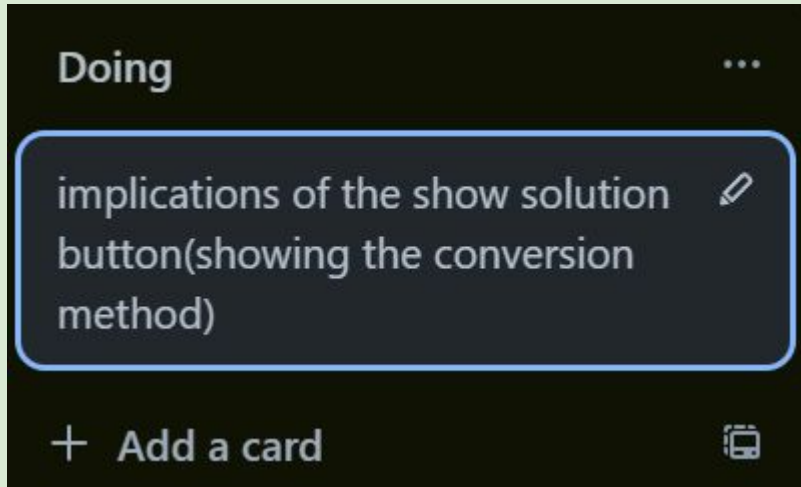
During the debugging process I can see how the program identify the units selected and use its relative conversion factor to convert the entered value

In this case:

value to convert=123
From unit=bytes
To units=kilobytes
Converted value=0.1201171875
Formatted value(results displayed)=0.12012

Component Planning...

Show Solution v1

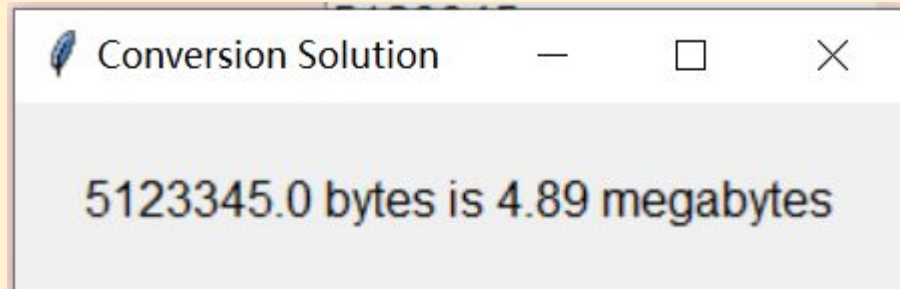
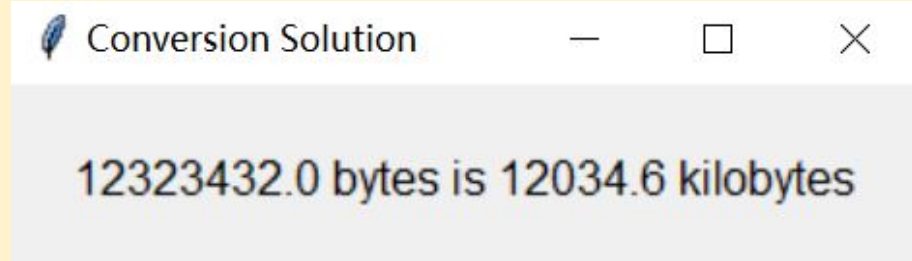


Component Test Plan <Edit Me>

Test Case	Expected Outcome
12323432 bytes to kilobytes	12034.6 kilobytes
5123345 bytes to megabytes	4.89 megabytes

Component Trialling

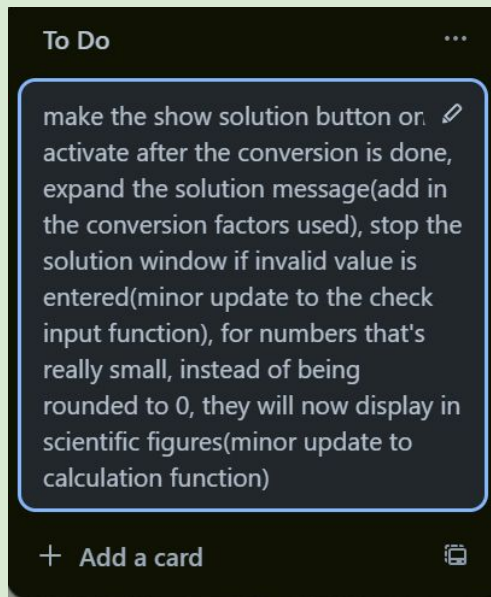
The trailing results are the same as my expected outcome.



Component Planning...

Show Solution v2

Huge update woohoo!

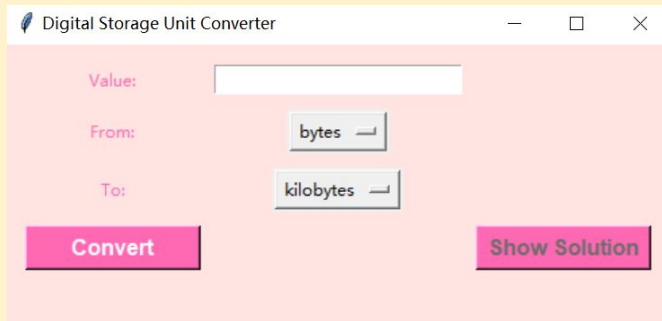


Component Test Plan <Edit Me>

Test Case	Expected Outcome
Before conversion	Show solution button disabled
After conversion	Show solution button activated
Entry grid empty	Display error message, disable show solution button
Converted value smaller than 5 decimal places	Display in scientific figures
Show solution window	Show conversion factors used

Component Trialling

The trailing results are the same as my expected outcome.



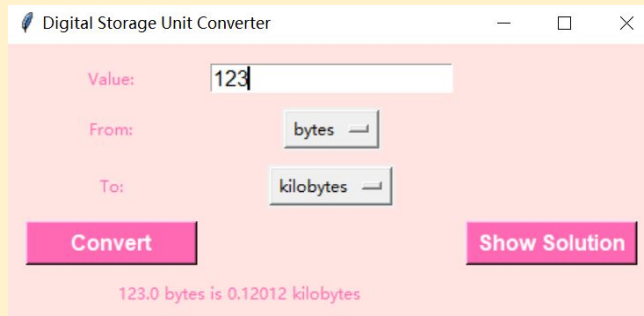
Digital Storage Unit Converter

Value:

From: bytes

To: kilobytes

Convert **Show Solution**



Digital Storage Unit Converter

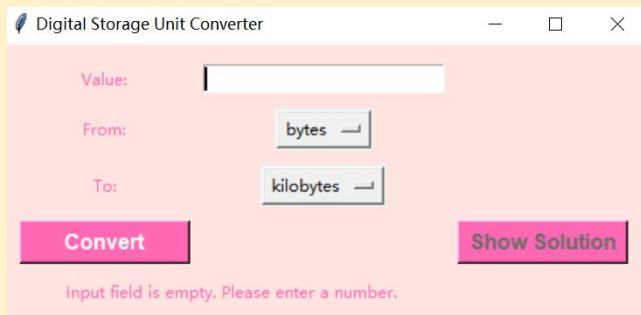
Value:

From: bytes

To: kilobytes

Convert **Show Solution**

123.0 bytes is 0.12012 kilobytes



Digital Storage Unit Converter

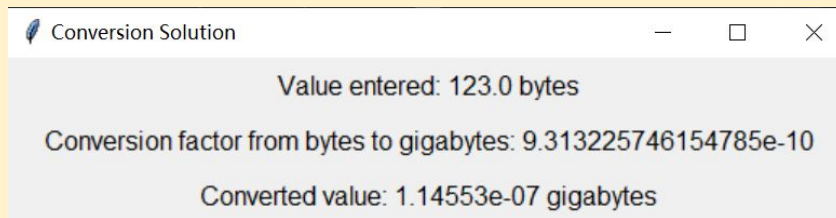
Value:

From: bytes

To: kilobytes

Convert **Show Solution**

Input field is empty. Please enter a number.



Conversion Solution

Value entered: 123.0 bytes

Conversion factor from bytes to gigabytes: 9.313225746154785e-10

Converted value: 1.14553e-07 gigabytes

Debugging of show solution gui

```
> 01 conversion_factor = {float} 0.0009765625
> 01 conversion_factors = {dict: 4} {'bytes': 1, 'gigabytes': 1073741824, 'kilobytes': 1024, 'megabytes': 1048576}
> 01 converted_value = {float} 0.1201171875
> 01 formatted_value = {str} '0.12012'
> 01 from_unit = {str} 'bytes'
> 01 self = {StorageConverter} <__main__.StorageConverter object at 0x0000029886CF72C0>
> 01 to_unit = {str} 'kilobytes'
> 01 value_to_convert = {float} 123.0
```

During the debugging process I can see how the program identify the variable that is going to display on the solution window

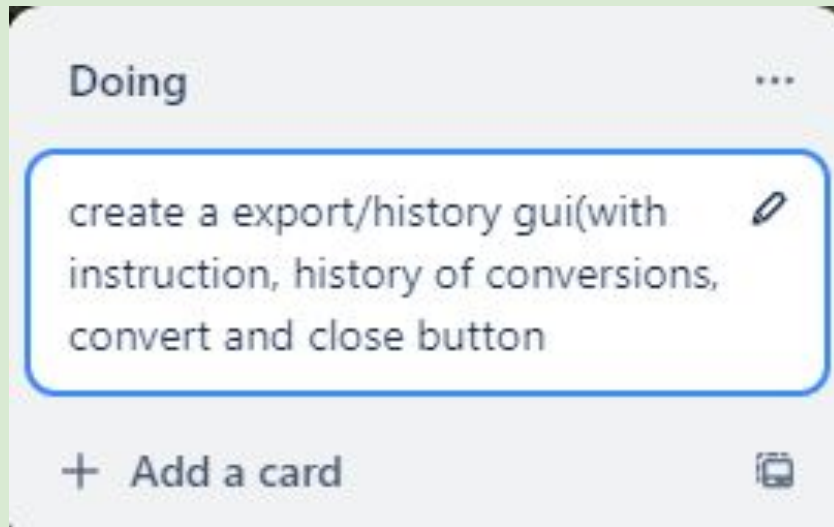
In this case:

value to convert=123
From unit=bytes
To units=kilobytes
Formatted value(results displayed)=0.12012
Conversion_factor = 0.0009765625

```
self.solution_details = [
    f"Value entered: {value_to_convert} {from_unit}",
    f"Conversion factor from {from_unit} to {to_unit}: {conversion_factor}",
    f"Converted value: {formatted_value} {to_unit}"
]
```

Component Planning...

History/ Export_v1



Component Test Plan <Edit Me>

Test Case	Expected Outcome
Before conversion	History/ export button disabled
After conversion	History/ export button activated Conversion history displayed in history export window
Entry grid empty and click convert button (history/export window)	Show error message “file name cannot be empty”
Dismiss button	Closes the history export window
Entry grid filled and click convert button(history/export window)	Show success message “history exported successfully”

Component Trialling

The trailing results are the same as my expected outcome.

Digital Storage Unit Converter

Value:

From:

To:

Convert **Show Solution** **Export/History**

Digital Storage Unit Converter

Value:

From:

To:

Convert **Show Solution** **Export/History**

123.0 bytes is 0.12012 kilobytes

Digital Storage Unit Converter

History / Export

Below is your calculation history.

All calculations are shown to the nearest degree.

123.0 bytes is 0.12012 kilobytes

Either choose a custom file name (and push <Export>) or simply push <Export> to save your calculations in a text file. If the filename already exists, it will be overwritten!

Export **Dismiss**

Digital Storage Unit Converter

History / Export

Below is your calculation history.

All calculations are shown to the nearest degree.

123.0 bytes is 0.12012 kilobytes

Either choose a custom file name (and push <Export>) or simply push <Export> to save your calculations in a text file. If the filename already exists, it will be overwritten!

Filename cannot be empty.

Export **Dismiss**

Digital Storage Unit Converter

History / Export

Below is your calculation history.

All calculations are shown to the nearest degree.

123.0 bytes is 0.12012 kilobytes

Either choose a custom file name (and push <Export>) or simply push <Export> to save your calculations in a text file. If the filename already exists, it will be overwritten!

History exported successfully!

Export **Dismiss**

Component Planning...

History/ Export_v2

Doing

...

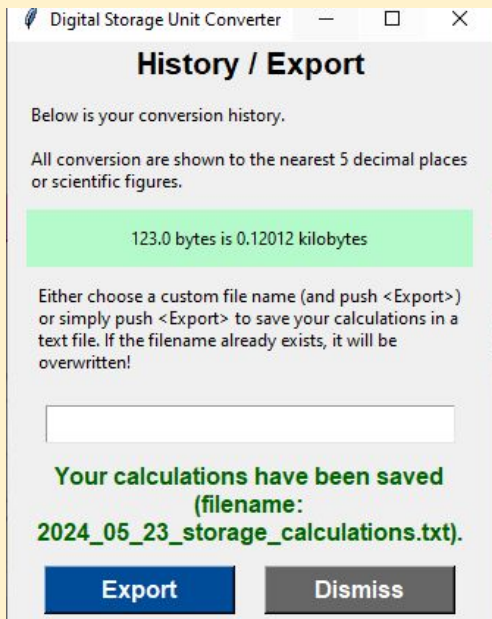
implicating the write_to_file
(exporting) function to history/
export window

Component Test Plan <Edit Me>



Test Case	Expected Outcome
Entry grid filled and click convert button (history/export window)	Success message “ Your calculations have been saved(filename: whatever the user entered in entry grid.txt)
Entry grid empty and click convert button (history/export window)	Success message “ Your calculations have been saved(filename: folder location.txt)
After click export button	History txt file saved in the folder History txt file show time and date the file is saved, and all conversion results are all aligned

Component Trialling

The trailing results are the same as my expected outcome.



```
**** Digital Storage Calculations ****
Generated: 23/05/2024
Here is your calculation history (oldest to newest)...
123.0   bytes   0.12012 kilobytes
```

 123	23/05/2024 10:31 am	Text Document	1 KB
 2024_05_23_storage_calculations	23/05/2024 10:31 am	Text Document	1 KB

Debugging of show history/export gui

During the debugging process I can see how the program stores the conversion and how many has been made .

In this case:

- Calc_list = 123.0 bytes is 0.12012 kilobytes',
 - '123.0 gigabytes is 125952.00000 megabytes
- Max_calcs = 50
- Num_calcs = 2

```
10 calc_background = (str) '#B4FACB'
01
> calc_list = (list: 2) ['123.0 bytes is 0.12012 kilobytes', '123.0 gigabytes is 125952.00000 megabytes']
10 calc_string_text = (str) '123.0 gigabytes is 125952.00000 megabytes\n123.0 bytes is 0.12012 kilobytes'
01
10 hist_text = (str) 'Below is your conversion history. \n\nAll conversion are shown to the nearest 5 decimal places or scientific figures.'
01
10 max_calcs = (int) 50
01
10 num_calcs = (int) 2
01
> partner = (StorageConvertor) <__main__.StorageConvertor object at 0x000001AAA0022C00>
10 save_text = (str) 'Either choose a custom file name (and push <Export>) or simply push <Export> to save your calculations in a text file. If the filename already exists, it will be overwritten!'
01
> self = (HistoryExport) <__main__.HistoryExport object at 0x000001AAA0050DA0>
10 showing_all = (str) 'Below is your conversion history.'
```


Component Planning...

History/ Import_v1

Doing

...

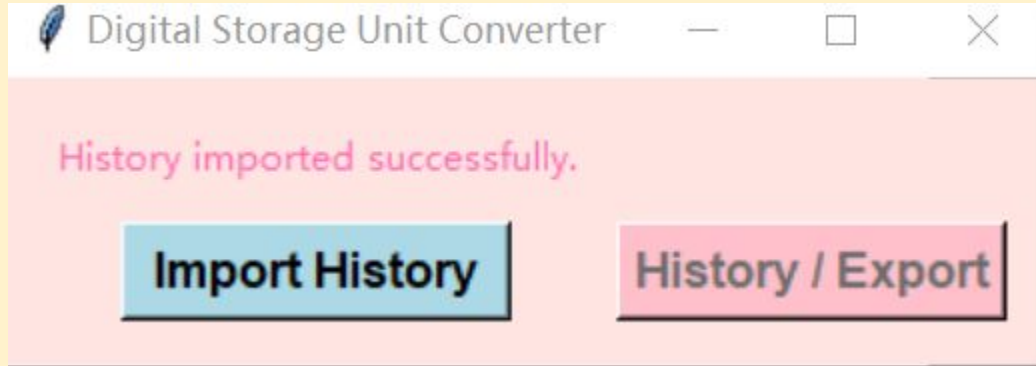
create a history/import component
that allow user to import previously
exported history results

Component Test Plan <Edit Me>

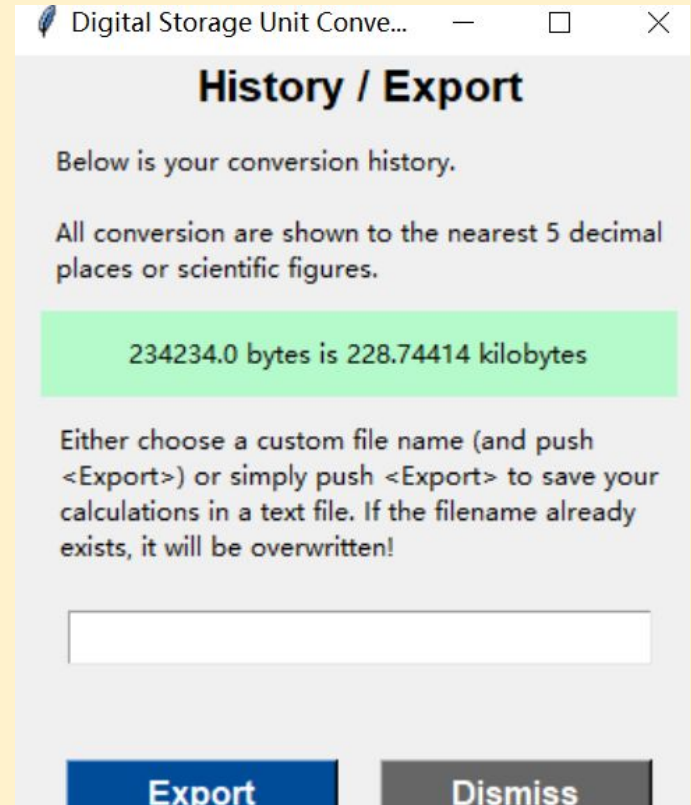
Test Case	Expected Outcome
Click on the history/import button	File explorer opens and ask the user to select the history file to be imported
After import	Show success message”history imported successfully” Imported history also display in the history grid in the history export gui

Component Trialling

The trailing results are the same as my expected outcome.

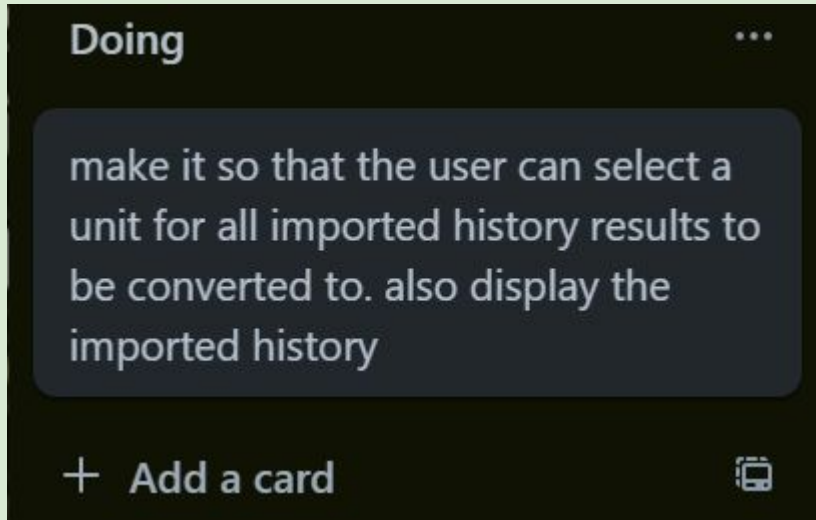


<https://drive.google.com/file/d/19X39xRfxVNmznlsus9gFcTI-596dM68/view?usp=sharing>



Component Planning...

History/ Import_v2



Component Test Plan <Edit Me>

Test Case	Expected Outcome
Click on the history/import button	Open history/import gui
Click on import button	File explorer opens and ask the user to select the history file to be imported
Click on export button	File explorer opens and ask the user to enter the name of the txt file to be saved as History txt file saved in the folder History txt file show time and date the file is saved, and all conversion results are all aligned(same format as the history export component)

Component Trialling

The trailing results are the same as my expected outcome.

<https://drive.google.com/file/d/1LaoN9PSwA3AnPZFQDvAuP5PZMBONuP1L/view?usp=sharing>

Component Testing Evidence <Edit Me>

Complete Program...

Assemble your components into a working program. On the slides that follow, please provide a test plan and evidence that your program works as expected.

If you are going for M / E, you also need to create extra slides showing how you have used your testing to improve the functionality of your program. This could mean having multiple test plans (and screenshots) showing several iterations of the assembled program.

Complete Program_v1

From the testing of the complete program_v1 and from user feedback i got from my peers, I was told the visual appearance of my program doesn't suit the purpose of a Digital Storage Unit Convertor, that the pinky colour theme seems to playful and unserious for a tool. So I decided to use high contrast colour theme but also not too overly colourful. I used Light cyan(#EAF6FF) and light black(#232528), the two colours are based of high contrasting white and black which suits the seriousness of a conversion tool, but both have a shade of blue in them, creating a contrasting harmony.

Complete Program_v1 <test plan>

Test plan:

- Run the program and everything should load as expected(buttons, entry boxes, dropdown boxes)
- Entry grid should allow user to enter any values
- When click on the convert button, if entry grid is empty or filled with non numerical values, a error message should be displayed. Only if a numeric value is entered the conversion result would display
- Both dropdown boxes should allow user to freely select the units they want
- Show solution button should be disabled before any conversion is made, and enabled after a conversion is made. When click on it, a new window should appear and shows the step process to the conversion.
- History/export should be disabled before any conversion is made, and enabled after a conversion is made. When click on it, a new window should appear and shows the previous conversion history. And below you can enter a name in the entry grid for the exported history txt file. There is also a dismiss button where you can use it to close the window.
- When click on the history/import button, a new window should appear, on this window you can import back in your previous exported history txt file by clicking on the import history button. On the right of this button, there is a drop down box where you can select a unit for all the results you just imported back in, this will allow you to convert all of the results you just imported back in them to another unit with just one click on the convert and export button.

Complete Program_v1 <testing evidence>

<https://drive.google.com/file/d/1Pozyp-3diYfQgX7NZv4B4DxhpZ6X82iG/view?usp=sharing>

Complete Program_v2

From the testing of the complete program_v1 and from user feedback i got from my peers, I was told that there weren't enough instructions or informations that tells him how to use the program. So i decide to add a instruction button, and when you click on the button it will open a instruction window that explains the functionality of each button and how to use the program

Complete Program_v2 <test plan>

Test plan:

- Run the program and everything should load as expected(buttons, entry boxes, dropdown boxes)
- Entry grid should allow user to enter any values
- when you click on the instruction button it will open a instruction window that explains the functionality of each button and how to use the program
- When click on the convert button, if entry grid is empty or filled with non numerical values, a error message should be displayed. Only if a numeric value is entered the conversion result would display
- Both dropdown boxes should allow user to freely select the units they want
- Show solution button should be disabled before any conversion is made, and enabled after a conversion is made. When click on it, a new window should appear and shows the step process to the conversion.
- History/export should be disabled before any conversion is made, and enabled after a conversion is made. When click on it, a new window should appear and shows the previous conversion history. And below you can enter a name in the entry grid for the exported history txt file. There is also a dismiss button where you can use it to close the window.
- When click on the history/import button, a new window should appear, on this window you can import back in your previous exported history txt file by clicking on the import history button. On the right of this button, there is a drop down box where you can select a unit for all the results you just imported back in, this will allow you to convert all of the results you just imported back in them to another unit with just one click on the convert and export button.

Component Planning...

Instruction gui

Doing

...

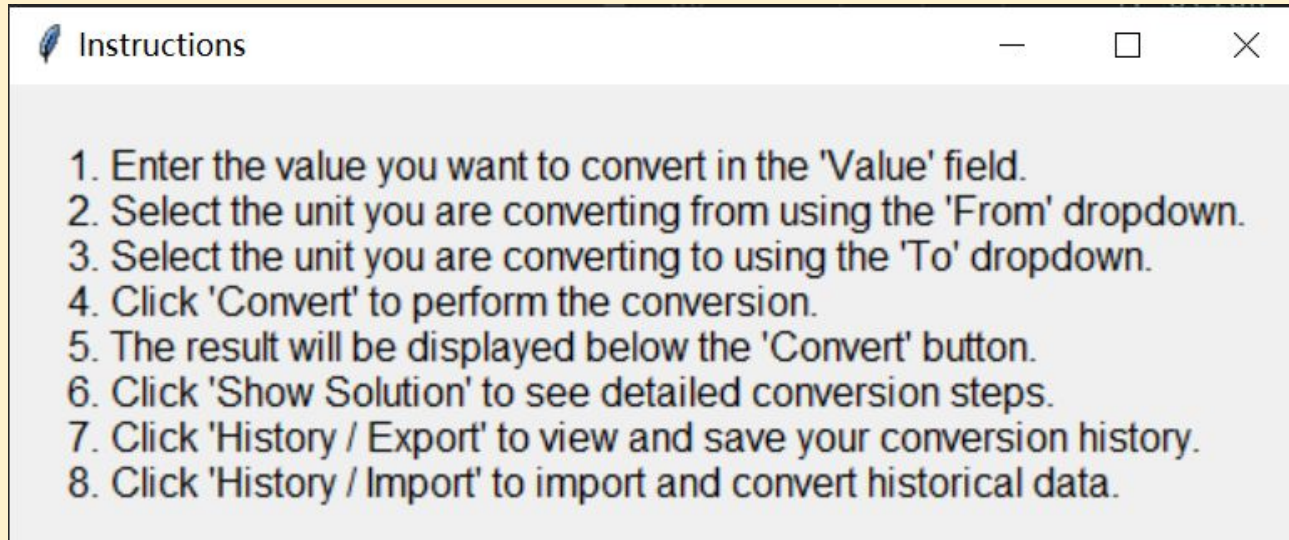
make a instruction component where
it explains the functionality of each
buttons and how to use the program

Component Test Plan <Edit Me>

Test Case	Expected Outcome
Click on the instruction button	Display a new window that has all the instruction on there

Component Trialling

The trailing results are the same as my expected outcome.



Complete Program_v2 <testing evidence>

https://drive.google.com/file/d/1_2V8FIRwN8lcYpyJU55ShbGs9_-CFy0/view?usp=sharing

Complete Program_v3

From the testing of the complete program_v2 and from user feedback i got from my peers, I found out that my history/export gui won't allow certain keywords that reserved for system file of Windows(con, con1, etc). so I will make some changes to the history export component that detects those windows reserved keyword, and display error messages when these keywords are entered. I was also told that most of my component gui can be opened many times, resulting a lot of component window being opened and slowing down the program. This will significantly affect the usability and functionality of the program. So i will make some changes to all the GUIs and only allow one to be opened.

Complete Program_v3 <test plan>

Test plan:

- Run the program and everything should load as expected(buttons, entry boxes, dropdown boxes)
- Entry grid should allow user to enter any values
- when you click on the instruction button it will open a instruction window that explains the functionality of each button and how to use the program
- When click on the convert button, if entry grid is empty or filled with non numerical values, a error message should be displayed. Only if a numeric value is entered the conversion result would display
- Both dropdown boxes should allow user to freely select the units they want
- Show solution button should be disabled before any conversion is made, and enabled after a conversion is made. When click on it, a new window should appear and shows the step process to the conversion.
- History/export should be disabled before any conversion is made, and enabled after a conversion is made. When click on it, a new window should appear and shows the previous conversion history. And below you can enter a name in the entry grid for the exported history txt file, except special symbols and windows reserved keywords. There is also a dismiss button where you can use it to close the window.
- When click on the history/import button, a new window should appear, on this window you can import back in your previous exported history txt file by clicking on the import history button. On the right of this button, there is a drop down box where you can select a unit for all the results you just imported back in, this will allow you to convert all of the results you just imported back in them to another unit with just one click on the convert and export button.
- all component GUIs can only have one opened at a time.

Complete Program_v3 changes

I listed the reserved names, and set that if the filename is one of the reserved names, the program will return a error message that say “sorry, {file name} is a system reserved name and cannot be used.

```
reserved_names = {"con", "aux", "nul", "prn", "com1", "com2", "com3", "com4", "com5", "com6", "com7", "com8",  
                  "com9", "lpt1", "lpt2", "lpt3", "lpt4", "lpt5", "lpt6", "lpt7", "lpt8", "lpt9"}  
if filename.lower() in reserved_names:  
    problem = f"Sorry, '{filename}' is a system reserved name and cannot be used."
```

Complete Program_v3 changes

```
class SolutionWindow:
    window_open = False

    def __init__(self, parent, solution_details):
        if SolutionWindow.window_open:
            return
        SolutionWindow.window_open = True

        self.solution_window = Toplevel(parent)
        self.solution_window.title("Conversion Solution")
        self.solution_window.protocol(name="WM_DELETE_WINDOW", self.on_close)

        for detail in solution_details:
            Label(self.solution_window, text=detail, font=("Arial", 12)).pack(padx=20, pady=5)

! usage
def on_close(self):
    SolutionWindow.window_open = False
    self.solution_window.destroy()
```

This code introduces class-level flags to ensure that each type of window can only be opened once at a time. The flags are set to 'true' when a window is opened and reset to 'false' when it is closed, preventing multiple instances of the same window from being opened simultaneously.

Complete Program_v3 <testing evidence>

<https://drive.google.com/file/d/1iG61lBguPeqkIfSlz4K3ROHrLq8LdVpp/view?usp=sharing>

Version Control



- calculation_v1.py
- calculation_v2.py
- check_input_v1.py
- check_input_v2.py
- completed.py
- completed_v2.py
- completed_v3.py
- convertor_gui_v1.py
- convertor_gui_v2.py
- convertor_gui_v3.py
- history_export_v1.py
- history_export_v2.py
- history_import_v1.py
- history_import_v2.py
- instruction.py
- show_solution_v1.py
- show_solution_v2.py

I have uploaded all versions of my component and main program onto github. Doing so helps me to spot and fix up errors I made during the development process. Most importantly, these different versions of components shows how my program has been improved.

I made individual slides about major changes I made in different version of components, With each version, I added new functionality and eliminated errors or problems that occurred during testing. In the final version, the program is fully refined and functions perfectly, handling both expected and unexpected cases.

Relevant Implications Addressed

You might wish to use this slide to explain how you have addressed the relevant implications if you have not done so in previous slides.

Comments go here

Complex Processes - Discussion

In the planning phase of my program, I explored multiple ideas that I could expand on with the components I learned from tutorials and decided on making a Digital Storage Unit Converter. I first created a program structure diagram using wireframe.cc, which provided a clear visual of the complete program. This diagram helped me set clear goals and understand the components I needed to develop. Additionally, I used a Trello board to plan out and record tasks for each component, which was instrumental in keeping track of my progress and reminding me what to do next. By breaking down the project into manageable tasks and setting deadlines, I could efficiently manage the trialling, testing, and refinement processes.

During the initial development phase, I integrated components I was familiar with from tutorials to ensure basic functionality. These components were tested and debugged during the tutorial phase, which gave me confidence in their reliability. This approach streamlined the development process, reducing the time needed for research and allowing me to focus more on testing and refining new components. Implementing known components not only enhanced the program's functionality but also improved usability by allowing me to dedicate more time to making these components user-friendly with clear instructions.

Complex Processes - Discussion

As I developed the program, I also researched and integrated new components, such as dropdown boxes for unit selection. There were many instances where integrating new components with the pre-existing ones posed challenges. For example, ensuring that the dropdown selections interacted correctly with the calculation functions required continuous testing and refinement. I used a combination of debugging tools and user testing to identify and fix issues, ensuring each component's functionality and user-friendliness.

User feedback was crucial in refining the program. During the user testing phase, a peer pointed out that the program lacked clear instructions, which left them confused about the purpose of each button. Based on this feedback, I implemented a detailed instruction GUI that explains the function of each button, significantly improving the program's usability. This addition ensured that future users would have a better understanding of how to navigate and use the program effectively. Additionally, I refined the color theme of the program based on user suggestions to enhance its aesthetics, making it more visually appealing and easier to use. My peer also pointed out that the history/export GUI does not permit certain reserved keywords used by Windows system files (e.g., con, con1). To address this issue, I updated the history export component to detect these reserved keywords and display appropriate error messages when such keywords are entered. Additionally, I was informed that multiple instances of the GUI components can be opened simultaneously, leading to performance issues and reducing the program's usability and functionality. To resolve this, I modified all GUI components to ensure only one instance can be opened at a time. These changes aim to enhance the overall user experience and ensure the program functions smoothly.

Complex Processes - Discussion

Reflecting on my development process, I realized that the planning phase could have been more efficient. Initially, I experimented with ideas like a donation function and other features that were ultimately unnecessary and time-consuming. These trials did not contribute to the final product and diverted my focus from more critical components. If I could redo this project, I would use that time to work on eye-catching and useful components or conduct more testing to further improve the functionality and usability of the program.

Another aspect I would change if I had more time would be to consolidate all the GUI elements into a single window. My current program consists of multiple GUI windows for different components, which can be confusing and annoying for users who have to deal with numerous pop-up windows. Combining these into a single window would streamline the user experience, making it more intuitive and enhancing the overall aesthetics of the program.

Despite these challenges, something I did well in the development process was the creation and integration of each component. Each component was thoroughly tested individually to ensure smooth operation, which made the final integration process straightforward. Consistent commenting in my code and comprehensive testing of each version of the components helped in quickly identifying and resolving any issues that arose during integration. This preparation ensured that combining the components into a fully functional program was a simple process of integrating defined functions into a shared class.

Complex Processes - Discussion

The thorough planning and use of project management tools like Trello and wireframe.cc played a crucial role in managing the project efficiently. They helped in organizing tasks, tracking progress, and incorporating user feedback effectively. The integration of user suggestions and continuous testing and refinement of components ensured the development of a high-quality digital technology outcome. Reflecting on my process, while there were areas for improvement, the overall project was successful due to the thorough planning, testing, and incorporation of feedback. These steps ensured that the final product was functional, user-friendly, and aesthetically pleasing.

Reflecting on my development process, I recognize that the iterative approach of planning, testing, and refining was crucial in achieving a high-quality outcome. The use of project management tools and the incorporation of user feedback were particularly effective in addressing usability and functionality issues. While there are always areas for improvement, such as consolidating GUI elements and refining the initial planning phase, the overall project demonstrates a successful application of digital technology development processes.

Complex Processes - Discussion

Aesthetics were also a key consideration. I used a high-contrast color theme to ensure readability and visual appeal. The color scheme consisted of light cyan (#EAF6FF) and light black (#232528), chosen for their high contrast and harmonious look. Buttons were centrally aligned and symmetric, enhancing the visual structure of the program. Large, bold text with contrasting colors was used to make the content easily readable, which is particularly important for users with visual impairments.

Functionality was a primary focus throughout the development process. I continuously tested each component to ensure it performed its intended tasks correctly. For example, the dropdown boxes for selecting units simplified the user experience by eliminating the need for manual input, reducing the risk of errors. The history import/export components were designed to allow users to easily save and retrieve their conversion history, enhancing the program's functionality and user convenience.

To address the usability aspect, I made sure the program was easy to navigate. I used descriptive labels like 'From:' and 'To:' to guide users through the conversion process. I also implemented clear error messages, such as notifying the user if they try to convert with empty entry fields or invalid values. These error messages helped users understand what went wrong and how to correct their input, improving their overall experience with the program.