App Report



Student Loan Calculator

Table of Contents

[Introduction 2](#_Toc5311381)

[Cover page 2](#_Toc5311382)

[Design\_Splash Screen 2](#_Toc5311383)

[Code\_Splash Screen 3](#_Toc5311384)

[First / Input Page 3](#_Toc5311385)

[Design\_Form 2 3](#_Toc5311386)

[Code\_Form 2 4](#_Toc5311387)

[Error handling 6](#_Toc5311388)

[Try & Catch and Loop 6](#_Toc5311389)

[Goto - Exitpoint 6](#_Toc5311390)

[Comments 7](#_Toc5311391)

[Second / Output Page 7](#_Toc5311392)

[Design\_Form 3 7](#_Toc5311393)

[Code\_Form 3 8](#_Toc5311394)

[“Quit” and “Reset” Button 8](#_Toc5311395)

[Application icons 10](#_Toc5311396)

[References 11](#_Toc5311397)

# Introduction

Through the module of Applications Development, this application was created for students to help give them clear overview of their student loans. Its structure is made so all the important information, that might concern them, is reliable. Inside the following documentation are provided screenshots along with clear step-by-step description of the key points of the application, both in terms of coding and design. The application’s design aim to be simplistic and “clean”, making it easy for the user to run the programme.

# Cover page

## Design\_Splash Screen

When we lunch the program, the first thing we encounter is the “Splash Screen”. Its purpose is to initiate a traditional “loading process” which will smoothly introduce the user to the app’s main screen.



*The Loading bar progressively filling up*

Figure 1. Splash Screen

## Code\_Splash Screen

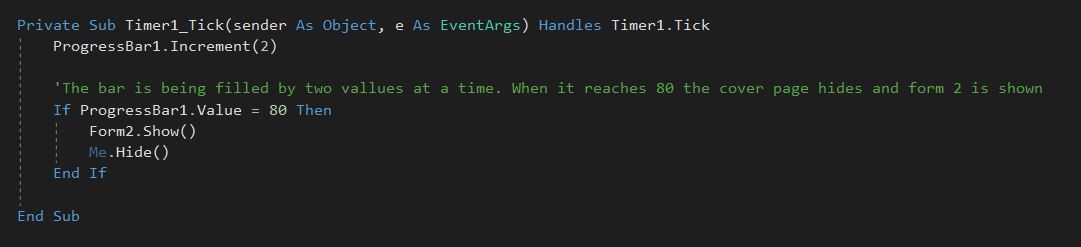
 A it is shown in Figure 2, the splash screen’s “Loading” time is affected by the number indicated by marker1. Marker 2 indicates the “steps” program is doing to fill the timer counter, thus effecting the speed of the bar.

Figure Timer

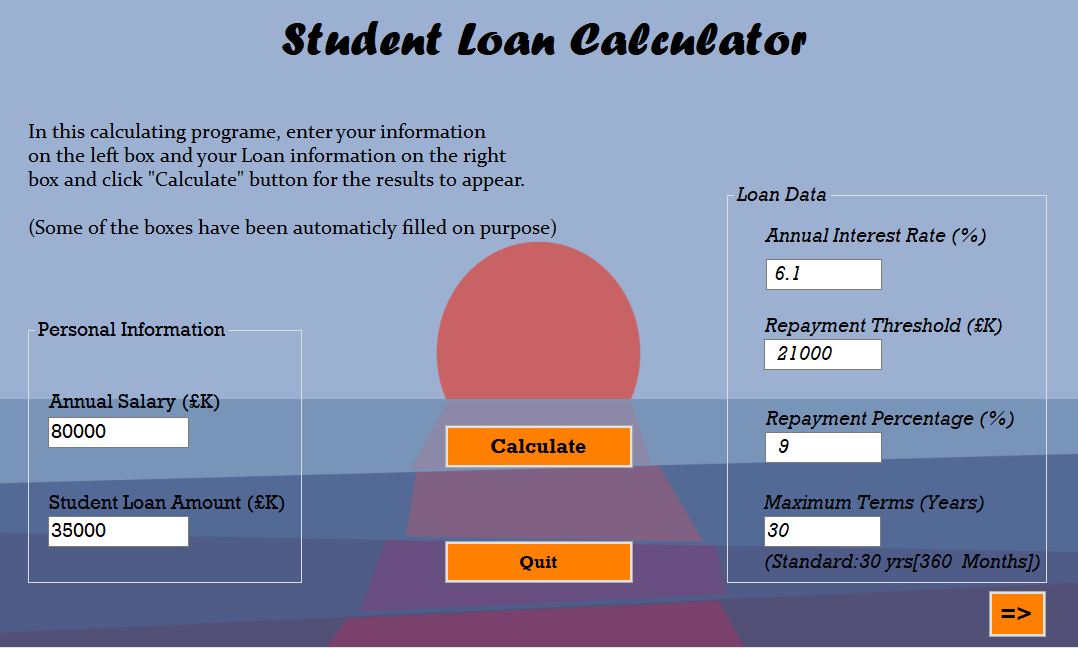
Marker1

Marker2

# First / Input Page

## Design\_Form 2

When the Splash Screen is finished loading, Form 2 appears. Form 2 is the main, input, form which also works as a “welcome” page as a small text provides a brief description of how the app works. The input text boxes provided represent all the important aspects that will make the calculations and provide the user with as many and accurate results as possible.



Marker 3

Figure Main / Input Page

Marker 5

Marker 4

Marker 8

Marker 7

Marker 6

A small text paragraph is provided describing briefly the main aspects of the programme as shown by Marker 3. Left Group Box (Marker 4) is the main input with Salary and Loan Amount whereas, right Group Box (Marker8) consists of already provided information such as interest rate and repayment threshold since those are the fixed current values (but they can, later on, be changed). The “Arrow” button (Marker 7) exists to give the ability to the user to move between input&output forms, being placed in the corner as its importance is not significant. On the other hand, “Calculate” button (Marker5) and “Quit” button (Marker 6) are placed in the centre of the screen in order to get the attention of the user and be clear to find following a simplistic layout.

## Code\_Form 2

The coding behind Form1 consists of different layers that follow a specific order. The main subject in the application in the “Calculate” button. All the important actions take place with a specific order, after the button is pressed. After we have entered our global variables, the first step if for the program to identify if the textbox input values are valid by ensuring their values are positive, numeric numbers and not empty spaces. The application’s design is meant to be as simple and easy to understand as possible, all according to Ben Schneiderman and his golden rules of simplified user interference

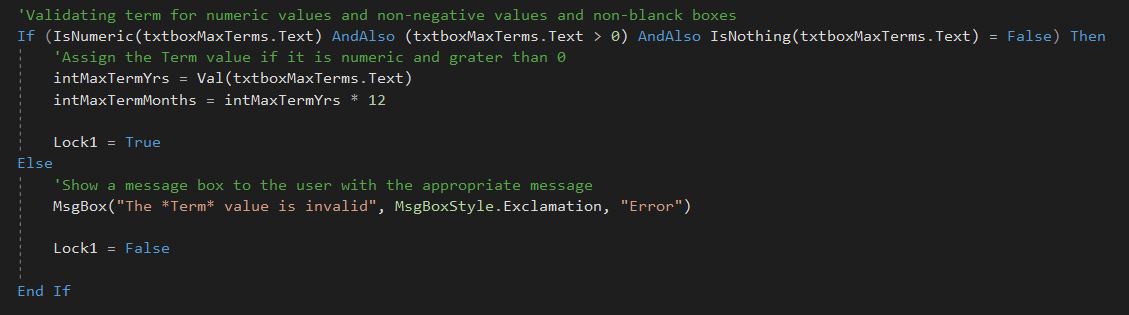
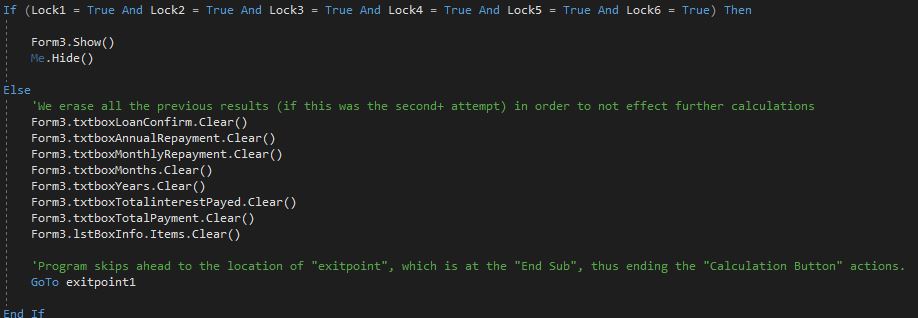
If an input error occurs, the programme needs to be able to handle it appropriately (skip calculations, won’t show form 2). For this reason, there is a “Lock” mechanism as shown in Figure 4 and Figure 5 below.

Figure Lock (a)

If the value is not correct, Lock is set *False*

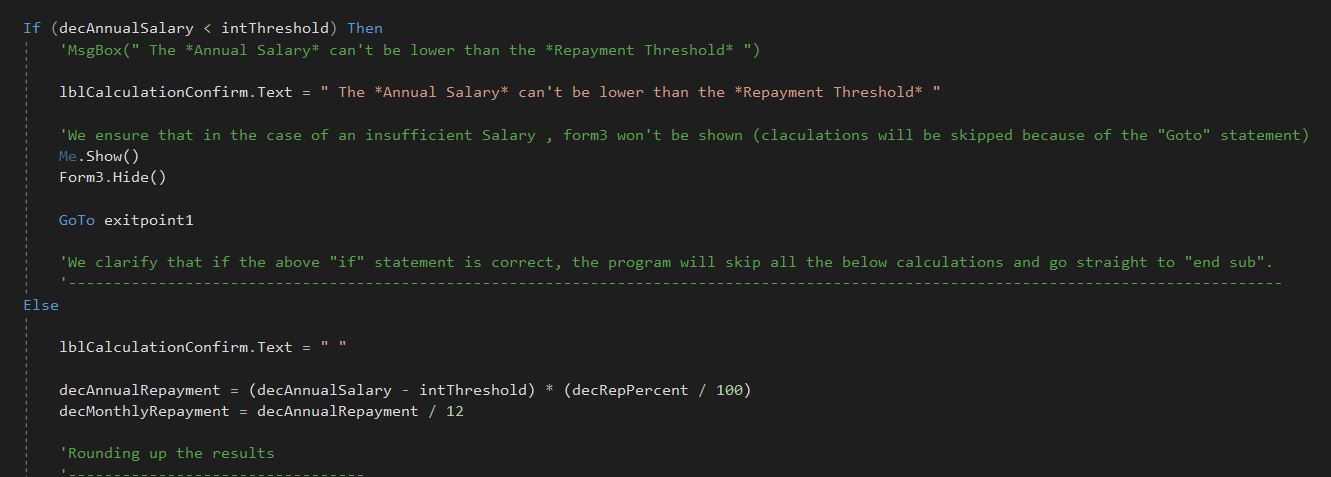
If the value provided is correct, Lock is set *True*

(Marker 9: ) If at least one of the values, inputted by the user, does not meet the requirements, the program will skip ahead everything up to the point the “Calculate” button’s actions end, showing only the error messages. This is accomplished by an error handling statement, analysed further down the assignment. Another action that is being taken is erasing the previous (if any) results already shown from previous calculations, to ensure they wont effect, in any way, future calculations.

Marker 9

All Values (each having their own Lock) need to be *True* for the programme to proceed

Figure Lock (b)

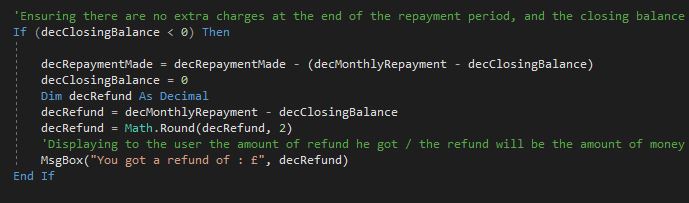
 The third step is another validation process (Figure 6) in which we verify if the Salary amount, the user entered, is higher than the repayment Threshold. If True then the calculation processes will begin, whereas if False, the program will skip the rest of the processes, only showing an appropriate error message.

Figure

# Error handling

## Try & Catch and Loop

There is also the use of error handling techniques such as “Try” & “Catch” which is used for the Loop statement. Inside the Loop statement, with the use of a counter, we are able to do multiple repetitive calculations to show monthly results to the user, with the aim of a more precise overall final output. The process continues until the counter(s) reach their given limit. The reason of the “Try” & “Catch” use is to prevent any types of errors from occurring during the calculations. Thus, achieving reliable results. Additional to the above, an important aspect of the Loop statement is another If statement, shown in Figure 7. With the use of that statement we are able to prevent surplus expenses at the final month of repayment. In the case of the Loan being payed earlier than the given term period, the program needs to be able to display, for the final month, 0 value for the Closing balance of the loan (Marker 10) and no further charges for the user (Marker 11). Also, the program will show a message to the user displaying the refund he got (Marker 12); since in that scenario, the final monthly repayment will be larger than the actual balance left to be replayed.



Marker 11

Marker 12

Marker 10

Figure . 3rd Validation ( If ) statement

## Goto - Exitpoint

For a program with the purpose of calculating Loans and money, certainty for the results and safety over errors is at most important. As already mentioned, in the application we use the error handling statement of “Try” & “Catch”. Additionally to that, there is also use of the “GoTo”-“exitpoint” statement. First, we mark when do we want the process to initiate. Then we need to mark the location the program will head to after the initiation. A clear example is an IF statement as shown by Figure 5. If the statement is FALSE then the program will ‘go’ to the “exitpoint” location, bypassing all the in-between aspects of the code.

In this case, the “GoTo” statement from Figure 5, has an exitpoint located at the end of the “Calculate\_Click” sub, right before the “End Sub” command. This indicates that if the IF statement is FALSE, thus triggering the “GoTo” statement, the program will skip all the rest of the code and will consider “End Sub” as the next step, ending the process of the “Calculate\_Click” (the process of when the user clicks the Calculate button).

Figure . "GotTo" - "exitpoint1"

## Comments

In addition to certainty and error-safety provided, the code must also be easy to read and most importantly, clear to understand. This can be achieved with the use of comments. The purpose of comments in the code is to act as instructions and help another individual who may be reading the code, to understand our way of thought, how and why we did things in a certain way. The more comments we provide through the code with clear, aimed and analytical content, the easier it is for us and other viewers to implement future changes and improvements to the code. Comments also act as notes, to us and other viewers, left to specify facts for future reading.

# Second / Output Page

## Design\_Form 3

For Form 3’s design, background is set with a, similar to Form 2’s, simple minimalistic theme with a mild blend of light and dark shades of blue, aiming to be relaxing for the viewer’s eye. Text boxes and buttons, all follow a circular layout leaving the centre empty in order to not make it look overcrowded and complicated for the user to locate each information provided by the boxes. Buttons have been placed in the corner and not the middle because their use is mostly secondary. We want the users to first clearly see all the information they need and then focus on whether they want to reset the process, go back and view their input page or quit the program.

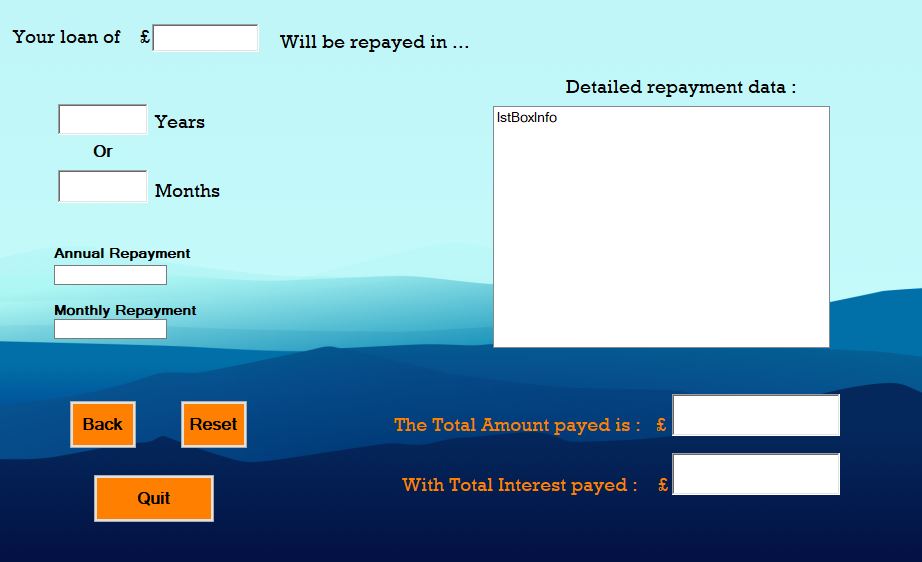


Figure 9. Output page

This part does not present any new information to the user. It has been included, for convenience reasons, to make it clear that the provided results relate to the given Loan, providing a more complete result screen.

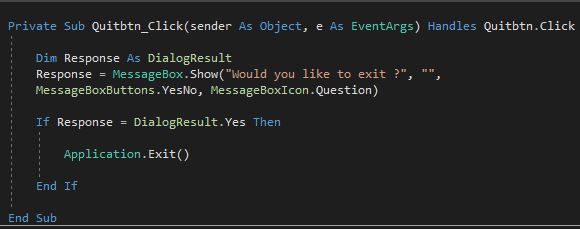
## Code\_Form 3

Considering that Form 3’s purpose is to only display results with a few buttons provided as well, the coding density is less than Form 2’s. All functions including the various calculations and actions are occurring inside form 2 and are programmed to show the results in the allocated boxes inside form 3. No text boxes are set to be editable by the user for the reason it is a results screen and its purpose is for display only; to prevent the user from changing any results. The following are the three provided buttons and how they work;

## “Quit” and “Reset” Button

The “Quit” button exists in both Form 2 (Main) and Form 3 (Output), with the same format.

When the user presses the “Quit” button, a message box will appear, requesting from the user to confirm their choice. That way, we avoid any accidental mis-clicks on the Quit button, forcing the user to restart the app and repeat the process.

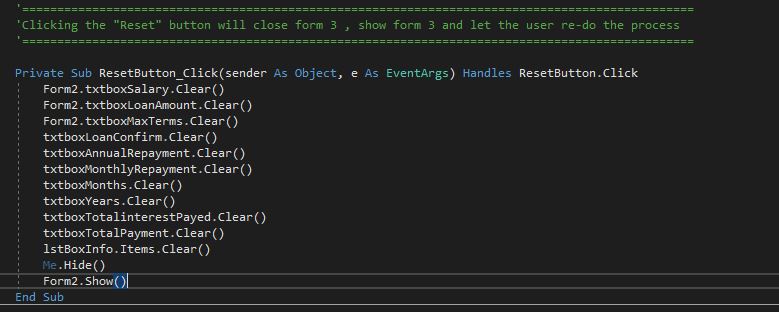
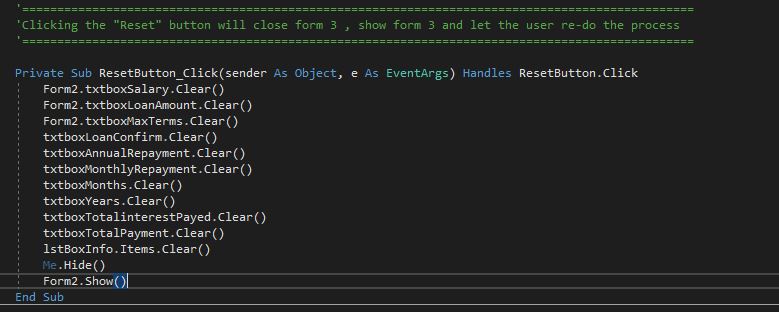


The program asks the user to confirm his action with a Yes or No Question

If the user’s response is “Yes: the program will shut down

Figure . Quit Button

The “Reset” button’s purpose is to provide the user with a fresh application. No results should be displayed, and the main input boxes should be cleared. A few input text boxes will remain as provided (by the user or the program itself) since they may be considered as “fixed numbers” and not require change.



First the program clears all output boxes from previous results …

…Then proceeds to appear the input form again and hide the output form

Figure . Reset Button

# Application icons

Both the main application icon and the included error messages own a specific icon which indicates their purpose. As it can be seen from Figure 12, in the user’s taskbar, our application will appear with its unique Calculator icon, relative to its purpose and making it distinct from different running apps. The error messages (Figure 13), have an exclamation mark as icon, indicating the situation of an error and something is wrong.

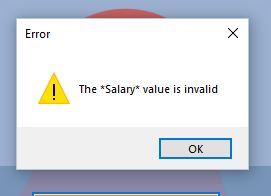


Figure . Application Main Icon

Figure . Error message icon

# References

Johnson, J. (2003). *Web bloopers : 60 common web design mistakes and how to avoid them.* Boston : Morgan Kaufmann.

Nielsen, J. (2001, October 31). *113 Design Guidelines for Homepage Usability.* Retrieved from https://www.nngroup.com/: https://www.nngroup.com/articles/113-design-guidelines-homepage-usability/

Sempf, B. (2005). *Visual Basic 2005 for Dummies.* Indianapolis, Indiana: Wiley Publishing, Inc.

Warren, G. (2019). *Tutorial: Get started with Visual Basic in Visual Studio.* Retrieved from https://img-prod-cms-rt-microsoft-com.akamaized.net/cms/api/am/imageFileData/RE1Mu3b?ver=5c31: https://docs.microsoft.com/en-us/visualstudio/get-started/visual-basic/tutorial-console?view=vs-2019

WONG, E. (2019). *Shneiderman’s Eight Golden Rules Will Help You Design Better Interfaces.* Retrieved from https://www.interaction-design.org/: https://www.interaction-design.org/literature/article/shneiderman-s-eight-golden-rules-will-help-you-design-better-interfaces