



# Unit 12 – Tasks 2 – 4 Software Design Report

SOPHIE MAY  
9800226832

# Table of Contents

---

Introduction.....	3
Purpose of the Program .....	3
Problem Definition .....	3
Requirements .....	4
Input Requirements.....	4
Process Requirements.....	4
Output Requirements.....	4
Business Requirements (Non-Functional Requirements) .....	4
Solutions .....	5
Bespoke Software.....	5
Off-The-Shelf Software.....	5
Customisable Software.....	5
Designs .....	6
GUI Designs.....	6
Introduction.....	6
1st Design .....	6
2nd Design .....	7
Components Table for 1st Design .....	7
Components Table for 2nd Design .....	10
Navigation for 1 <sup>st</sup> Design .....	12
Navigation for 2 <sup>nd</sup> Design .....	13
Implemented Requirements for 1 <sup>st</sup> Design .....	13
Input .....	13
Process.....	13
Output .....	13
Implemented Requirements for 2 <sup>nd</sup> Design .....	13
Input .....	13
Process.....	14
Output .....	14
Data Validation for 1 <sup>st</sup> Design.....	14
Data Validation for 2 <sup>nd</sup> Design .....	14
Data Dictionary.....	14
Introduction.....	14
What is a variable? .....	15
Size of a variable.....	15
Processes .....	15

Store User Data .....	15
Update Exchange Rate (optional).....	16
Find the Current Exchange Rate .....	16
Calculate Exchanged Amount.....	16
Display Result .....	16
Reset/Clear Data.....	16
Flow Chart .....	16
What is an algorithm? .....	16
What is a flowchart?.....	17
What does a flowchart show? .....	17
Why is a flowchart important in programming? .....	17
Test Plan .....	17
Justifications .....	19
Task 3.....	20
Original Code .....	21
Test Log.....	0
Updated Code.....	0
User Interface .....	3
Altered Code.....	0
Feedback.....	0

# Introduction

---

Planning is a very important part of any project, before beginning a project, it is important to design the product to show any errors or alter the product so that the final product will be completed on time, and to a high standard.

Without a design or a plan to work towards, we won't be able progress with the project as we do not know what we will need to create the product.

Projecting a timescale is also very important, without planning how much time we need to spend on each area of the project to work out how much time the project will require.

## Purpose of the Program

---

The company (a local travel agent called "Perfect getaways"), has requested a design for a currency exchange rate calculator program, this program has been designed to convert an amount in British pounds to the equivalent amount in a foreign currency (such as Euro, US Dollar, Bangkok Dollars etc.)

Additionally, this program will also be able to convert foreign currency into British pounds. Once the program has made the required/proposed calculation, the amount will be displayed on screen.

## Problem Definition

---

A problem definition is an explanation of any issues that occur (or that may occur in the future), that must be solved or overcome.

- 1) The travel agency manually works out the currency exchange by hand, so they may make mathematical errors (human errors), which could lead to unhappy customers and loss of revenue.
- 2) Cannot make bookings online, the company use a dumb terminal, this is outdated and loses the company revenue as most customers book online these days.
- 3) The system is outdated and is not time-efficient, so it takes longer to convert the currency which could prove frustrating for both the customer and the employee, which could cause the company to lose the booking.
- 4) Customers have said that they are not happy with the service that the travel agents company has provided and this will most likely result in the company losing customers and revenue which will ultimately impact their reputation and they will lose business.
- 5) If an employee from the company makes an error when calculating the currency exchange by hand, the company may lose revenue, which could ultimately result in that employee losing his/her job.

# Requirements

---

A requirement is a specified target or a condition that must be adhered to. For example, a requirement in the specified brief (which was designed for the program) was to convert British currency into the foreign currency equivalent. This is a non-negotiable specification that must be implemented into the programs core design.

A 'functional requirement' within a program is defined as the purpose of the program, so it is the function or action that the program executes. For example, a requirement of the program is that the program must include an input feature to insert the required data information in order to perform the required calculation.

## Input Requirements

The first input requirement is that there is a field available, to input the amount value to be exchanged.

The second input requirement is the currency type (that the program will convert the first amount into, for example, US Dollars).

The third input requirement is the currency to exchange from (output type that the original amount changed to, for example, Euros).

The fourth input requirement is the calculation button, to instruct the program to calculate the exchange rate based on the (above) information given by the user.

## Process Requirements

The process requirement is the calculation process itself, this is the calculation for the exchange rate, converting the original value to the new, exchanged value.

## Output Requirements

The output requirement is to display the answer of the processed calculation, in the specified answer output field.

Another output would be to display an error message to the user if they make an error when inputting data into the calculator.

## Business Requirements (Non-Functional Requirements)

These are the requirements that the business would like to achieve in order to boost profits, improve staff performance and reduce costs to the business. These are non-functional requirements and they include any additional criteria that the business would like the program or system to do to aid the business.

- The currency exchange program must reduce the amount of errors made by staff during the exchange calculation of the currency to improve the service provided to customers.
- The new program should not require the use of pens and paper, or the need for staff members to intervene with the calculations in the program, as the formulas for the calculator must be built into the program and it must do all the calculation work in the program.
- The program must improve customer service by reducing the wait times for the currency exchange services they provide.
- The program must be simple and easy to use to ensure that the staff members can operate the program.

# Solutions

---

There are three possible solutions that the client can choose from, these are bespoke software, off-the-shelf software and customisable software.

## Bespoke Software

Bespoke software is software that is tailor-made and developed for an individual user, it is designed, developed and maintained for that users' needs and requirements.

### Advantages of this software are:

- The software is tailored to the clients' needs and can be made to include extra features to improve productivity and accuracy.
- Bespoke software is scalable and can be altered and changed for the client as their needs change and their business grows.

### Disadvantages of this software are:

- The software is much more expensive than off-the-shelf software.
- The software must be designed and developed from scratch so it takes time to implement this system for the client.

## Off-The-Shelf Software

Off-the-shelf software is software that is ready-made for mass production and distribution to a broad range of customers. It cannot be altered.

### Advantages of this software are:

- This software is cheaper to purchase than bespoke software.
- Off-the-shelf software is easy to use as it is designed to be used by a wider audience so it is designed to be more user friendly than bespoke software.

### Disadvantages of this software are:

- The help available is not as readily available as bespoke software which is a direct line to the developer of the system.
- The software cannot be altered to suit the individual needs or requirements of the user.

## Customisable Software

Customisable software is software that is already made but can be customised to the needs of the user. It is a cheaper alternative to bespoke software as long as the user has the time to modify it for their individual needs.

# Designs

## GUI Designs

### Introduction

For the benefit of the client, I have produced two designs for the exchange rate calculator program, one is a green and dark-grey, horizontal design and the other is a contrasting, orange and green vertical design. In this report, I have detailed the features and components of each design.

The dark-grey and green design allows the user to enter the amount they wish to exchange into a text field, select the currency of this amount and then select which currency they wish to change the amount into. It also comes with a clear button to allow the user to clear all data from text fields and other exchange rate field to enter a currency exchange rate of their choosing, this is good for updated rates and other currencies that are not listed as an option.

I plan to take the green and dark-grey design to production, as I feel this design fulfils the client's specified requirements. The orange and green vertical design, will be discarded as it is visually and functionally inferior to the other design, the font is difficult to see and the colours, orange and green, are not suitable for the program as they affect its usability for the user.

### 1st Design

The screenshot shows a window titled "Currency Exchange Calculator" with a dark grey background and a green title bar. The interface includes the following elements:

- 1**: Title bar "Currency Exchange Rate Calculator" in white text on a green background.
- 2**: Label "Enter Amount" above a text field containing "546.00".
- 3**: Label "Select Currency From" above a dropdown menu showing "GBP".
- 4**: Label "Select Currency To" above a dropdown menu showing "Euro".
- 5**: Label "Other Exchange Rate" above an empty text field.
- 6**: Label "Result" above a text field containing "612.22".
- 12**: A green "Calculate" button.
- 13**: A grey "Clear" button.
- 14**: An error message box with the text "\*Incorrect data type entered".

## 2nd Design



Components Table for 1st Design

No.	Components Type	Components Name	Text Type	Font Type	Font Colour	Font Size	Background Colour
1	Label: "Currency Exchange Rate Calculator"  This tells the user what the purpose of the program is and what the program is.	lblTitle	String	Times New Roman	#ffffff	28pt	#64a538
2	Label: "Enter Amount"  This tells the user where they need to input the amount they wish to convert.	lblEnter	String	Times New Roman	#ffffff	14pt	N/A
3	Label: "Select Currency From"  This tells the user that they need to select the currency type for the input amount that they wish to convert.	lblFrom	String	Times New Roman	#ffffff	14pt	N/A
4	Label: "Select Currency To"	lblTo	String	Times New Roman	#ffffff	14pt	N/A



	<p>This tells the user that they need to select the currency type for the output result.</p> <p>A currency must be selected in order for the program to display the correct exchanged amount in the output box.</p>						
5	<p>Label: "Other Exchange Rate"</p> <p>This tells the user that they can enter a specific numerical value into this box text field in order to alter/update the calculation process in order to calculate a specific rate.</p>	lblOther	String	Times New Roman	#ffffff	14pt	N/A
6	<p>Label: "Result"</p> <p>This tells the user the amount that has been exchanged and where the resulting output amount is to be displayed.</p>	lblResult	String	Times New Roman	#ffffff	14pt	N/A
7	<p>Label: "Calculate"</p> <p>This tells the user to press this button in order to start the calculation process.</p>	lblCalculate	String	Times New Roman	#181717	14pt	N/A
8	<p>Label: "Clear"</p> <p>This tells the user to press the button in order to clear the text field data.</p>	lblClear	String	Times New Roman	#181717	14pt	N/A
9	<p>Button: "Calculate"</p> <p>User presses the button and it triggers the assigned response (calculation process).</p>	btnCalc	N/A	Times New Roman	N/A	14pt	#64a538
10	<p>Button: "Clear"</p> <p>User presses this button and it triggers the assigned response (removes data from text fields).</p>	btnClear	N/A	Times New Roman	N/A	14pt	#d0cece
11	<p>Message Box: Error Message Box</p>	msgError	String	Times New Roman	#181717	14pt	#ffffff

	This is a pop up error message box to display a message to the user when an incorrect data type error has occurred (only appears when the calculation button is pressed).						
12	Text Box: Enter Amount (Input field)  This is used to enter the original amount/number data for the calculation.	txtAmount	Number	Times New Roman	#181717	14pt	#ffffff
13	Text Box: Other Exchange Rate (Alternative Currency Rate field)  This is used to enter another currency format for the calculation and can be used for either the currency to be exchanged from a certain type or the currency to be exchanged to a new currency type.	txtOther	Number	Times New Roman	#181717	14pt	#ffffff
14	Text Box: Result (Output field)  This is to allow the program to generate the exchanged/converted amount to display to the user (once the calculation process is complete).	txtOuput	Number	Times New Roman	#181717	14pt	#ffffff
15	Combo-box: Select Currency From  This is a drop-down list of currency types (Euro, GBP, USD, AUD, Other) for the user to select to specify which currency they wish to convert from.	cmbFrom	String	Times New Roman	#181717	14pt	#ffffff
16	Combo-box: Select Currency To This will drop down list to select a  This is a drop-down list of currency types (Euro, GBP, USD, AUD, Other) for the user to select to specify which currency they wish to convert to.	cmbTo	String	Times New Roman	#181717	14pt	#ffffff
17	Background	N/A	N/A	N/A	N/A	N/A	#595959

This is the main background screen (dark grey).

## Components Table for 2nd Design

No.	Components Type	Components Name	Text Type	Font Type	Font Colour	Font Size	Background Colour
1	<p>Label: "Calculator"</p> <p>This tells the user what the purpose of the program is and what the program is.</p>	lblTitle	String	Harlow Solid Italic	#9c813	28pt	#ED7D31
2	<p>Label: "Amount"</p> <p>This tells the user where they need to input the amount they wish to convert.</p>	lblEnter	String	Algerian	#of18oa	14pt	N/A
3	<p>Label: "Currency"</p> <p>This tells the user that they need to select the currency type for the input amount that they wish to convert.</p>	lblFrom	String	Algerian	#of18oa	14pt	N/A
4	<p>Label: "Currency"</p> <p>This tells the user that they need to select the currency type for the output result.</p> <p>A currency must be selected in order for the program to display the correct exchanged amount in the output box.</p>	lblTo	String	Algerian	#of18oa	14pt	N/A
5	<p>Label: "Other Exchange Rate"</p> <p>This tells the user that they can enter a specific numerical value into this box text field in order to alter/update the calculation process in order to calculate a specific rate.</p>	lblOther	String	Algerian	#of18oa	14pt	N/A

6	<p>Label: "Output"</p> <p>This tells the user the amount that has been exchanged and where the resulting output amount is to be displayed.</p>	lblResult	String	Algerian	#of18oa	14pt	N/A
7	<p>Label: "Calculate"</p> <p>This tells the user to press this button in order to start the calculation process.</p>	lblCalculate	String	Calibri (Body)	#5e943a	18pt	N/A
8	<p>Label: "Clear"</p> <p>This tells the user to press the button in order to clear the text field data.</p>	lblClear	String	Calibri (Body)	#5e943a	18pt	N/A
9	<p>Button: "Calculate"</p> <p>User presses the button and it triggers the assigned response (calculation process).</p>	btnBlue	N/A	Calibri (Body)	N/A	18pt	#6bae3e
10	<p>Button: "Clear"</p> <p>User presses this button and it triggers the assigned response (removes data from text fields).</p>	lblClear	N/A	Calibri (Body)	N/A	18pt	#6bae3e
11	<p>Text Box: Amount (Input field)</p> <p>This is used to enter the original amount/number data for the calculation.</p>	txtInput	Number	Times New Roman	ffffff	18pt	#ed7d31
12	<p>Text Box: Other Exchange Rate (Alternative Currency Rate field)</p> <p>This is used to enter another currency format for the calculation and can be used for either the currency to be exchanged from a certain type or the</p>	txtOther	Number	Times New Roman	ffffff	18pt	#ed7d31

	currency to be exchanged to a new currency type.						
13	Text Box: Output field  This is to allow the program to generate the exchanged/converted amount to display to the user (once the calculation process is complete).	txtOuput	Number	Times New Roman	#ffffff	18pt	#ed7d31
14	Combo-box: Currency  This is a drop-down list of currency types (Euro, GBP, USD, AUD, Other) for the user to select to specify which currency they wish to convert from.	cmbFrom	String	Times New Roman	#ffffff	18pt	#ed7d31
15	Combo-box: Currency This will drop down list to select a  This is a drop-down list of currency types (Euro, GBP, USD, AUD, Other) for the user to select to specify which currency they wish to convert to.	cmbTo	String	Times New Roman	#ffffff	18pt	#ed7d31
16	Foreground  This is the main background screen (green).	N/A	N/A	N/A	N/A	N/A	#70AD47
16	Background  This is the outside background screen (black).	N/A	N/A	N/A	N/A	N/A	#333333

## Navigation for 1<sup>st</sup> Design

First you must enter the amount you wish to exchange, into the “Enter Amount” text field. This data must be written in numerical format (e.g. 50.00). You must then select the “Select Currency From” selection field and select the type of currency that you wish to exchange from (e.g. GBP). Then you must select the type of currency you wish to change this amount into (e.g. AUD), this is in the “Select Currency To” selection field.

Optionally, you can enter a decimal exchange rate for the program to calculate with instead of the options provided in the currency selection fields. This can be used for a specific currency rate or for updated currency rates. Simply select the “other” option in the “Select Currency To” field and enter the rate into the “other exchange rate” field. Select the “Calculate” button

to calculate the result. The result will then be displayed in the “Result” field. To clear all/any data entered into this program, you can select the “Clear” button, which will reset the program to allow the user to use the program again.

## Navigation for 2<sup>nd</sup> Design

---

First you must enter the amount you wish to exchange, into the “Amount” text field. This data must be written in numerical format (e.g. 50.00). You must then select the “currency” selection field and select the type of currency that you wish to exchange from (e.g. GBP). Then you must select the type of currency you wish to change this amount into (e.g. AUD).

Optionally, you can enter a decimal exchange rate for the program to calculate with instead of the options provided in the currency selection fields. This can be used for a specific currency rate or for updated currency rates. Simply select the “other” option in the currency to be exchanged field and enter the rate into the other exchange rate section. Select the calculate button to calculate the result. The result will then be displayed in the output field.

## Implemented Requirements for 1<sup>st</sup> Design

---

### Input

Requirement to input amount was implemented using a text field with a label to tell the user what to do.

Requirement to input currency type from was implemented using a text field with a label to tell the user what to do

Requirement to input currency type to was implemented using a text field with a label to tell the user what to do.

Requirement to instruct program to calculate the exchange rate was implemented by a button with a label to tell the user what to do.

Requirement to input alternative exchange rate was implemented using a text field with a label to tell the user what to do.

### Process

Requirement to process the exchange rate calculation was implemented using a calculation button with a label to tell the program to calculate and process the data entered by the user.

### Output

Requirement to output the result of the processed calculation in the specified answer output field was implemented using a text field with a label to tell the user what to do.

Requirement to output/display the error message to the user was implemented using a message pop-up box with instructions/error message upon clicking the calculate button.

Requirement to output the clear/reset action for all the data field boxes in the program was implemented using a button with a “clear” label to tell the user what to do.

## Implemented Requirements for 2<sup>nd</sup> Design

---

### Input

Requirement to input amount was implemented using a text field with a label to tell the user what to do.

Requirement to input currency type from was implemented using a text field with a label to tell the user what to do

Requirement to input currency type to was implemented using a text field with a label to tell the user what to do.

Requirement to instruct program to calculate the exchange rate was implemented by a button with a label to tell the user what to do.

Requirement to input alternative exchange rate was implemented using a text field with a label to tell the user what to do.

## Process

Requirement to process the exchange rate calculation was implemented using a calculation button with a label to tell the program to calculate and process the data entered by the user.

## Output

Requirement to output the result of the processed calculation in the specified answer output field was implemented using a text field with a label to tell the user what to do.

Requirement to output/display the error message to the user was implemented using a message pop-up box with instructions/error message upon clicking the calculate button.

---

## Data Validation for 1<sup>st</sup> Design

In the enter amount text field, only number data can be entered, and the calculator will produce a pop-up error message box to the user if the data is written in any other format.

In the other exchange rate text field, only number data can be entered, and the calculator will produce a pop-up error message box to the user if the data is written in any other format.

In the select currency from selection box, a currency must be selected from this selection list, if no currency type is selected then the calculator will produce a pop-up error message box to the user if the data has not been selected.

In the select currency to selection box, a currency must be selected from this selection list, if no currency type is selected then the calculator will produce a pop-up error message box to the user if the data has not been selected.

---

## Data Validation for 2<sup>nd</sup> Design

In the amount text field, only number data can be entered, and the calculator will produce a pop-up error message box to the user if the data is written in any other format.

In the other exchange rate text field, only number data can be entered, and the calculator will produce a pop-up error message box to the user if the data is written in any other format.

In the select currency selection box, a currency must be selected from this selection list, if no currency type is selected then the calculator will produce a pop-up error message box to the user if the data has not been selected.

In the select currency selection box, a currency must be selected from this selection list, if no currency type is selected then the calculator will produce a pop-up error message box to the user if the data has not been selected.

---

# Data Dictionary

## Introduction

A data dictionary contains a list of variables that will be used to store the data that the user will input for the program to calculate. This data dictionary shows what each variable is and what the size of the variables are. This dictionary contains all the data types used within the program I plan to create.

## What is a variable?

Data is stored and referenced using a variable (which can be thought of as a container). A variable is used to store information, which, can be used as a reference to allow the program to carry out certain functions and commands. Variables can be changed and altered depending on the set conditions or information stored within a program.

## Size of a variable

A data type dictates how large or small the variable will be. We use different sized variables for storing different types of data, such as a field having a minimum or maximum character entered in a text field.

Boolean – This data type is for true or false statements and holds 1 byte.

Decimal – This data type is used for storing money and holds 8 bytes.

String – This data type is used for storing letters (depending on the number of characters) and holds 2 bytes per letter.

Integer – This data type is used for storing whole numbers and holds 2 bytes.

Variable	Data Type	Use	Size
<b>enterAmount</b>	Decimal	Storing amount to convert/exchange	8 bytes
<b>otherExchange</b>	Decimal	Storing alternative exchange rate to exchange from or into.	8 bytes
<b>outputResult</b>	Decimal	Storing the result/answer.	8 bytes
<b>currencyFrom</b>	String	Storing the letters/characters/text for the currency (AUD, Euro, GBP), for the entered amount.	2 bytes per letter
<b>currencyTo</b>	String	Storing the letters/characters/text for the currency (AUD, Euro, GBP) for the program to convert the entered amount into.	2 bytes per letter
<b>messageError</b>	String	Pop up display text that states when the user has made an error, used for storing letters, characters, text.	2 bytes per letter

# Processes

---

## Store User Data

- Amount to be exchanged
- Type of currency to exchange from
- Type of currency to exchange to
- (Optional) Other/alternative exchange rate

This process will store input data from the user into the program. It will store the amount to be exchanged, the type of currency to exchange from, the type of currency to exchange to and an optional process of other/alternative exchange rate to be exchanged from or to.



## Update Exchange Rate (optional)

- (Optional) Other/alternative exchange rate
- Type of currency to exchange from (other)
- Type of currency to exchange to (other)

This process is only to be used when the user needs an alternative or updated exchange rate to the selection offered to the user under the type of currency to be exchanged from and to selection fields. The user can enter this data (an updated exchange rate or an alternative exchange rate), in the “Other Exchange Rate” text field and this data will then be stored and used within the program’s calculation process. This will over-ride any existing calculation settings stored within the program for the stored currency values. The program will then use this entered rate to calculate the new exchange rate.

## Find the Current Exchange Rate

- (Optional) Other/alternative exchange rate
- Type of currency to exchange from (other)
- Type of currency to exchange to (other)

This process is only to be used when the user needs an alternative or updated exchange rate to the selection offered to the user under the type of currency to be exchanged from and to selection fields. The user can enter this data (an updated exchange rate or an alternative exchange rate), in the “Other Exchange Rate” text field and this data will then be stored and used within the program’s calculation process. This will over-ride any existing calculation settings stored within the program for the stored currency values. The program will then use this entered rate to calculate the new exchange rate.

## Calculate Exchanged Amount

This exchanged amount will be calculated using the data entered into the program by the user. Once this data has been sent for processing (when the user presses the calculate button), the program will calculate the amount to be exchanged multiplied by the currency rate to be exchanged into.

The program will use the amount to be exchanged, the currency type to be exchanged from and the currency type to be exchanged to calculate the end result.

## Display Result

This process will display the result to the user.

## Reset/Clear Data

The clear button will clear all the data fields containing data and will reset the programming ready for use.

# Flow Chart

---

## What is an algorithm?

An algorithm is a list of step-by-step instructions that, when followed, will solve a problem. The two main techniques for producing an algorithm are pseudo-code and flowcharts.

## What is a flowchart?

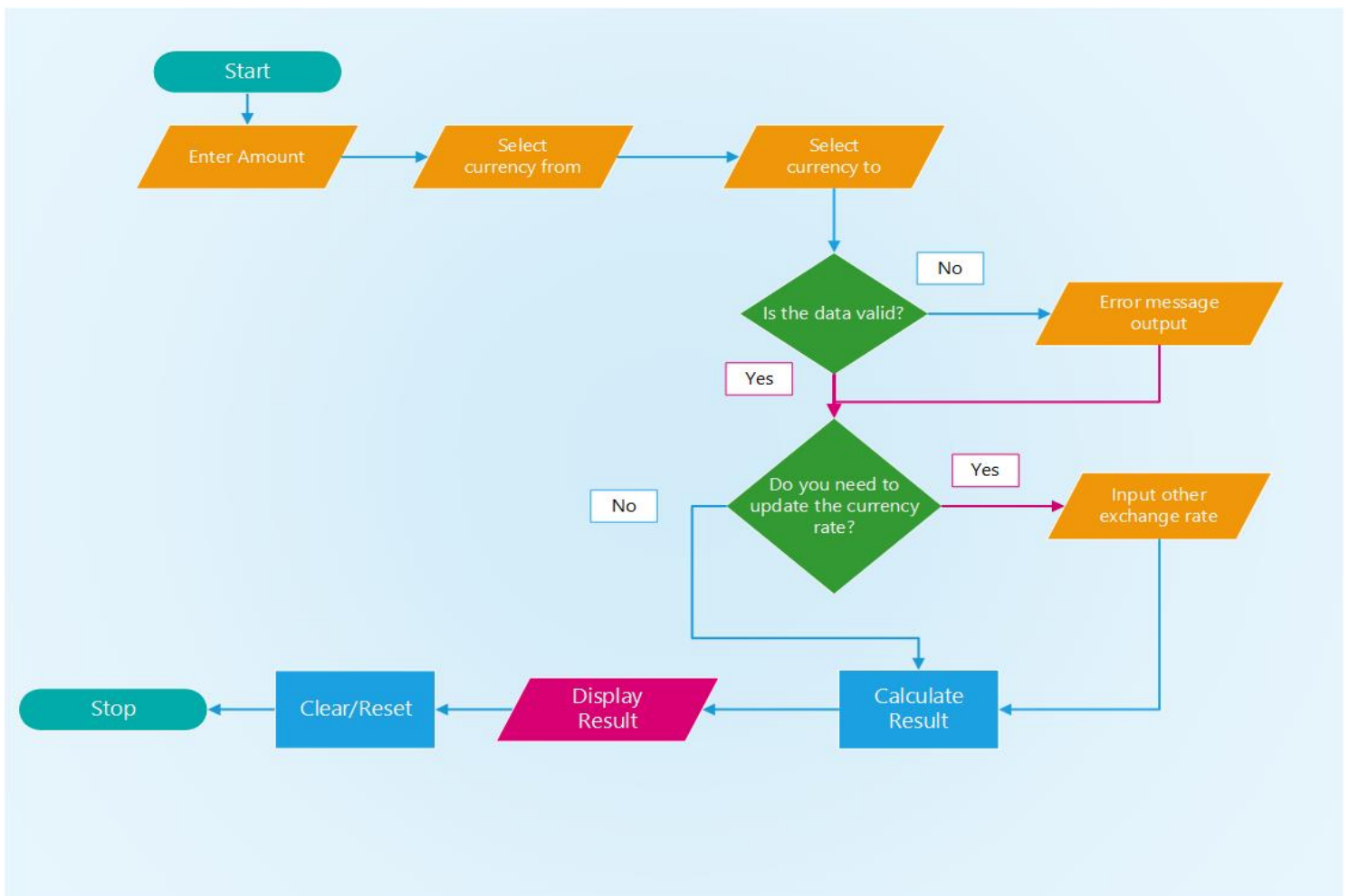
A flowchart is a diagram that shows the breakdown of a task or a system into steps. A flowchart is made-up of a series of symbols with connecting arrows to show the steps in an algorithm to achieve an end goal.

## What does a flowchart show?

A flowchart shows a step-by-step diagram of the tasks taken to achieve an end goal.

## Why is a flowchart important in programming?

A flowchart is important and is often used to show prospective clients/employers how a program is going to operate. A flowchart is easy to understand and allows the client/employer to understand and make alterations to the program just by looking at the flowchart.



## Test Plan

A test plan is a table that lists all the tests that the program needs to pass in order to be ready to use. A test plan will record a list of tests to be carried out, the expectations of the developer when each test is carried out and it can also record the actual data from the tests.

Test No.	Test Description	Test Data	Expected Result	Actual Result	Screenshots
1	Program loads – Open the program and the program should load without issue.	None	The program should load without issue.		
2	Check that you can enter decimal data into “Enter Amount” text field.	Decimal Data/Numbers	The text field should allow the user to enter numbers into the “Enter Amount” field.		
3	Check that the user can select a “Currency From” in the combo box.	String data (eg GBP)	That the user can select any currency from the combo box.		
4	Check that the user can select a “Currency To” in the combo box.	String data (eg Euro)	That the user can select any currency from the combo box.		
5	Check that exchange rate works.	GBP to Euro Amount £10 $10 * 1.143 = 11.43$	That €11.43 is the answer/result.		
6	Check that exchange rate works.	Euro to GBP Amount €10 $10 * 0.875 = 8.75$	That £8.75 is the answer/result.		
7	Check that exchange rate works.	GBP to AUD Amount £10 $10 * 1.835 = 18.35$	That \$18.35 is the answer/result.		
8	Check that exchange rate works.	AUD to GBP Amount \$10 $10 * 0.545 = 5.45$	That £5.45 is the answer/result.		
9	Check that exchange rate works.	GBP to USD Amount £10 $10 * 1.398 = 13.98$	That \$13.98 is the answer/result.		
10	Check that exchange rate works.	USD to GBP Amount \$10 $10 * 0.715 = 7.15$	That £7.15 is the answer/result.		
11	Check invalid user input.	Invalid data	If a letter is typed into a text field that only allows decimal/number data, the program will display an error message.		
12	Updated/other exchange rate.	GBP to Other (1.52). Amount is £10 * 1.52 = 15.20	The calculation should store the updated/other exchange rate into the program to perform the calculation, the result of this example is 15.20.		

# Justifications

---

The two designs that have been produced are polar opposites and couldn't be more different. On the design that has been chosen, the colours have a fairly urban theme, and consist of a dark-grey background with white font, white text boxes with accents of a playful green shade on the title and calculator button.

The use of white as the font colour allows the text to stand out clearly against the dark-grey and green background colours, this allows the user to clearly read the text due to the good contrast between the colours. The selected font styles and sizes also reinforce this key requirement in the design, as it increases the programs accessibility and overall usability for the user.

The level of detail in the chosen design allows the client to envision the end product, and labels have been used within the design to clearly demonstrate the features of the program to the client. The client has requested that the program is able to exchange currencies from GBP to at least three other foreign currencies and must be able to alter the exchange rate to updated rates, this design has been created to fulfil these needs and contains additional features such as a clear button and an error message box.

The design that has been selected for production has a horizontal layout, which has been created to aid usability as the program's layout follows the natural movement of the human eye (left to right) with minimal effort.

On the alternative design, the colours clash, neon orange and a sickly shade of green jump out at the user and make it difficult for the user to look at. The title's font colour blends in with the vibrant orange background, so the user cannot read the title. The choice of font style, colour and size all make it extremely difficult for the average user to read (or even see), which does not bode well for anyone with visual impairments.

The layout of the alternative design is vertical and feels uncomfortable to look at, this is due to the way people read (from left to right, top to bottom) which means that vertical layout seems unnatural and inconvenient to the human eye. Therefore, once a user opens this program and looks at how illogical the layout of the program is, it is very likely that they will not like the design from a visual perspective and opt for another program.

Another major flaw with the second design is the lack of an error message pop-up box, the program must contain this as part of the client's requirements and if this aspect is not fulfilled the client may choose another company to develop the program (as their needs will have not been met).

Overall, the selected grey, white and green design is clearly the superior choice and does indeed fit with the requirements stated by the client during the briefing, and with the support and approval of the client, we plan to see the design through to production.

# Task 3

---

# Original Code

---

```
public partial class Form1 : Form
{
    //A variable is a container which stores data.

    //Here I have declared a variable called currencyFrom and set it as a string datatype with
    speech marks.
    //This variable will hold and store the data from the currencyFrom combo-box for the program
    to retrieve later on.
    string currencyFrom = "";

    //Here I have declared a variable called currencyTo and set it as a string datatype with
    speech marks.
    //This variable will hold and store the data from the currencyTo combo-box for the program
    to retrieve later on.
    string currencyTo = "";

    //Here I have declared a variable called amount and set it as a decimal datatype and set it
    to 0.
    //This variable will hold and store the data from the amount textbox for the program to
    retrieve later on.
    decimal amount = 0;

    //Here I have declared a variable called exchangeRate and set it as a decimal datatype and
    set it to 0.
    //This variable will hold and store the data from the other exchange rate textbox for the
    program to retrieve later on.
    decimal exchangeRate = 0;

    //Here I have declared a variable called result and set it as a decimal datatype and set it
    to 0.
    //This variable will output the calculation result to the result textbox for the program to
    display to the user.
    decimal result = 0;

    // An array is a data structure, which can store a fixed-size collection of elements of the
    same data type.
    //An array is used to store a collection of data.
    //Here I have declared the array for my currency rates.
    //To allow the program to retrieve the data assigned to the currency to and from.
    //We use an array to store a collection of data so that it can be accessed or retrieved by
    the program.
    private string[,] rateArray =
    {
        { "GBP", "USD", "1.34" },
        { "USD", "GBP", "0.75" },
        { "GBP", "EURO", "1.14" },
        { "EURO", "GBP", "0.88" },
        { "GBP", "AUD", "1.75" },
        { "AUD", "GBP", "0.57" }};

    public Form1()
    {
        InitializeComponent();
    }

    private void label6_Click(object sender, EventArgs e)
    {
    }
}
```

```

private void btnCalc_Click(object sender, EventArgs e)
{
    //This is the code to trigger an action once the calculate button is clicked.
    //This will trigger the method called storeData to run.
    //The button is used to trigger an action in order to get the program to do something.
    //In this case the method storeData is used to store the data that has been entered by
the user.
    //This will then trigger the chkRate method calculation process of the program.
    storeData();
}

public void storeData()
{
    //This gets the data from the combo-box for the currency from.
    //And sets /stores it in the variable for the currency from.
    currencyFrom = cmbFrom.Text;

    //This gets the data from the combo-box for the currency To.
    //And sets /stores it in the variable for the currency To.
    currencyTo = cmbTo.Text;

    //This gets the amount entered in by the user and stores it into the variable amount.
    amount = Convert.ToDecimal(txtEnter.Text);

    //This will trigger the method chkRate to run.
    chkRate();
}

public void chkRate()
{
    //This is an if statement which tells the program that if the string (txtExchangeRate),
    //is null or empty, then the following conditions will be carried out.
    //A string is null if it has not been assigned a value.
    //A string is empty if it is explicitly assigned an empty string ("") or String.Empty.
    if (!string.IsNullOrEmpty(txtExchangeRate.Text))
    {
        //This converts the value of the data into a decimal datatype.
        exchangeRate = Convert.ToDecimal(txtExchangeRate.Text);

        //This triggers the updateRate method to run.
        updateRate();
    }
    //This is an else statement which tells the program what to do if the if statement
criteria has not been met.
    else
    {
        //This triggers the findExchangeRate method to run.
        findExchangeRate();
    }
}

public void updateRate()
{
    //Here I have declared a variable called count within the updateRate method.
    //The variable has a datatype of int (meaning integer) with an initial value of 0.
    int count = 0;

    while (count < rateArray.GetUpperBound(0))
    {
        if (currencyFrom.Equals(rateArray[count, 0]) && (currencyTo.Equals(rateArray[count,
1])))
        {
            //Stores the exchange rate variable into the array.
            rateArray[count, 0] = exchangeRate.ToString();
            break;
        }
    }
}

```

```

    }
    //Increment count value by 1.
    count =(count + 1);

    //Triggers the calculate method to run.
    calculate();
}
}

public void findExchangeRate()
{
    //Here I have declared a variable called count within the findExchangeRate method.
    //The variable has a datatype of int (meaning integer) with an initial value of 0.
    int count = 0;

    while (count < rateArray.GetUpperBound(0))
    {
        if (currencyFrom.Equals(rateArray[count, 0]) && (currencyTo.Equals(rateArray[count,
1])))
        {
            exchangeRate = Convert.ToDecimal(0);

            //Converts one datatype into another datatype.
            exchangeRate = Convert.ToDecimal(txtExchangeRate.Text);

            Convert.ToDecimal(0);
            break;
        }
        count = (count + 1);
    }
}

public void calculate()
{
    //Multiply the amount by the exchangeRate and display the answer in the result variable.
    result = amount * exchangeRate;

    //Trigger the calculate method to run.
    display();
}

public void display()
{
    txtResult.Text = result.ToString();
}

private void btnClear_Click(object sender, EventArgs e)
{
}

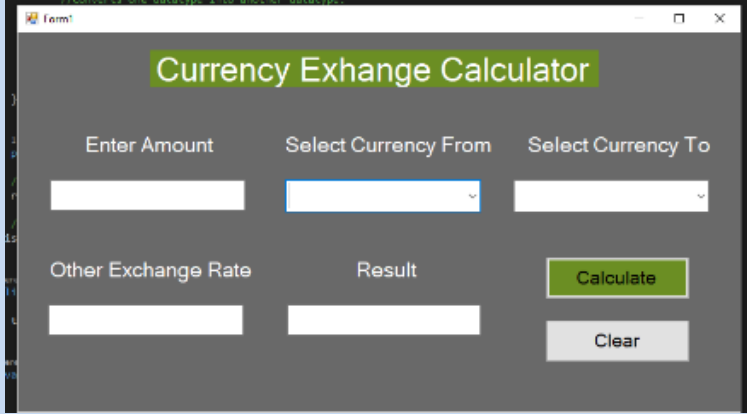
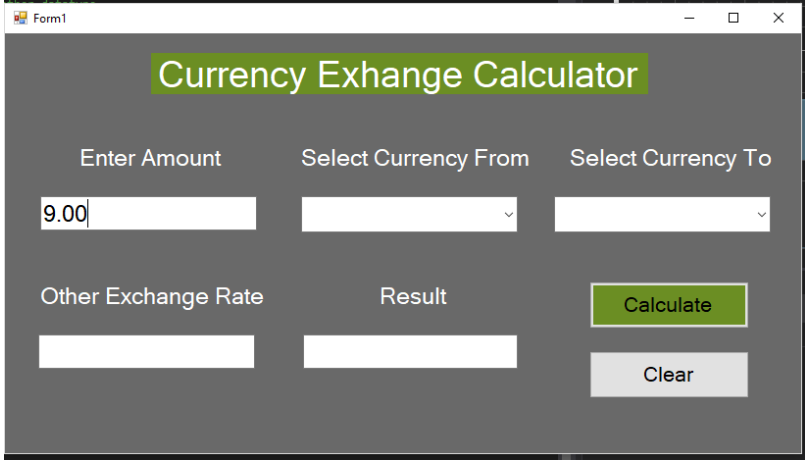
private void cmbFrom_SelectedIndexChanged(object sender, EventArgs e)
{
}

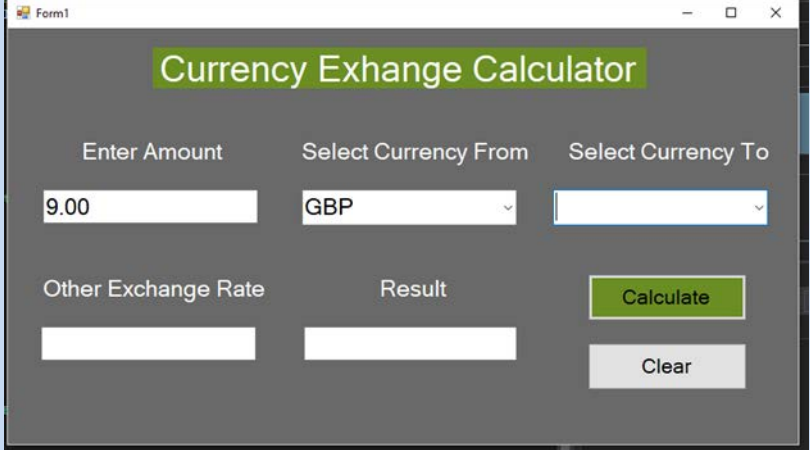
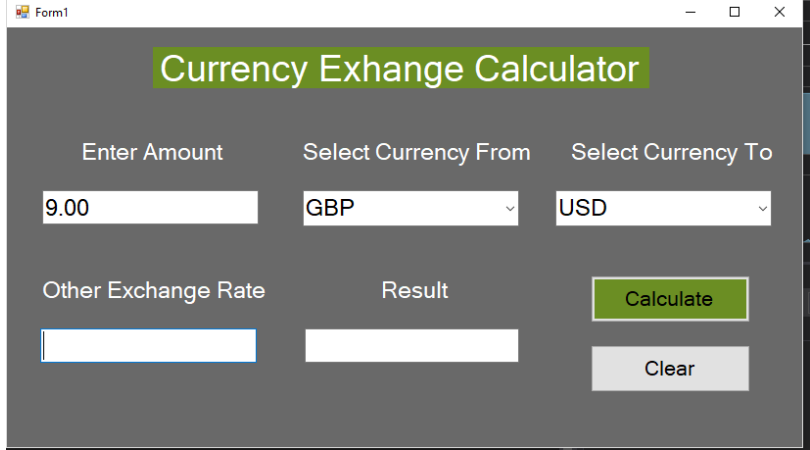
private void cmbTo_SelectedIndexChanged(object sender, EventArgs e)
{
}
}

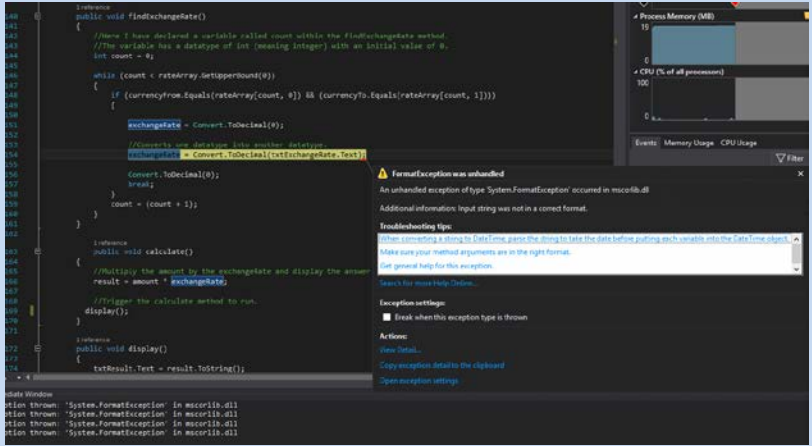
```



# Test Log

Test No.	Test Description	Test Data	Expected Result	Actual Result	Screenshots
1	Program loads – Open the program and the program should load without issue.	None	The program should load without issue.	The program loaded without any issues.	
2	Check that you can enter decimal data into "Enter Amount" text field.	Decimal Data/Numbers	The text field should allow the user to enter numbers into the "Enter Amount" field.	The "Enter Amount" text field allowed me to enter in the decimal data.	

<p><b>3</b></p>	<p>Check that the user can select a "Currency From" in the combo box.</p>	<p>String data (e.g. GBP)</p>	<p>That the user can select any currency from the combo box.</p>	<p>The combo selection box for currency from, allowed me to select from a drop down list of four possible currencies (AUD, EURO, GBP, USD), as it is supposed to.</p>	
<p><b>4</b></p>	<p>Check that the user can select a "Currency To" in the combo box.</p>	<p>String data (eg Euro)</p>	<p>That the user can select any currency from the combo box.</p>	<p>The combo selection box for currency to, allowed me to select from a drop down list of four possible currencies (AUD, EURO, GBP, USD), as it is supposed to.</p>	

5	Check that exchange rate works.	GBP to Euro Amount £10 $10 * 1.14 = 11.40$	That €11.40 is the answer/result.	The program is crashing when I press the calculate button.	
5.1	Check that exchange rate works.	GBP to Euro Amount £10 $10 * 1.14 = 11.40$	That €11.40 is the answer/result.	Before: The program is crashing when I press the calculate button.	<p>Before</p> <pre> public void findExchangeRate() {     //Here I have declared a variable called     //count within the findExchangeRate method.     //The variable has a datatype of int (meaning     //integer) with an initial value of 0.     int count = 0;      while (count &lt; rateArray.GetUpperBound(0))     {         if (currencyFrom.Equals(rateArray[count, 0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1])))         {             exchangeRate = Convert.ToDecimal(0);              //Converts one datatype into another             //datatype.             exchangeRate = Convert.ToDecimal(txtExchangeRate.Text);              Convert.ToDecimal(0);             break;         }         count = (count + 1);     } } </pre>

				<p>After: Here is the code after I altered it. I changed the (txtExchangeRate.Text) to (rateArray[count, 2]), this stopped the program from crashing but it didn't fix the problem as the result is still not displaying.</p>	<p>After</p> <pre> public void findExchangeRate() {     //Here I have declared a variable called     count within the findExchangeRate method.     //The variable has a datatype of int (meaning     integer) with an initial value of 0.     int count = 0;      while (count &lt; rateArray.GetUpperBound(0))     {         if (currencyFrom.Equals(rateArray[count,         0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1])))         {             //Converts one datatype into another             datatype.             exchangeRate =             Convert.ToDecimal(rateArray[count, 2]);              break;         }         count = (count + 1);     } } </pre>
5.2	Check that exchange rate works.	GBP to Euro Amount £10 10* 1.14 = 11.40	That €11.40 is the answer/result.	<p>Before: Here is the code after I altered it. I changed the (txtExchangeRate.Text) to (rateArray[count, 2]), this stopped the program from crashing but it didn't fix the problem as the result is still not displaying.</p>	<p>Before</p> <pre> public void findExchangeRate() {     //Here I have declared a variable called     count within the findExchangeRate method.     //The variable has a datatype of int (meaning     integer) with an initial value of 0.     int count = 0;      while (count &lt; rateArray.GetUpperBound(0))     {         if (currencyFrom.Equals(rateArray[count,         0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1])))         {             //Retrieves the value from the array             using count.             //It then converts the retrieved             value datatype into another datatype (decimal).             //It then stores it in the variable             for the exchange rate. </pre>

After:  
Here is the code after I altered it. I declared the calculate variable at the bottom of the code which runs the calculation method and then also runs the display method which fixes the issue of the program not displaying the answer. The program is adding extra zeros to the answer but the answer is still correct.

```
        exchangeRate =  
Convert.ToDecimal(rateArray[count, 2]);  
  
        break;  
    }  
    count = (count + 1);  
}  
}  
  
After  
  
public void findExchangeRate()  
{  
    //Here I have declared a variable called  
count within the findExchangeRate method.  
    //The variable has a datatype of int (meaning  
integer) with an initial value of 0.  
    int count = 0;  
  
    while (count < rateArray.GetUpperBound(0))  
    {  
        if (currencyFrom.Equals(rateArray[count,  
0]) && (currencyTo.Equals(rateArray[count, 1])))  
        {  
            //Retrieves the value from the array  
using count.  
            //It then converts the retrieved  
value datatype into another datatype (decimal).  
            //It then stores it in the variable  
for the exchange rate.  
            exchangeRate =  
Convert.ToDecimal(rateArray[count, 2]);  
  
            break;  
        }  
        count = (count + 1);  
    }  
    calculate();  
}
```

--	--	--	--	--	--

6	Check that exchange rate works.	Euro to GBP Amount €10 10* 0.88 = 8.88	That £8.88 is the answer/result.	The program is crashing when I press the calculate button.	
---	---------------------------------	--	----------------------------------	--	--

6.1	Check that exchange rate works.	Euro to GBP Amount €10 10* 0.88 = 8.88	That £8.88 is the answer/result.	Before: The program is crashing when I press the calculate button.	Before <pre> public void findExchangeRate() {     //Here I have declared a variable called     count within the findExchangeRate method.     //The variable has a datatype of int (meaning     integer) with an initial value of 0.     int count = 0;      while (count &lt; rateArray.GetUpperBound(0))     { </pre>
-----	---------------------------------	--	----------------------------------	---	---

After:  
Here is the code after I altered it. I declared the calculate variable at the bottom of the code which runs the calculation method and then also runs the display method which fixes the issue of the program not displaying the answer. The program is adding extra zeros to the answer but the answer is still correct.

```
        if (currencyFrom.Equals(rateArray[count, 0]) && (currencyTo.Equals(rateArray[count, 1])))
        {
            //Retrieves the value from the array
            using count.
            //It then converts the retrieved
            value datatype into another datatype (decimal).
            //It then stores it in the variable
            for the exchange rate.
            exchangeRate =
            Convert.ToDecimal(rateArray[count, 2]);

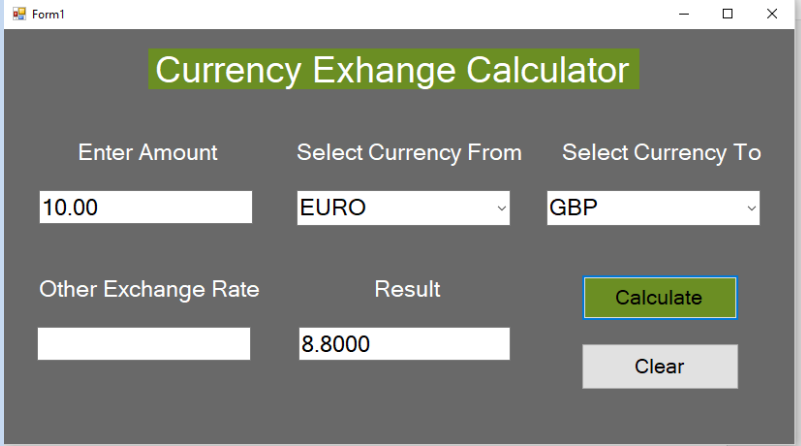
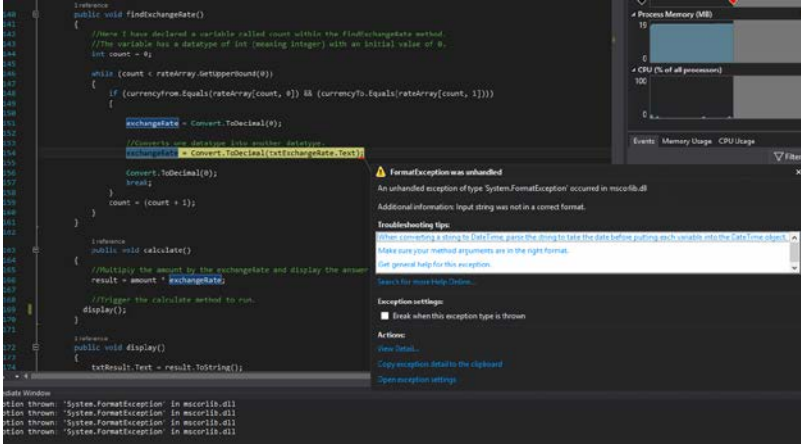
            break;
        }
        count = (count + 1);
    }
}
```

After

```
public void findExchangeRate()
{
    //Here I have declared a variable called
    count within the findExchangeRate method.
    //The variable has a datatype of int (meaning
    integer) with an initial value of 0.
    int count = 0;

    while (count < rateArray.GetUpperBound(0))
    {
        if (currencyFrom.Equals(rateArray[count, 0]) && (currencyTo.Equals(rateArray[count, 1])))
        {
            //Retrieves the value from the array
            using count.
            //It then converts the retrieved
            value datatype into another datatype (decimal).
            //It then stores it in the variable
            for the exchange rate.
            exchangeRate =
            Convert.ToDecimal(rateArray[count, 2]);

            break;
        }
    }
}
```

					<pre> count = (count + 1); } calculate(); } </pre> 
7	Check that exchange rate works.	GBP to AUD Amount £10 10* 1.75 = 17.50	That \$17.50 is the answer/result.	The program is crashing when I press the calculate button.	
7.1	Check that exchange rate works.	GBP to AUD Amount £10 10* 1.75 = 17.50	That \$17.50 is the answer/result.	Before: The program is crashing when I press the calculate button.	Before <pre> public void findExchangeRate() {     //Here I have declared a variable called count within the findExchangeRate method </pre>



After:  
Here is the code after I altered it. I declared the calculate variable at the bottom of the code which runs the calculation method and then also runs the display method which fixes the issue of the program not displaying the answer. The program is adding extra zeros to the answer but the answer is still correct.

```
//The variable has a datatype of int (meaning integer) with an initial value of 0.  
int count = 0;  
  
while (count < rateArray.GetUpperBound(0))  
{  
    if (currencyFrom.Equals(rateArray[count, 0]) && (currencyTo.Equals(rateArray[count, 1])))  
    {  
        //Retrieves the value from the array using count.  
        //It then converts the retrieved value datatype into another datatype (decimal).  
        //It then stores it in the variable for the exchange rate.  
        exchangeRate =  
Convert.ToDecimal(rateArray[count, 2]);  
  
        break;  
    }  
    count = (count + 1);  
}  
}
```

After

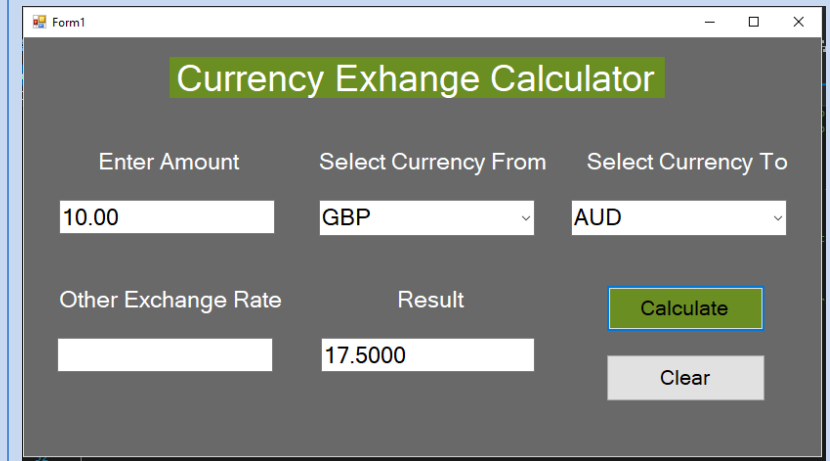
```
public void findExchangeRate()  
{  
    //Here I have declared a variable called count within the findExchangeRate method.  
    //The variable has a datatype of int (meaning integer) with an initial value of 0.  
    int count = 0;  
  
    while (count < rateArray.GetUpperBound(0))  
    {  
        if (currencyFrom.Equals(rateArray[count, 0]) && (currencyTo.Equals(rateArray[count, 1])))  
        {  
            //Retrieves the value from the array using count.  
            //It then converts the retrieved value datatype into another datatype (decimal).  
            exchangeRate =  
Convert.ToDecimal(rateArray[count, 2]);  
  
            break;  
        }  
        count = (count + 1);  
    }  
}
```

```

//It then stores it in the variable
for the exchange rate.
        exchangeRate =
Convert.ToDecimal(rateArray[count, 2]);

        break;
    }
    count = (count + 1);
}
calculate();
}

```



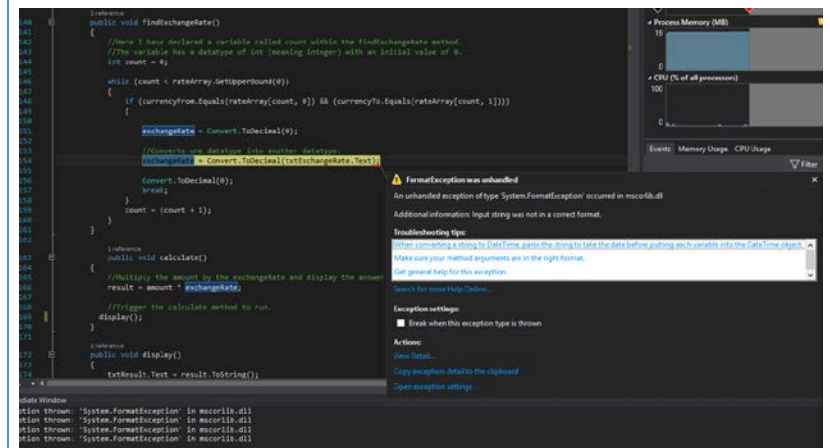
8

Check that exchange rate works.

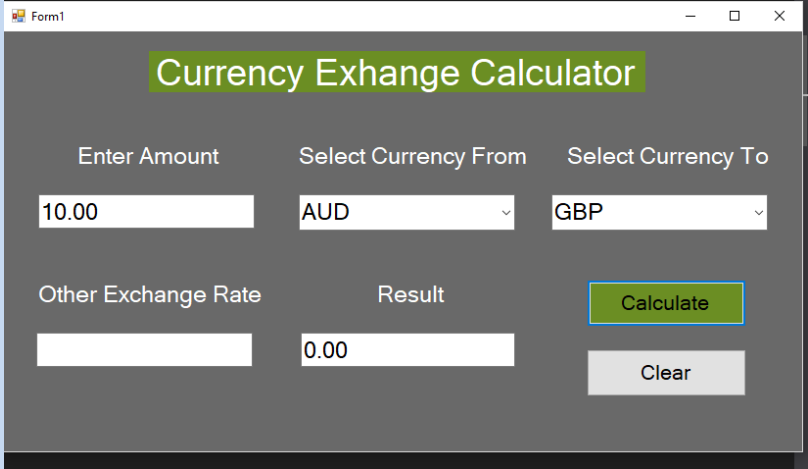
AUD to GBP  
Amount \$10  
 $10 * 0.57 = 5.70$

That £5.70 is the answer/result.

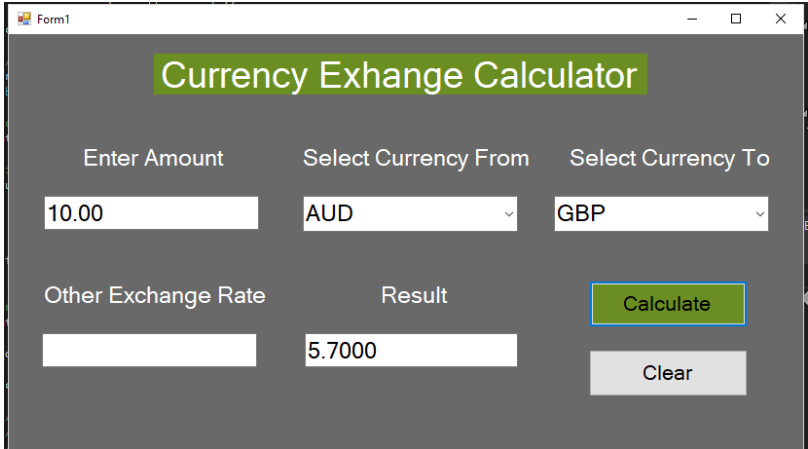
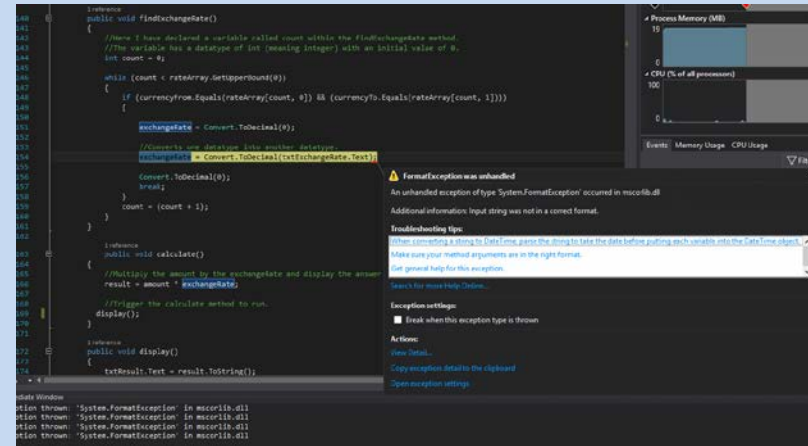
The program is crashing when I press the calculate button.



<p><b>8.1</b></p>	<p>Check that exchange rate works.</p>	<p>AUD to GBP Amount \$10 <math>10 * 0.57 = 5.70</math></p>	<p>That £5.70 is the answer/result.</p>	<p>Before: The program is crashing when I press the calculate button.</p> <p>After: Here is the code after I altered it. I declared the calculate variable at the bottom of the code which runs the calculation method and then also runs the display method. However, the answer is now displaying as 0.00 instead of £5.70.</p>	<p>Before</p> <pre> public void findExchangeRate() {     //Here I have declared a variable called     count within the findExchangeRate method.     //The variable has a datatype of int (meaning     integer) with an initial value of 0.     int count = 0;      while (count &lt; rateArray.GetUpperBound(0))     {         if (currencyFrom.Equals(rateArray[count,         0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1])))         {             //Retrieves the value from the array             using count.             //It then converts the retrieved             value datatype into another datatype (decimal).             //It then stores it in the variable             for the exchange rate.             exchangeRate =             Convert.ToDecimal(rateArray[count, 2]);              break;         }         count = (count + 1);     } } </pre> <p>After</p> <pre> public void findExchangeRate() {     //Here I have declared a variable called     count within the findExchangeRate method.     //The variable has a datatype of int (meaning     integer) with an initial value of 0.     int count = 0;      while (count &lt; rateArray.GetUpperBound(0))     {         if (currencyFrom.Equals(rateArray[count,         0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1])))         { </pre>
-------------------	--	---	---	---	--

					<pre> //Retrieves the value from the array using count. //It then converts the retrieved value datatype into another datatype (decimal). //It then stores it in the variable for the exchange rate. exchangeRate = Convert.ToDecimal(rateArray[count, 2]);  break; } count = (count + 1); } calculate(); } </pre> 
8.2	Check that exchange rate works.	AUD to GBP Amount \$10 $10 * 0.57 = 5.70$	That £5.70 is the answer/result.	<p>Before:</p> <p>I altered the code but it still would not display the correct answer for an exchange from AUD to GBP.</p>	<p>Before</p> <pre> public void findExchangeRate() { //Here I have declared a variable called count within the findExchangeRate method. //The variable has a datatype of int (meaning integer) with an initial value of 0. int count = 0;  while (count &lt; rateArray.GetUpperBound(0)) { </pre>

				<p>After:</p> <pre>        if (currencyFrom.Equals(rateArray[count, 0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1])))         {             //Retrieves the value from the array             using count.             //It then converts the retrieved             value datatype into another datatype (decimal).             //It then stores it in the variable             for the exchange rate.             exchangeRate =             Convert.ToDecimal(rateArray[count, 2]);              break;         }         count = (count + 1);     }     calculate(); }</pre> <p>After</p> <pre>public void findExchangeRate() {     //Here I have declared a variable called     count within the findExchangeRate method.     //The variable has a datatype of int (meaning     integer) with an initial value of 0.     int count = 0;      while (count &lt;= rateArray.GetUpperBound(0))     {         if (currencyFrom.Equals(rateArray[count, 0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1])))         {             //Retrieves the value from the array             using count.             //It then converts the retrieved             value datatype into another datatype (decimal).             //It then stores it in the variable             for the exchange rate.             exchangeRate =             Convert.ToDecimal(rateArray[count, 2]);              break;         }         count = (count + 1);     } }</pre>
--	--	--	--	---

					<pre> } calculate();  } </pre> 
9	Check that exchange rate works.	GBP to USD Amount £10 10* 1.34 = 13.40	That \$13.40 is the answer/result.	The program is crashing when I press the calculate button.	
9.1	Check that exchange rate works.	GBP to USD Amount £10 10* 1.34 = 13.40	That \$13.40 is the answer/result.	<p>Before: The program is crashing when I press the calculate button.</p> <p>After:</p>	<p>Before</p> <pre> public void findExchangeRate() {     //Here I have declared a variable called     count within the findExchangeRate method.     //The variable has a datatype of int (meaning     integer) with an initial value of 0.     int count = 0; </pre>

Here is the code after I altered it. I declared the calculate variable at the bottom of the code which runs the calculation method and then also runs the display method which fixes the issue of the program not displaying the answer. The program is adding extra zeros to the answer but the answer is still correct.

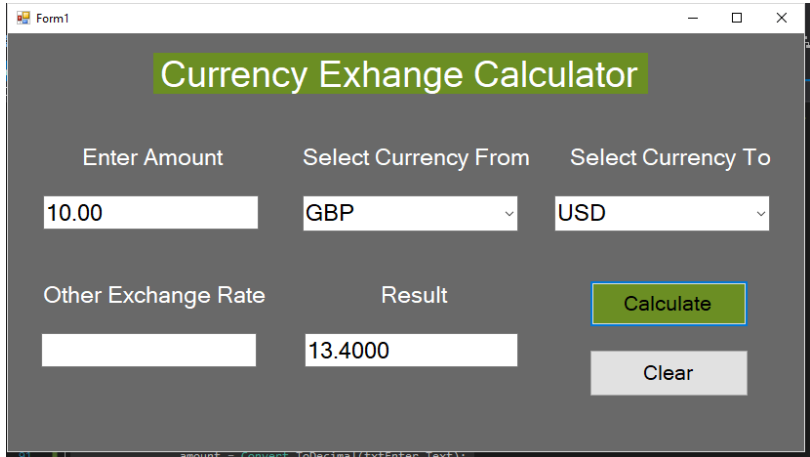
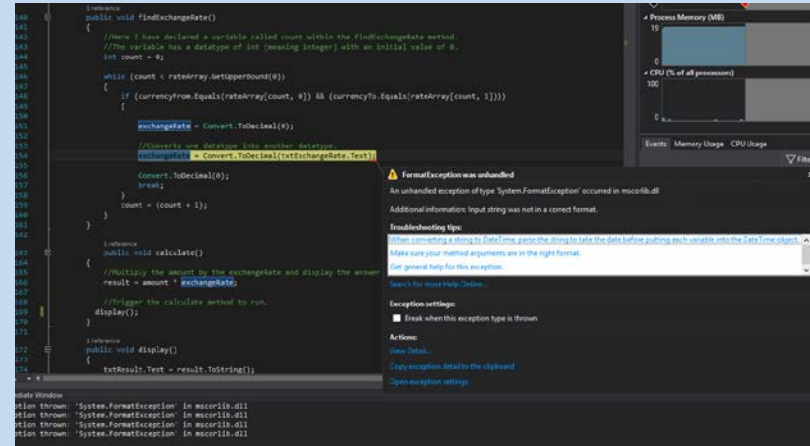
```
while (count < rateArray.GetUpperBound(0))
{
    if (currencyFrom.Equals(rateArray[count, 0]) && (currencyTo.Equals(rateArray[count, 1])))
    {
        //Retrieves the value from the array
        using count.
        //It then converts the retrieved
        value datatype into another datatype (decimal).
        //It then stores it in the variable
        for the exchange rate.
        exchangeRate =
        Convert.ToDecimal(rateArray[count, 2]);

        break;
    }
    count = (count + 1);
}
}
```

After

```
public void findExchangeRate()
{
    //Here I have declared a variable called
    count within the findExchangeRate method.
    //The variable has a datatype of int (meaning
    integer) with an initial value of 0.
    int count = 0;

    while (count < rateArray.GetUpperBound(0))
    {
        if (currencyFrom.Equals(rateArray[count, 0]) && (currencyTo.Equals(rateArray[count, 1])))
        {
            //Retrieves the value from the array
            using count.
            //It then converts the retrieved
            value datatype into another datatype (decimal).
            //It then stores it in the variable
            for the exchange rate.
            exchangeRate =
            Convert.ToDecimal(rateArray[count, 2]);
```

					<pre>                 break;             }             count = (count + 1);         }         calculate();     } </pre> 
10	Check that exchange rate works.	USD to GBP Amount \$10 $10 * 0.75 = 7.5$	That £7.50 is the answer/result.	The program is crashing when I press the calculate button.	
10.1	Check that exchange rate works.	USD to GBP Amount \$10 $10 * 0.75 = 7.5$	That £7.50 is the answer/result.	Before: The program is crashing when I press the calculate button.	Before <pre> public void findExchangeRate() {     //Here I have declared a variable called count within the findExchangeRate method     //The variable has a datatype of int (meaning integer) with an initial value of 0.     int count = 0;     while (count &lt; rateArray.GetUpperBound(0))     {         if (currencyFrom.Equals(rateArray[count, 0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1])))         {             exchangeRate = Convert.ToDecimal(0);             //convert to decimal             exchangeRate = Convert.ToDecimal(rateArray[count, 2]);             count = (count + 1);         }     }     //display the amount by the exchange rate and display the answer     result = amount * exchangeRate;     //trigger the calculate method to run.     display(); } </pre>



After:  
Here is the code after I altered it. I declared the calculate variable at the bottom of the code which runs the calculation method and then also runs the display method which fixes the issue of the program not displaying the answer. The program is adding extra zeros to the answer but the answer is still correct.

```
//The variable has a datatype of int (meaning integer) with an initial value of 0.
int count = 0;

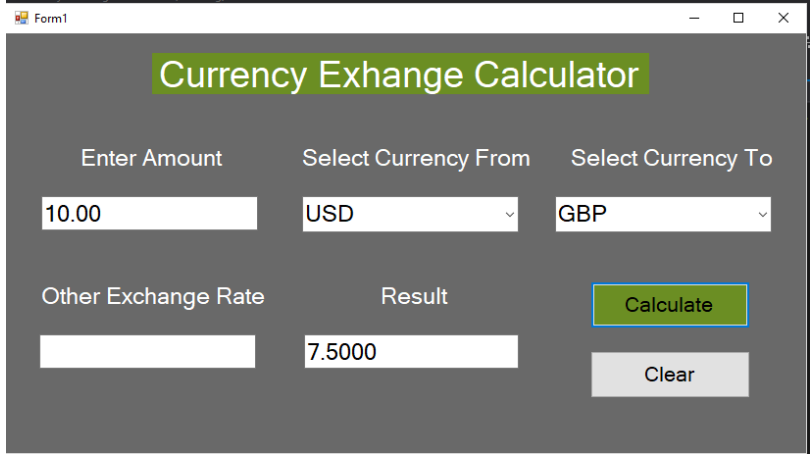
while (count < rateArray.GetUpperBound(0))
{
    if (currencyFrom.Equals(rateArray[count, 0]) && (currencyTo.Equals(rateArray[count, 1])))
    {
        //Retrieves the value from the array using count.
        //It then converts the retrieved value datatype into another datatype (decimal).
        //It then stores it in the variable for the exchange rate.
        exchangeRate =
Convert.ToDecimal(rateArray[count, 2]);

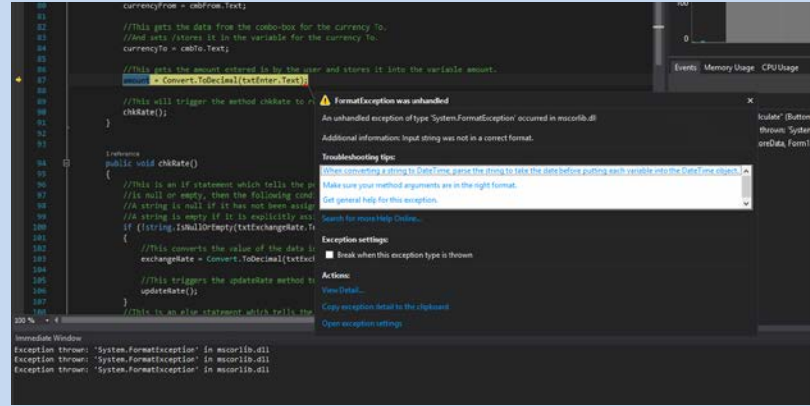
        break;
    }
    count = (count + 1);
}
}
```

After

```
public void findExchangeRate()
{
    //Here I have declared a variable called count within the findExchangeRate method.
    //The variable has a datatype of int (meaning integer) with an initial value of 0.
    int count = 0;

    while (count < rateArray.GetUpperBound(0))
    {
        if (currencyFrom.Equals(rateArray[count, 0]) && (currencyTo.Equals(rateArray[count, 1])))
        {
            //Retrieves the value from the array using count.
            //It then converts the retrieved value datatype into another datatype (decimal).
```

					<pre> //It then stores it in the variable for the exchange rate.         exchangeRate = Convert.ToDecimal(rateArray[count, 2]);          break;     }     count = (count + 1); } calculate(); } </pre> 
--	--	--	--	--	--

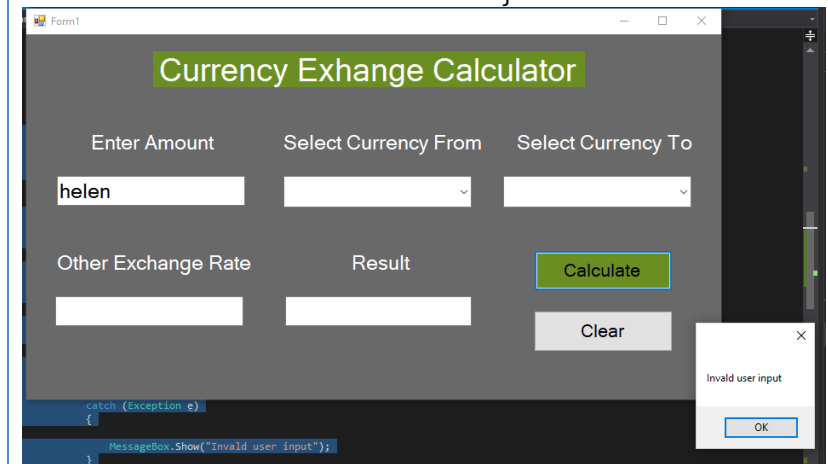
11	Check invalid user input.	Invalid data	If a letter is typed into a text field that only allows decimal/number data, the program will display an error message.	The program crashes and does not display an error message.	
----	---------------------------	--------------	---	--	--

<p><b>11.1</b></p>	<p>Check invalid user input.</p>	<p>Invalid data</p>	<p>If a letter is typed into a text field that only allows decimal/number data, the program will display an error message.</p>	<p>Before: The program crashes and does not display an error message.</p> <p>After: I altered the code to include a try/catch statement which also includes a message in a pop-up message box. The message says "Invalid user input".</p>	<p>Before</p> <pre> public void storeData() {     //This gets the data from the combo-box for the currency from.     //And sets /stores it in the variable for the currency from.     currencyFrom = cmbFrom.Text;      //This gets the data from the combo-box for the currency To.     //And sets /stores it in the variable for the currency To.     currencyTo = cmbTo.Text;      //This gets the amount entered in by the user and stores it into the variable amount.     amount = Convert.ToDecimal(txtEnter.Text);      //This will trigger the method chkRate to run.     chkRate(); } </pre> <p>After</p> <pre> public void storeData() {     try     {         //This gets the data from the combo-box for the currency from.         //And sets /stores it in the variable for the currency from.         currencyFrom = cmbFrom.Text;          //This gets the data from the combo-box for the currency To. </pre>
--------------------	----------------------------------	---------------------	--	---	---

```
//And sets /stores it in the variable for
the currency To.
currencyTo = cmbTo.Text;

//This gets the amount entered in by the
user and stores it into the variable amount.
amount =
Convert.ToDecimal(txtEnter.Text);

//This will trigger the method chkRate to
run.
chkRate();
}
catch (Exception e)
{
    MessageBox.Show("Invalid user input");
}
}
```



12

Updated/other exchange rate.

GBP to Other (1.52). Amount is £10 \* 1.52 = 15.20

The calculation should store the updated/other exchange rate into the program to perform the calculation, the result of this example is 15.20.

The exchange rate box does work, but it is adding on too many zeros which are unnecessary.

The screenshot shows a window titled 'Form1' with the following interface:

- Title:** Currency Exchange Calculator
- Enter Amount:** Text box containing '10.00'
- Select Currency From:** Dropdown menu showing 'GBP'
- Select Currency To:** Dropdown menu showing 'AUD'
- Other Exchange Rate:** Text box containing '1.52'
- Result:** Text box containing '15.2000'
- Buttons:** A green 'Calculate' button and a grey 'Clear' button.

# Updated Code

---

```
namespace Currency_Exchange_Calculator
{
    public partial class Form1 : Form
    {
        //A variable is a container which stores data.

        //Here I have declared a variable called currencyFrom and set it as a string datatype with
        //speech marks.
        //This variable will hold and store the data from the currencyFrom combo-box for the program
        //to retrieve later on.
        string currencyFrom = "";

        //Here I have declared a variable called currencyTo and set it as a string datatype with
        //speech marks.
        //This variable will hold and store the data from the currencyTo combo-box for the program
        //to retrieve later on.
        string currencyTo = "";

        //Here I have declared a variable called amount and set it as a decimal datatype and set it
        //to 0.
        //This variable will hold and store the data from the amount textbox for the program to
        //retrieve later on.
        decimal amount = 0;

        //Here I have declared a variable called exchangeRate and set it as a decimal datatype and
        //set it to 0.
        //This variable will hold and store the data from the other exchange rate textbox for the
        //program to retrieve later on.
        decimal exchangeRate = 0;

        //Here I have declared a variable called result and set it as a decimal datatype and set it
        //to 0.
        //This variable will output the calculation result to the result textbox for the program to
        //display to the user.
        decimal result = 0;

        // An array is a data structure, which can store a fixed-size collection of elements of the
        //same data type.
        //An array is used to store a collection of data.
        //Here I have declared the array for my currency rates.
        //To allow the program to retrieve the data assigned to the currency to and from.
        //We use an array to store a collection of data so that it can be accessed or retrieved by
        //the program.
        private string[,] rateArray =
        {
            { "GBP", "USD", "1.34" },
            { "USD", "GBP", "0.75" },
            { "GBP", "EURO", "1.14" },
            { "EURO", "GBP", "0.88" },
            { "GBP", "AUD", "1.75" },
            { "AUD", "GBP", "0.57" }};

        public Form1()
        {
            InitializeComponent();
        }

        private void label6_Click(object sender, EventArgs e)
        {

```

```

}

private void btnCalc_Click(object sender, EventArgs e)
{
    //This is the code to trigger an action once the calculate button is clicked.
    //This will trigger the method called storeData to run.
    //The button is used to trigger an action in order to get the program to do something.
    //In this case the method storeData is used to store the data that has been entered by
the user.
    //This will then trigger the chkRate method calculation process of the program.
    storeData();
}

public void storeData()
{
    //Try statement is used to encapsulate a region of code.
    //If any code throws an exception within that try block, the exception will be handled
by the corresponding catch statement.
    try
    {
        //This gets the data from the combo-box for the currency from.
        //And sets /stores it in the variable for the currency from.
        currencyFrom = cmbFrom.Text;

        //This gets the data from the combo-box for the currency To.
        //And sets /stores it in the variable for the currency To.
        currencyTo = cmbTo.Text;

        //This gets the amount entered in by the user and stores it into the variable
amount.
        amount = Convert.ToDecimal(txtEnter.Text);

        //This will trigger the method chkRate to run.
        chkRate();
    }
    //When an exception occurs, the Catch block of code is executed.
    //This is where the program is able to handle the exception.
    //When the incorrect type of data is input into the data boxes by the user, it will be
handled in the catch part of the statement.
    //The program will display the message box showing the message "Invalid user input".
    catch (Exception e)
    {
        MessageBox.Show("Invalid user input");
    }
}

public void chkRate()
{
    //This is an if statement which tells the program that if the string (txtExchangeRate),
    //is null or empty, then the following conditions will be carried out.
    //A string is null if it has not been assigned a value.
    //A string is empty if it is explicitly assigned an empty string ("") or String.Empty.
    if (!string.IsNullOrEmpty(txtExchangeRate.Text))
    {
        //This converts the value of the data into a decimal datatype.
        exchangeRate = Convert.ToDecimal(txtExchangeRate.Text);

        //This triggers the updateRate method to run.
        updateRate();
    }
    //This is an else statement which tells the program what to do if the if statement
criteria has not been met.
    else
    {

```

```

        //This triggers the findExchangeRate method to run.
        findExchangeRate();
    }
}

public void updateRate()
{
    //Here I have declared a variable called count within the updateRate method.
    //The variable has a datatype of int (meaning integer) with an initial value of 0.
    int count = 0;

    while (count < rateArray.GetUpperBound(0))
    {
        1])))
        if (currencyFrom.Equals(rateArray[count, 0]) && (currencyTo.Equals(rateArray[count,
            {
                //Stores the exchange rate variable into the array.
                rateArray[count, 2]= exchangeRate.ToString();
                break;
            }
            //Increment count value by 1.
            count =(count + 1);

            //Triggers the calculate method to run.
            calculate();
        }
    }

    public void findExchangeRate()
    {
        //Here I have declared a variable called count within the findExchangeRate method.
        //The variable has a datatype of int (meaning integer) with an initial value of 0.
        int count = 0;

        while (count <= rateArray.GetUpperBound(0))
        {
            1])))
            if (currencyFrom.Equals(rateArray[count, 0]) && (currencyTo.Equals(rateArray[count,
                {
                    //Retrieves the value from the array using count.
                    //It then converts the retrieved value datatype into another datatype (decimal).
                    //It then stores it in the variable for the exchange rate.
                    exchangeRate = Convert.ToDecimal(rateArray[count, 2]);
                    break;
                }
                //Increment count value by 1.
                count = (count + 1);
            }
            //Triggers the calculate method to run.
            calculate();
        }

        public void calculate()
        {
            //Multiply the amount by the exchangeRate and display the answer in the result variable.
            result = amount * exchangeRate;

            //Trigger the calculate method to run.
            display();
        }

        public void display()
        {
            //Displays the result in the result text field.
            txtResult.Text = result.ToString();
        }
    }
}

```

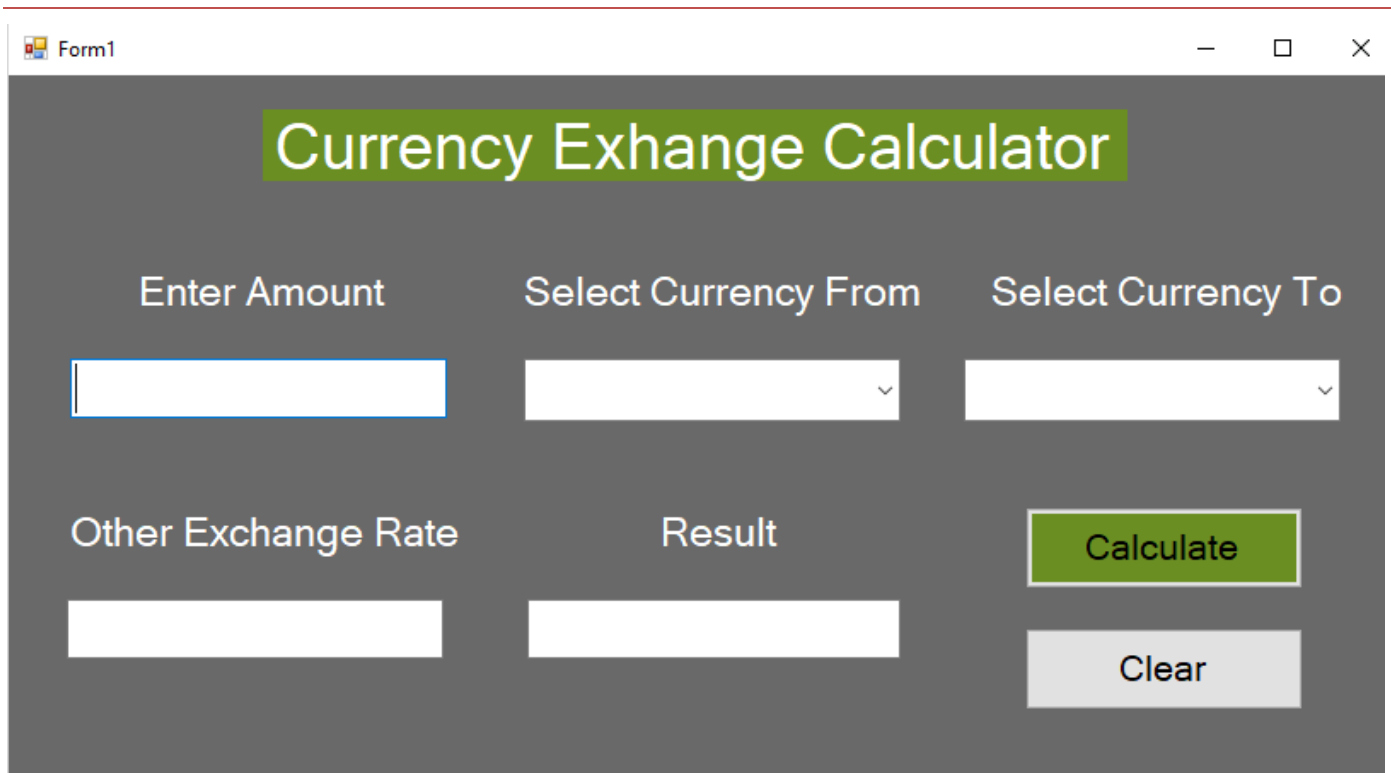


```
private void btnClear_Click(object sender, EventArgs e)
{
}

private void cmbFrom_SelectedIndexChanged(object sender, EventArgs e)
{
}

private void cmbTo_SelectedIndexChanged(object sender, EventArgs e)
{
}
}
}
```

## User Interface



The screenshot displays a Windows application window titled "Form1" with standard minimize, maximize, and close buttons. The main content area has a dark gray background and features a green header bar with the text "Currency Exchange Calculator". Below the header, the interface is organized into several sections:

- Enter Amount:** A white text input field.
- Select Currency From:** A white dropdown menu with a downward arrow.
- Select Currency To:** A white dropdown menu with a downward arrow.
- Other Exchange Rate:** A white text input field.
- Result:** A white text input field.
- Calculate:** A green button with white text.
- Clear:** A light gray button with black text.

# Altered Code

Description	Code Before Alterations	Altered Code	Why I Altered This Code
<p>This is test no 5 - 5.2, where I checked that the exchange rate works from GBP to EURO.</p>	<pre> public void findExchangeRate() {     //Here I have declared a variable     called count within the findExchangeRate method.     //The variable has a datatype of int     (meaning integer) with an initial value of 0.     int count = 0;      while (count &lt; rateArray.GetUpperBound(0))     {         if (currencyFrom.Equals(rateArray[count, 0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1])))         {              exchangeRate = Convert.ToDecimal(0);              //Converts one datatype into another datatype.             exchangeRate = Convert.ToDecimal(txtExchangeRate.Text);              Convert.ToDecimal(0);             break;         }         count = (count + 1);     } } </pre>	<p>After</p> <pre> public void findExchangeRate() {     //Here I have declared a variable called count within the findExchangeRate method.     //The variable has a datatype of int     (meaning integer) with an initial value of 0.     int count = 0;      while (count &lt;= rateArray.GetUpperBound(0))     {         if (currencyFrom.Equals(rateArray[count, 0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1])))         {              //Retrieves the value from the array using count.             //It then converts the retrieved value datatype into another datatype (decimal).             //It then stores it in the variable for the exchange rate.             exchangeRate = Convert.ToDecimal(rateArray[count, 2]);              break;         }         count = (count + 1);     }     calculate(); } </pre>	<p>I altered this code because, the program kept crashing when I pressed the calculate button when I tried to test if the program could calculate and display the exchange rate from GBP to Euro.</p> <p>The issue was that the program was not retrieving the data from the array and then executing the calculate method, once I had placed these in my code, the exchange rate worked.</p>
<p>This is test no 6 - 6.1, where I checked that the exchange</p>	<pre> public void findExchangeRate() {     //Here I have declared a variable called count within the findExchangeRate method. </pre>	<pre> public void findExchangeRate() {     //Here I have declared a variable called count within the findExchangeRate method. </pre>	<p>I altered this code because, the program kept crashing when I pressed the calculate button when I tried to test if the program could</p>

<p>rate works from EURO to GBP.</p>	<pre> //The variable has a datatype of int (meaning integer) with an initial value of 0. int count = 0;  while (count &lt; rateArray.GetUpperBound(0)) {     if (currencyFrom.Equals(rateArray[count, 0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1])))     {          exchangeRate = Convert.ToDecimal(0);  //Converts one datatype into another datatype. exchangeRate = Convert.ToDecimal(txtExchangeRate.Text);  Convert.ToDecimal(0); break;     }     count = (count + 1); } } </pre>	<pre> //The variable has a datatype of int (meaning integer) with an initial value of 0. int count = 0;  while (count &lt;= rateArray.GetUpperBound(0)) {     if (currencyFrom.Equals(rateArray[count, 0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1])))     {         //Retrieves the value from the array using count. //It then converts the retrieved value datatype into another datatype (decimal). //It then stores it in the variable for the exchange rate. exchangeRate = Convert.ToDecimal(rateArray[count, 2]);          break;     }     count = (count + 1); } calculate(); } </pre>	<p>calculate and display the exchange rate from EURO to GBP.</p> <p>The issue was that the program was not retrieving the data from the array and then executing the calculate method, once I had placed these in my code, the exchange rate worked.</p>
<p>This is test no 7 – 7.1, where I checked that the exchange rate works from GBP to AUD.</p>	<pre> public void findExchangeRate() {     //Here I have declared a variable called count within the findExchangeRate method. //The variable has a datatype of int (meaning integer) with an initial value of 0. int count = 0;  while (count &lt; rateArray.GetUpperBound(0)) {     if (currencyFrom.Equals(rateArray[count, 0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1])))     { </pre>	<pre> public void findExchangeRate() {     //Here I have declared a variable called count within the findExchangeRate method. //The variable has a datatype of int (meaning integer) with an initial value of 0. int count = 0;  while (count &lt;= rateArray.GetUpperBound(0)) {     if (currencyFrom.Equals(rateArray[count, 0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1])))     { </pre>	<p>I altered this code because, the program kept crashing when I pressed the calculate button when I tried to test if the program could calculate and display the exchange rate from GBP to AUD.</p> <p>The issue was that the program was not retrieving the data from the array and then executing the calculate method, once I had placed these in my code, the exchange rate worked.</p>

	<pre> exchangeRate = Convert.ToDecimal(0);  //Converts one datatype into another datatype. exchangeRate = Convert.ToDecimal(txtExchangeRate.Text);  Convert.ToDecimal(0); break; } count = (count + 1); } } </pre>	<pre> //Retrieves the value from the array using count. //It then converts the retrieved value datatype into another datatype (decimal). //It then stores it in the variable for the exchange rate. exchangeRate = Convert.ToDecimal(rateArray[count, 2]);  break; } count = (count + 1); } calculate(); } </pre>	
<p>This is test no 8 – 8.2, where I checked that the exchange rate works from AUD to GBP.</p>	<pre> public void findExchangeRate() { //Here I have declared a variable called count within the findExchangeRate method. //The variable has a datatype of int (meaning integer) with an initial value of 0. int count = 0;  while (count &lt; rateArray.GetUpperBound(0)) { if (currencyFrom.Equals(rateArray[count, 0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1]))) {  exchangeRate = Convert.ToDecimal(0);  //Converts one datatype into another datatype. exchangeRate = Convert.ToDecimal(txtExchangeRate.Text);  Convert.ToDecimal(0); break; } count = (count + 1); } } </pre>	<pre> public void findExchangeRate() { //Here I have declared a variable called count within the findExchangeRate method. //The variable has a datatype of int (meaning integer) with an initial value of 0. int count = 0;  while (count &lt;= rateArray.GetUpperBound(0)) { if (currencyFrom.Equals(rateArray[count, 0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1]))) {  //Retrieves the value from the array using count. //It then converts the retrieved value datatype into another datatype (decimal). //It then stores it in the variable for the exchange rate. exchangeRate = Convert.ToDecimal(rateArray[count, 2]);  break; } count = (count + 1); } calculate(); } </pre>	<p>I altered this code because, the program kept crashing when I pressed the calculate button when I tried to test if the program could calculate and display the exchange rate from AUD to GBP.</p> <p>The issue was that the program was not retrieving the data from the array and then executing the calculate method because I had not set the count in the while statement to less than or equal to, once I had placed the equals sign next to the less than sign in my code, the exchange rate worked.</p>

	<pre> } </pre>	<pre> } </pre>	
<p>This is test no 9 – 9.1, where I checked that the exchange rate works from GBP to USD.</p>	<pre> public void findExchangeRate() {     //Here I have declared a variable     called count within the findExchangeRate method.     //The variable has a datatype of int     (meaning integer) with an initial value of 0.     int count = 0;      while (count &lt; rateArray.GetUpperBound(0))     {         if (currencyFrom.Equals(rateArray[count, 0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1])))         {             exchangeRate = Convert.ToDecimal(0);              //Converts one datatype into another datatype.             exchangeRate = Convert.ToDecimal(txtExchangeRate.Text);              Convert.ToDecimal(0);             break;         }         count = (count + 1);     } } </pre>	<pre> public void findExchangeRate() {     //Here I have declared a variable called count within the findExchangeRate method.     //The variable has a datatype of int     (meaning integer) with an initial value of 0.     int count = 0;      while (count &lt;= rateArray.GetUpperBound(0))     {         if (currencyFrom.Equals(rateArray[count, 0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1])))         {             //Retrieves the value from the array using count.             //It then converts the retrieved value datatype into another datatype (decimal).             //It then stores it in the variable for the exchange rate.             exchangeRate = Convert.ToDecimal(rateArray[count, 2]);              break;         }         count = (count + 1);     }     calculate(); } </pre>	<p>I altered this code because, the program kept crashing when I pressed the calculate button when I tried to test if the program could calculate and display the exchange rate from GBP to USD.</p> <p>The issue was that the program was not retrieving the data from the array and then executing the calculate method, once I had placed these in my code, the exchange rate worked.</p>
<p>This is test no 10 – 10.1, where I checked that the exchange rate works from USD to GBP.</p>	<pre> public void findExchangeRate() {     //Here I have declared a variable     called count within the findExchangeRate method.     //The variable has a datatype of int     (meaning integer) with an initial value of 0.     int count = 0;      while (count &lt; rateArray.GetUpperBound(0))     { </pre>	<pre> public void findExchangeRate() {     //Here I have declared a variable called count within the findExchangeRate method.     //The variable has a datatype of int     (meaning integer) with an initial value of 0.     int count = 0;      while (count &lt;= rateArray.GetUpperBound(0))     { </pre>	<p>I altered this code because, the program kept crashing when I pressed the calculate button when I tried to test if the program could calculate and display the exchange rate from USD to GBP.</p> <p>The issue was that the program was not retrieving the data from the array and then executing the calculate</p>

	<pre> if (currencyFrom.Equals(rateArray[count, 0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1]))) {     exchangeRate = Convert.ToDecimal(0);  //Converts one datatype into another datatype. exchangeRate = Convert.ToDecimal(txtExchangeRate.Text);  Convert.ToDecimal(0); break; } count = (count + 1); } } </pre>	<pre> if (currencyFrom.Equals(rateArray[count, 0]) &amp;&amp; (currencyTo.Equals(rateArray[count, 1]))) {     //Retrieves the value from the array using count. //It then converts the retrieved value datatype into another datatype (decimal). //It then stores it in the variable for the exchange rate. exchangeRate = Convert.ToDecimal(rateArray[count, 2]);  break; } count = (count + 1); } calculate(); } </pre>	<p>method, once I had placed these in my code, the exchange rate worked.</p>
<p>This is test no 11 – 11.1, where I checked invalid user input displays a message box when the user enters the wrong type of data.</p>	<pre> public void storeData() {     //This gets the data from the combo- box for the currency from. //And sets /stores it in the variable for the currency from. currencyFrom = cmbFrom.Text;  //This gets the data from the combo- box for the currency To. //And sets /stores it in the variable for the currency To. currencyTo = cmbTo.Text;  //This gets the amount entered in by the user and stores it into the variable amount. amount = Convert.ToDecimal(txtEnter.Text);  //This will trigger the method chkRate to run. chkRate(); } </pre>	<pre> public void storeData() {     try     {         //This gets the data from the combo- box for the currency from. //And sets /stores it in the variable for the currency from. currencyFrom = cmbFrom.Text;  //This gets the data from the combo- box for the currency To. //And sets /stores it in the variable for the currency To. currencyTo = cmbTo.Text;  //This gets the amount entered in by the user and stores it into the variable amount. amount = Convert.ToDecimal(txtEnter.Text);  //This will trigger the method chkRate to run. </pre>	<p>I altered this code because, the program kept crashing when I entered the wrong data type into the program.</p> <p>The issue was that the program was not displaying an error message box to the user.</p> <p>I altered the code to include a try/catch statement which also includes a message in a pop-up message box. The message says “Invalid user input”.</p>

```
        chkRate();
    }
    catch (Exception e)
    {
        MessageBox.Show("Invalid user
input");
    }
}
```

# Feedback

---

Name of Reviewer: James Saunders

Date of Review: 28/06/2018

Name of Program: Currency Exchange Calculator

Interface	Reviewer-Good	Reviewer-Bad	Reviewers-Comments	Programmers Comment
Font (Size, Colour, Contrast )	Font type was easy to read on background.	The labels are not inlined with boxes.	Adjust position.	I disagree, the labels are centre-aligned with the boxes and some of the titles are longer than others, this is in-keeping with my design and I wish to keep them as they are, as I do not feel the reviewer has given any valid reason to suggest otherwise.
English (Spelling, Grammar, Capitals etc)	All good.	Nothing bad.	Nothing to change.	Happy with the feedback.
Background	Colours look good together.	The background is a little plain, could do with some brighter colours.	Possibly change the colours to something brighter.	I disagree, the program is to be used to functionality and usability purposes, I feel the visual look is fairly irrelevant and the colours are fairly standard. I like the grey background with the accents of green, I didn't want to use blue which is the most used colour in programs and I felt a white



				background was too plain.
Layout (textboxes, labels, buttons, combo boxes)	Simple and logical.	Labels not consistently placed.	Adjust position.	The labels are consistently placed, I do not understand what the reviewer is talking about.
Easy of Use	It is very easy to use.	Clear button not functioning.  And when exchanging currencies: AUD – AUD, AUD – EURO, AUD – USD, GBP – GBP, GBP, USD – USD, USD – AUD, USD – EURO, EURO – EURO, EURO – AUD, EURO – USD, these choices do not display the answer to the user.	Program the clear function.  Program the rest of the currencies to display the answer to the user.	I plan to complete these in my refinements of the program.
<b>Program Structure Quality</b>				
Comments to help read the code	Well commented throughout.	Nothing.	Can't suggest improvements.	Happy with this feedback.
Indentation	Easy to follow.	Nothing.	Nothing.	Happy with this feedback.
Method names are meaningful	Yes, easy to understand.	Nothing.	Nothing.	Happy with this feedback.
Variable names meaningful	Yes, explains its use just from the name.	Nothing.	Nothing.	Happy with this feedback.
Easy to read code	Yes, easy to follow structure.	Unneeded events in code.	Remove unneeded code.	Happy with this feedback.
<b>Functionality</b>				
Exchange Rate correctly worked	Works correctly.	When exchanging currencies: AUD – AUD, AUD – EURO, AUD – USD, GBP – GBP,	Add extra exchange rates.	Happy with this feedback, changed will be made in the refinements.

		GBP, USD – USD, USD – AUD, USD – EURO, EURO – EURO, EURO – AUD, EURO – USD, these choices do not display the answer to the user.		
Result of Calculation formatted	Result shown correctly.	Too many zeros showing in the result.	Limit the number of zeros to 2 decimal places.	I am happy with this feedback, and the changes will be made during refinements.
Other	N/A	N/A	N/A	Nothing to comment on.
<b>Quality</b>				
Easy to use	Easy to use	N/A	N/A	Happy with the feedback.
Robust (works every time)	Program works without crashing, when the data type is invalid and displays a message box.	And when exchanging currencies: AUD – AUD, AUD – EURO, AUD – USD, GBP – GBP, GBP, USD – USD, USD – AUD, USD – EURO, EURO – EURO, EURO – AUD, EURO – USD, these choices do not display the answer to the user.	Add extra calculations.	Happy with the feedback, I plan to make these changes in the refinements.
Reliable(produces Correct answers)	Works fine	And when exchanging currencies: AUD – AUD, AUD – EURO, AUD – USD, GBP – GBP, GBP, USD – USD, USD – AUD, USD – EURO, EURO – EURO, EURO –	Add extra calculations.	Happy with the feedback, as I have said, I plan to make these changes in the refinements.

		AUD, EURO – USD, these choices do not display the answer to the user.		
--	--	---	--	--

Name of Reviewer: Harry Blair

Date of Review: 28/06/18

Name of Program: Currency Exchange Calculator

Interface	Reviewer-Good	Reviewer-Bad	Reviewers-Comments	Programmers Comment
Font (Size, Colour, Contrast )	The font blended in with the design of the app and brought the simplicity of the navigation and use of the app.	No bad things.	No adjusting comment.	I am very content with this feedback.
English (Spelling, Grammar, Capitals etc)	All the grammar, works in conjunction with their corresponding combobox and textbox.	The spelling of the title, mainly 'The Exchange Calculator', is spelt incorrectly.	Fix the spelling of the title for your app.	I am very happy with this feedback. And I plan to change this.
Background	I like the background colour and it works super effectively with the surrounding colours.	No bad things.	No adjusting comment.	I am very pleased with this feedback.
Layout (textboxes, labels, buttons, combo boxes	The layout of the app is of the highest quality and looks proficient and professional.	No bad things.	No adjusting comment.	I am very happy with this feedback.

Easy of Use	The app is very easy to navigate around and is very simplistic to use.	No bad things.	No adjusting comment.	I am very happy with this feedback.
<b>Program Structure Quality</b>				
Comments to help read the code	Comments are accurate and precise, whilst helping the user to understand the code.	No bad things.	No adjusting comment.	I am very pleased with this feedback.
Indentation	Indentation is good.	No bad things.	No adjusting comment.	I am very pleased with this feedback.
Method names are meaningful	Method names are good and correlates correctly.	No bad things.	No adjusting comment.	I am very happy with this feedback.
Variable names meaningful	Variable names are good and are meaningful.	No bad things.	No adjusting comment.	I am content with this feedback.
Easy to read code	Code is laid out neatly and visible.	No bad things.	No adjusting comment.	I am happy with this feedback.
<b>Functionality</b>				
Exchange Rate correctly worked	Exchange rate works correctly.	Clear button doesn't work and some of the exchange rates don't: AUD – AUD, AUD – EURO, AUD – USD, GBP – GBP, GBP, USD – USD, USD – AUD, USD – EURO, EURO – EURO, EURO – AUD, EURO – USD.	Clear button needs to be fixed.  The currencies in your array need to be added in.	I plan to complete these in my refinements of the program.
Result of Calculation formatted	Results on the calculator works and is very fun to use.	No bad things.	No adjusting comment.	I am very happy with this feedback.
Other	No other comments	No bad things.	No adjusting comment.	Nothing to comment on.

Quality				
Easy to use	The navigation is easy to use and very simplistic.	No bad things.	No adjusting comment.	I am very happy with this feedback.
Robust (works every time)	The calculator works.	No bad things.	No adjusting comment.	I am very pleased with this feedback.
Reliable(produces Correct answers)	The answers are correct and reliable.	No bad things.	No adjusting comment.	I am very content with this feedback.

# Improvements

What I changed and why I made these changes	Before Screenshot	After Screenshot
<p>I originally planned to have a clear button but decided against it due to the fact that I didn't know how to code it.</p> <p>However, after the feedback I received, I decided to research how to code the clear button and find out that it was relatively straight-forward to implement.</p> <p>I basically just researched how to clear a combo box and a text box and simply entered the names of the text boxes and added <code>.clear()</code>; to the end of each one to instruct it to clear. For the combo-boxes, I again entered the names of the two combo boxes and added <code>.Text = ""</code>; which instructs the combo boxes to clear.</p>	<pre data-bbox="528 363 1218 467">private void btnClear_Click(object sender, EventArgs e) { } </pre>	<pre data-bbox="1256 363 2136 751">private void btnClear_Click(object sender, EventArgs e) {     //This is the method to clear the data entered by the user.     //This is the line of code to clear the enter amount textbox.     txtEnter.Clear();     //This is the line of code to clear the exchange rate text box.     txtExchangeRate.Clear();     //This is the line of code to clear the result text box.     txtResult.Clear();     //This is the line of code to clear the currency from combo box.     cmbFrom.Text = "";     //This is the line of code to clear the currency to combo box.     cmbTo.Text = ""; } </pre>

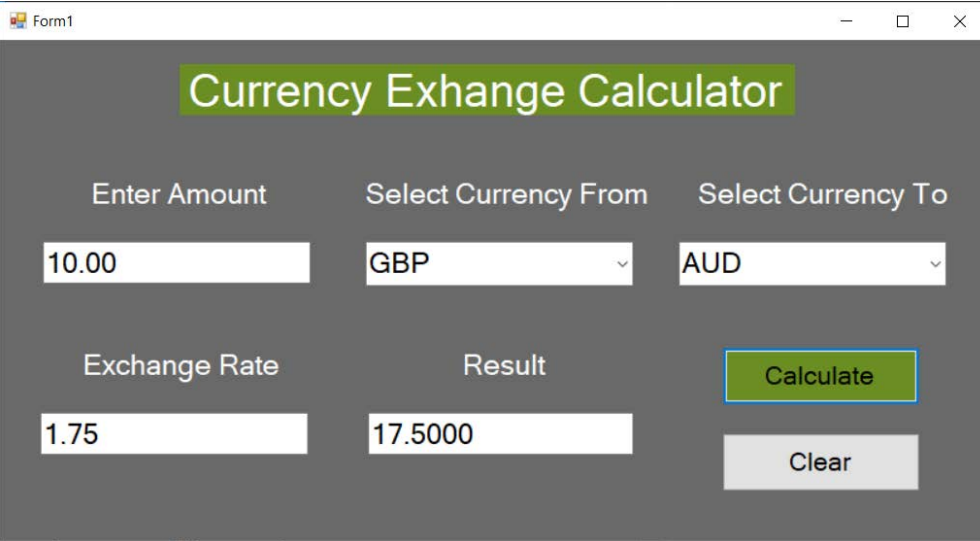
# Task 4

---

# Review Original Requirements P6

---

The client (a local travel agent) requested we design and develop a currency exchange rate calculator to calculate the rates for its customers. The program has to allow the user to enter an amount in British pounds and calculate the equivalent amount in a foreign currency (such as Australian Dollars). The program should then allow the user to enter an amount in a foreign currency (such as USD, United States Dollars) and calculate the equivalent amount into British pounds. The user must be also be able to update the exchange rates and the exchanged amount must be shown alongside the exchange rate.



The screenshot shows a window titled "Form1" with a dark grey background. At the top, the text "Currency Exchange Calculator" is displayed in a green box. Below this, there are three input fields: "Enter Amount" containing "10.00", "Select Currency From" containing "GBP", and "Select Currency To" containing "AUD". Below these, there are two more input fields: "Exchange Rate" containing "1.75" and "Result" containing "17.5000". To the right of the "Result" field is a green "Calculate" button, and below it is a grey "Clear" button.

As shown in the screenshot above, the user can enter an amount into the "Enter Amount" text box. The user is then able to select the currency to exchange from and to via the combo selection boxes. The user has a choice of four currencies AUD, EURO, GBP and USD to exchange from and to, this fulfils the requirement for at least three different currencies as requested in the client's briefing. Once the calculate button is pressed, the calculation result is shown in the result text box and the exchange rate used in the calculation is displayed in the exchange rate text box as requested by the client. In the test log the program was tested and screenshot evidence was provided to show the individual tests that were made to show the program functions as it was originally meant to. The program fulfils the requirements and purpose stated in the brief made by the client by providing an accurate calculation of the exchange rate with the full functions available to the user. It also contains a clear button for additional functionality, by clearing the entered data ready for the next user.