130089872

Pond Calc assigment.

Contents

[Table of Figures 2](#_Toc61349666)

[Pseudocode and flow chart 3](#_Toc61349667)

[Introduction 3](#_Toc61349668)

[Admin Login 3](#_Toc61349669)

[Pseudocode 3](#_Toc61349670)

[Admin Menu 6](#_Toc61349671)

[Pseudocode 6](#_Toc61349672)

[Pond calculator measurements 9](#_Toc61349673)

[Pseudocode 9](#_Toc61349674)

[Test log and Evaluation 12](#_Toc61349675)

[Test log 12](#_Toc61349676)

[Appendices 16](#_Toc61349677)

# Table of Figures

[Figure 1 Admin Login 4](#_Toc61349683)

[Figure 2 Login UI 5](file:///C:\Users\dales\Dropbox\My%20PC%20(LAPTOP-D0497OJ7)\Desktop\Dales%20course%20work\year%201\greg%20program%20fundamentals\codin\Dale%20Smallwood%20Programing%20Fundementals%20assigment%202.docx#_Toc61349684)

[Figure 3 Admin Menu 7](#_Toc61349685)

[Figure 4 Admin menu default UI 8](#_Toc61349686)

[Figure 5 New user section 8](file:///C:\Users\dales\Dropbox\My%20PC%20(LAPTOP-D0497OJ7)\Desktop\Dales%20course%20work\year%201\greg%20program%20fundamentals\codin\Dale%20Smallwood%20Programing%20Fundementals%20assigment%202.docx#_Toc61349687)

[Figure 6 Length validation flow chart 10](#_Toc61349688)

[Figure 7 Pond Calc UI 11](file:///C:\Users\dales\Dropbox\My%20PC%20(LAPTOP-D0497OJ7)\Desktop\Dales%20course%20work\year%201\greg%20program%20fundamentals\codin\Dale%20Smallwood%20Programing%20Fundementals%20assigment%202.docx#_Toc61349689)

[Figure 8 Test description 13](#_Toc61349690)

[Figure 9 Test Expected Results 14](#_Toc61349691)

[Figure 10 Test results and did they work 15](#_Toc61349692)

# Pseudocode and flow chart

## Introduction

Inside this document is the pseudo code and flow charts of the pond calculator with descriptions of the UI and explaining how it functions.

## Admin Login

As shown below (fig1) this pseudo code is so the program can determine that a log in is either successful or unsuccessful. Upon being successful, the current page would close and display the ‘Admin Page’. When unsuccessful a text box would appear saying that the login is invalid.

### Pseudocode

if user is adminuser and password is adminpassword

show the login form

hide previous form

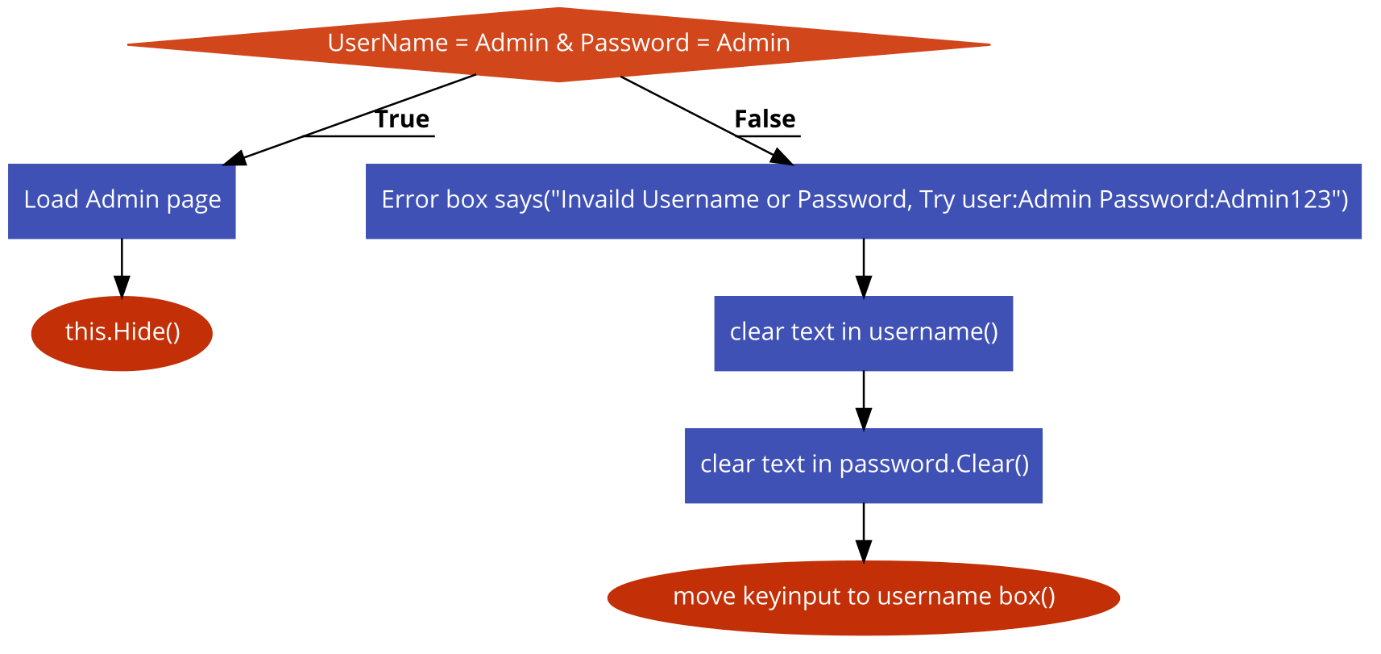
else

Show a message box “invalid username and password”

Clear text input for username and password

Move key input back to username box

Figure Admin Login



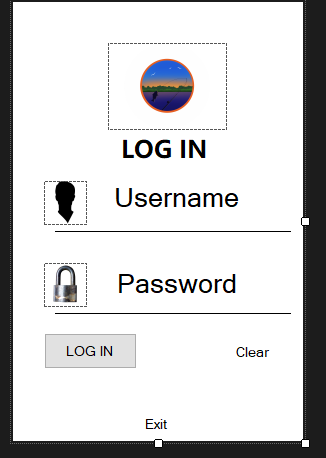


Figure Login UI

User inputs the admin username here

User inputs the admin password here

Clicking this clears the text on the username and password section

Clicking this button checks for the required information

Clicking this exits the application

## Admin Menu

This flowchart (fig3) may look confusing at first glance, but once explained in this description it is extremely easy to understand. The purpose of this code is to prevent the user from being confused and overwhelmed with unneeded information as default the application is displayed as (fig4) where it is limited what can be altered. This provides the user with a list of options upon selecting an option and entering the correct password. The information related to that option will become active (fig5) allowing the user to input or continue to the next form once the progress bar is filled.

### Pseudocode

If NewUser is checked and correct password

Show NewUser section and return other sections to their default UI

If RemoveUser is checked and correct password

Show RemoveUser section and return other sections to their default UI

If None is checked and correct password

Show Continue button and return all sections to their default UI

Figure Admin Menu

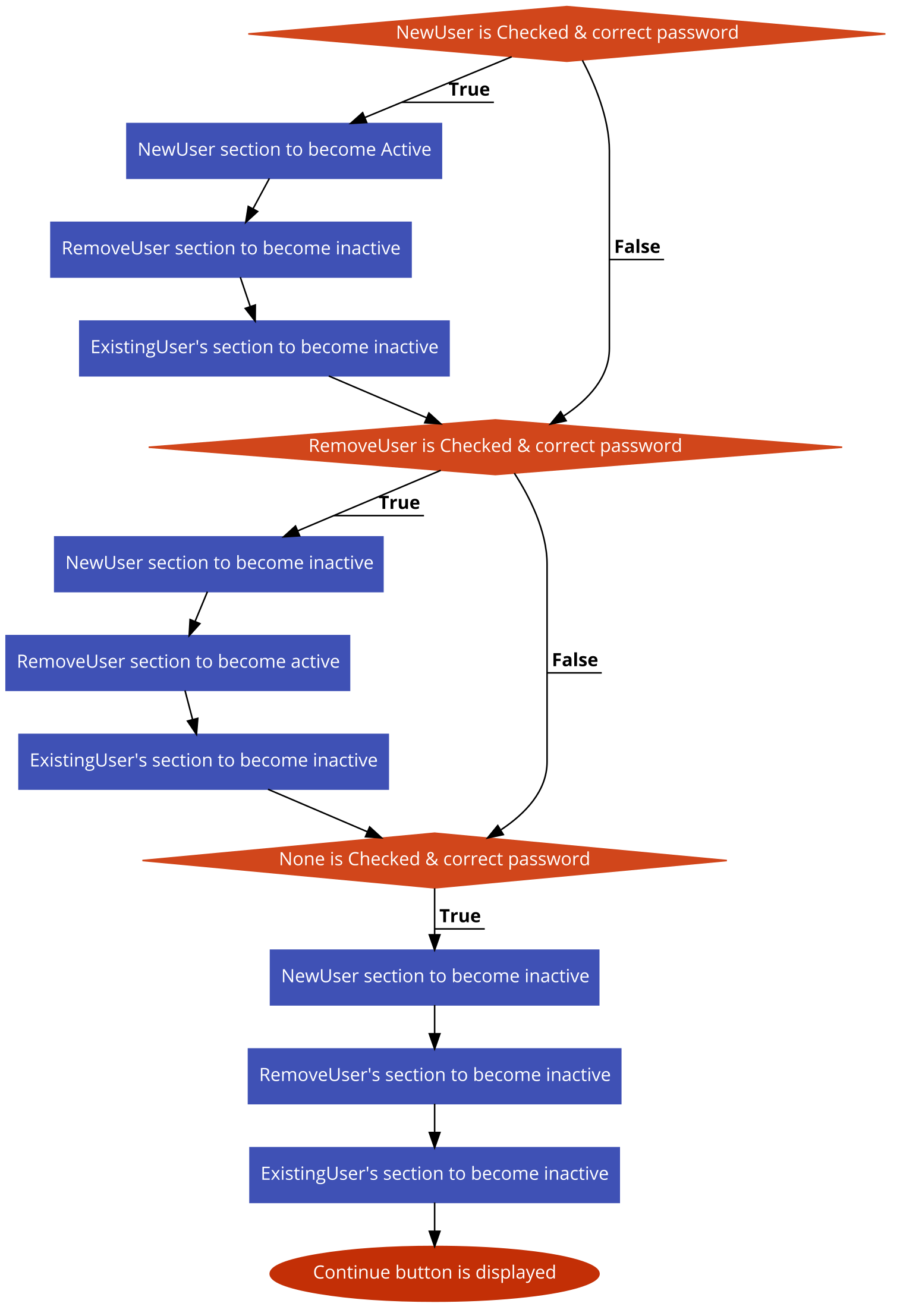
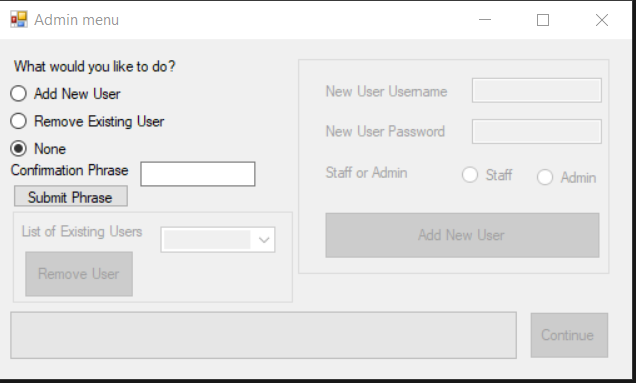


Figure Admin menu default UI



Add New User section displaying information

Required password for input to function

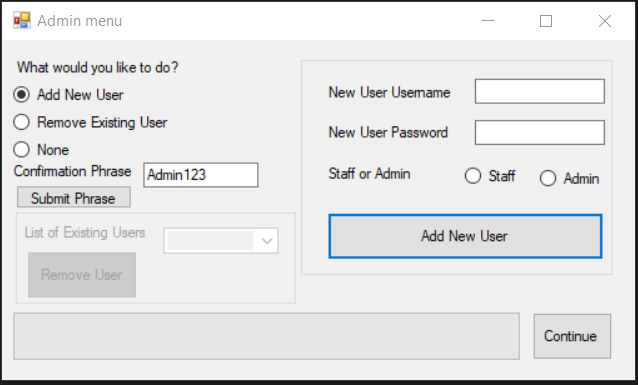


Figure New user section

Select an option to display information

Progress bar this must be full before application allows user to progress.

bar begins to move once the user clicks continue.

Remove User section displayed as currently inactive

Once progress bar is full clicking this button loads Pond Quote

Confirm selection

## Pond calculator measurements

When inputting the length of the pond it must be between one and ten meters. This flow chart (fig 6) demonstrates the validation of the length input (fig 7). This is also used for the width and depth when the requirements of the application progresses. But if the requirements are not met a message box will appear explaining the fault.

### Pseudocode

If pond length box is blank

Show a message box “please enter desired length”

If pond length box is less than 1

Show a message box “length cannot be less than 1 meter”

If pond length is more than 10

Show a message box “length cannot be greater than 10”

If pond length is between 1 and 10

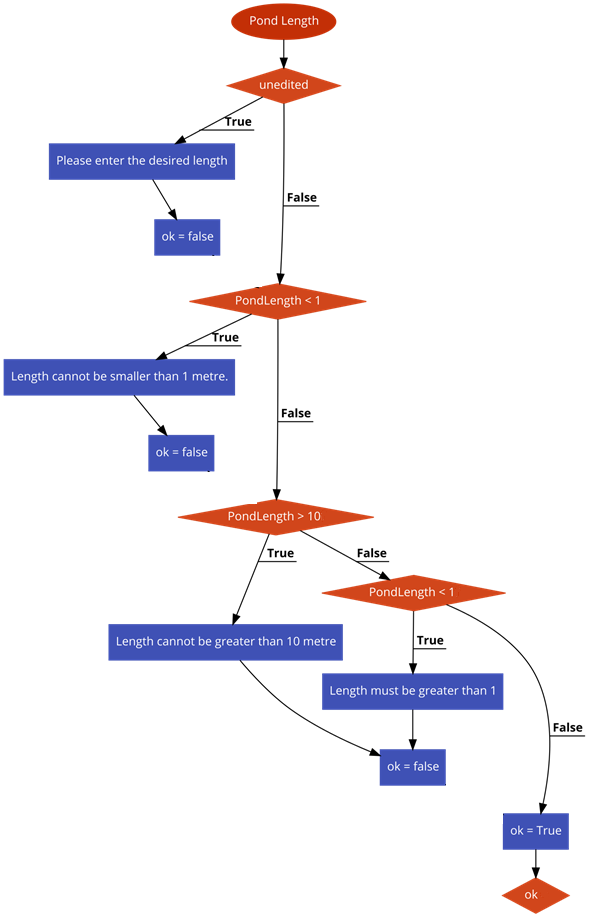
Check width of pond

And repeat the above statements

Check depth of pond

And repeat the above statements

Figure Length validation flow chart



Clear’s and resets all existing information

Drop down box of different pond edges

User inputs measurements

Close application

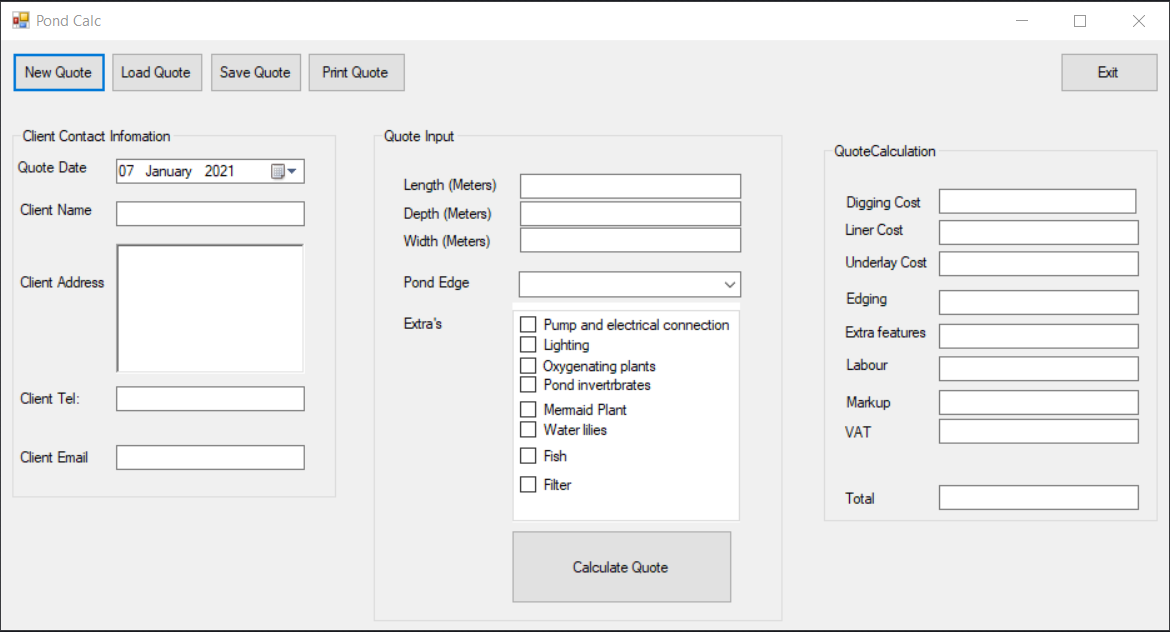


Figure Pond Calc UI

If all validation is approved display costs in quote calculation

All clients contact information to be inputted here

A tick box that adds the cost of features to the quote

Box to display all quote related information

# Test log and Evaluation

After creating a test log on excel and checking the application thoroughly a few issues were discovered along the way and some have not been abled to be resolved at the current moment, they are commented out of the active code and a explanation is along side it.

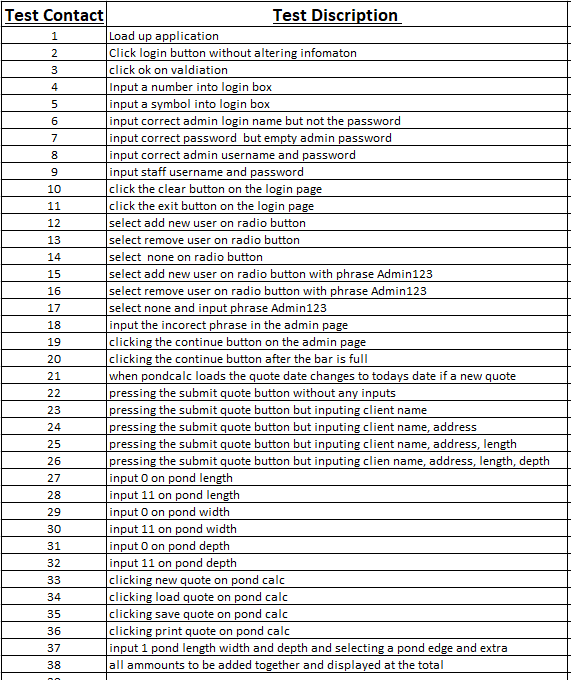
The main issue with the mobile phone and email variable’s is the developer wanted to use C# 8.0 but does not have access to the code at this current time, if the developer did have access to this these variables would not be broken, other than this the application runs smoothly and has extra features such as an admin page.

The admin page contains some dummy sections of code for adding and removing users but does contain a simplistic style that does not overload the user with information along side this it also contains a confirmation password system and a progress bar. The application was tested by a user who also lacks the knowledge of using in depth technology and new systems.

Another issue with the application was saving and loading the application this was attempted and to be stored in a folder but this was not resolved.

## Test log

Figure Test description



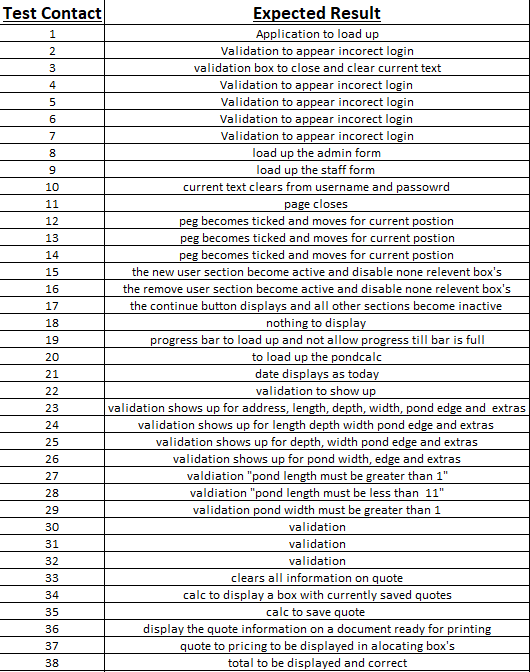


Figure Test Expected Results

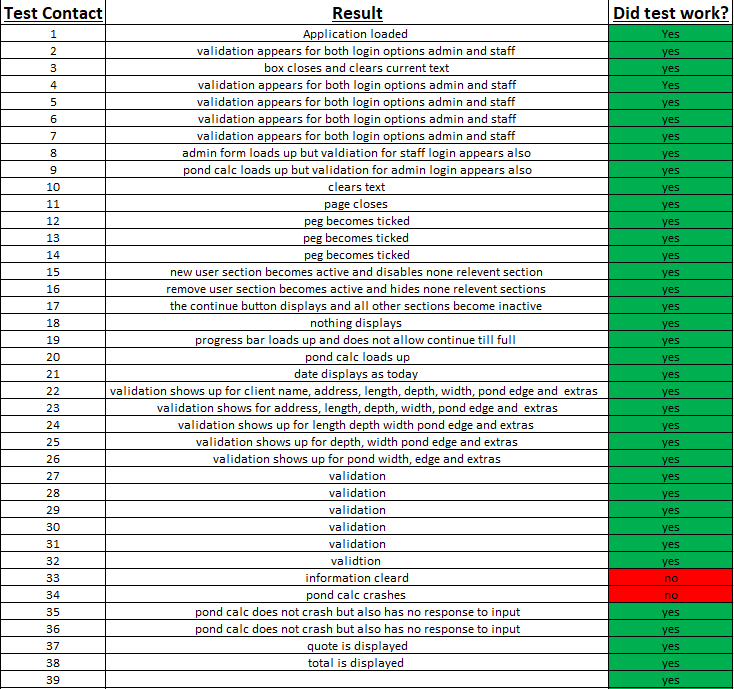


Figure Test results and did they work

# Appendices

Login page

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Pondcalc\_dale\_smallwood\_130089872\_lv4\_2020

{

public partial class Form1 : Form

{

string[] AdminUsers = { "Admin"}; //Admin usernames

string[] AdminPassword = { "Admin123"}; // Admin Sasswords

string[] StaffUsers = {"Dale", "Jeff", "Frank" };//Staff Usernames

string[] StaffPassword = { "Staff123" }; //Staff Passwords

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

if (txtUserName.Text == AdminUsers[0] && txtPassword.Text == AdminPassword[0])

{

new Form2().Show();

this.Hide();

}

else

{

MessageBox.Show("Invaild Username or Password, Try user:Admin Password:Admin123");

txtUserName.Clear();

txtPassword.Clear();

txtUserName.Focus();

}

if (txtUserName.Text == StaffUsers[0] && txtPassword.Text == StaffPassword[0])

{

new PondInput().Show();

this.Hide();

}

else

{

MessageBox.Show("Invaild Username or Password, Try user:Dale Password:Staff123");

txtUserName.Clear();

txtPassword.Clear();

txtUserName.Focus();

}

}

private void Clear\_Click(object sender, EventArgs e)

{

txtUserName.Clear();

txtPassword.Clear();

txtUserName.Focus();

}

private void label2\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void txtUserName\_TextChanged(object sender, EventArgs e)

{

}

}

}

Admin page

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Pondcalc\_dale\_smallwood\_130089872\_lv4\_2020

{

public partial class Form2 : Form

{

string[] SubmitConfirm = { "Admin123" }; //Admin Confermation input

public Form2()

{

InitializeComponent();

}

private void NewUser\_CheckedChanged(object sender, EventArgs e)

{

}

private void Remove\_CheckedChanged(object sender, EventArgs e)

{

}

private void None\_CheckedChanged(object sender, EventArgs e)

{

}

private void PhraseSubmit\_Click(object sender, EventArgs e)

{//Disables uneeded infomation for required actions

if (NewUser.Checked && Confirmation.Text == SubmitConfirm[0])

{

GroupBoxNewUser.Enabled = true;

GroupBoxRemoveUser.Enabled = false;

ExistingUsers.Enabled = false;

RemoveUser.Enabled = false;

SubmitNewUser.Enabled = true;

Continue.Enabled = false;

}

if (Remove.Checked && Confirmation.Text == SubmitConfirm[0])

{

GroupBoxNewUser.Enabled = false;

GroupBoxRemoveUser.Enabled = true;

ExistingUsers.Enabled = true;

RemoveUser.Enabled = true;

SubmitNewUser.Enabled = false;

Continue.Enabled = false;

}

if (None.Checked && Confirmation.Text == SubmitConfirm[0])

{

GroupBoxNewUser.Enabled = false;

GroupBoxRemoveUser.Enabled = false;

ExistingUsers.Enabled = false;

RemoveUser.Enabled = false;

SubmitNewUser.Enabled = false;

Continue.Enabled = true;

}

}

private void Confirmation\_TextChanged(object sender, EventArgs e)

{

}

private void GroupBoxNewUser\_Enter(object sender, EventArgs e)

{

}

private void SubmitNewUser\_Click(object sender, EventArgs e)

{

Continue.Enabled = true;

}

private void GroupBoxRemoveUser\_Enter(object sender, EventArgs e)

{

}

private void RemoveUser\_Click(object sender, EventArgs e)

{

Continue.Enabled = true;

}

private void AdminProgress\_Click(object sender, EventArgs e)

{

}

private void AdminMenuTimer\_Tick(object sender, EventArgs e)

{

this.AdminProgress.Increment(1);

}

private void Continue\_Click(object sender, EventArgs e)

{

this.AdminMenuTimer.Start();

if (AdminProgress.Value >= AdminProgress.Maximum)

{

// Stop timer

AdminMenuTimer.Stop();

new PondInput().Show();

this.Hide();

}

}

private void Form2\_Load(object sender, EventArgs e)

{

}

}

}

Pond Calc

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Drawing.Printing;

using System.IO;

using System.Linq;

using System.Windows.Forms;

namespace Pondcalc\_dale\_smallwood\_130089872\_lv4\_2020

{

public partial class PondInput : Form

{

private PrintDocument printDocument1;

private object printPreviewDialog1;

public PondInput()

{

InitializeComponent();

}

private void Caculate\_Click(object sender, EventArgs e)

{

Boolean alertcheck = false;

bool ok = true;

//client name validation

if (Name.Text == "")

{

MessageBox.Show("Please input clients name");

alertcheck = true;

ok = false;

return;

}

//client adress validation

if (Address.Text == "")

{

MessageBox.Show("Please input clients Address");

alertcheck = true;

ok = false;

return;

}

/\*static bool PhoneNumb(string number)

\* This section is comment'd out as i currently do not have c# 8.0 i am on 7.3

\* but this statment is to check if a moblie phone number is input'd into the pondcalc under the client moblie section.

{

return number[0] == '0' &number[1] == '7' && number.Length == 11 && IsDigit(number);

}

static bool IsDigit(string input)

{

foreach (char c in input)

{

if (c < '0' || c > '9')

return false;

}

return true;

}\*/

string clientEmail = Email.Text;

{

/\* if(ValidEmail(clientEmail))

{

alertcheck = false;

ok = true;

return;

}

else

{

MessageBox.Show("Email is Invalid");

alertcheck = true;

ok = false;

return;

}

/\*private static bool ValidEmail(string clientEmail)

{

try

{

MailAdress mail = new MailAdress(clientEmail);

return true;

}

catch (Exception e)

{

return false;

}

}\*/

}

float PondLength = 0;

if (!float.TryParse(tbPondLength.Text, out PondLength))

{

/\*Validation for length of pond \*/

if (alertcheck == false)

{

MessageBox.Show("Please enter the desired length");

alertcheck = true;

ok = false;

return;

}

}

if (PondLength < 1f)

{

if (alertcheck == false)

{

MessageBox.Show("Length cannot be smaller than 1 metre.");

alertcheck = true;

ok = false;

return;

}

}

if (PondLength > 10f)

{

if (alertcheck == false)

{

MessageBox.Show("Length cannot be greater than 10 metre.");

alertcheck = true;

ok = false;

return;

}

}

else if (PondLength < 1f)

{

MessageBox.Show("Length must be greater than 1");

ok = false;

return;

}

float PondWidth = 0;

if (!float.TryParse(tbPondWidth.Text, out PondWidth))

{

/\*Validation for Width of pond \*/

if (alertcheck == false)

{

MessageBox.Show("Please enter the desired Width");

alertcheck = true;

ok = false;

return;

}

}

if (PondWidth < 1f)

{

if (alertcheck == false)

{

MessageBox.Show("Width cannot be smaller than 1 metre.");

alertcheck = true;

ok = false;

return;

}

}

if (PondWidth > 10f)

{

if (alertcheck == false)

{

MessageBox.Show("Width cannot be greater than 10 metre.");

alertcheck = true;

ok = false;

return;

}

}

else if (PondWidth < 1f)

{

MessageBox.Show("Width must be greater than 1");

ok = false;

return;

}

float PondDepth = 0;

if (!float.TryParse(tbPondDepth.Text, out PondDepth))

{

/\*Validation for Depth of pond \*/

if (alertcheck == false)

{

MessageBox.Show("Please enter the desired Depth");

alertcheck = true;

ok = false;

return;

}

}

if (PondDepth < 1f)

{

if (alertcheck == false)

{

MessageBox.Show("Depth cannot be smaller than 1 metre.");

alertcheck = true;

ok = false;

return;

}

}

if (PondDepth > 10f)

{

if (alertcheck == false)

{

MessageBox.Show("Depth cannot be greater than 10 metre.");

alertcheck = true;

ok = false;

return;

}

}

else if (PondDepth < 1f)

{

MessageBox.Show("Depth must be greater than 1.");

ok = false;

return;

}

if (ok == true)

{

// Mesurment var's

float Perimeter = 0.0f;

float Area = 0.0f;

float Volume = 0.0f;

//calculation

Perimeter = 2 \* (PondLength + PondWidth);

Area = (PondLength \* PondWidth);

Volume = (PondLength \* PondWidth \* PondDepth);

//Dig cost calculation

float DigCost = PondLength \* PondWidth \* PondDepth \* 50.00f;

tbDigCost.Text = "£" + DigCost.ToString("0.00");

//Liner calculation

float Liner = (PondLength + 2 \* PondDepth + 1) \* (PondWidth + 2 \* PondDepth + 1) \* 3.00f;

tbLinerCost.Text = "£" + Liner.ToString("0.00");

//Underlay calculation

float Underlay = (PondLength + 2 \* PondDepth + 1) \* (PondWidth + 2 \* PondDepth + 1) \* 2.00f;

tbUnderlayCost.Text = "£" + Underlay.ToString("0.00");

//Edge calculations

float edgeCost = 0;

//Edge var's

if (PondEdge.Text == "Concrete paving 0.5m x 0.5m") { edgeCost = (Perimeter \* 2) \* 0.75f; }

if (PondEdge.Text == "Stone paving 0.5m x 0.5m") { edgeCost = (Perimeter \* 2) \* 2.20f; }

if (PondEdge.Text == "Stone paving 0.5m x 0.75m") { edgeCost = (Perimeter \* 2) \* 3.50f; }

if (PondEdge.Text == "Plain concrete 0.5m") { edgeCost = (Perimeter \* 2) \* 4.00f; }

if (PondEdge.Text == "Brick paving 0.5m") { edgeCost = (Perimeter \* 2) \* 20.00f; }

if (PondEdge.Text == "Decking 0.5m") { edgeCost = (Perimeter \* 2) \* 20.00f; }

if (PondEdge.Text == "Wall 0.5m") { edgeCost = (Perimeter \* 2) \* 60.00f; }

// else if (PondEdge.Text) "" { MessageBox.Show("Please select a Edge."); alertcheck = true; ok = false; } <- this var needs fixing

//Edge result

tbEdging.Text = "£" + edgeCost.ToString("0.00");

//Extra features calculation

float ExtraCost = 0;

if (Pump.Checked == true) { ExtraCost += +100f; }

if (Lights.Checked == true) { ExtraCost += +75f; }

if (Filter.Checked == true) { ExtraCost += +20f; }

if (OxygenPlant.Checked == true) { ExtraCost = (Volume \* 5) \* 0.75f; }

if (Lilles.Checked == true) { ExtraCost = (float)Math.Floor(Area \* 2); }

if (Invert.Checked == true) { ExtraCost = (float)Math.Floor(Area \* 3); }

if (Mermaid\_Plant.Checked == true) { ExtraCost = Perimeter \* 4 \* 2; }

if (Fish.Checked == true) { ExtraCost = Volume \* 10 \* 0.5f; }

//Display extra cost

tbExtraCost.Text = "£" + ExtraCost.ToString("0.00");

//labour

float labour = 200f;

tbLabourCost.Text = "£" + labour.ToString("0.00");

//Markup

float Markup = (Liner + edgeCost + Underlay + ExtraCost /100) \* 0.25f ;

tbMarkup.Text = "£" + Markup.ToString("0.00");

//vat

float VAT = (DigCost + Liner + Underlay +ExtraCost + edgeCost +labour + Markup /100) \* 0.20f;

tbVatCost.Text = "£" + VAT.ToString("0.00");

//total

float Total = (DigCost + Liner + Underlay + ExtraCost + edgeCost + labour + Markup + VAT);

tbTotalCost.Text = "£" + Total.ToString("0.00");

}

}

private void Exit\_Click(object sender, EventArgs e)

{

Close();

}

private void PondLength\_TextChanged(object sender, EventArgs e)

{

}

private void PondDepth\_TextChanged(object sender, EventArgs e)

{

}

private void PondWidth\_TextChanged(object sender, EventArgs e)

{

}

private void PondEdge\_SelectedIndexChanged(object sender, EventArgs e)

{

}

//calculations

//Extra's calculation

private void Lilles\_CheckedChanged(object sender, EventArgs e)

{

}

private void Mermaid\_Plant\_CheckedChanged(object sender, EventArgs e)

{

}

private void New\_Click(object sender, EventArgs e)

{

Name.Clear();

Address.Clear();

PhoneNumb.Clear();

Email.Clear();

tbPondLength.Clear();

tbPondDepth.Clear();

tbPondWidth.Clear();

PondEdge.ResetText();

ExtraBox.ResetText();

tbDigCost.Clear();

tbEdging.Clear();

tbExtraCost.Clear();

tbLabourCost.Clear();

tbVatCost.Clear();

tbTotalCost.Clear();

tbLinerCost.Clear();

tbUnderlayCost.Clear();

tbMarkup.Clear();

}

private void Name\_TextChanged(object sender, EventArgs e)

{

}

private void DigCost\_TextChanged(object sender, EventArgs e)

{

}

private void ExtraTotal\_TextChanged(object sender, EventArgs e)

{

}

private void Save\_Click(object sender, EventArgs e)

{

// Output.add()

}

private void Load\_Click(object sender, EventArgs e)

{

string FilePath = @"C:\Users\dales\Dropbox\My PC (LAPTOP-D0497OJ7)\Desktop\Dales course work\year 1\greg program fundamentals\codin\Pond Calc lv4 infomation input.txt";

// string[] Input = File.ReadAllLines(FilePath);

List<string> Input = new List<string>();

Input = File.ReadAllLines(FilePath).ToList();

foreach (string line in Input)

{

}

}

private void printDocument1\_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)

{

e.Graphics.DrawString(this.QuoteDate.Text, this.QuoteDate.Font, Brushes.Black, new Point(10, 10));

}

private void Print\_Click(object sender, EventArgs e)

{

// PrintPreviewDialog.Document = printDocument1;

// PrintPreviewDialog.ShowDialog(); printDocument1.PrintPage += printDocument1\_PrintPage;

}

}

}