UNIVERSITY OF WINCHESTER

BS1220 Introduction to Software Design & Development

Summative – Portfolio

University of Winchester – Business, Law and Digital Technologies
BSc (Hons) Computer Science

Important Information for this Summative:

- This is the written report.
- A single python file 'prototype.py' should accompany this document.

Grade A: 72%

Student ID: 2101168

Version 1.0.4

Word Count: 3,432 / 3,500
Monday, 23rd May 2022

Contents

1 Project Initiation

1.1 Introduction

A department store provides customer quotes for gift wrapping; however, it is assumed that this is done manually as an employee would measure the size of the present or given the measurement and workout the price of this using a standard calculator. This system would need to be improved for efficiency, as manually calculating these prices uses up the customer's time.

1.2 Scope

Defining the scope of this project is important to prevent scope creep – which is 'devoting project time and resources to unapproved changes' according to Santos (2021). To prevent this, deliverables have been created to stick to a goal without changing it.

For this project, an application will be developed in Python using the tkinter GUI framework to provide a quick and easy way to calculate prices of the department's gift-wrapping services. This will include the ability to choose between three present types, two wrapping paper types, six colours, and whether the customer wants to add a bow and/or gift card with a message. This application will show a preview of the wrapping paper with the colour applied. This application should include functionality to have multiple quotes in one order as well as the ability to save the quotes to an external file.

1.3 Deliverables

For this development cycle, an application with a graphical user interface must be created. This should be delivered to Dr Claire Ancient by the 23rd of May 2022. Failure to do so will continually hold back the department in terms of efficiency.

1.4 Acceptance Criteria

Before the start of development, the system will need to have a proposed methodology, as well as requirements to break down the project into stages. In addition, the system will need to be well designed to give a guided path to the creation of final application. This will need to be approved before starting the programming aspect of the application.

1.5 Exclusions

The project will not include the ability to handle payments or purchase items and will only be used to show multiple quotes within an order. This system will neither have the functionality to interact with the banking service that customer wishes to use.

1.6 Constraints

One constraint is that Python's tkinter module must be used as it the default interface framework that does not require any extra extensions or installation. This will make it easy to maintain the compatibility between different operating systems that the parent company of the department wishes to use.

1.7 Assumptions

One main assumption is that the application will only be used by the employees working within the department and not by the customer.

1.8 Preposed Methodology

A methodology is required for project to be successful. This is because a software development methodology gives you path of steps to follow within the lifecycle of the development of the application.

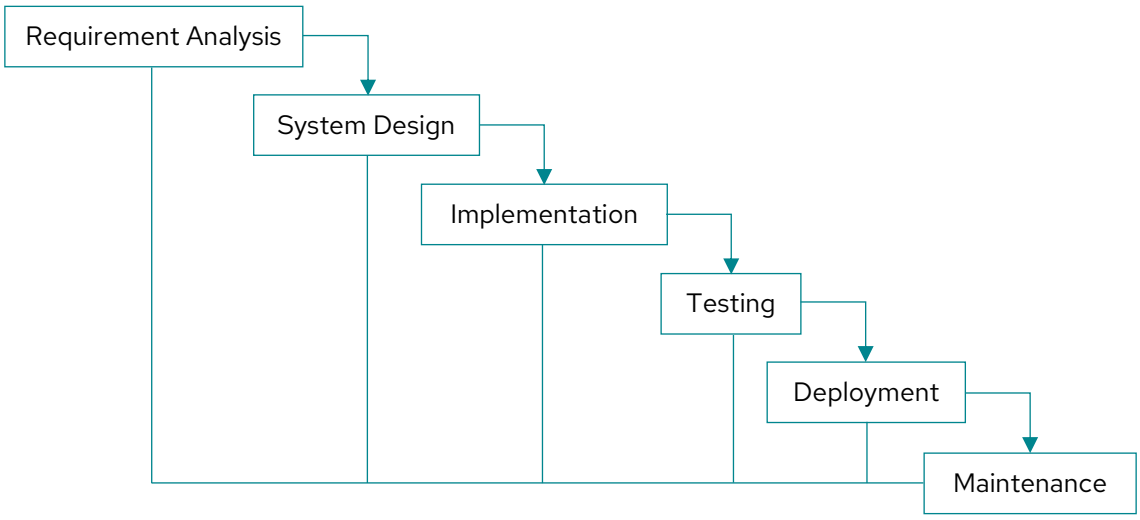


Figure 1: Waterfall methodology adapted from *tutorialspoint* (n.d.).

The software development methodology that is going to be used for this project is going to be Waterfall. According to *Lucidchart* (n.d.), the Waterfall model has a 'clear structure'. This can be seen as a strength because this allows clear steps through development. This will help with the project as a linear approach is best when starting a project. This will allow to focus each section individually allowing it to be completed to the best standard and plan in the time management between each stage.

One of the negatives is that this methodology is not suited for a large project according to Gaille (2020). This is best for the project as it is small.

An agile methodology is not suited for this type of project as these methodologies do not have 'a clear vision of what the "final product" looks like.' As mentioned by Lynn (n.d.). It is important to know what the final product looks like to know what you're achieving as your result, since this application has goal.

This is a suitable methodology for the case study as the result needs to be a 'proof of concept' and the Waterfall is a process that can be used to start a development cycle as it offers a 'clear goal' that department may be looking for and the fact that it 'hits the deadlines' so the department will know what date this development cycle will finish. (*Kitchen*, n.d.).

1.9 Requirements

1.9.1 Gathering Requirements

To gather requirements, extensive analysis has been taken place on the case study to ensure that all the necessary requirements have been created. This is because, there is a need to stick to the case study to provide results that meet what end user is looking for in this application.

For the functional requirements, the features the program should include were considered as well as some functionality that would help the end user to with using the application. An example of this would be choosing present types for requirements made from the client. In addition, a requirement may include to be able to edit a quote after being added to an order.

For the non-functional requirements, the usability of the user interface and anything to do with the end user's experience of using the application. This could involve researching on how other focus on this aspect of development as these non-functional requirements could have common points and goals.

1.9.2 Functional Requirements

Defining functional requirement is important show what is priority within the scope of the project. These requirements are prioritised using the MoSCoW technique.

The MoSCoW technique 'framework is easy to master and use because it's based on simple principles' explained by Korolev (2021), making it a good decision for small projects. As well as this method was hinted within the case study.

Table 1, seen below, shows these prioritised functional requirements.

ID	Title	Detailed Description	Owner	Priority	Deliver by
FR_01	Ability to choose present types.	The end user must be given three options of present type: Cube, Cuboid, and Cylinder.	Dr Claire Ancient	Must	23/05/2022
FR_02	Ability to preview the shape type chosen	The program could be able to preview the shape used within a quote.	Dr Claire Ancient	Could	23/05/2022
FR_03	Ability to enter present dimensions.	The end user must be given text boxes to enter the dimensions of the present type selected	Dr Claire Ancient	Must	23/05/2022
FR_04	Ability to calculate the area needed.	The program should be able to calculate the recommended needed for the purchase.	Dr Claire Ancient	Must	23/05/2022

ID	Title	Detailed Description	Owner	Priority	Deliver by
FR_05	Ability to choose wrapping paper type.	The end user must be given a choice of a cheap variant of the wrapping paper as well as an expensive variant.	Dr Claire Ancient	Must	23/05/2022
FR_06	Ability to choose wrapping paper colour.	The end user must be given a choice of six pre-set colours of wrapping paper the company offers.	Dr Claire Ancient	Must	23/05/2022
FR_07	Ability to preview the wrapping paper design and colour.	The program must be able to show the design of chosen wrapping paper type as well as the colour.	Dr Claire Ancient	Must	23/05/2022
FR_08	Ability to add a bow.	The end user must be able to add the bow item onto the quote.	Dr Claire Ancient	Must	23/05/2022
FR_09	Ability to add a gift card with a message.	The end user must be able to add a gift card as well as be able to type the message to be printed.	Dr Claire Ancient	Must	23/05/2022
FR_10	Ability to calculate price of the wrapping paper area that may be used.	The program must be able to calculate the price of wrapping paper based on the area and the type.	Dr Claire Ancient	Must	23/05/2022
FR_11	Ability to calculate the price of the message within a gift card.	The program must be able to calculate the price of the gift card, taking a count of the number of characters within the message.	Dr Claire Ancient	Must	23/05/2022
FR_12	Ability to calculate the price of a quote.	The program must be able to calculate the price of a single quote, based on the options the end user has chosen.	Dr Claire Ancient	Must	23/05/2022
FR_13	Ability to calculate the total of multiple quotes.	The program must be able to calculate the total price of all the quotes.	Dr Claire Ancient	Must	23/05/2022

ID	Title	Detailed Description	Owner	Priority	Deliver by
FR_14	Ability to edit quotes that have been created.	The program should be able to allow the end user to edit existing quotes on the order to allow any changes and save them.	Dr Claire Ancient	Should	23/05/2022
FR_15	Ability to delete quotes that on the list.	The program should be able to allow the end user to delete a quote from the order.	Dr Claire Ancient	Should	23/05/2022
FR_16	Ability to export the quotes to an external file.	The program could allow the end user to export the quotes to an external file.	Dr Claire Ancient	Could	23/05/2022

Table 1: Functional Requirements.

1.9.3 Non-Functional Requirements

Non-functional requirements 'provides ... the ability to assess the quality of the software [produced]' as mentioned by Crawford (n.d.). Assessing the quality may increase the end-user's satisfaction of the product overall.

Table 2, shown below, shows the non-functional requirements of the final product.

ID	Title	Detailed Description
NFR_01	User Experience	The user experience must be logical and easy to understand and navigate without the need of instructions or help guide.
NFR_02	Usability	The user interface must follow the WCAG 1.4.3 guideline to make the application accessible to any employee within the department. This should include a colour scheme that must be at least meet the AA standard.
NFR_03	Maintainability	Any extra functionality the department would like to add must be easy to implement over the application's lifetime.
NFR_04	Performance	The application must be light on system resources as well as responsive to the interaction from the user.

Table 2: Non-Functional Requirements.

1.9.4 Future Requirements

Towards the future, once the project has been completed, there may be additional requirements that may be added based on employee feedback on how the system has been implemented. This employee feedback is important as this is a program that will be used day to day that will need to be further tailored to the department's needs.

In addition to this, extra functionality may be added upon use of the application, which could include the change of pricing for the wrapping paper, gift cards and bow or the department may decide to add more colours and wrapping paper designs.

2 Design

2.1 Use Case Diagram

Figure 2, shown below, is a use case diagram that models the employee interacting with the system. As mentioned by *Visual Paradigm* (n.d. a) 'A use case diagram ... does not show the [details] ... [and that] it only summarizes some of the relationships' in no particular order.

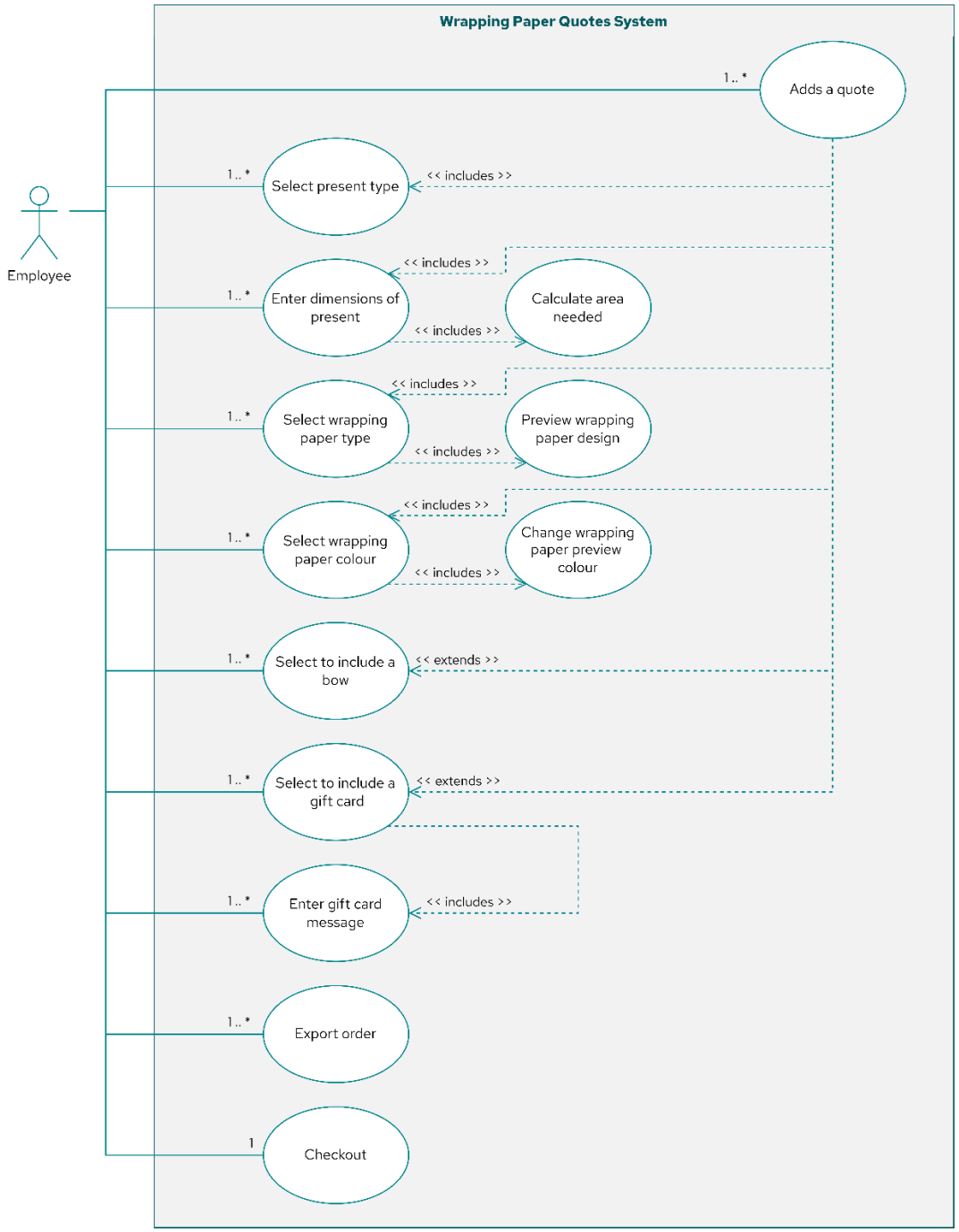


Figure 2: Use Case Diagram.

2.2 Activity Diagram

Figure 3, below, shows an activity diagram of how the department employee will interact with the system, '[describing] dynamic aspects of the system'. (*Visual Paradigm*, n.d. b)

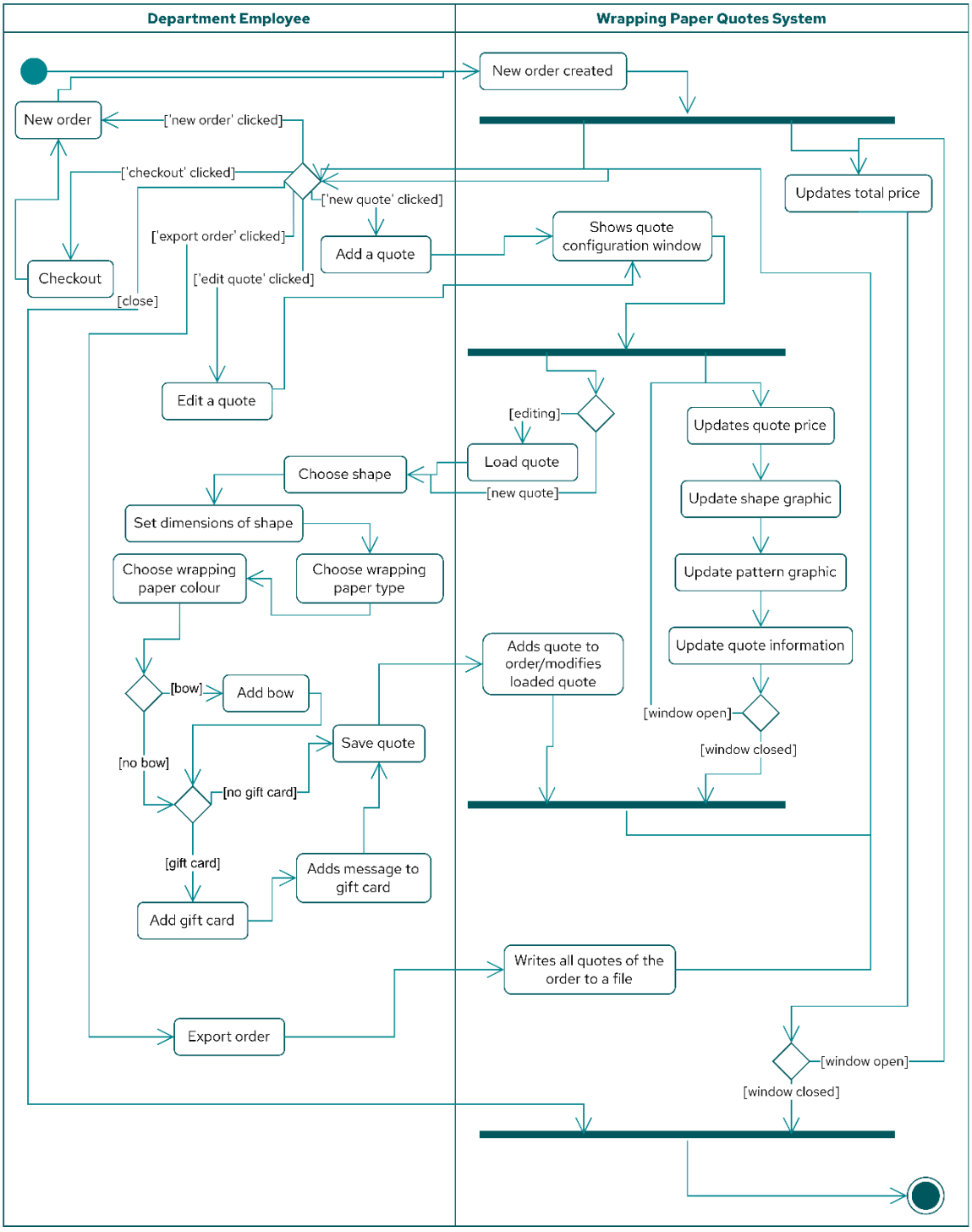


Figure 3: Activity Diagram.

2.3 Class Diagram

Figure 4, below, shows the classes used within the system and the relationships between them. This diagram shows how the objects relate to each other within the system.

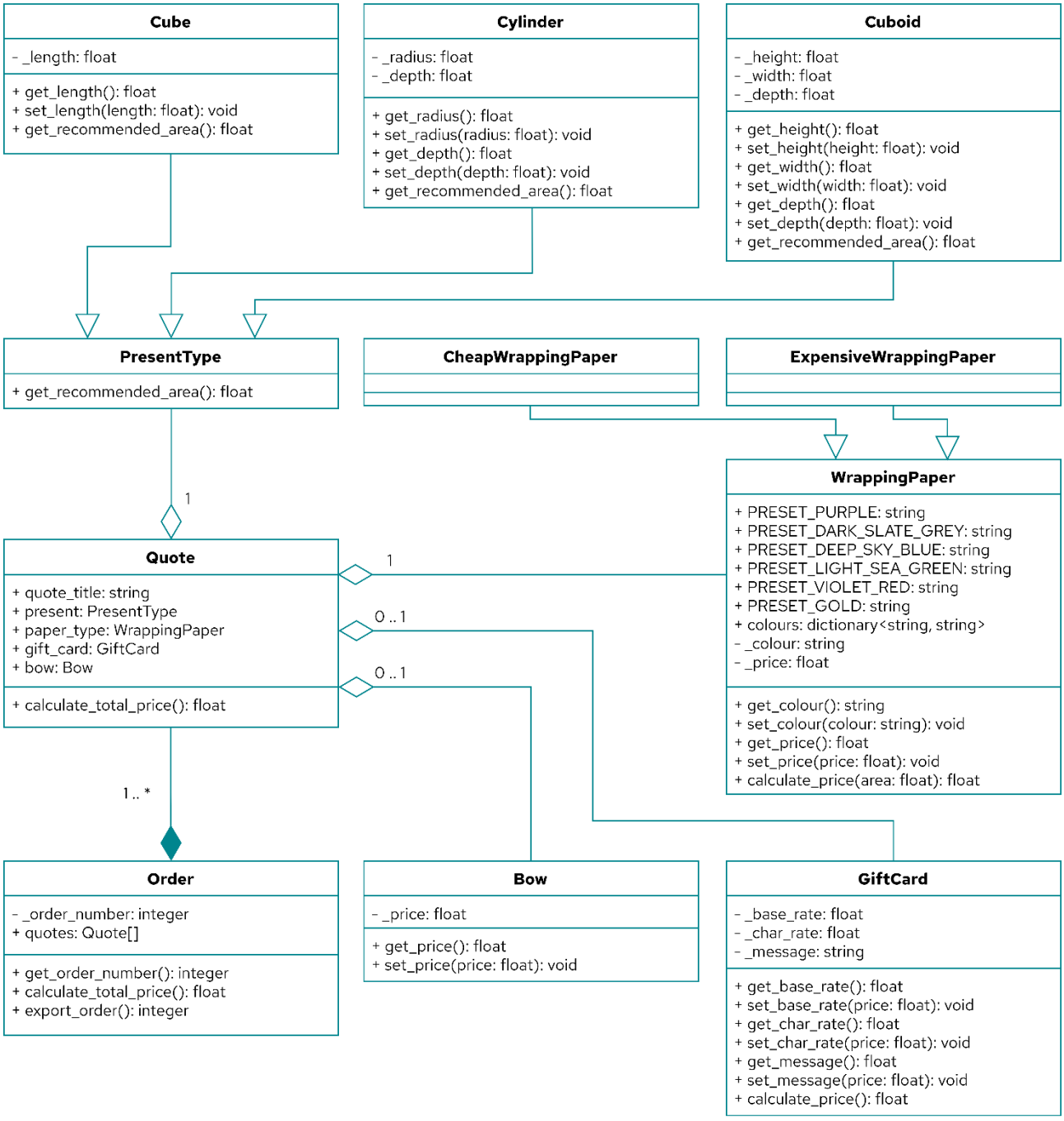


Figure 4: Class Diagram

2.4 Sequence Diagram

Figure 5 shows a sequence diagram of the messages between the employee, user interface and the system, '[capturing] the interaction … that detail how operations are carried out'. (*Visual Paradigm*, n.d. c)

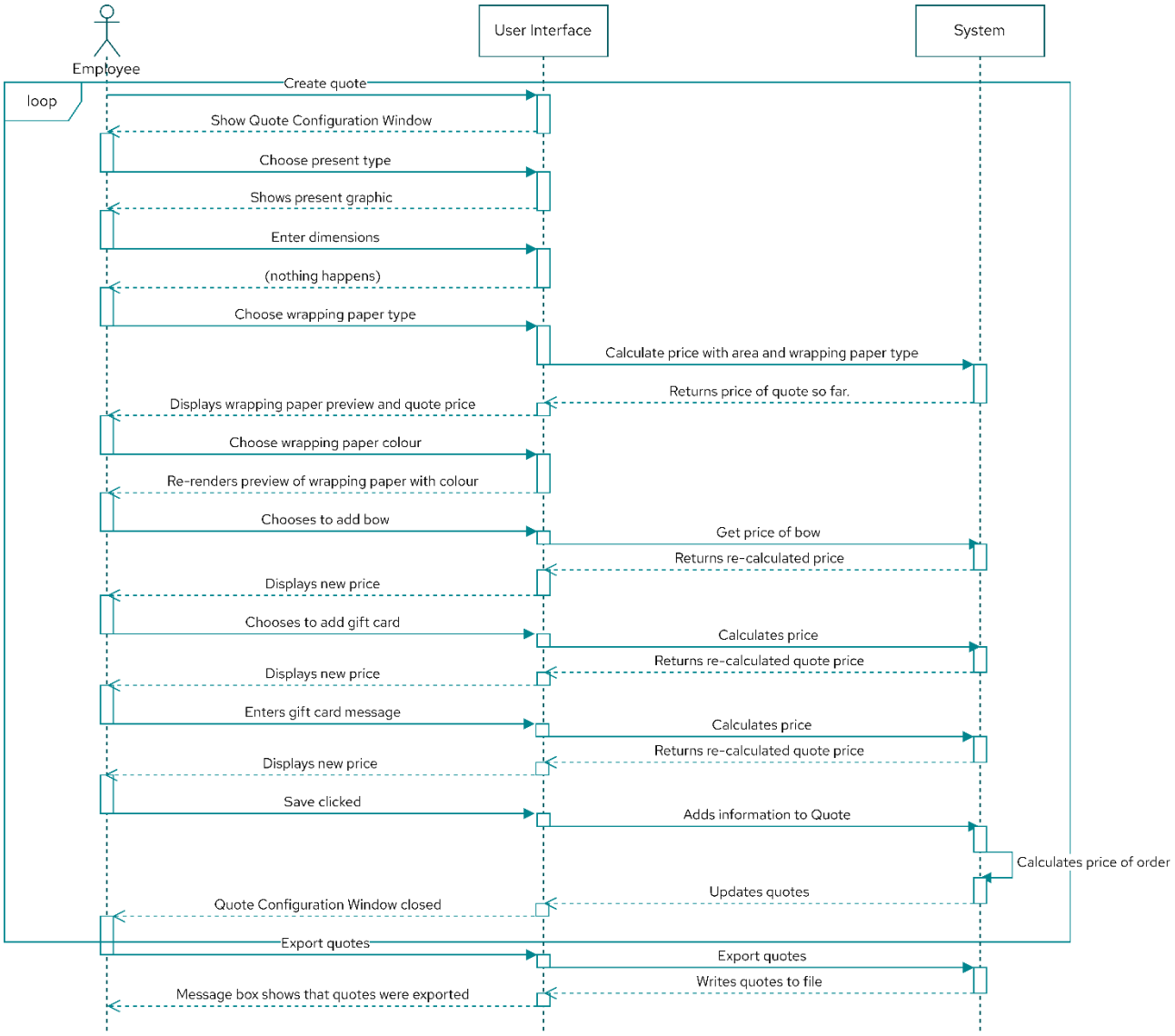


Figure 5: Sequence Diagram

2.5 Interface Design

2.5.1 Wireframe of the User Interface

Figure 6 shows the Main Window of the application. In summary, this window shows all the current quotes within the order and actions that can be called within the order.

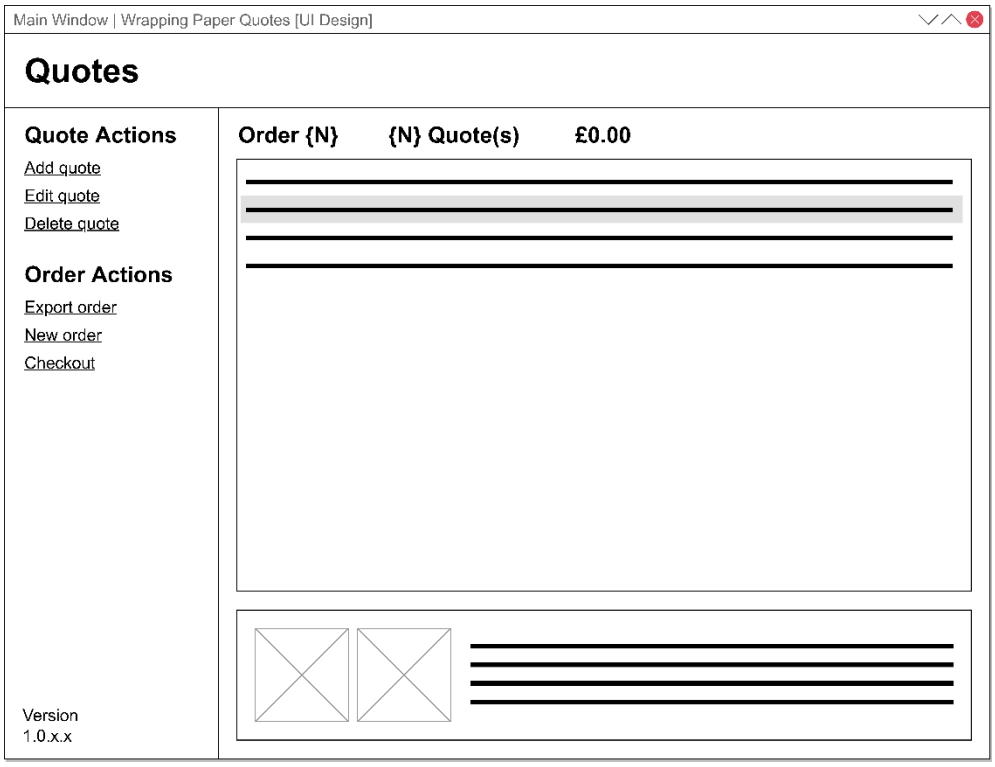


Figure 6: Wireframe of the Main Window.

The design of this window consists of a header, sidebar, and the main content. The header shows the title of the window. The sidebar shows a list of options categorised between actions that affect the quotes or the order. This sidebar design is somewhat reminiscent of the Windows 11's 'Principles of good navigation' (*Microsoft*, 2021) as seen in the figure below.

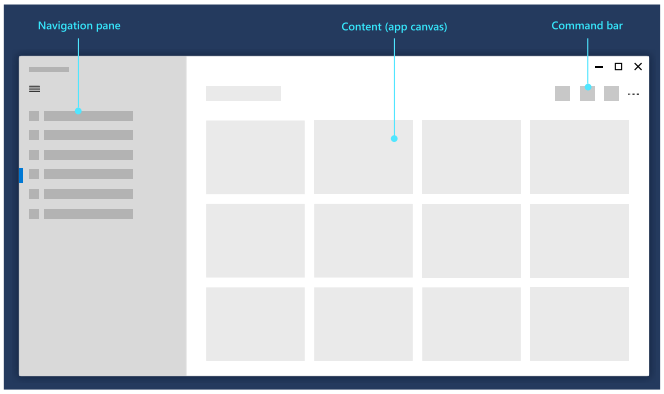


Figure 7: Principles of good navigation (*Microsoft*, 2021)

However, the actions are on the sidebar instead of having a command bar. The main content of this window shows a title showing the order number, number of quotes and the total calculated price of the order. Below

this is a list-box of the list of quotes within that order. While selecting any of these quotes, the panel below this will show a preview of the shape and wrapping paper selected within that one quote as well as the dimensions of the shape, gift card information and if a bow was added.

Another window to this application is the Quote Configuration Window, shown in figure 8, which allows the editing and creation of quotes with the information provided by the end user.

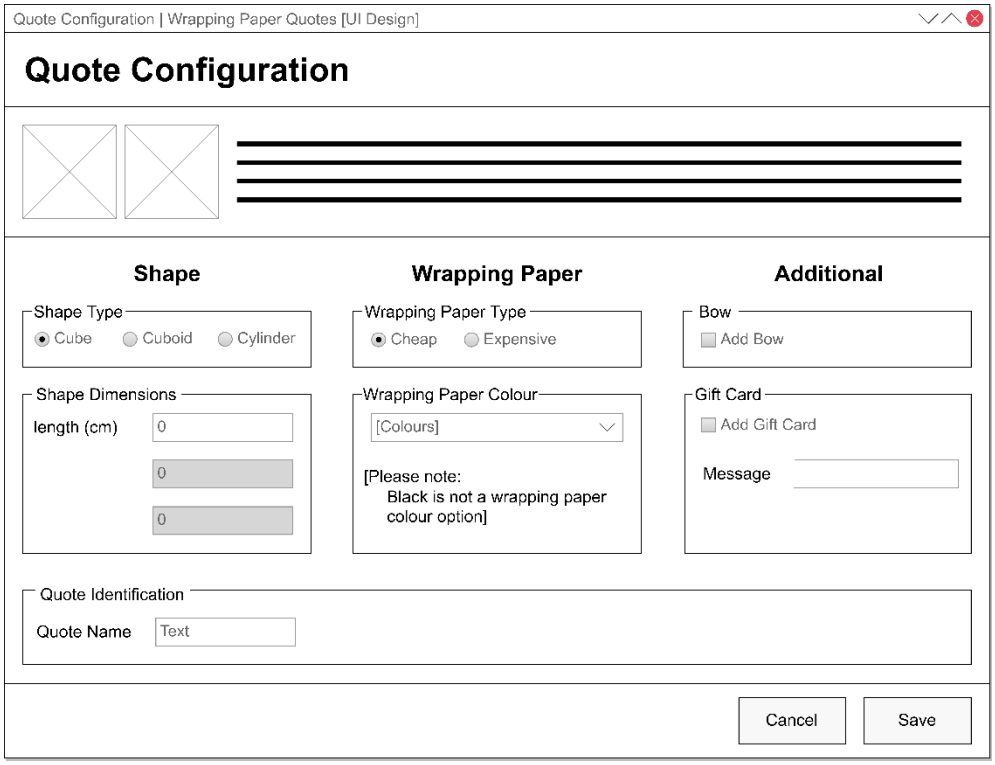


Figure 8: Wireframe of the Quote Configuration Window.

This window consists of a preview pane showing the quote's shape chosen as well as the wrapping paper. This helps visually see the details of the main part of quote. Below this are all the options separated into three columns: shape, wrapping paper and additional. This will help the user easily find the option that they are looking for. Below this is a quote name identifier textbox to allow the end user to name the quote for easier recall.

2.5.2 POUR Guidelines

This user interface must follow the POUR guidelines by the WCAG. As mentioned by *WebAIM* (2021) that these guidelines 'were invented to make life easier for people with disabilities'.

2.5.2.1 Perceivability

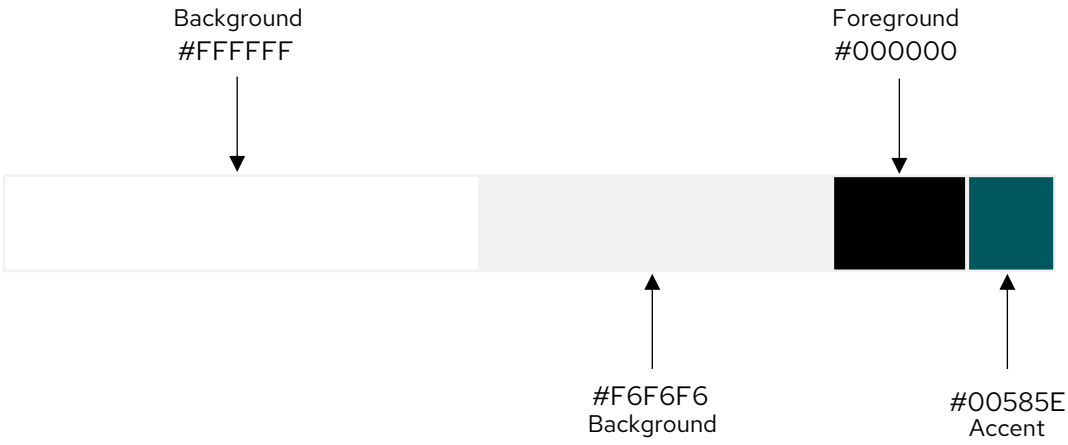


Figure 9: Colour Scheme

To make this application perceivable, the user interface colour scheme, shown in figure 9, was considered to make sure the contrast ratio of the background and foreground would be readable to someone visually impaired. This contrast ratio was then tested with the WebAIM Contrast Checker, as seen below in Table 3.

Foreground Colour	Background Colour	Preview		Contrast Ratio	Normal Text		Large Text		Graphical Objects
					AA	AAA	AA	AAA	AA
#000000	#FFFFFF	Text.	**Text.**	21:1	✓	✓	✓	✓	✓
#00585E	#FFFFFF	Text.	**Text.**	8.21:1	✓	✓	✓	✓	✓
#000000	#F6F6F6	Text.	**Text.**	19.43:1	✓	✓	✓	✓	✓
#00585E	#F6F6F6	Text.	**Text.**	7.59:1	✓	✓	✓	✓	✓

Table 3: Contrast Colour Checks

Figure 9 also shows the approximate ratio of the colours that will be used within the application. The two background colours will help to give more a structure between the widgets within the interface.

2.5.2.2 Operability

In terms of using the application, it can be used with completely the keyboard, by using tab, space and arrow keys. This functionality will help people who cannot use a pointer on the screen.

In addition to this there are no time limits for tasks to be done within the program, so the end user can use the application at their own pace. If the end user makes a mistake, message boxes will be used to inform the user and give the user another chance. These message boxes must be informative to tell the end user exactly what is wrong in an understandable format. Adding on this, if the end user makes a mistake within the quote, it can be edited again. This will help the end user in case the customer has any changes later on to their quote. As well as asking to export or save on a closing of a window, so that the customer can keep a copy as well as the department if needed.

2.5.2.3 Understandability

The understandability of the application has been considered through the widgets labelled clearly within the multiple windows that the application has. For people who have cognitive disabilities, the amount of information to comprehend and remember has been kept to a minimum by not including long amounts of text.

2.5.2.4 Robustness

The application will need to be backwards compatible to ensure that devices without internet access, that are not getting regularly updated, can still work with the latest version of the application. As mentioned by *WebAIM* (2021) '[end-users] all expect the [application] to work. When [it] ... does not support their technologies, they get frustrated and may never return.' This increases the importance of compatibility of older versions of the python framework as end-users may just expect it to work. In short terms, first impressions will matter to the department.

3 Functional Testing

3.1 Testing Table

The Table 4, below, shows the tests conducted to ensure that the application works correctly and as intended. All features tested have been separated into categories, between the UI, System & Error Handling – these all have subcategories.

As well as this, the input data with ('') quotation marks show something is already in the user interface, ("") speech marks show something is being typed and ([]) square brackets shows a generic action to be taken place. (->) shows another action to be followed.

Table 5 shows the evidence of this tests through the use of screenshots (Figures 10-61).

Test Ref.	Feature to be Tested	Input Data / Actions	Expect Outcome	Actual Outcome	Test Successful
Application Version 1.0.14-sc2					
1	**UI (Window)**: Displaying the Main Window.	N/A. (On Start-up)	The main window should be shown	Works as intended.	Yes
2	**UI (Button)**: Click Event.	'Add quote'	The quote configuration window should be open simultaneously as the main window.	Works as intended.	Yes
3	**UI (Button)**: Click Event.	'Add quote' -> 'Cancel'	The quote configuration window should close without any message boxes or options.	Works as intended.	Yes
4	**UI (Button) & Error Handling (Incompleteness)**: Click Event.	'Add quote' -> 'Save'	An error message should be shown saying that the quote is not complete	Works as intended.	Yes
5	**UI (Window)**: Closing Event.	'Add quote' -> 'X' on native title-bar.	A message box should show, asking whether the quote should be saved	Works as intended.	Yes
5.1	**UI (Message Box, Button)**: Click Event	'Cancel'	The message box should close, keeping the	Works as intended.	Yes

Test Ref.	Feature to be Tested	Input Data / Actions	Expect Outcome	Actual Outcome	Test Successful
			quote configuration window open.		
5.2	**UI (Message Box, Button)**: Click Event	'No'	The message box should close, closing the quote configuration window.	Works as intended.	Yes
5.3	**UI (Message Box, Button)**: Click Event	'Yes'	The message box should close, going through the same procedure as the save button.	Works as intended.	Yes
6	**UI (Radio Button, Canvas)**: Click Event	'Add quote' -> 'Cube'	A graphic of a cube shape should be shown, and the 2 textboxes must be greyed out, labelling only the 'length'.	Works as intended.	Yes
6.1	**UI (Button) & Error Handling (Incompleteness)**: Click Event.	'Save'	A message box should appear informing the end user that the quote is not complete.	Works as intended.	Yes
6.2	**UI (Entry) & System (Calculations):** Number Entry.	"1"	Nothing should happen, only that you have typed in a number.	Works as intended.	Yes
6.3	**UI (Button) & Error Handling (Incompleteness):** Click Event	'Save'	A message box should appear informing the end user that the quote is not complete.	Works as intended.	Yes
6.4	**UI (Radio Button, Canvas) & System (Calculations):** Click Event.	'Cheap'	The price, dimensions and area should now be shown. The cheap wrapping paper preview	Works as intended.	Yes

Test Ref.	Feature to be Tested	Input Data / Actions	Expect Outcome	Actual Outcome	Test Successful
			should be rendered within the preview. (Pattern 6)		
6.5	**UI (Button) & Error Handling (Incompleteness):** Click Event.	'Save'	A message box should appear informing the end user that the quote has not got a valid colour selected.	Works as intended.	Yes
6.6	**UI (Combo Box, Canvas):** Choosing from List of Values.	'Deep Sky Blue'	The preview of the wrapping paper should be re-rendered to show the colour chosen.	Works as intended.	Yes
6.7	**UI (Entry):** Text Entered.	"Quote Test"	The quote name should also be displayed within the preview section of the window.	Works as intended.	Yes
6.8	**UI (Button) & System (Item Management, Update, Calculations):** Button Clicked.	'Save'	The window should close without any error, adding it to the list box of the Main Window. The Main Window should display the correct number of quotes on the order as well as the correct price.	Works as intended.	Yes
7	**UI (List Box, Canvas, Label):** Quote Selection.	[Quote Clicked On]	The Main Window should display the quote in an easier to read format, previewing the shape and wrapping paper as well towards the bottom of the window.	Works as intended.	Yes

Test Ref.	Feature to be Tested	Input Data / Actions	Expect Outcome	Actual Outcome	Test Successful
8.1	**UI (Radio Button, Canvas)**: Click Event	'Add quote -> 'Cuboid'	A graphic of the cuboid shape should be shown, and the 3 textboxes should all be labelled	Works as intended.	Yes
8.2	**UI (Entry Boxes, Label):** Number Entry.	Width: "14" Height: "4" Depth: "9"	Nothing should happen. The entered should not output an error.	Works as intended.	Yes
8.3	**UI (Radio Button, Canvas) & System (Calculations):** Click Event.	'Expensive'	A preview of the expensive wrapping would be rendered. As well as this the area and price should now be shown. (Pattern 8)	Works as intended.	Yes
8.4	**UI (Combo Box, Canvas):** Choosing from List of Values.	'Violet Red'	The wrapping paper preview should be re-rendered to show the colour chosen.	Works as intended.	Yes
8.5	**UI (Check Box) & System (Item Management):** Click Event	Bow [Ticked]	The price should be adjusted as the bow has been added.	Works as intended.	Yes
8.6	**UI (Check Box) & System (Item Management):** Click Event	Gift Card [Ticked]	The price should be adjusted as the gift card has been added.	Works as intended.	Yes
8.7	**UI (Entry) & System (Item Management, Update):** Text Entry.	"Happy Birthday!"	The price should be updated as characters are entered.	Works as intended.	Yes
8.8	**UI (Button) & System (Item Management, Update, Calculations):** Click Event.	'Save'	The system should successfully add the quote to the list box on the main window as	Works as intended.	Yes

Test Ref.	Feature to be Tested	Input Data / Actions	Expect Outcome	Actual Outcome	Test Successful
			the quote configuration window closes. The price of this quote should be £9.55 as indicated within the case study.		
9.1	**UI (Radio Button, Canvas)**: Click Event	'Add quote' -> 'Cylinder'	A graphic of the cylinder shape should be shown, and 1 textbox should be greyed out, others should be labelled radius and depth.	Works as intended.	Yes
9.2	**UI (Radio Button, Canvas):** Click Event.	'Cheap'	The cheap wrapping paper design should be rendered within a canvas.	Works as intended.	Yes
9.3	**UI (Combo Box, Canvas):** Choosing from List of Values.	'Dark Slate Grey'	The wrapping paper preview should be re-rendered to show the colour chosen.	Works as intended.	Yes
9.4	**UI (Button) & Error Handling (Incompleteness):** Click Event.	'Save'	A message box should be shown to inform the end user that the dimensions within the entry boxes are invalid.	Works as intended.	Yes
9.5	**UI (Entry) & System (Calculations) & Error Handling (Unexpected Inputs):** Text Entry.	Radius: "five" Depth: "one"	The preview area of the window should tell the end user that the dimensions are invalid.	Works as intended.	Yes
9.6	**UI (Button) & Error Handling (Unexpected Inputs):** Text Entry.	'Save'	A message box should be shown to inform the end user that the dimensions within	Works as intended.	Yes

Test Ref.	Feature to be Tested	Input Data / Actions	Expect Outcome	Actual Outcome	Test Successful
			the entry boxes are invalid.		
9.7	**UI (Entry) & System (Calculations):** Number Entry	Radius: "1.5" Depth: "-5"	The preview area of the window should show the dimensions entered as well as the area and calculated price.	Works as intended.	Yes
9.8	**UI (Entry)**: Text Entry	"Another Test Quote"	The preview area should update to show the name of the quote.	Works as intended.	Yes
9.9	**UI (Button) & System (Item Management, Update, Calculations):** Button Clicked.	'Save'	The system should successfully add the quote to the list box on the main window as the quote configuration window closes.	Works as intended.	Yes
10	**UI (Button) & System (Exporting):** Click Event.	'Export quotes'	The system should be able to export the quotes to the relative directory, displaying the text file in a receipt like structure.	Works as intended.	Yes
11	**UI (Button) & System (Item Management, Update, Calculations):** Click Event.	[Select a Quote] -> 'Delete quote'	The system should be able to delete the quote from the list box, recalculating the total price of the order.	Works as intended.	Yes
12	**UI (Button) & Error Handling (Invalid Operation):** Click Event.	'Delete quote' [no quote selected]	A message box should show informing the end user that there is not a selected quote to edit.	Works as intended.	Yes

Test Ref.	Feature to be Tested	Input Data / Actions	Expect Outcome	Actual Outcome	Test Successful
13	**UI (Button, Window):** Click Event.	[Select a Quote] -> 'Edit quote'	The quote configuration window should show up, showing the same information as show within the quote.	Works as intended.	Yes
14	**UI (Button) & Error Handling (Invalid Operation):** Click Event.	[Select currently editing quote] -> 'Delete quote'	A message box should show informing the user that quote is currently being edited.	Works as intended.	Yes
15	**UI (Entry, Button) & System (Item Management):** Text Entry & Click Event	[currently editing quote] Quote name: "The Brief Test Quote" -> 'Save'	The quote should update its name within the main window.	Works as intended.	Yes
16	**UI (Button, Message Box) & System (Item Management):** Click Events.	'New order', 'No'	A message box should show asking if the end user wants the export before starting a new order. The order should be incremented and have no quotes in the list box.	Did not work as intended. Program created a blank order, without asking about exporting. It is supposed to ask if any changes have been made after exporting or if it has not been exported.	Partially. *Fixed in Patch A.*
Application Version 1.0.14a-sc2					
16a	**UI (Button, Message Box) & System (Item Management):** Click Events.	'New order', 'No'	A message box should show asking if the end user wants the export before starting a new order. The order should be incremented and	Works as intended	Yes

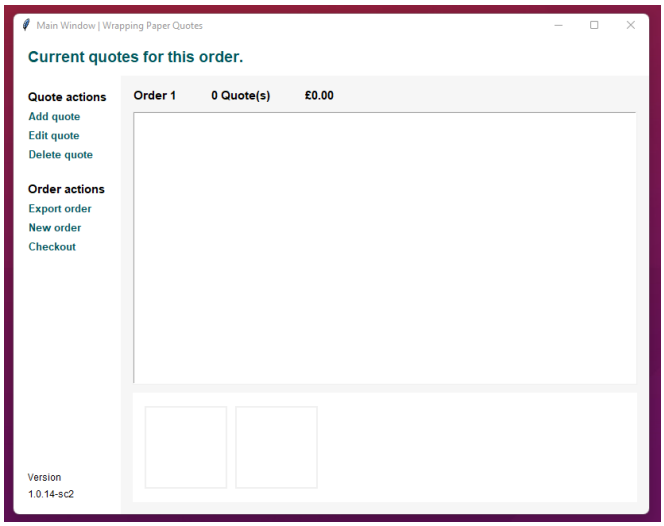
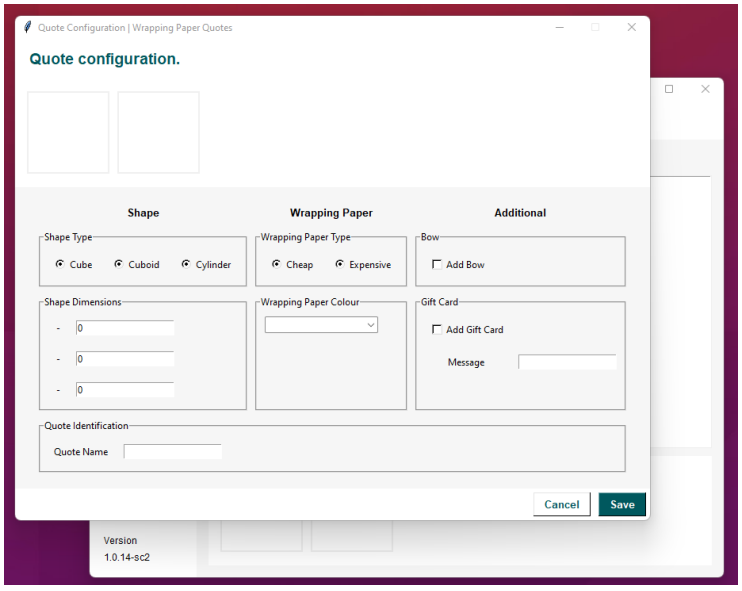
Test Ref.	Feature to be Tested	Input Data / Actions	Expect Outcome	Actual Outcome	Test Successful
			have no quotes in the list box.		
17	**UI (Button) & System (Exporting):** Click Event.	'Export order'	A message box should show informing the end user that an empty order cannot be exported.	Works as intended	Yes
18	**UI (Button, Message Box) & System (Item Management):** Click Event.	'New order'	No message box should show as the order is empty and the order number should be incremented.	Works as intended	Yes
19	**UI (List Box, Canvas):** Choosing from list of values.	[All colours to be tested]	The colours should be translated to the correct colours from the human readable format of colours.	Works as intended.	Yes
19.1	**UI (List Box, Canvas):** Colour Testing	'Purple'	The colour shown on the canvas should be translated to 'purple'.	Works as intended.	Yes
19.2	**UI (List Box, Canvas):** Colour Testing	'Dark Slate Grey'	The colour shown on the canvas should be translated to 'DarkSlateGray4'.	Works as intended.	Yes *Patch B Changes: The UI now says 'Dark Slate Grey 4'*
19.3	**UI (List Box, Canvas):** Colour Testing	'Deep Sky Blue'	The colour shown on the canvas should be translated to 'deep sky blue'.	Works as intended.	Yes
19.4	**UI (List Box, Canvas):** Colour Testing	'Light Sea Green'	The colour shown on the canvas should be	Works as intended.	Yes

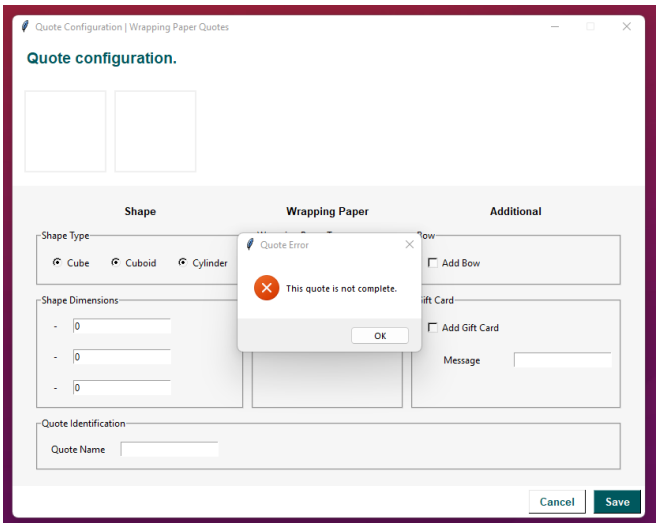
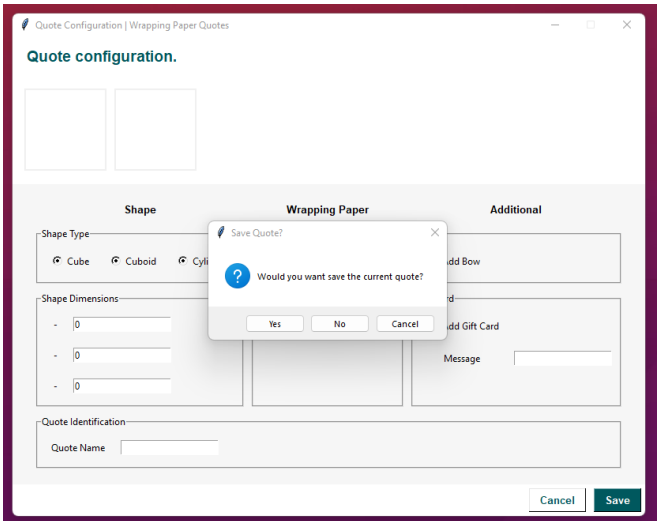
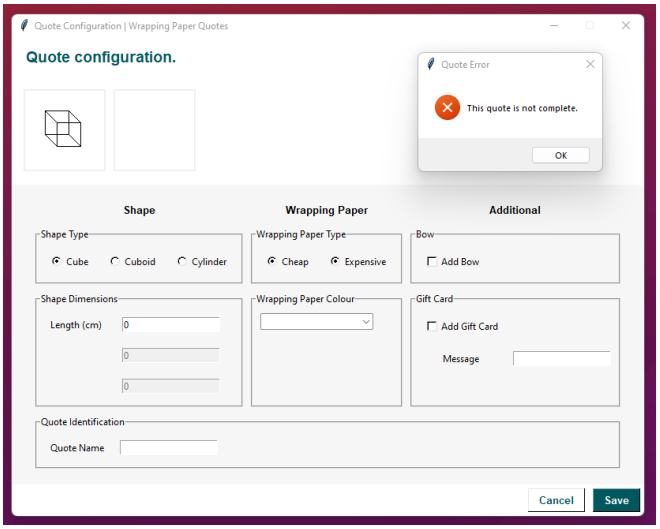
Test Ref.	Feature to be Tested	Input Data / Actions	Expect Outcome	Actual Outcome	Test Successful
			translated to 'light sea green'.		
19.5	**UI (List Box, Canvas):** Colour Testing	'Violet Red'	The colour shown on the canvas should be translated to 'VioletRed2'.	Works as intended.	Yes *Patch B Changes: The UI now says 'Violet Red 2'*
19.6	**UI (List Box, Canvas):** Colour Testing	'Gold'	The colour shown on the canvas should be translated to 'gold'.	Works as intended.	Yes
20	**UI (Button) & System (Exporting):** Click Event.	[Create a quote] -> 'Save' -> 'Checkout'	The application should be able to export the order to the relative directory as well as starting an application. Since this is a proof of concept. It is a simulation of what might happen. It does not include a banking system.	Works as intended.	Yes

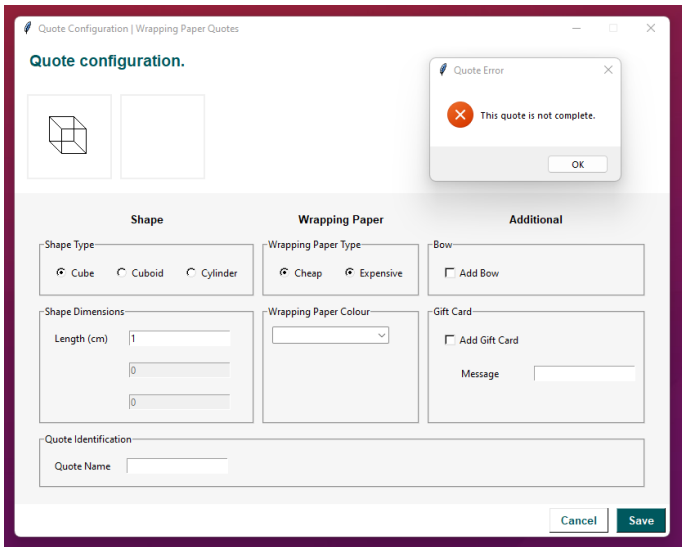
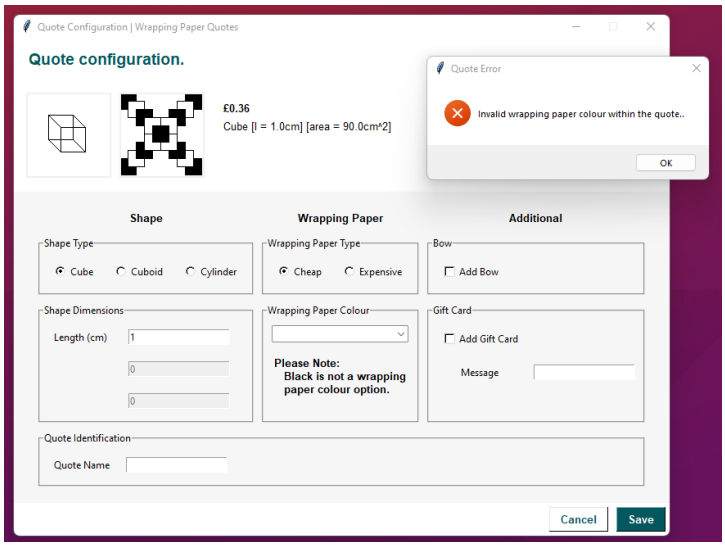
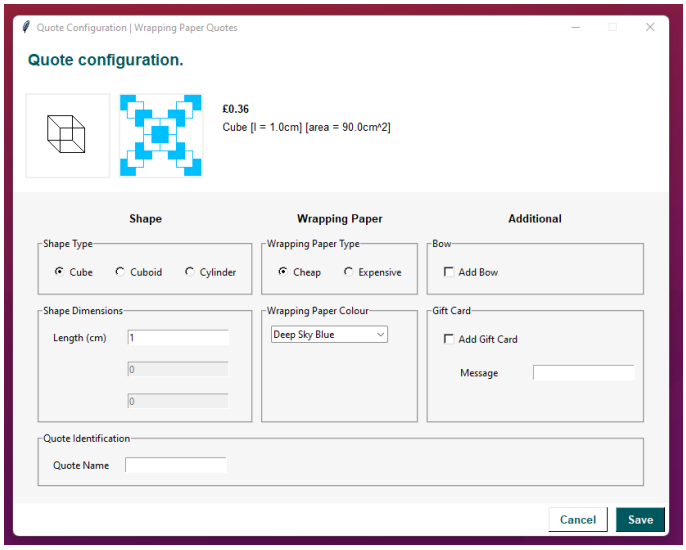
Table 4: Functional Tests Conducted

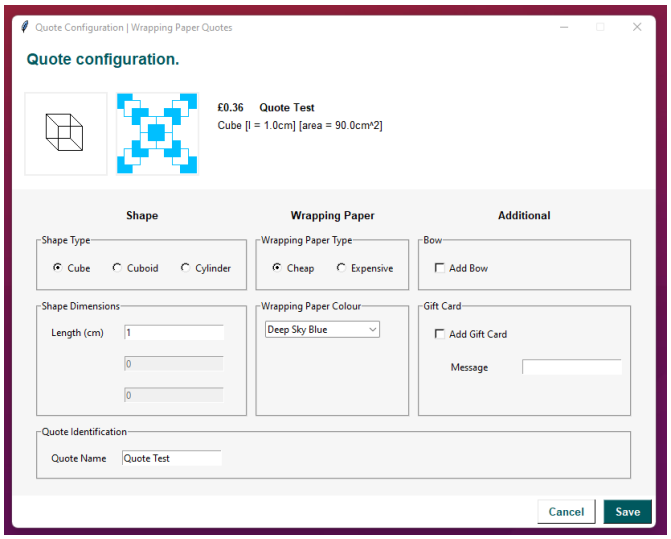
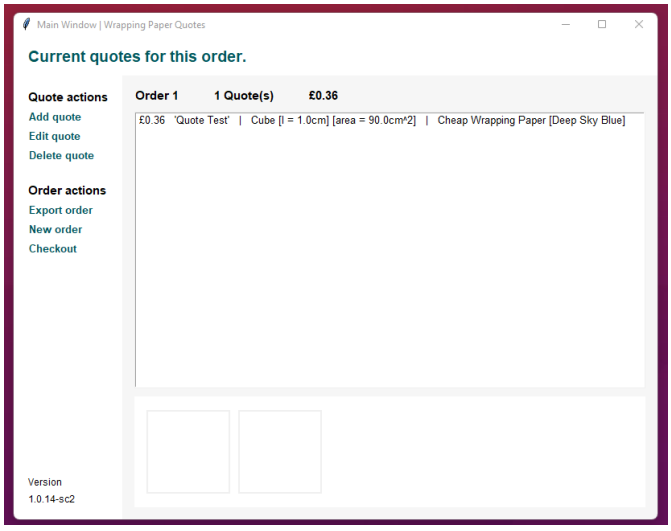
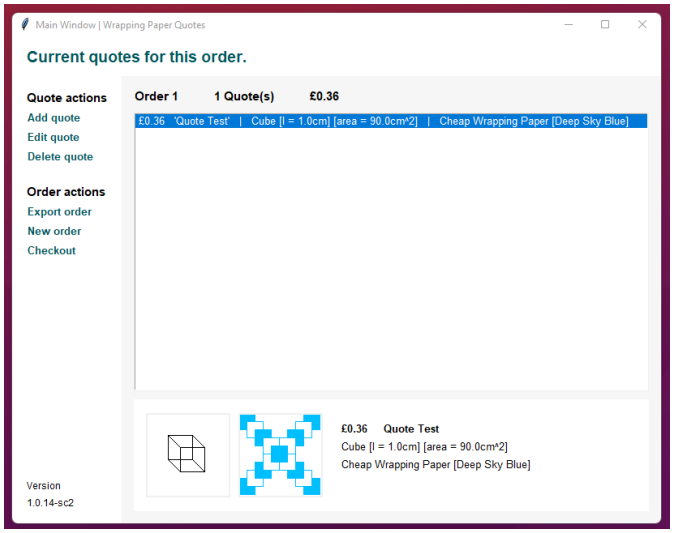
3.2 Screenshot Evidence

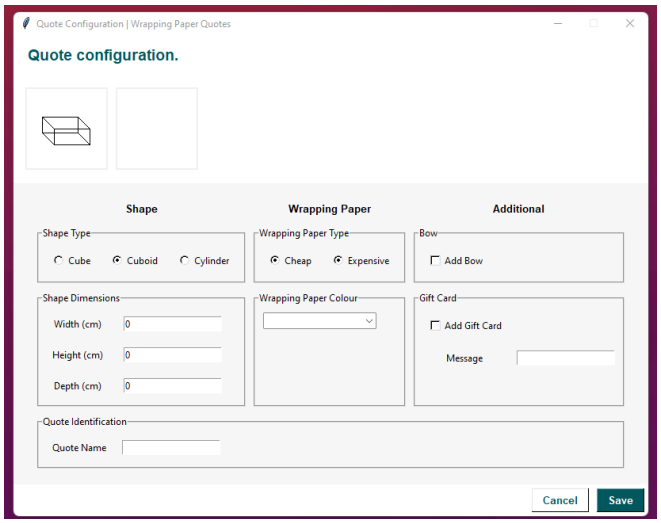
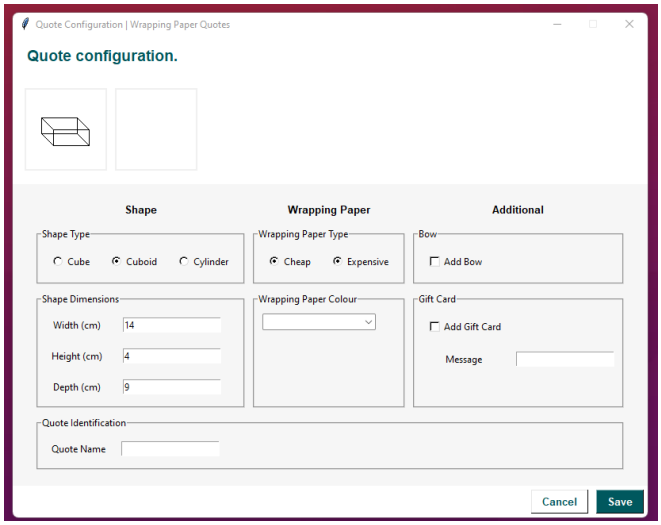
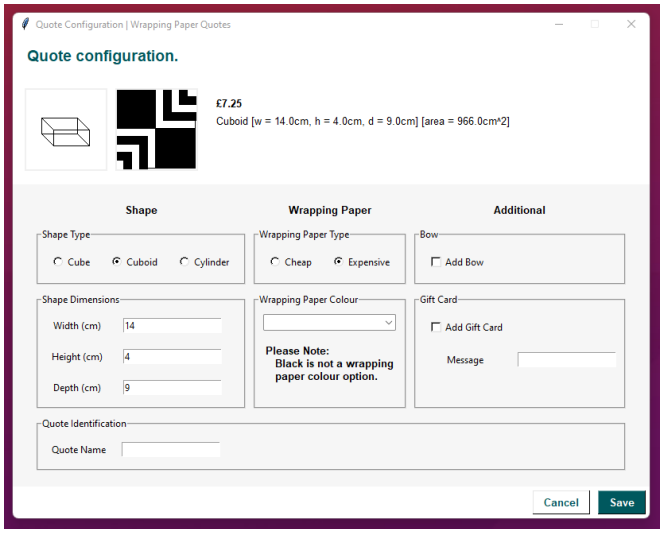
Table 5, below, shows the evidence of the functional test that have been taken place. The test reference column refers back to the tests within Table 4.

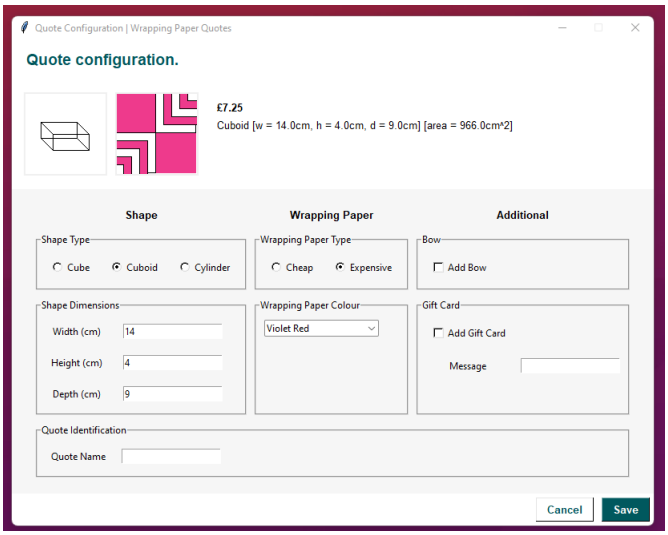
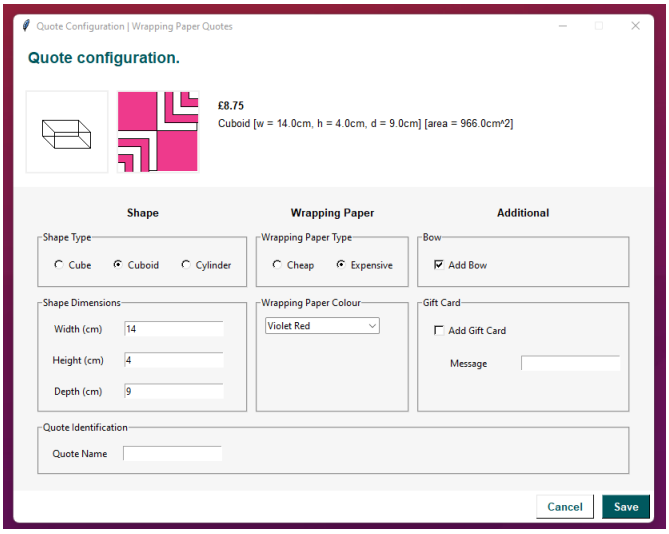
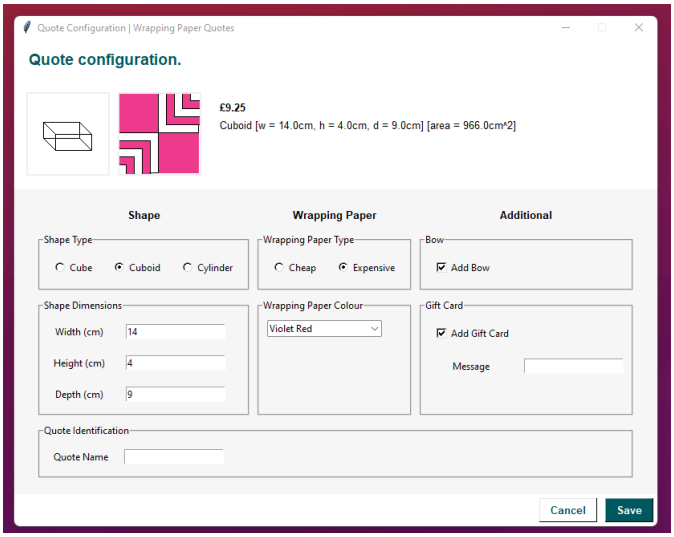
Test Reference	Screenshot and Explanations
1 *Figure 10*	 The screenshot beside shows the Main Window shown on the initial start-up of the application.
2 *Figure 11*	 The screenshot above shows both the Main Window and the Quote Configuration Window open.
3	**!** | *Test 3 could not be show as you cannot evidence a window closing.*

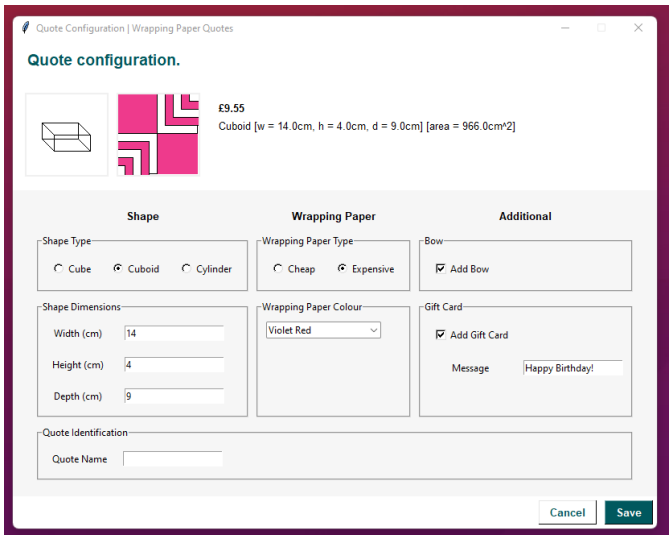
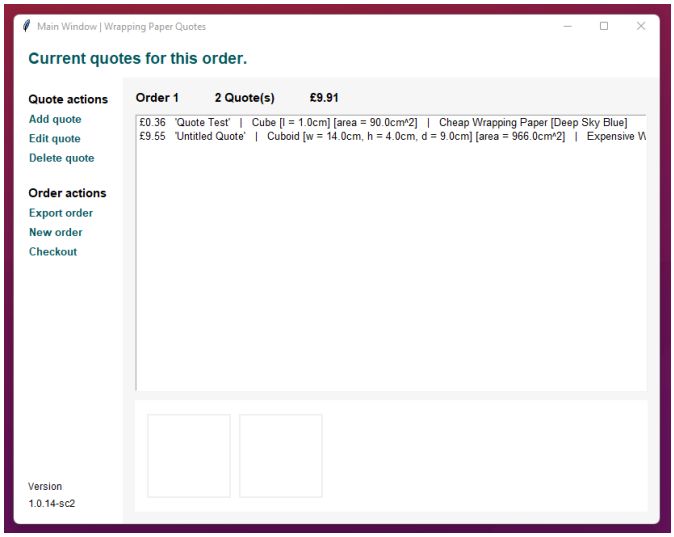
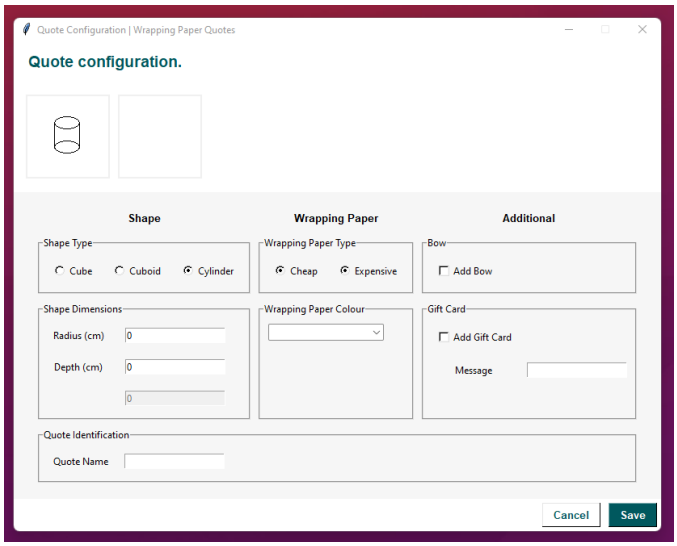
Test Reference	Screenshot and Explanations	
4 *Figure 12*		The message shown after just clicking save within the Quote Configuration Window.
5 *Figure 13*		On closing the quote configuration window, this message box is shown to ask the user whether the quote should be saved or pressing the 'x' was accidental.
6, 6.1 *Figure 14*		The quote configuration window will not let the end user save an unfinished quote. Figure 14 shows that only the present type was selected. A graphic of this shape can be seen within the first canvas.

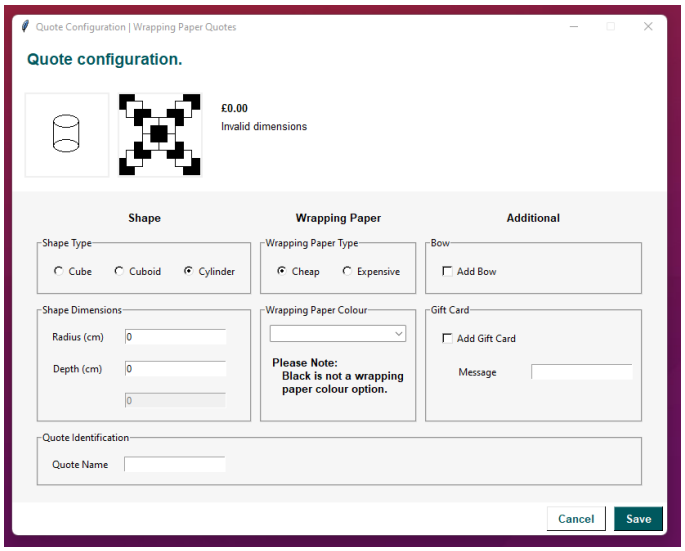
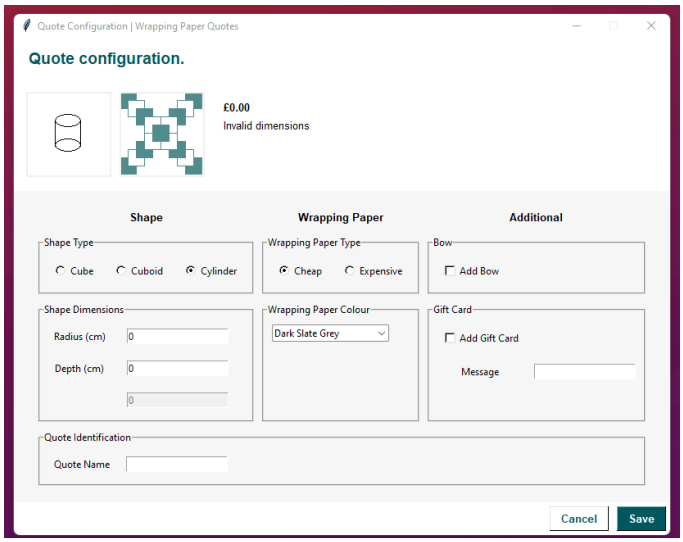
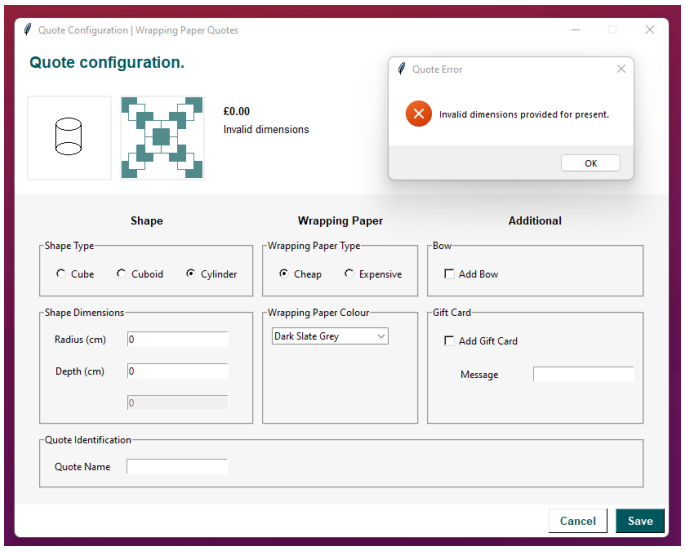
Test Reference	Screenshot and Explanations
6.2, 6.3 *Figure 15*		As mentioned before, the quote configuration window will not let the end user save an unfinished quote. Figure 15 shows the dimensions have now been entered.
6.4, 6.5 *Figure 16*		Figure 16 again shows an unfinished quote; however, the message starts getting more specific as the end user has not chosen a colour. As well as this, the price is calculated since the end user has chosen the wrapping paper type. Colour does not affect the price of the quote.
6.6 *Figure 17*		Figure 17 shows the quote, where the canvas has the colour rendered wrapping paper.

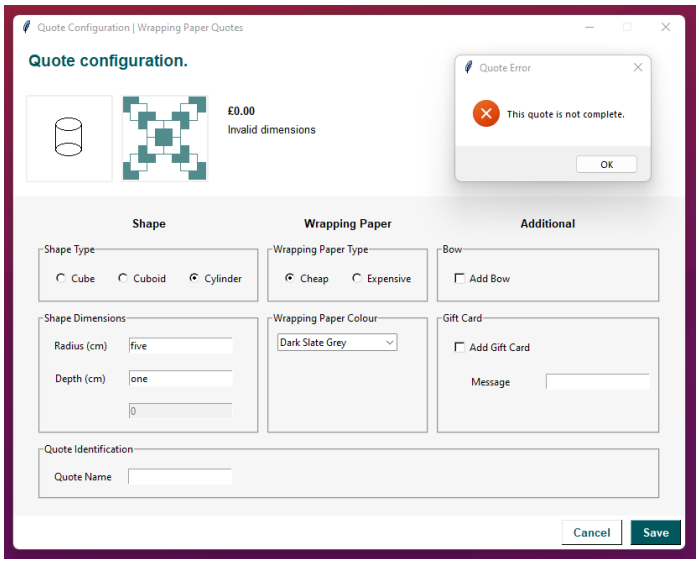
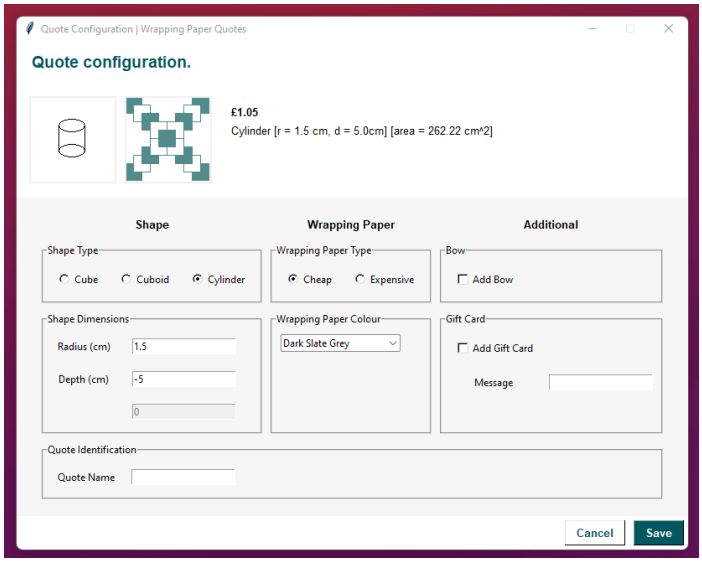
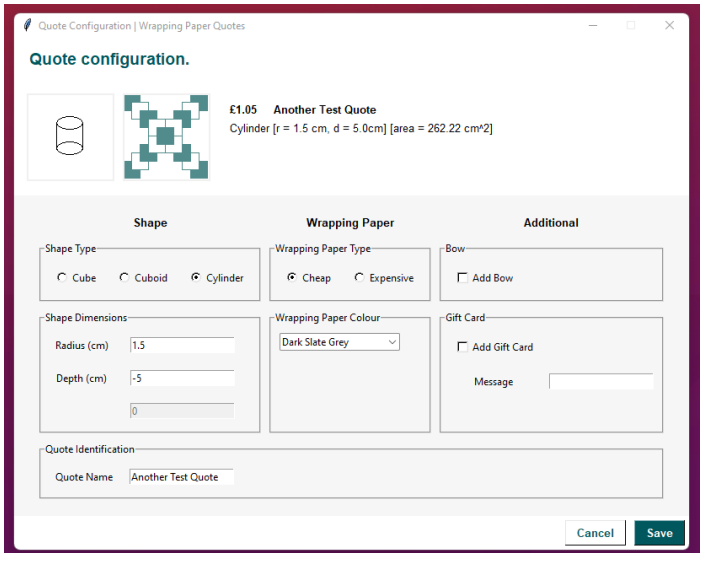
Test Reference	Screenshot and Explanations	
6.7 *Figure 18*		Figure 18 shows the quote being named for easier recall. This name is also shown next to the price.
6.8 *Figure 19*		Figure 19 shows the saved quote within the Main Window in the list box. The title text of the content has been updated to show the number of quotes and the total price of the order.
7 *Figure 20*		Figure 20 shows the functionality of the selection of quotes within a list box. All the information is shown within the list box entry and can be fully seen if the window was maximised; however, the preview that is in the quote configuration window is below this list box to separate out the information for easier readability.

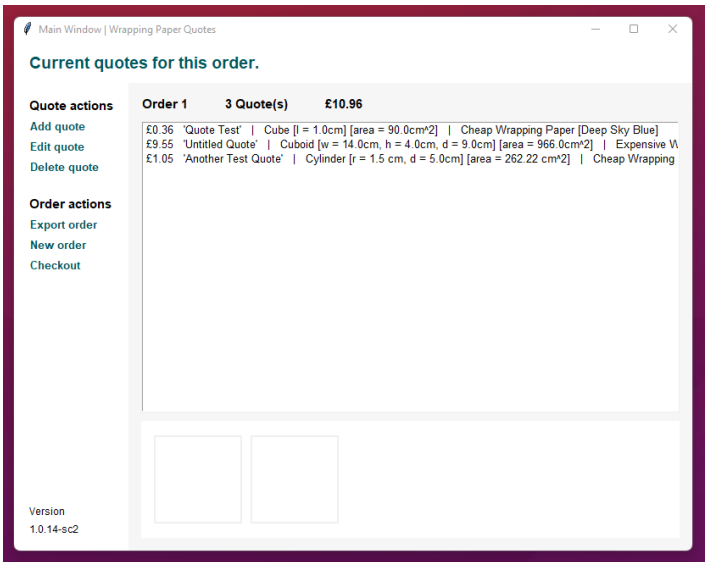
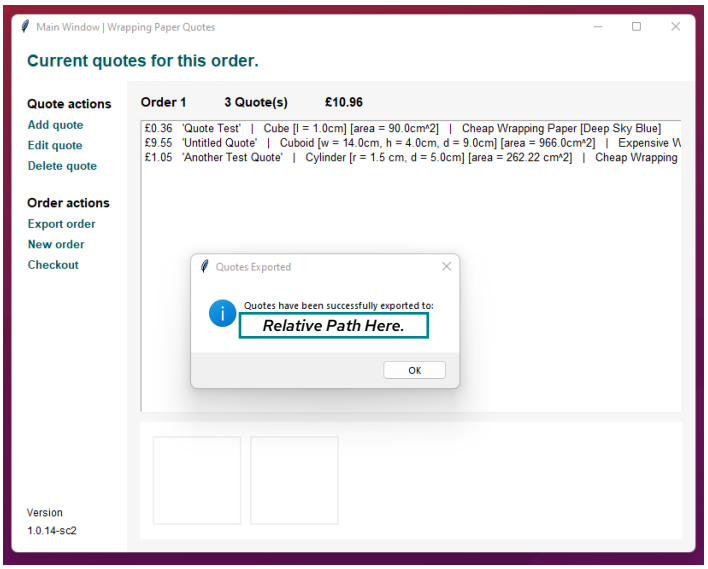
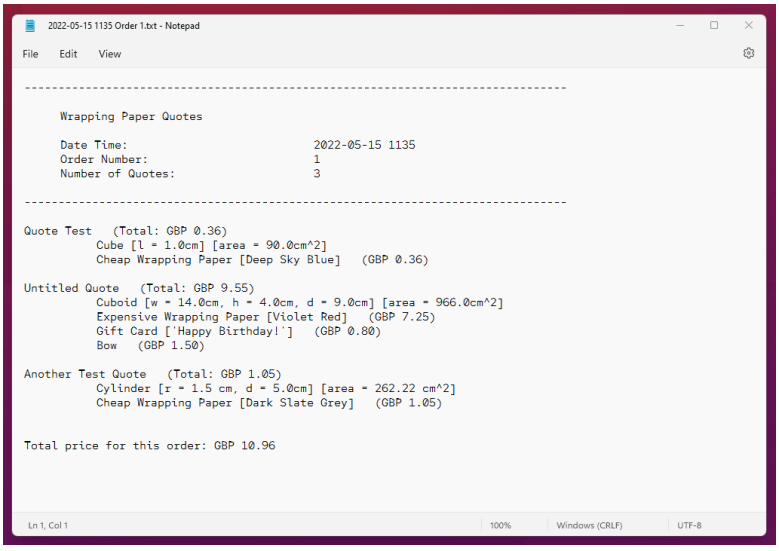
Test Reference	Screenshot and Explanations	
8.1 *Figure 21*		Figure 21 shows the graphic of the cuboid on the first canvas within the quote configuration window.
8.2 *Figure 22*		Figure 22 shows the dimensions entered. This has no effect as the wrapping paper has not be selected to determine the price.
8.3 *Figure 23*		Figure 23 shows the wrapping paper has been selected, however, the is a note informing the that black is not a colour option. The price has also been calculated.

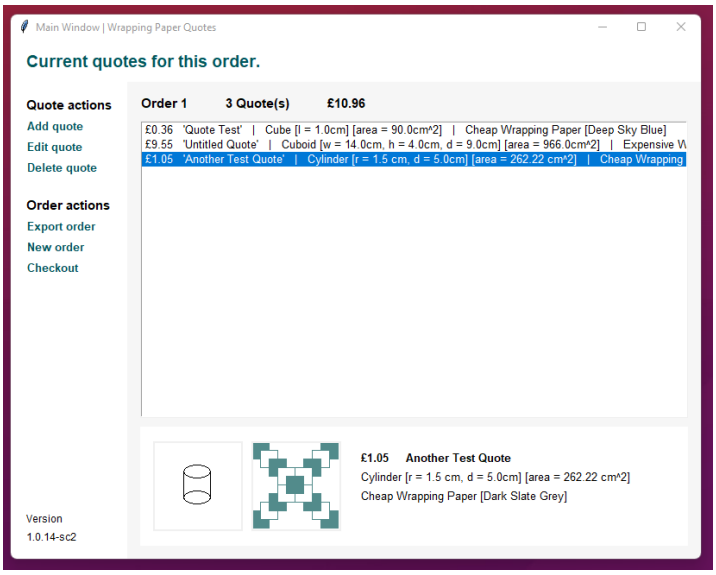
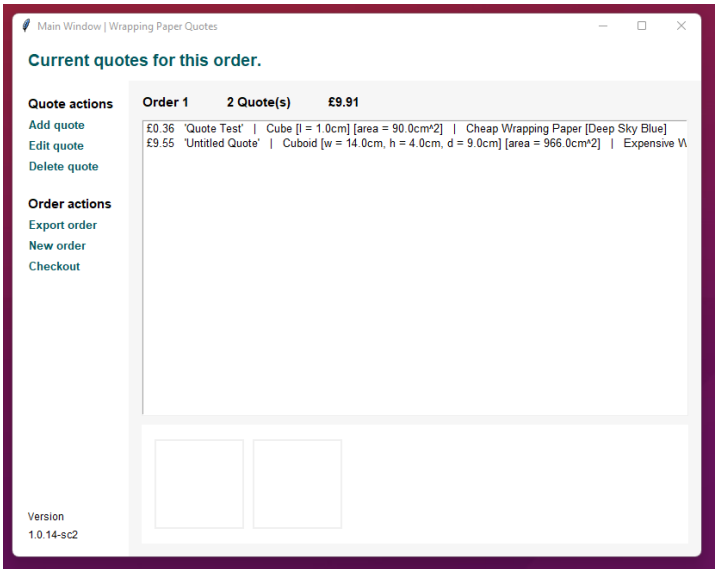
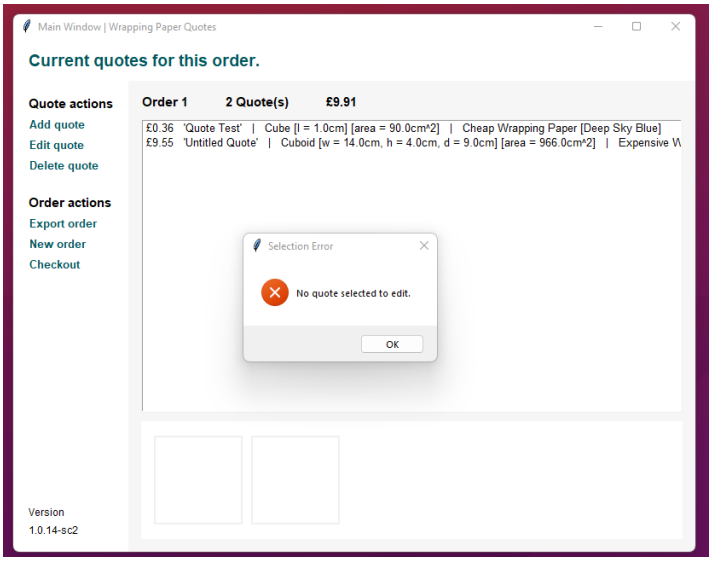
Test Reference	Screenshot and Explanations	
8.4 *Figure 24*		Figure 24 shows the re-rendered the expensive version of wrapping paper preview with colour chosen.
8.5 *Figure 25*		Figure 25 shows that the bow has been added, this shown within the price increase and the tick box being ticked.
8.6 *Figure 26*		Figure 26 shows the gift card added within the quote and the pricing has been updated to reflect this.

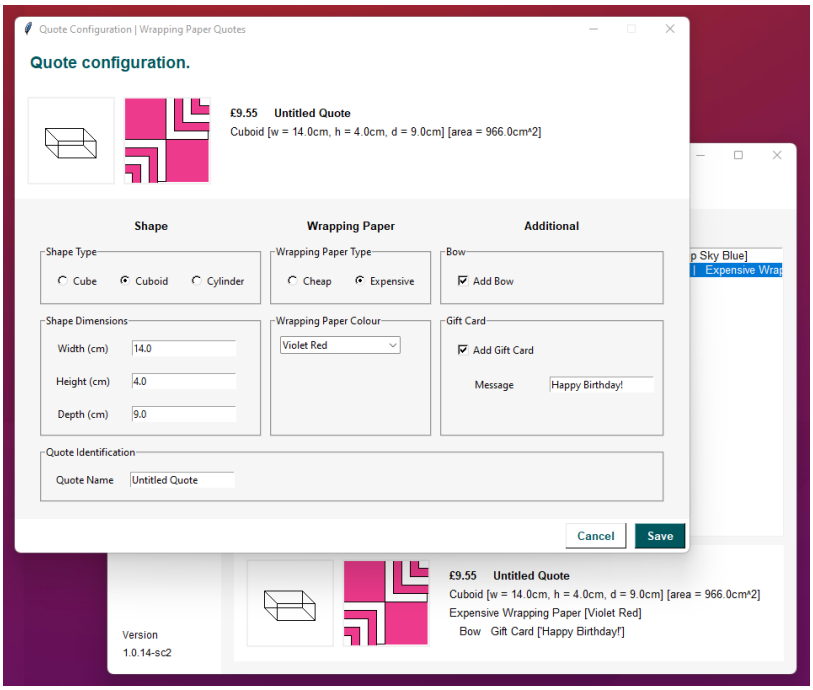
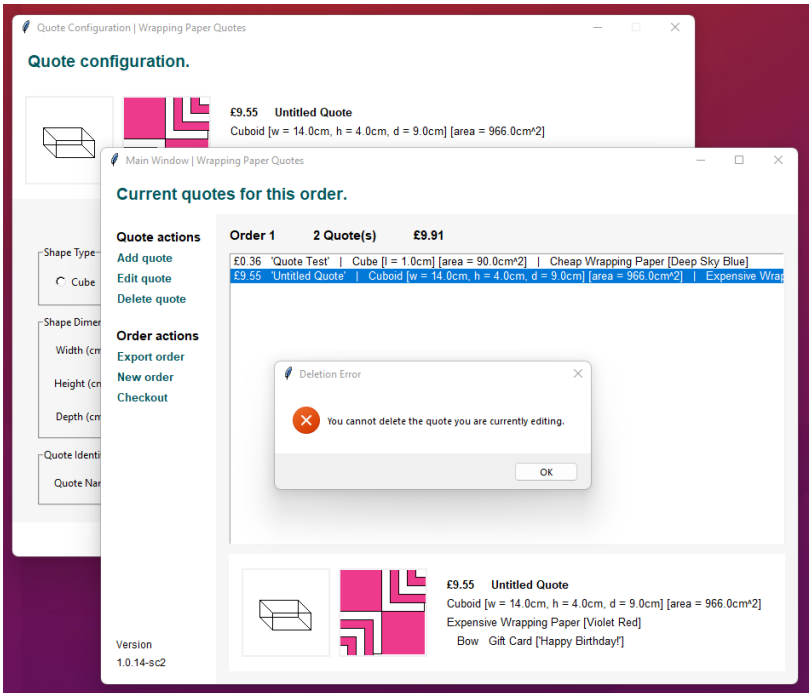
Test Reference	Screenshot and Explanations
8.7 *Figure 27*		Figure 27 shows the text entered to be printed on the gift card. The pricing has also been updated to show this.
8.8 *Figure 28*		Figure 28 shows that the quote has been saved successfully. This quote was not named so the placeholder text 'Untitled Quote' was put in the place of the name.
9.1 *Figure 29*		Figure 29 shows the cylinder graphic on the canvas.

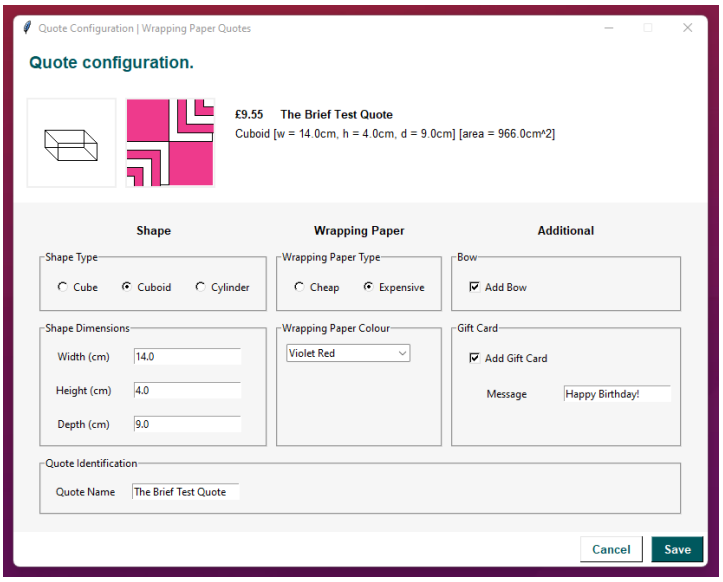
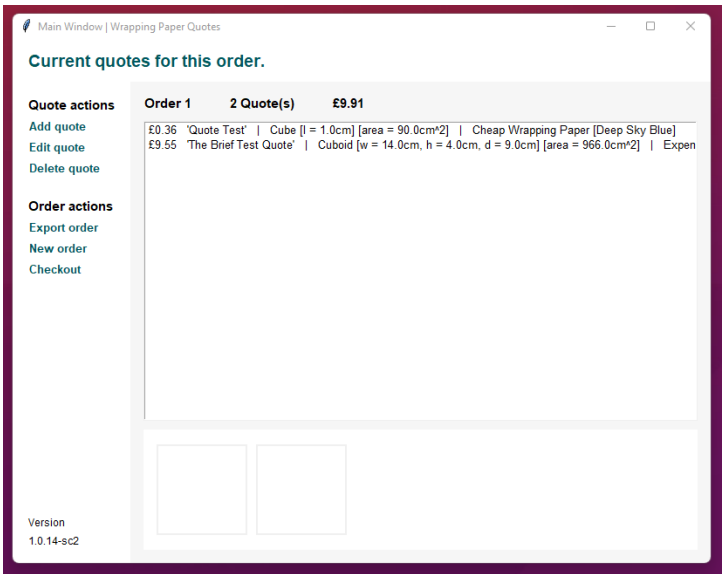
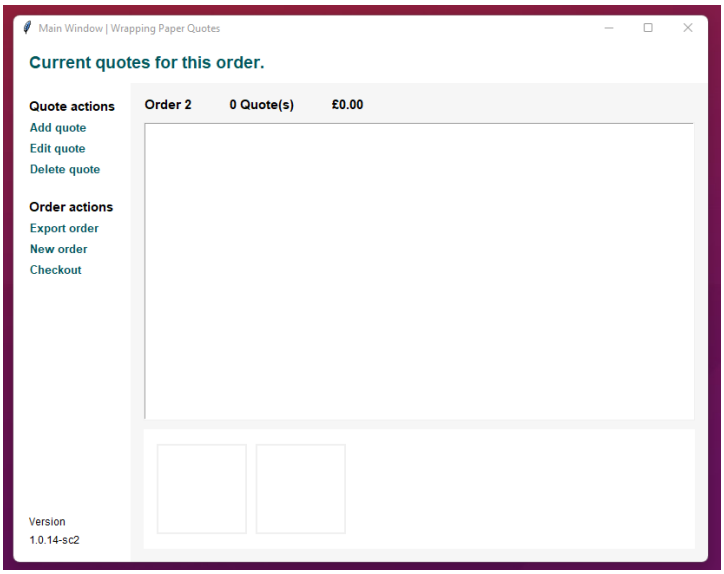
Test Reference	Screenshot and Explanations	
9.2 *Figure 30*		Figure 30 shows the cheap wrapping paper rendered as black and white as the colour has not been chosen.
9.3 *Figure 31*		Figure 31 shows the re-rendered wrapping paper colour of 'DarkSlateGray4'
9.4 *Figure 32*		Figure 32 shows a error messages informing the user that the dimensions entered are invalid.

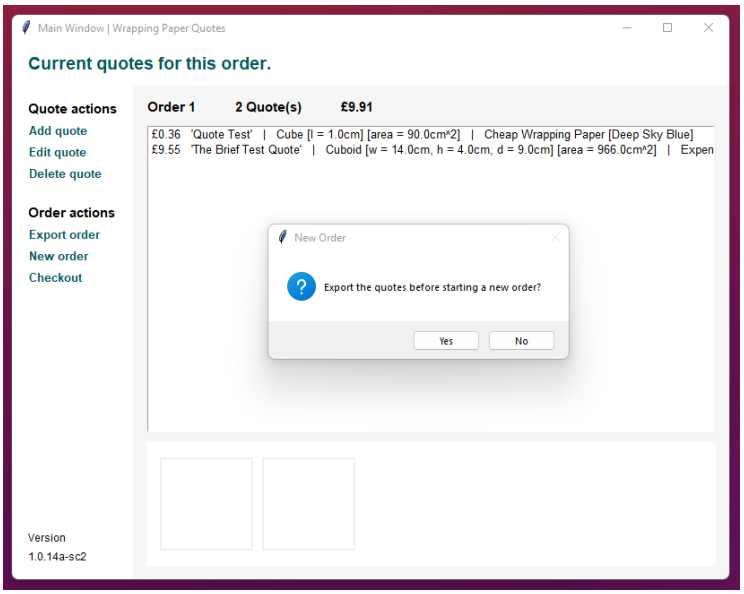
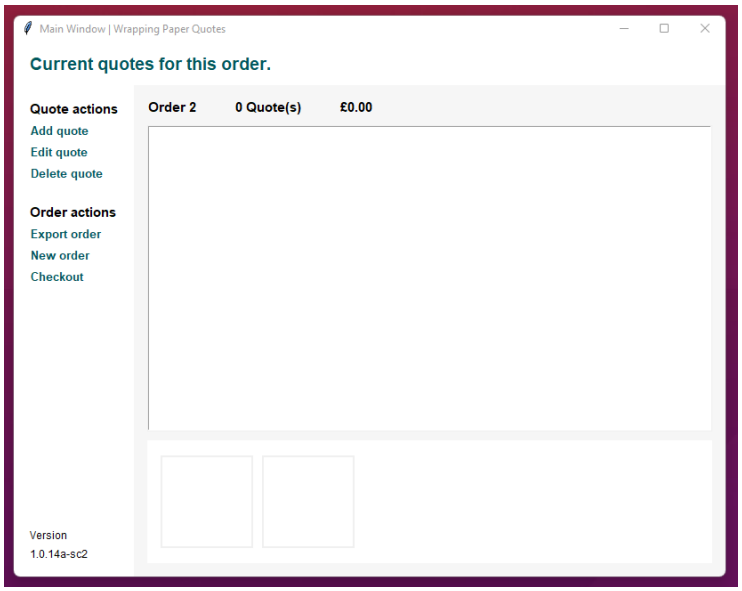
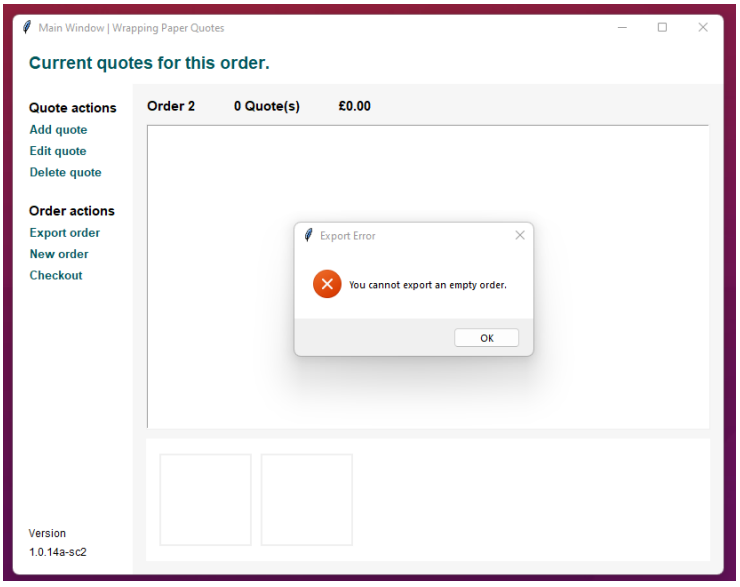
Test Reference	Screenshot and Explanations	
9.5, 9.6 *Figure 33*		Figure 33 shows the numbers entered in text and not figures to show the invalid entry handling of the application.
9.7 *Figure 34*		Figure 34 shows the ability to handle floating point / decimal numbers as well as negatives. The program takes the absolute value of the number entered to allow entry of negatives. e.g. | -5 | = 5
9.8 *Figure 35*		Figure 35 shows the names of the quote being entered.

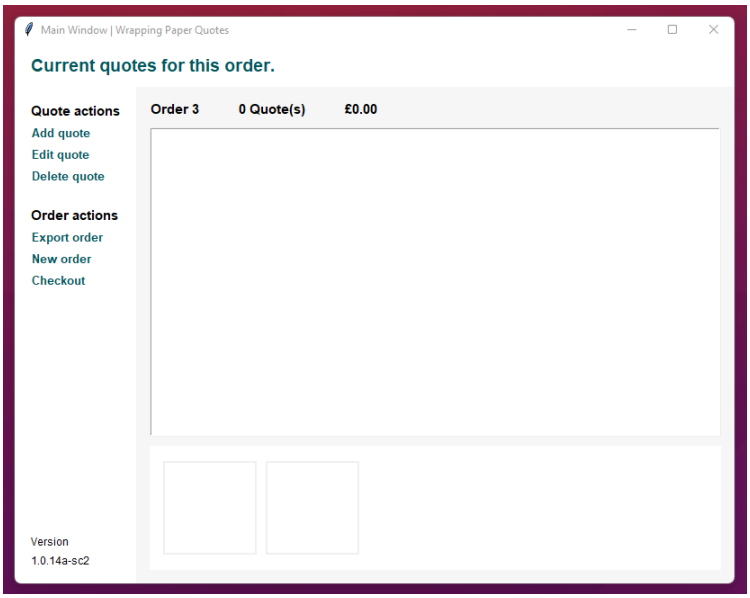
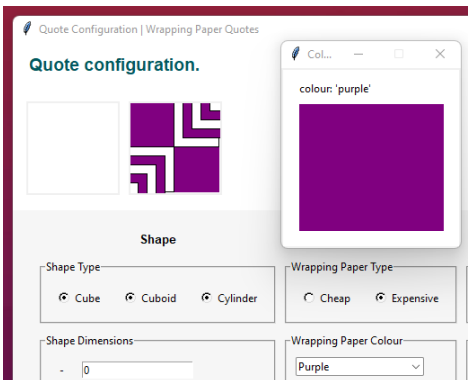
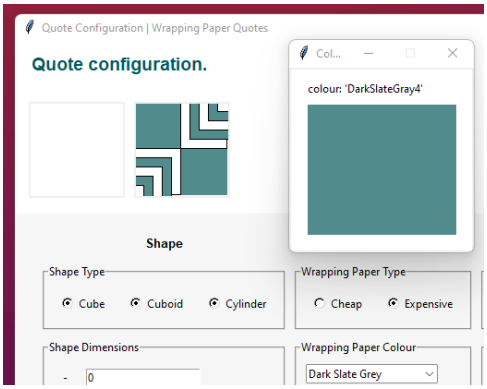
Test Reference	Screenshot and Explanations
9.9 *Figure 36*		Figure 36 shows three quotes within the current order, and the quote added was successfully saved.
10 *Figure 37*		Figure 37 shows a message box showing that the quotes have been successfully exported to the relative directory. The figure shows the user path.
Figure 38		Figure 38 shows the exported file on Notepad.

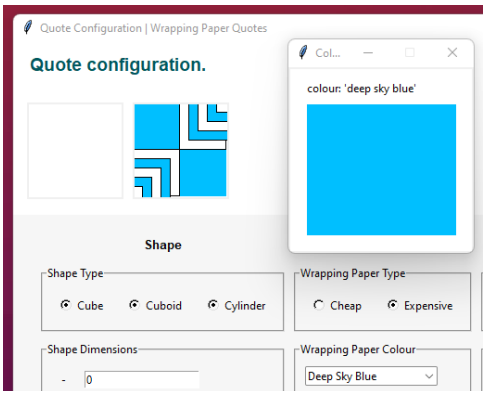
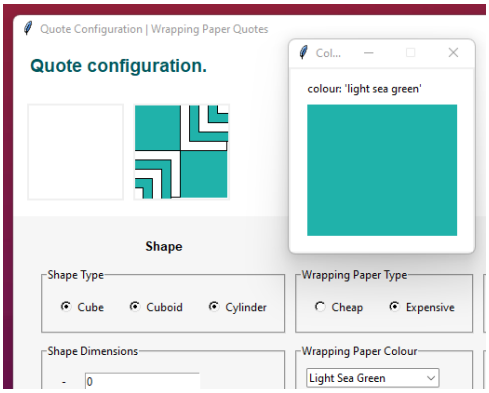
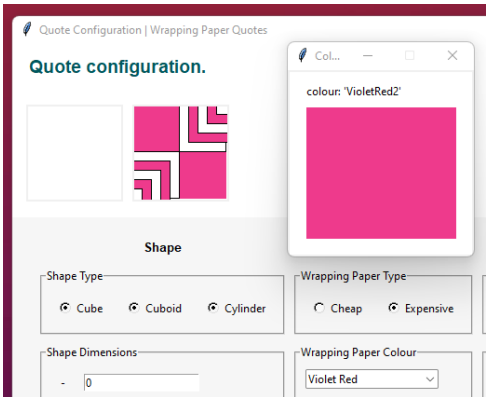
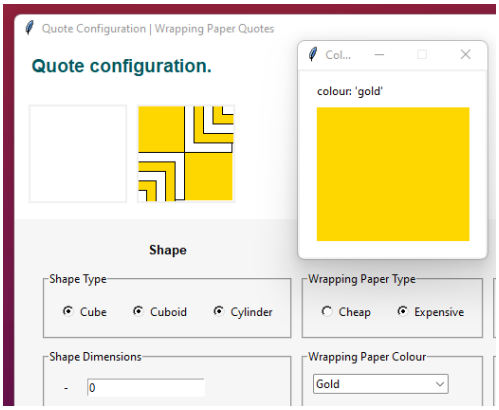
Test Reference	Screenshot and Explanations
11 *Figure 39*	 Figure 39 shows the selection of a quote to mark for deletion.
Figure 40	 Figure 40 shows that the quote has been deleted after clicking 'Delete quote'.
12 *Figure 41*	 Figure 41 shows an error when clicking 'Edit quote' due to the fact that no quote was selected.

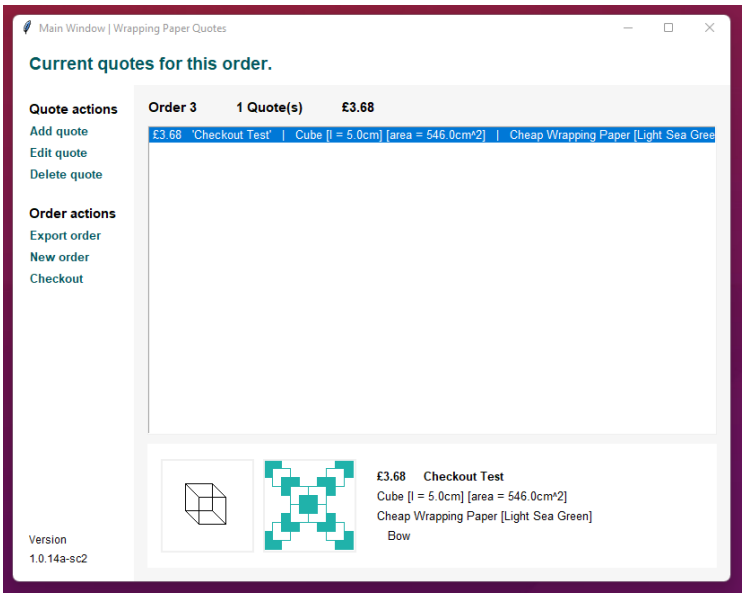
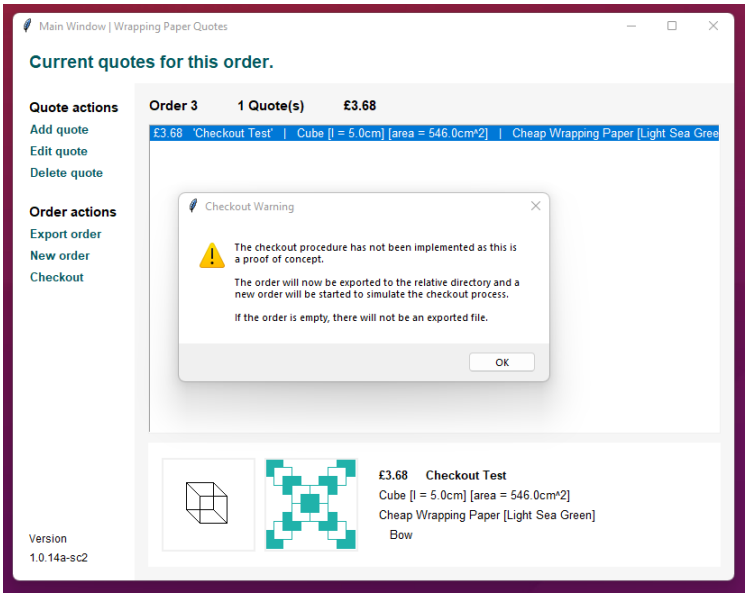
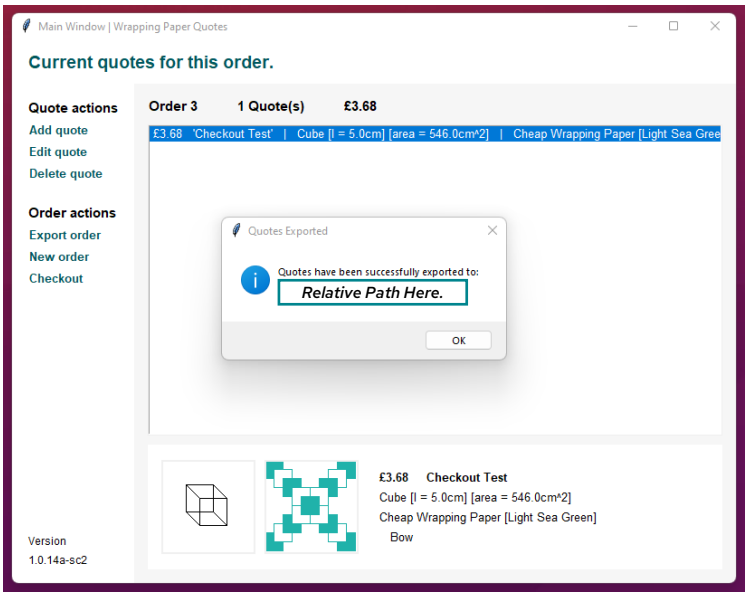
Test Reference	Screenshot and Explanations
13 *Figure 42*		Figure 42 shows the Quote Configuration Window showing the details of a quote after clicking on 'Edit quote' with a quote selected.
14 *Figure 43*		Figure 43 shows the program ability to know which quote is currently being edited and will not allow you to delete it.

Test Reference	Screenshot and Explanations
15 *Figure 44*	 Figure 44 shows the quote name being changed for this specific quote.
Figure 45	 Figure 45 shows the quote has been updated successfully with the new quote name.
16 *Figure 46*	 Figure 46 shows the a new order has been starting. This is seen from the incremented order number. There should have been a message box asking if the order should be exported as the order has changed since the last export.

Test Reference	Screenshot and Explanations
16a *Figure 47*	 *Updated Version (1.0.14a-sc2) which fixes the issue found in test ref. 16* Figure 47 show the message box if the order has been modified after exporting or has not been exported once.
Figure 48	 Figure 48 shows a new order after the user answers the message box.
17 *Figure 49*	 Figure 49 shows a message box informing the user that an empty order cannot be exported.

Test Reference	Screenshot and Explanations
18 *Figure 50*	 Figure 50 shows that a new order has been started. No message box was shown asking about exporting the previous order due to the previous order being empty.
19	**!** | *For these sub-tests, another application was made to show the colour and compare it to colour found within the program. This is to make sure the application is translating the colours correctly.*
19.1 *Figure 51*	 'Purple' >> 'purple'
19.2 *Figure 52*	 'Dark Slate Grey' >> 'DarkSlateGrey4' *The user interface now says 'Dark Slate Grey 4' in version 1.0.14b-sc2*

Test Reference	Screenshot and Explanations	
19.3 *Figure 53*		'Deep Sky Blue' >> 'deep sky blue'
19.4 *Figure 54*		'Light Sea Green' >> 'light sea green'
19.5 *Figure 55*		'Violet Red' >> 'VioletRed2' *The user interface now says 'Violet Red 2' in version 1.0.14b-sc2*
19.6 *Figure 56*		'Gold' >> 'gold'

Test Reference	Screenshot and Explanations
20 *Figure 57*	 Figure 57 shows a new quote created to show the implementation of the checkout process as this is a proof of concept.
Figure 58	 Figure 58 shows the warning given to the user when the 'Checkout' button is clicked. This warning explains the process of how it has been implemented.
Figure 59	 Figure 59 shows that the quote has been exported.

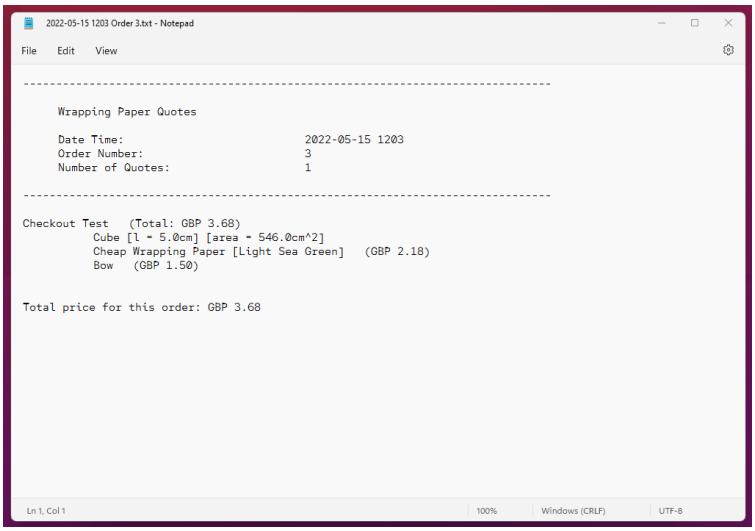
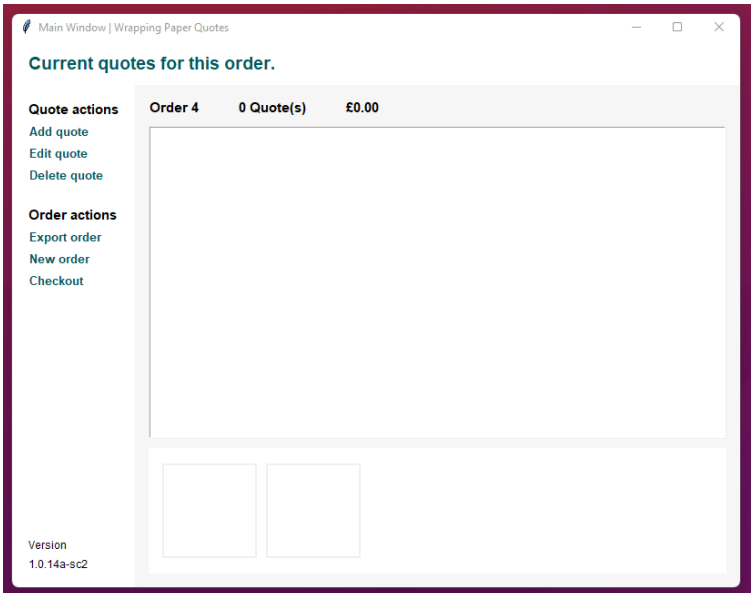
Test Reference	Screenshot and Explanations
20 continued... *Figure 60*	 Figure 60 shows the exported quote.
Figure 61	 Figure 61 shows that a new order has been started.

Table 5: Functional Testing Evidence Screenshots

4 Reflection

The Module

During this module, I have learnt how a project can be documented as well as improved my ability to program as I had got more practice in to continue further developing programming concepts.

The process of documenting the development process was probably the hardest thing, as I have not really ever done this. During my previous development project, I had experienced explaining the brief and solution, the context, objectives/requirements, flowcharts and class diagrams, the user interface, and testing. This module further developed these skills as well as different ways to show interactions besides flowcharts and class diagrams.

During the programming portion of the module, I did learn a few things involving types and using getters and setters within the object as well as outside. I also spent the time comparing the code displayed within the demos and how I would have done it. This is because, I have tendency to 'over-engineer' the code as I have a habit of implementing error handling, so the program does not break over an incorrect input or change. One thing I wanted on work on during this assignment is efficiency by avoiding continuous repetition.

The Assignment

First section of the assignment – the project initiation – was the part that went well compared to the other sections, as I knew what I was doing. I wrote an introduction to give context, which I did assume parts of why the department may need a piece of software or 'proof of concept' to calculate quotes. After this, explaining the scope, deliverables, acceptance criteria, exclusions, constraints, and assumptions. This is a similar structure of the project scope statement found on Canvas. This helped me to analyse the case study to ensure that I understood it fully and was able to propose a methodology as well as create relevant requirements to the case study. The creation of requirements also went well as I found it simple as I understood the case study after all the analysis. I kept the documentation in this order as it felt the most logical to me.

The next section being the design section was the most difficult for me as I had to focus on creating a multitude of diagrams, which I was not one hundred percent clear on. I was to create the diagram including only the system, which did confuse me at first, as I always thought the system was the functionality part and anything graphical or visual is not. Out of all the diagrams, I found the sequence diagram the hardest to create, as I found it difficult to show the objects interacting with each other. To combat this, I showed the interactions and messages between the actor - the department employee, the graphical user interface, and the system as a whole. For the user interface, I came up with multiple designs and chose one. Personally, I am not exactly happy with it, as I would like to change it slightly. I was also restricted as I could not include any images with the buttons to help visually. I went with a sidebar design as I see that in other applications. I made sure that the contrast between background and foreground was enough for visually impaired people as the main accessibility feature. In addition to this, I kept the application text minimal but still informative of any issues or errors that may occur during operation.

During the programming part, I made sure that I closely followed PEP8, the style guide for python, with the help of a python extension called 'pycodestyle'. At first, I always got warnings and errors on blank lines and whitespace, the character limit for a line and other formatting issues, but after a while, I got used to making sure that the code met all the guidelines. Besides this, I decided to keep the system's functionality separate to the user interface, making sure every function was modular and efficient. This would allow easy access to the functionality, which can work independently from the graphical user interface. This means that a developer could remove the code forming the interface and re-write the interface without affecting the functionality. This justifies the use of PEP484, type hints, as this helps with readability of code, if anyone

needs to work with system itself, understanding the data entered. Making the implementation of the interface was something I found difficult as I was not familiar with the tkinter framework. I first experimented with this framework to create some GUIs to see what can be made and how the widgets and layouts worked. This is noted in Appendix A as well as the changes between versions. After this, I felt most comfortable with using the pack layout manager, however, I did use grid within a frame quote options as it was easier. I kept re-writing the interface due to the fact that I felt overwhelmed by the amount of code, if I needed to go through to fix anything. To solve this, I separated everything into functions, including how widgets are initialised.

During the testing part, I was not sure how I would structure the tests in the given table. There is not much evidence problem solving as I came to testing the program, as it only failed one. This failure was a logic error that I easily spotted once I came across it, due to the context. I incremented the version to distinguish a difference to know that the issue had been fixed and that I was now testing the newer version as I continued with the rest of the tests. There is not much because during the programming portion, I spent time working on the error handling, which can be seen through the testing evidence screenshots. I did change the colour names displayed in the GUI after testing, but as this did affect the operability of the program, I decided to put a note in the corresponding test reference instead of re-doing it, that it has changed in a later version. In addition to this, I knew there were certain issues with the application at the time that did not affect the operation of the end user. These issues are listed in Appendix B.

To improve application, I would recreate the interface, but making sure that this more efficient as I am going along as I have a better understanding of the tkinter framework. I would use the translator inter-operation class to do more of the checks to avoid further repetition of code. I think if I was able to use multiple files, I would be able make the sections more modular and may overcome the issue of re-writing the entire interface constantly. In addition, I would like to change the design to remove the sidebar have a command bar ribbon below the header of the window. More accessibility features could be added such as the ‘are you sure’ when deleting a quote. As well as this, being able to import previous quotes and continue to be able to export quotes in a human readable format and in a program readable format to allow an easier to change existing orders.

In conclusion, this module helped develop some skills, more to do with documentation then programming. I managed to learn the tkinter framework. As I have gone on to doing programming projects, the more I have improved in coding and helped consolidate my knowledge on object-orientated programming and how it is implemented within a program.

5 References

Crawford, D. (n.d.) *What are Non-Functional Requirements?* Available at: https://www.modernrequirements.com/blogs/pillar/what-are-non-functional-requirements/ [Accessed 25 March 2022]

Gaille, L. (2020) *15 Advantages and Disadvantages of a Waterfall Model*. Available at: https://vittana.org/15-advantages-and-disadvantages-of-a-waterfall-model [Accessed 07 February 2022]

Kitchen (n.d.) *10 Advantages of Waterfall Model – When, Why, and How to Use It*. Available at: https://kitchen.co/blog/advantages-waterfall-model/ [Accessed 22 May 2022]

Korolev, S. (2021) *MoSCoW Method: How to Make the Best of Prioritization*. Available at https://railsware.com/blog/moscow-prioritization/ [Accessed 25 March 2022]

Lucidchart (n.d.) *The Pros and Cons of Waterfall Methodology*. Available at: https://www.lucidchart.com/blog/pros-and-cons-of-waterfall-methodology [Accessed 03 February 2022]

Lynn, R. (n.d.) *Disadvantages of Agile*. Available at: https://www.planview.com/resources/articles/disadvantages-agile/ [Accessed 07 February 2022]

Microsoft (2021) *Navigation design basics for Windows apps*. Available at: https://docs.microsoft.com/en-us/windows/apps/design/basics/navigation-basics [Accessed 24 April 2022]

Santos, J. M. D. (2021) *What is Scope Creep in Project Management?* Available at: https://project-management.com/scope-creep/ [Accessed 13 February 2022]

tutorialspoint (n.d.) *SDLC – Waterfall Model*. Available at: https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm [Accessed 02 February 2022]

Visual Paradigm (n. d. a) *What is Use Case Diagram?* Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/ [Accessed 21 May 2022]

Visual Paradigm (n. d. b) *What is Activity Diagram?* Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/ [Accessed 22 May 2022]

Visual Paradigm (n.d. c) *What is Sequence Diagram?* Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/ [Accessed 22 May 2022]

WebAIM (2021) *Constructing a POUR Website – Putting People at the Center of the Process*. Available at: https://webaim.org/articles/pour/ [Accessed 24 April 2022]

6 Appendices

6.1 Appendix A: Application Version History

Please note that 1.0.1 to 1.0.6, was experimental testing the tkinter elements to see how they interacted with each other and making sure the system was structure worked correctly without dependence on the graphical user interface.

Version	Changes
1.0.1	• A test version for the Main Window.
1.0.2	• Canvas playground to construct wrapping paper and shape graphics.
1.0.3	• Added the ability to customize a quote.
1.0.4	• Ability to support multiple quotes was added. • System structure was complete. • The Main Window had two frames for editing and viewing the quotes. • The quote summary pane was added. • Test quotes were added.
1.0.5	• Editing frame screen was removed; no functionality was added. • The quote non-graphic summary pane was added. • INTERNAL: doc strings were removed.
1.0.6	• All elements of the User Interface were removed. • New 'pence to pounds' converter function was added
1.0.7	• New user interface was created. • Navigation bar was added. • New Actions sidebar. • List box to show the list of quotes. • New Window added for editing or creating a quote. • New quote summary pane was added. • Ability to preview the chosen wrapping paper with colour and graphic of the shape chosen. • Ability to name the quote was added. • Application Metadata are now global constants.
1.0.8	• The actions sidebar has two sections 'Quote actions' and 'Order actions'. • Ability to update quotes list manually was added via 'Update quotes' button. • 'Options' button was removed. • Actions sidebar button colour changed for better readability. • 'Total price' was added to the Main Window. • Navigation bar foreground and background colours switched. • The editing quote window was renamed to the quote configuration window. • The price of the current editing quote was added to the 'Quote Configuration' window. • Test quotes were added. • __str__ method now used to return the summary of the objects within the system to display within the user interface.

	• New round number method was added for accurate rounding using the decimal import. • The quote summary pane was added to the Main Window to preview quote would needing to go the editing window • Quote Summary Pane was completed.
1.0.9	• The get recommended area function will now call the base class to simplify working the area with the height and the width of the net of the present, taking them in as keyword arguments. • The 'Translator' class was introduced to continue to remove repeated code. This is now an inter-operation layer between the system and the user interface. • New method to get the date and time as a string was added. • Ability to export the order. • List box of quote now automatically updates. • Added typing import for type hinting a list – Fixed the incompatibility of using an older version of Python. • INTERNAL: Code now specifies if an argument is positional or a keyword, with '/' or '*'.
1.0.10	• Additional calculation error checks were implemented
1.0.11	• 'Total price' panel was removed. Now included within the title of main content frame, as well as the order number and number of quotes.
1.0.12	• Area recommended to calculate cylinder was corrected. • More detailed error messages added. • Navigation bar text colour was changed for optimal readability.
1.0.13-sc1	• The test quotes were removed. • Quote to string method was redone. • Quotes list box no longer includes the quote index.
1.0.14-sc2	• The 'Update quote' button was removed, as it was no longer needed. The 'New quote' button was changed to 'Add quote'. • The exported order now shows the number of quotes within the order.
1.0.14a-sc2	• Fixed an issue of the export message box not showing when the user starts a new order, when the order has been changed after a previous export.

1.0.14b-sc2	• Ensured that all debugging symbols have been removed.
	• The Navigation Bar is now internally named the Header of the Window.
	• The close button on the Quote Configuration Window fixed the saving ability.
	• Human readable colour names updated to reflect the ones in the case study
	  ○ Violet Red >> "Violet Red 2"
	  ○ Dark Slate Grey >> "Dark Slate Grey 4"

Table 6: Application Version History

6.2 Appendix B: Application Limitations & Known Issues

ID	Details
KI_01	Cascading Error Messages Reproduction Steps: Click 'Add quote' to get the Quote Configuration Window. When a quote is not complete or has incorrect inputs, you can click the 'Save' button multiple times.
KI_02	The program is able to calculate the price of the wrapping paper required if one of the dimensions are filled for the cylinder or two of the dimensions are filled for the cuboid. Reproduction Steps: Click 'Add quote' to get the Quote Configuration Window. Choose a wrapping paper type. EITHER Choose cuboid and type numerical values into two of the entry boxes (anyone of them), leaving one entry box with 0 or no value at all OR Choose cylinder and type numerical values into one of the entry boxes (anyone of them), leaving one entry box with 0 or no value at all Partial Fix: The Quote Configuration Window will not allow you to save the quote without all dimension fields are filled in correctly.
KI_03	Error Closing: Main Window becomes blank and while Quote Configuration is open. Not been able to reproduce this.
L_01	Python Version Requires Python 3.8.0 or Higher for the program to work.
L_02	The Quotes List Box on the Main Window. Having a lot of quotes within the list box. The widget does not have a scrollbar bound to it. The end user will need to use the arrow keys up and down or click drag the mouse for this. The summary string of quotes are cut off within the list box; this window can be maximised to combat this. The Quote Summary Pane also shows the same information but in a more structed format with previews.

Table 7: Known Issues and Limitations within the Application

END OF SUMMATIVE

Feedback

What you did well:

- Very good consideration of the POUR guidelines to support your development of the application
- An extensive test plan, which is well evidenced
- A very good overview of the system, with a clear understanding of how it needs to be developed
- Clear and easy to use interface

Aspects of this assignment you could develop and improve:

- It is good that you have error handling within your interface, but, this could be further developed by integrating more descriptive feedback – for example, I entered a letter where a number should be and the error simply told me the order was incomplete
- For your future requirements, think about how you would gather the data. It is good that you will get information from the employees, but make sure you are also considering the methods you would use to achieve this.
- Think about your inheritance – it is good that you have included it, but the subclasses must add something to the superclass. Currently, your cheap and expensive wallpaper don't add anything to the superclass

Something to think about for your future learning:

- POUR is focussed on Accessibility. I would recommend, that you take a look at the usability aspects as well. This will further enhance the justification of your interface designs
- Try to include more consideration in your reflection about what you would do differently next time with a view to supporting your personal development over time.