Carbon Footprint Calculator

IT PAT

Johan Nel – Gr 12-5

HOËRSKOOL SECUNDA

Contents

Please note: This is a planning document. Some changes in the final program might not be reflected in this document.

Task 0: Research

Topic

The topic of my PAT is Save our planet. I will write a program that shows you your carbon footprint, and how you can reduce it. This can help reduce climate change and global warming.

Purpose of program

I will use Delphi 2010 to write a program that will allow admins to change and select the carbon emissions of certain devices, objects and power generating methods, and the normal users to view and update their carbon footprint information. They'll also get tips on how to reduce their carbon footprint if it's too high.

Possible solution

My program will allow users to register to use the program. This will automatically add them to the database. They will then be able to select and use the program to see their carbon footprint. The admins will be able to change, add and update the carbon emissions of certain devices, objects and power generating methods.

Scope

My program won't allow users to sign up and log in on the internet. Users can only access the information on the computer with the database. Information about your carbon footprint may not be 100% accurate.

Task 1: Task definition and Users

Task 1A: Scenario and Scope - Define the Task

People all over the world are unaware of the impact of their activities on our planet. My aim is to code a program that will inform them about climate change, let them calculate their carbon footprint based on electricity use, and then tell them how they can reduce it to ***save our planet***. When they know their carbon footprint, they should be encouraged to reduce it. The program will also give the user tips on how to reduce their carbon footprint.

It will not be difficult to calculate your carbon footprint as a normal user. You will just need to enter values such as electricity units used, or litres of fuel consumed, and your footprint will be calculated for you automatically. The program will be able to calculate these values (units and litres) for you too if you do not know them off the top of your head.

Administrators will be able to set the carbon emission values in the admin side of the program, so that the calculated carbon footprint can be as accurate and consistent as possible. Administrators will also be able to create score sheets and see averages of the carbon footprints of normal users.

Even though this program has a large scope, some functionality will not be implemented at this stage. This includes but is not limited to:

- The program will not be able to send information to external devices such as other computers, web browsers and phones to make it easier for the user to sign up.
- No real-time data will be collected from the internet. The carbon emissions will need to be updated and entered manually by normal users and admins.
- Users cannot edit any of their information or entries after it has been submitted

The program will require me to use the skills I've learnt with Delphi programming to ensure that users can easily calculate their carbon footprint. It will also ensure that the users, and administrators are aware of the overall carbon footprint, and how it can be reduced.

Task 1B: User Requirements

- **User: Administrator**

Description	
Role	The administrator will use this program to log in using username and a password. When logged in, they will be able to use the program to enter the carbon emissions for power generating methods, such as main grid electricity and generators. They will also be able to calculate the average carbon footprint of normal users.
Activity	The administrator will be able to: • Create an account (sign up) • Log in • View data in databases • Enter and change the carbon emissions for generating methods • Calculate and print carbon footprints of users • Retrieve the highest and lowest carbon footprint entry of each user. • Sort and Print the carbon footprints in order

Value	We need administrators to update the carbon emissions for devices, objects and power generating methods, to ensure that the program stays as up to date as possible, accurate and consistent as possible. It will also allow the administrator to retrieve useful information like the highest, lowest and average carbon footprints of normal users.
Limitations	Administrators cannot edit the normal user's personal information. Other limitations are defined in the scope of the project.

- **`User: Normal user`**

Description	
Role	Normal users will use this program to log in using a username and a password. When logged in, they will be able to calculate their carbon footprint by entering values such as electricity units used, or litres of fuel consumed. They will then be able to see their carbon footprint, how it compares to others, and how it can be reduced to help save our planet.
Activity	Normal users will be able to: • Create an account (sign up) • Log in • Enter information to retrieve their carbon footprint • View and update their carbon footprint • See how their carbon footprint compares to the footprint of other users • See how their carbon footprint changes over time • Get tips on how their carbon footprint can be reduced
Value	Normal users are needed to gather data about carbon footprints. When a normal user knows that their carbon footprint is, they can start to reduce it, helping us to save our planet. Without normal users, the program will not have any use. The program makes it easier to view and calculate your carbon footprint.
Limitations	Normal users cannot edit their own account information. They also cannot view other's data or change the carbon emissions for emissions for devices, objects and power generating methods.

- **User: Developer**

Description	
Role	A developer is not a conventional user of the program. They don't use the program at all, but they code it, maintain it, and update its text files. The developer will be the one fixing errors, helping normal users and administrators to use the program, and add new features to the program.
Activity	The developer will be able to: • Maintain the program • Add new features to the program • Update the database and text files (if needed)
Value	Developers maintain the program. Without them, the program will be an abandoned project.
Limitations	Technically, Developers do not use the program. If they were to use the program, their user type would be that of an Administrator.

Task 2: Database Design

The Database is called **CarbonFootprintDB.mdb**. There's an example of the Database in the *Phase 1* folder, and the Database that is used by the Program can be found in the *Main Program Carbon Footprint* folder inside the *Phase 2* folder.

Key:
🔑 Primary Key
🗝 Foreign Key
🔑🔑 *If two Primary keys are used, it is a composite key.*

tblUsers

Field Name	Data Type	
Username	Short Text	The username used to log in
Password	Short Text	The password used to log in
FirstName	Short Text	The user's name
LastName	Short Text	The user's surname
BirthDate	Date/Time	The user's birth date
Email	Short Text	The user's email adress
ContactNo	Short Text	The user's phone number
Country	Short Text	The country the user lives in
Gender	Short Text	M: Male ||or|| F: Female ||or|| U: Unspecified
UserType	Short Text	A: Administrator ||or|| N: Normal User

Field Name	Data Type	Field size	Required?
Username 🔑	Short Text / String	20	Yes
Password	Short Text / String	30	Yes
FirstName	Short Text / String	15	Yes
LastName	Short Text / String	15	Yes
BirthDate	Date	Format: Short Date yyyy/MM/dd	Yes
Email	Short Text / String	25	Yes
ContactNo	Short Text / String	10	Yes
Country	Short Text / String	30	Yes
Gender	Short Text / String	1	Yes
UserType	Short Text / String	1	Yes

Extract:

Username	Password	FirstName	LastName	BirthDate	Email	ContactNo	Country	Gender	UserType
DefaultUser	ｷﾛしへ山回|-|	Default	User	2000/01/01	default@gmail.com	0000000000	South Africa	U	N
JohanNel	ｷﾛしへ山回|-|	Johan	Nel	2005/11/23	jnel677@gmail.com	0662240908	South Africa	M	A
SibuDeBruin	ﾈヨンハフヨフミﾈヨラ	Sibusiso	Ngubane	2005/05/18	Sibu@gmail.com	1234567890	South Africa	F	N

tblEmission

Field Name	Data Type	
EmissionType	Short Text	The type of emission
CO2Emission	Number	How much CO2e is emitted per [unit]

Field Name	Data Type	Field size	Required?
EmissionType 🔑	Short Text / String	50	Yes
CO2Emission	Number / Real	Single	Yes

Extract:

EmissionType	CO2Emission
E Biomass	0.74
E Coal	0.82
E Eskom	1.06
E Geotermal	0.038
E Hydropower	0.024
E Natural gas	0.49
E Nuclear	0.012
E Ocean	0.017
E Solar	0.039
E Wind	0.0115
F First Entry	0

tblGenerator

Field Name	Data Type	
GenName	Short Text	Generator name
GenBrand	Short Text	Generator brand
Type	Short Text	Type of generator (Fuel type)
Capacity	Number	Generating capacity in kW or kVA
EmissionType	Short Text	The type of emission

Field Name	Data Type	Field size	Required?
GenName 🔑	Short Text / String	20	Yes
GenBrand 🔑	Short Text / String	20	Yes
Type	Short Text / String	10	Yes
Capacity	Number / Real	Single	Yes
EmissionType 🗝	Short Text / String	50	Yes

Extract:

GenName	GenBrand	Type	Capacity	EmissionType
BP10S-G	Generac	LP Gas	10	G Generac BP10S-G 10.00 (L)
Afric	MAC	LP Gas	5	G MAC Afric 5.00 (L)
BPD5000L	PAW	Diesel	10	G PAW BPD5000L (D)
BPD5000S	PAW	Diesel	5	G PAW BPD5000S (D)
Diesel Generator	NoBrand	Diesel	7.5	G NoBrand Diesel Generator 7.50 (D)
LP Gas Duel Fuel	RedRhino	LP Gas	3.5	G RedRhino LP Gas Duel Fuel 3.50 (L)

tblElectricity

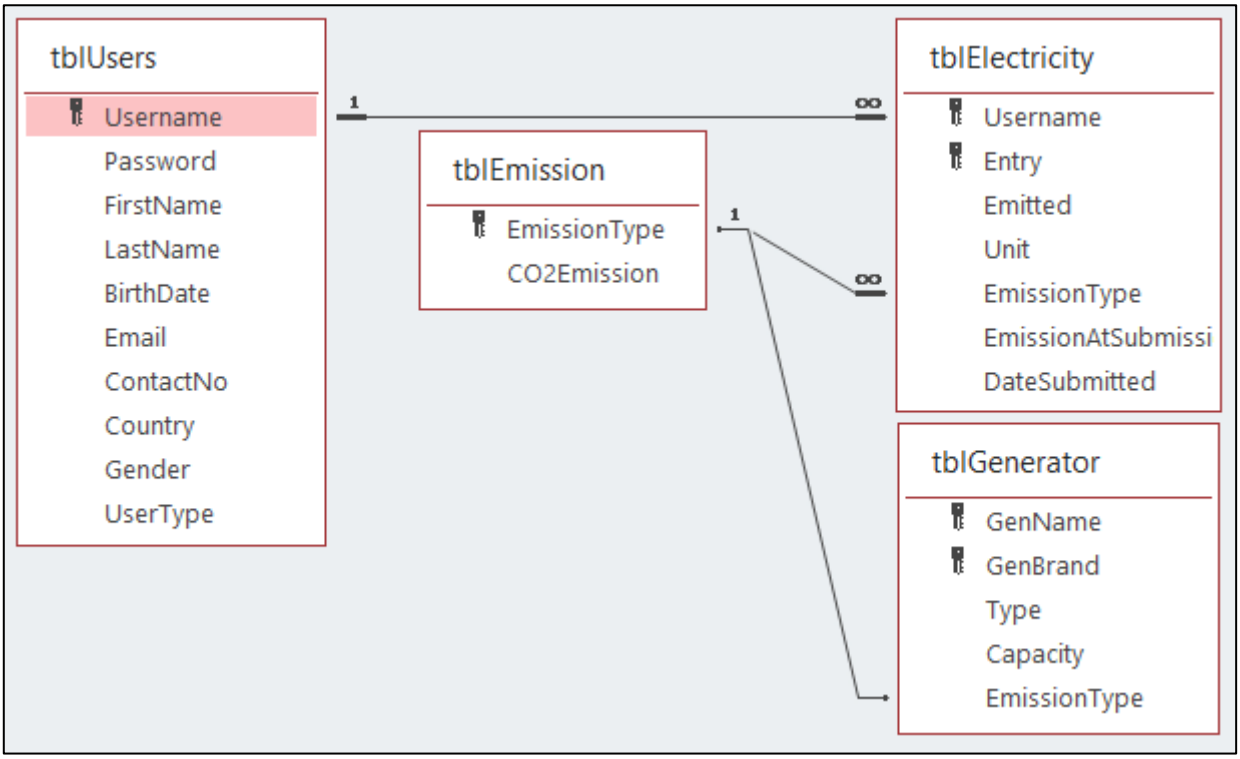
Field Name	Data Type	Descri
Username	Short Text	The user's username
Entry	Number	The number of the entry in the database
Emitted	Number	The amount of CO2e emitted by this entry
Unit	Number	The number of [unit] consumed (Usually Litres or Units)
EmissionType	Short Text	The type of emission
EmissionAtSubmission	Number	The CO2 Emission per [unit] at the date submitted
DateSubmitted	Date/Time	The date of submission

Field Name	Data Type	Field size	Required?
Username 🔑	Short Text / String	20	Yes
Entry 🔑	Number / Integer	Byte	Yes
Emitted	Number / Real	Single	Yes
Unit	Number / Integer	Integer	Yes
EmissionType 🗝	Short Text / String	50	Yes
EmissionAtSubmission	Number / Real	Single	Yes
DateSubmitted	Date	Format: Short Date yyyy/MM/dd	Yes

Extract:

Username	Entry	Emitted	Unit	EmissionType	EmissionAtSubmission	DateSubmitted
DefaultUser	0	0	0	F First Entry	0	2023/07/28
DefaultUser	1	1.06	1	E Eskom	1.06	2023/07/28
DefaultUser	2	2.12	2	E Eskom	1.06	2023/07/06
DefaultUser	3	0.19	5	E Geotermal	0.038	2023/07/28
DefaultUser	4	1.06	1	E Eskom	1.06	2023/07/28

Relationship Description



Username: Primary key 🔑 in tblUsers and Foreign key🗝 in tblElectricity.

EmissionType: Primary key 🔑 in tblEmission, Foreign key🗝 in tblElectricity, and Linked to tblGenerator.

Username & Entry: Composite key 🔑 🔑 in tblElectricity.

GenName & GenBrand: Composite key 🔑 🔑 in tblGenerator.

Database Manipulation

The database will be used and manipulated appropriately to ensure that the program's requirements are met.

- A user record, containing the username, password, name, surname, date of birth, email, contactno, country, gender, usertype – **tblUsers**.
- The program will use **tblUsers** to check if the user entered the correct information when logging in.
- Normal users can view their carbon footprint and how it has changed over time on the User Dashboard. This information is retrieved from **tblElectricity** by the program.
- Normal users can calculate their carbon footprint from Main Electricity by selecting an EmissionType in **tblEmission** (starts with "E").
- Normal users can add a generator to **tblGenerator** on their form.
- Normal users can calculate their carbon foorprint from Generators by selecting an EmissionType from **tblEmissions** (starts with "G"(.
- Administrators can create reports of **tblElectricity**.
- Administrators can edit generators in **tblGenerators**.
- Administrators can add and edit EmissionTypes in **tblEmission**.

Task 3: Data Dictionary

Task 3A: Classes and Objects

- **clsFactLoader | objFactLoader | TFactLoader**

The FactLoader object loads facts from a Fact text file into an array, and then keeps track of which facts have been returned, and which facts have not. It automatically returns a random fact, or a fact in order (depending on function used).

This object is used on Login and Signup screen as fun facts and is also used to give tips to reduce a user's carbon footprint on the Normal user's screen. It cannot retrieve facts or tips online.

Attributes

```
   Constants
-  MaxFacts = 1000;
-  NoFileErrorMsg = 'Error: No Facts Found';

   Variables
-  farrFacts: Array[1..MaxFacts] of String;
-  farrUsedFacts: Array[1..MaxFacts] of Boolean;
-  farrHeadings: Array[1..MaxFacts] of String;
-  fFactCount: Integer;
-  fLastUsed : Integer;
```

Methods

```
+  constructor Create(sFile: String);
   //This is used to create the object.
   //sFile specifies the file name where the facts will be loaded
   // from

+  function GetFact: String;
   //This function retrieves a random fact from farrFacts,
   // and returns it.
   //Facts will only be repeated once all facts have been returned.

+  function GetNextFact: String;
   //This function retrieves a fact in order from farrFacts,
   // and returns it. A random fact will not be returned.
   //Facts cannot be repeated.

+  function GetNextHeading: String;
   //This function retrieves the heading from farrHeadings,
   // and returns it.

+  function GetFactCount: Integer;
   //Returns the value of fFactCount
```

What processing will be done AND How will processing be done?

```
constructor Create(sFile: String);YearsBetween(dDOB, Date);
var
  FFactFile : TextFile;
  sFact : String;
  k, iBut : Integer;
begin
  //Assign File
  AssignFile(FFactFile,'Facts/'+sFile+'.txt');

  //Test if the file Exists
  if FileExists('Facts/'+sFile+'.txt') = False then //Show Error if File does not
  Exist
  begin
    iBut := MessageDlg('Could not find Fact file! Continuing without
    Facts.',mtError,[mbOk],0);
    farrFacts[1] := NoFileErrorMsg;
    fFactCount := 1;
    Exit;
  end;//if FileExists = False

  Reset(FFactFile);
```

```
  //Initialise
  fLastUsed := 0;
  fFactCount := 0;

  //Load Facts into Arrays
  if Pos('Facts',sFile) <> 0 then //If file name contains 'Facts', no headings
  will be used
  begin
    while (NOT(EOF(FFactFile))) AND (fFactCount < MaxFacts) do
    begin
      ReadLn(FFactFile, sFact);
      inc(fFactCount,1);
      farrHeadings[fFactCount] := '';
      farrFacts[fFactCount] := sFact;

    end;//while
  end//if
  else //If the file name does not contain 'Facts', headings will be used
  begin
    while (NOT(EOF(FFactFile))) AND (fFactCount < MaxFacts) do
    begin
      ReadLn(FFactFile, sFact);
      if (Length(sFact) <> 0) AND (sFact[1] = '*') then
      begin
        inc(fFactCount,1);
        farrHeadings[fFactCount] := Copy(sFact,2);
      end else
      if (Length(sFact) <> 0) AND (sFact[1] <> '*') then
        farrFacts[fFactCount] := sFact;
    end;//while
  end;//else

  //Change all the Facts to not used
  for k := 1 to fFactCount do
  begin
    farrUsedFacts[k] := False;
  end;//for k

  CloseFile(FFactFile);

end;
```

```
function GetFact: String;
var
  k, iFact : Integer;
  bReset : Boolean;
begin
  iFact := RandomRange(1,fFactCount+1);

  while farrUsedFacts[iFact] = True do //If the Fact is already used, get a new Fac
  begin
    iFact := RandomRange(1,fFactCount+1)
  end;//while

  farrUsedFacts[iFact] := True;
  Result := farrFacts[iFact];//Return value

{$REGION 'Check if all Facts are used'}
  bReset := True;
  for k := 1 to fFactCount do
  begin
    if farrUsedFacts[k] = False then
      bReset := False;
  end;//for

  if bReset = True then
  begin
    for k := 1 to fFactCount do
      farrUsedFacts[k] := False;
  end;//if
{$ENDREGION}
end;
```

```
function GetNextFact: String;
begin
  result := farrFacts[fLastUsed];
end;
```

```
function GetNextHeading: String;
begin
  inc(fLastUsed);
  farrUsedFacts[fLastUsed] := True;
  result := farrHeadings[fLastUsed];
end;
```

```
function GetFactCount: Integer;
begin
  result := fFactCount;
end;
```

- **`clsGenerator | objGenerator | TGenerator`**

The Generator object is used when a generator is selected or created on tbsElectricity. It is used to identify the Generator in the database, and to calculate the number of litres used. It also calculates the amount of CO2e produced, that can be added to their carbon footprint. It relies on the database and user entered information to work. Data cannot be retrieved from the internet.

Attributes

Variables

```
- fName : String;
- fBrand : String;
- fType : String;
- fCapacity : Real;
- fEmissions : Real;
- fConsumed : Integer;
```

Methods

```
+ constructor Create(sName, sBrand , sType : String; rCapacity,
  rEmissions : Real);
  //This is used to create the object using the parameters.

+ Function CalculateEmissions:Real;
  //Calculates the amount of CO2e produced using the fConsumed and
  // fEmissions attributes

+ Procedure SetConsumed(iLitres:Integer);
  //Sets the amount of Litres consumed by setting fConsumed to
  // the value of iLitres

+ Function GetConsumed:Integer;
  //Returns the value of fConsumed

+ Function GetEmission:Real;
  //Returns the value of fEmissions

+ Function CreateEmissionType:String;
  //Creates the emission type that is used in the Database, using
  // the object's attributes

+ Function ToString:String;
  //Returns the object's attributes so that it can be displayed in
  // a richedit

+ Function CalculateLiters(iMin:Integer):Integer;
  //Calculates the amount of litres consumed, using fType and iMin
```

What processing will be done AND How will processing be done?

```
constructor Create(sName, sBrand , sType : String; rCapacity, rEmissions :
Real);
 begin
    fName := sName;
    fBrand := sBrand;
    fType := sType;
    fCapacity := rCapacity;
    fEmissions := rEmissions;
    fConsumed := 0;
 end;
```

```
Function CalculateEmissions:Real;
begin
   result := fConsumed * fEmissions;
end;
```

```
Procedure SetConsumed(iLitres:Integer);
begin
   fConsumed := iLitres;
end;
```

```
Function GetConsumed:Integer;
begin
   result := fConsumed;
end;
```

```
Function GetEmission:Real;
begin
   result := fEmissions;
end;
```

```
Function CreateEmissionType:String;
begin
   result := 'G '+fBrand+' '+fName+' '+FloatToStrF(fCapacity,ffFixed,5,2)+
   ('+fType[1]+')';
end;
```

```
Function ToString:String;
begin
   result := '';
   result := result + 'Name: '+#9+fName;
   result := result +#10+ 'Brand: '+#9+fBrand;
   result := result +#10+ 'Type: '+#9+fType;
   result := result +#10+ 'Capacity: '+#9+FloatToStrF(fCapacity,ffFixed,10,2)+' kVA
   or kW';
   result := result +#10+ 'Emissions: '+#9+FloatToStrF(fEmissions,ffFixed,10,2)+' k
   CO2e per Liter consumed';
end;
```

```
Function CalculateLiters(iMin:Integer):Integer;
var
  rConsumption : Real;
begin
  //For illustration only: Average fuel consumed per kVA per hour, according t
  generator type
  rConsumption := 0;
  if fType = 'Diesel' then
    rConsumption := 0.35;
  if fType = 'Petrol' then
    rConsumption := 0.45;
  if fType = 'LP Gas' then
    rConsumption := 0.4;
  if fType = 'Natural Gas' then
    rConsumption := 0.3;

  result := Ceil(fCapacity * rConsumption * iMin/60);

end;
```

Task 3B: Text Files

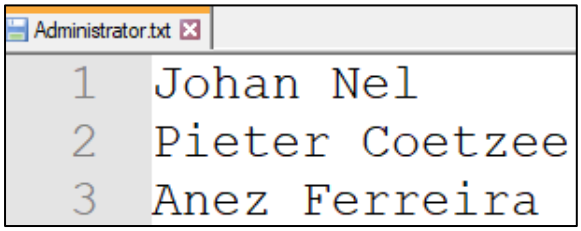
Multiple text files are used in the program to store data for later use and to keep record.

- **Administrator.txt**

  Format: [Name] + [Surname]

  When a user signs up, the program will first loop through the text file. If the name and surname the user entered when signing up corresponds with a name and surname in the text file, it means that they are on the administrator list for the program. When signup is complete, the program will give the user the option to become an administrator or normal user.

  Extract:



- **Countries.txt**

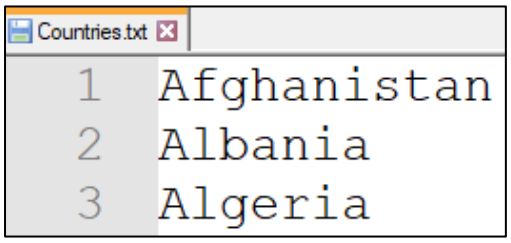
  Format: [Country name]

  This text file stores all country names in the world. When the form is activated, the program loads all the countries contained in this text file into the combo boxes that are used to select the country that you live in. The countries are sorted alphabetically in the combo boxes.

  In case new countries get established, they can be added to this text file to be loaded into the combo boxes as well. The hardcoded limit for the number of countries is 250, but this can be changed by a developer.

  A country's name cannot be longer than 30 characters, as specified in the database.

  Extract:

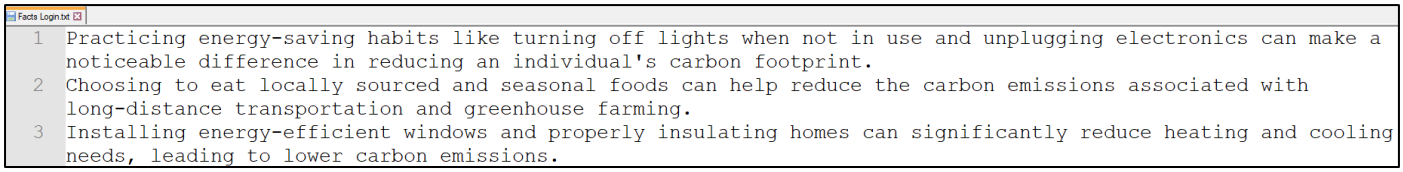


- **Facts\[FactFile].txt**

  Format: [Fact]

  This text file stores fun facts that can be loaded into components on the form. When the form is activated, the program loads all the facts in this text file into an Object (objFactLoader) that will automatically cycle through them on the Main Form (frmSignupLogin). It is also used for the tips that tells Normal users how they can reduce their carbon footprint.

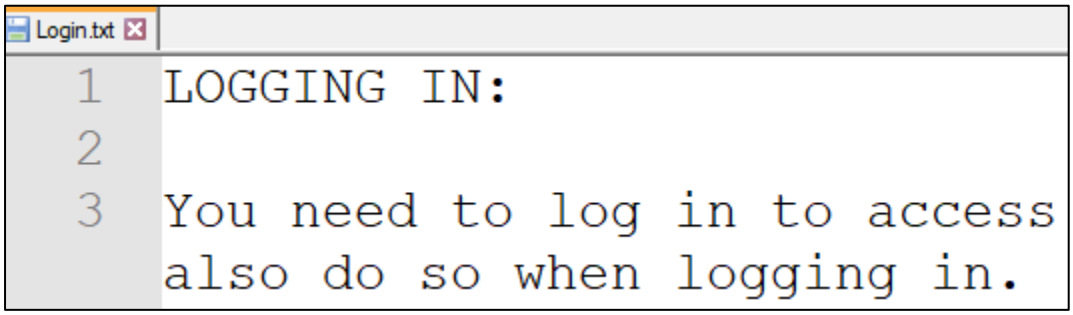
  Extract:



- **Help\[HelpFile].txt**

  These text files are used to load information for the help button. Instead of storing help information in the program code, it is stored in these files. When a help button is clicked, the program will use a procedure to load the text file and display its contents in a MessageDlg. It is used to inform the user on how to use the program when they're unsure.

  Extract:



Task 3C: Arrays

- **Arrays to sort the Countries.txt file**

  Purpose

  To ensure that users (normal and administrator) can easily find the country they live in, the program will always sort the contents of the Countries.txt text file before it loads it into all the item combo boxes.

  Populate the array

  I will use code to retrieve all the country names in the Countries.txt text file, to be loaded into the array.

  Example:

Pos	1	2	3	4	5	…250
arrCountries	Afghanistan	Albania	Algeria	Andorra	Angola	…

  Declaration

```
arrCountries : array[1..250] of String
```

- **Arrays to load and display facts from a Fact text file**

  Purpose

  These arrays are used in ObjFactLoader, to load the facts used on Login and Signup screen as fun facts and the tips on the Normal user's screen.

Populate the array

I will use Facts\[FactFile].txt to populate the arrays.

Example:

Pos	1	2	3	4	5	…MaxFacts
farrFacts	Choosing to…	Opting for…	Investing in…	Educating …	…	MaxFacts
farrHeadings	Reduce…	Use…	Turn off…	Invest…	…	MaxFacts
farrUsedFacts	False	False	False	False	…	MaxFacts

Declaration

```
farrFacts: Array[1..MaxFacts] of String;
farrUsedFacts: Array[1..MaxFacts] of Boolean;
farrHeadings: Array[1..MaxFacts] of String;
```

Task 3C: Subroutines

Multiple *important* subroutines will be used to make repeated code easier to use:

- **Encrypt**

Purpose:

This simple **function** will be used to encrypt and decrypt passwords. Its purpose is to prevent the developers from seeing the user's original password. The function receives the password string as input, and then returns a string of the password encrypted or decrypted.

Copy of code:

```
Result := sPW;
for k := 1 to Length(sPW) do
   Result[k] := chr(NOT(ord(sPW[k])));
```

Explanation:

The line of code is executed for every character of the password. First, it retrieves the character's ASCII value using *Ord* (a built-in function). It then uses the *NOT* operator to 'invert' this value. When you use the *NOT* operator on a number, it reruns the negative of the number, and minuses one. After it 'inverted' the character's ASCII code, it uses *Chr* (a built-in function) to return a new and encrypted or decrypted character. These characters are usually Chinese or Arabic when used for encryption.

Example:

`sPW[k] = A;`

The ASCII value of A is 65. The program performs the NOT operator.

Formula: –(65) –1 = –66

After it 'inverted' the ASCII value, it returns a character which cannot be understood. This new and encrypted character is ̈. The character is now encrypted.

Now, in reverse:

`sPW[k] = ̈;`

In reverse, the same process happens. The ASCII value, -66, gets 'inverted' with NOT.

Formula: – (–66) –1 = 65.

The character corresponding to the ASCII value of 65 character is A. The character is now decrypted.

Declaration:

```
function Encrypt(sPW:String):String;
```

- **Help**

Purpose:

This simple **procedure** is used to load help message from the Help text files. It shows the help message as an Information MessageDlg.

Copy of code:

```
//Test if the help text file is there
if FileExists('Help\'+sFile+'.txt') = False then
begin //if not show error
    iBut := MessageDlg('Help file not found! Contact an admin to fix this
error.',mtError,[mbAbort],0);
    Exit;
end;
  //If it is, assign and reset file pointer
  sMessage := '';
  AssignFile(FHelp,'Help\'+sFile+'.txt');
  Reset(FHelp);
  //generate a string help message
while NOT EOF(FHelp) do
begin
  ReadLn(FHelp,sLine);
  if sMessage = '' then //if it is the first line in the message, don't add
a 'line' in the message
  begin
    sMessage := sLine;
  end//if
  else
  begin
    sMessage := sMessage + #10 + sLine;
  end; //else
  end;//while
  //show the retrieved help message
  iBut := MessageDlg(sMessage,mtInformation,[mbOk],0);
  Closefile(FHelp);
```

Explanation:

When this procedure is called, it will first test if the help text file exists. If it does not, an error message it showed, and the procedure is exited.

If it does exist, the procedure assigns the text file to a variable. It then starts to generate the help message using a while loop. If the message is empty, it will not add a line/paragraph marker. If it contains text, it will add a paragraph marker for each new line that it read.

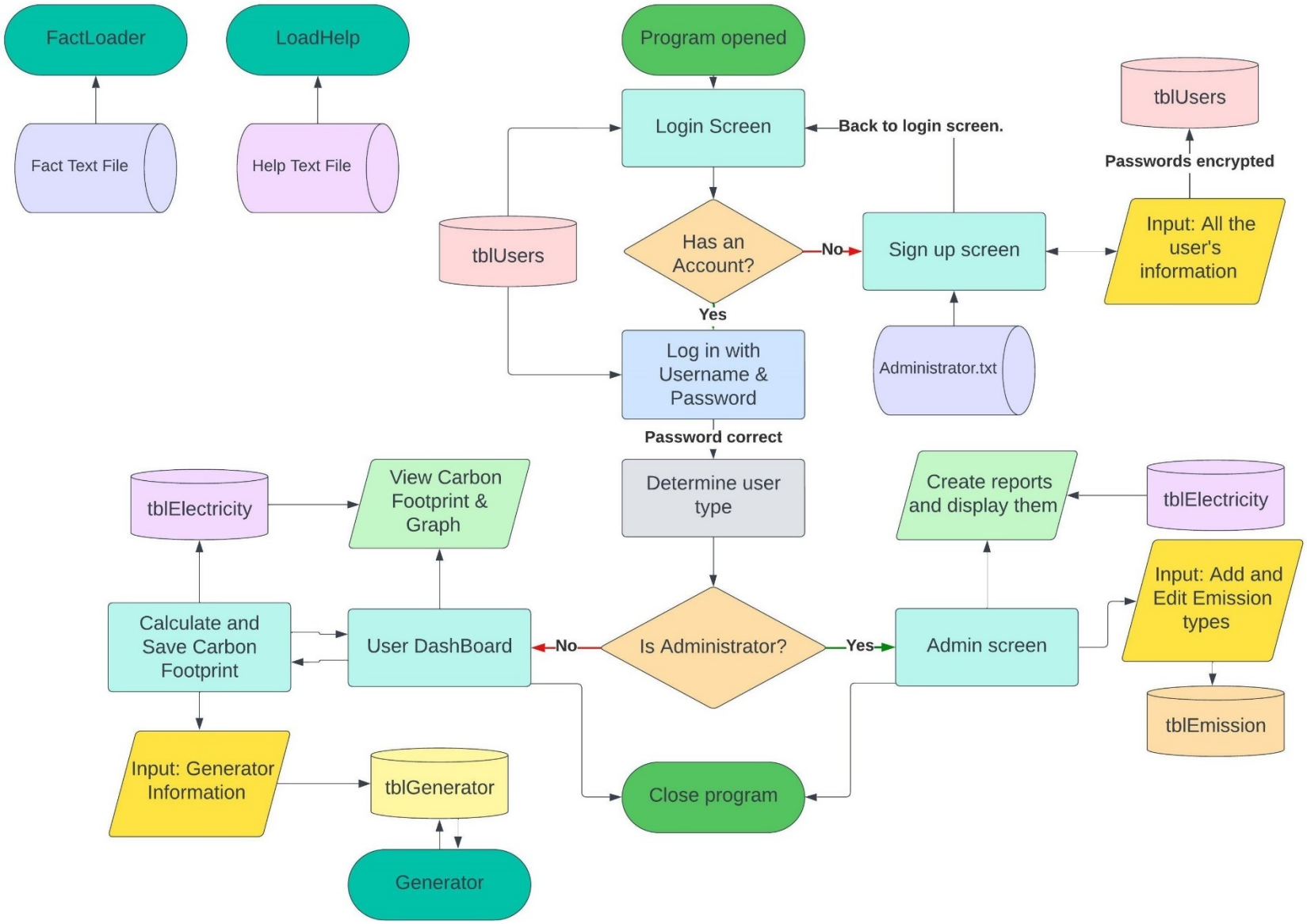
Afterwards it will show the help message with a MessageDlg and close the text file.

Declaration:

```
procedure Help(sFile:String);
```

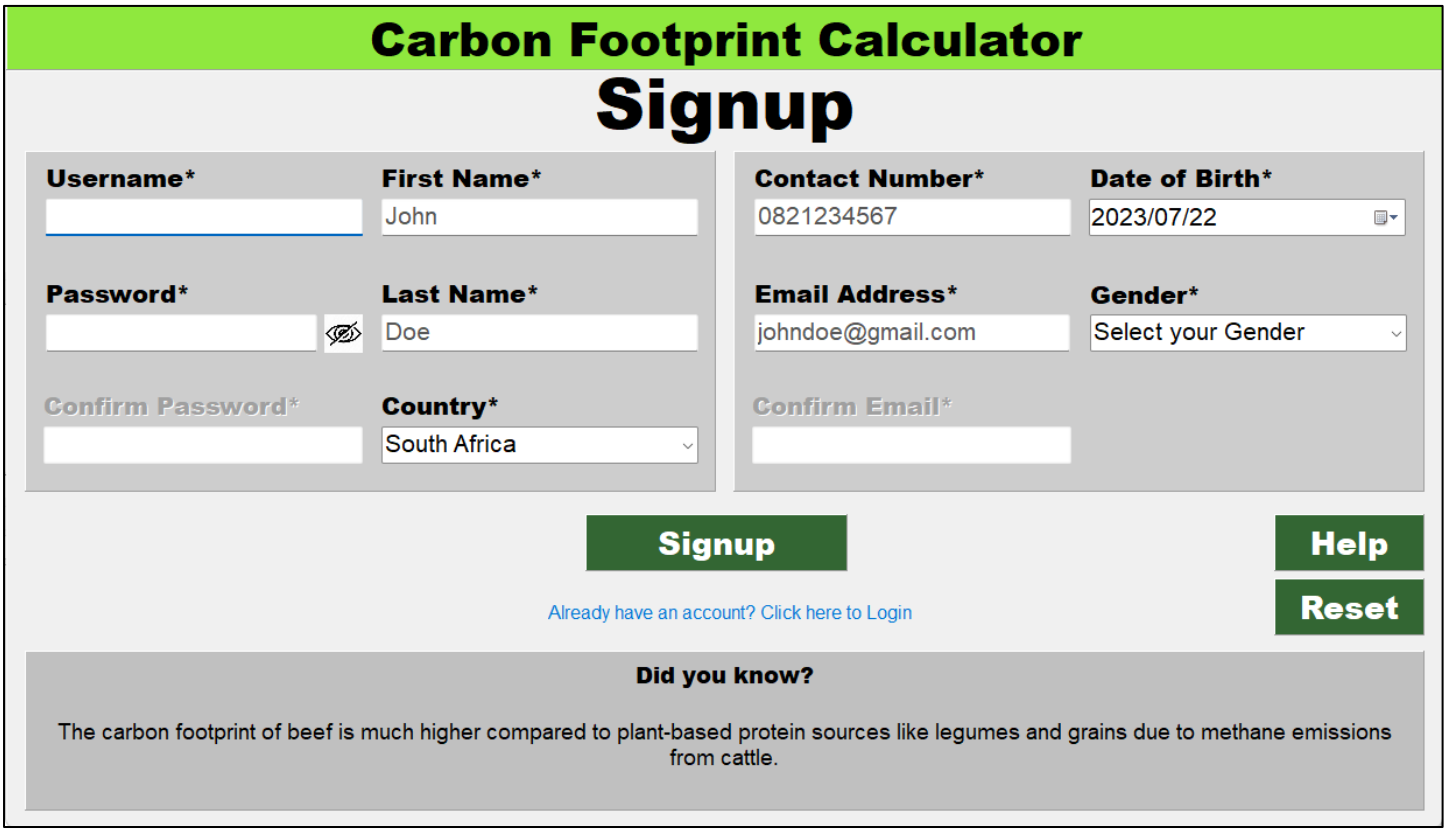
Task 4: Flow Diagram and GUI

Task 4A: Flow Diagram

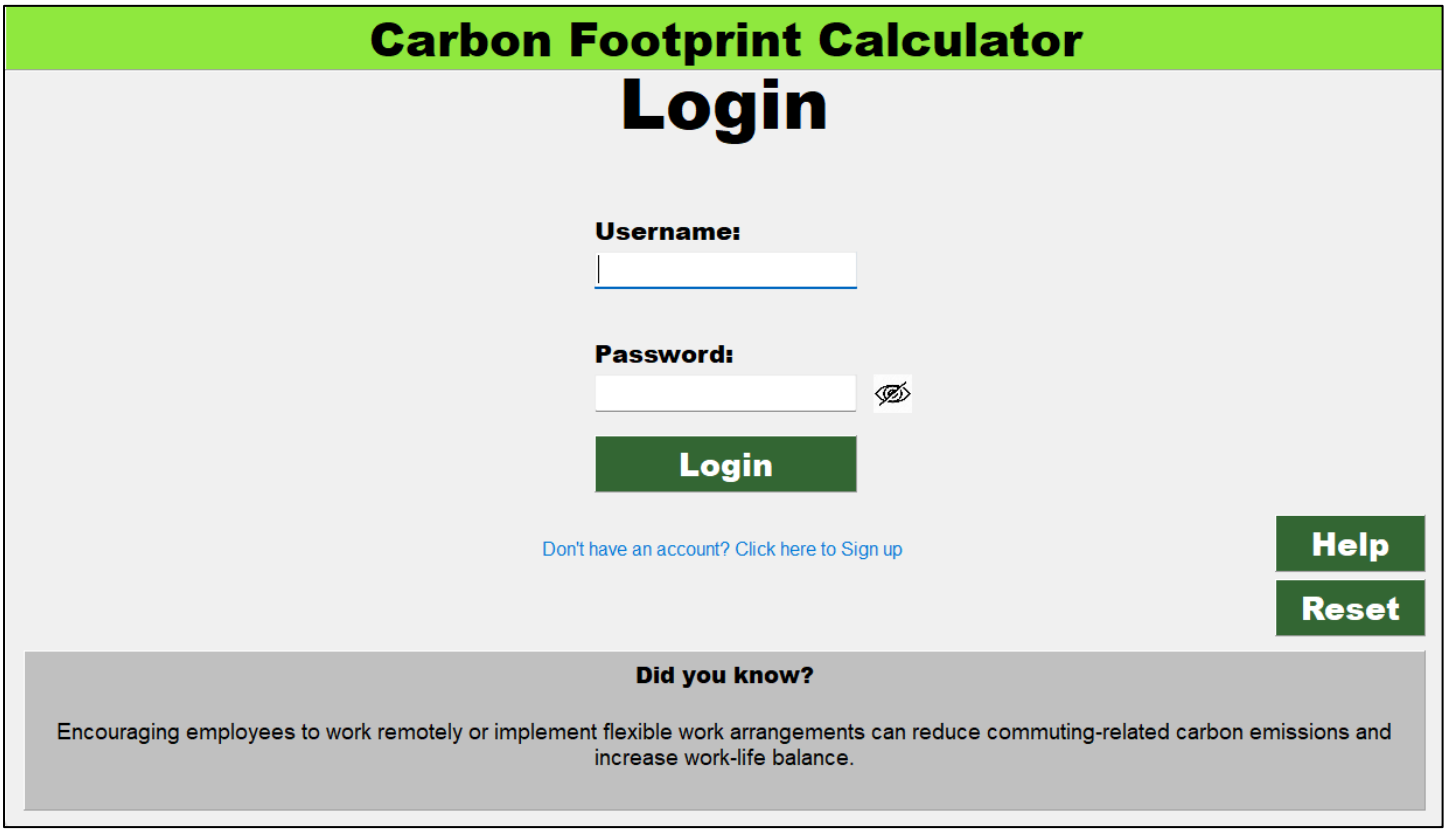


Task 4B: Graphical User Interface (GUI)

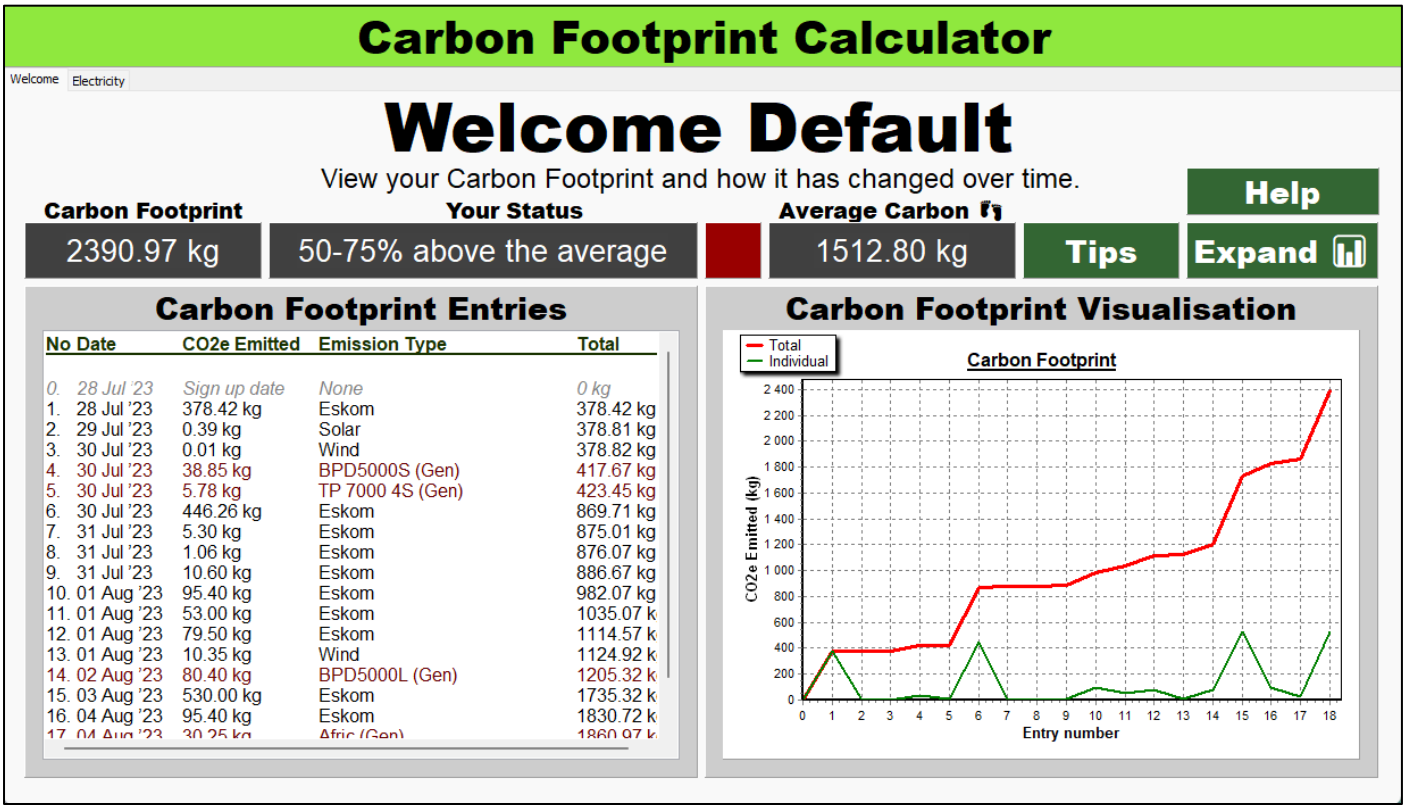
- **frmSignupLogin – pnlSignUp**



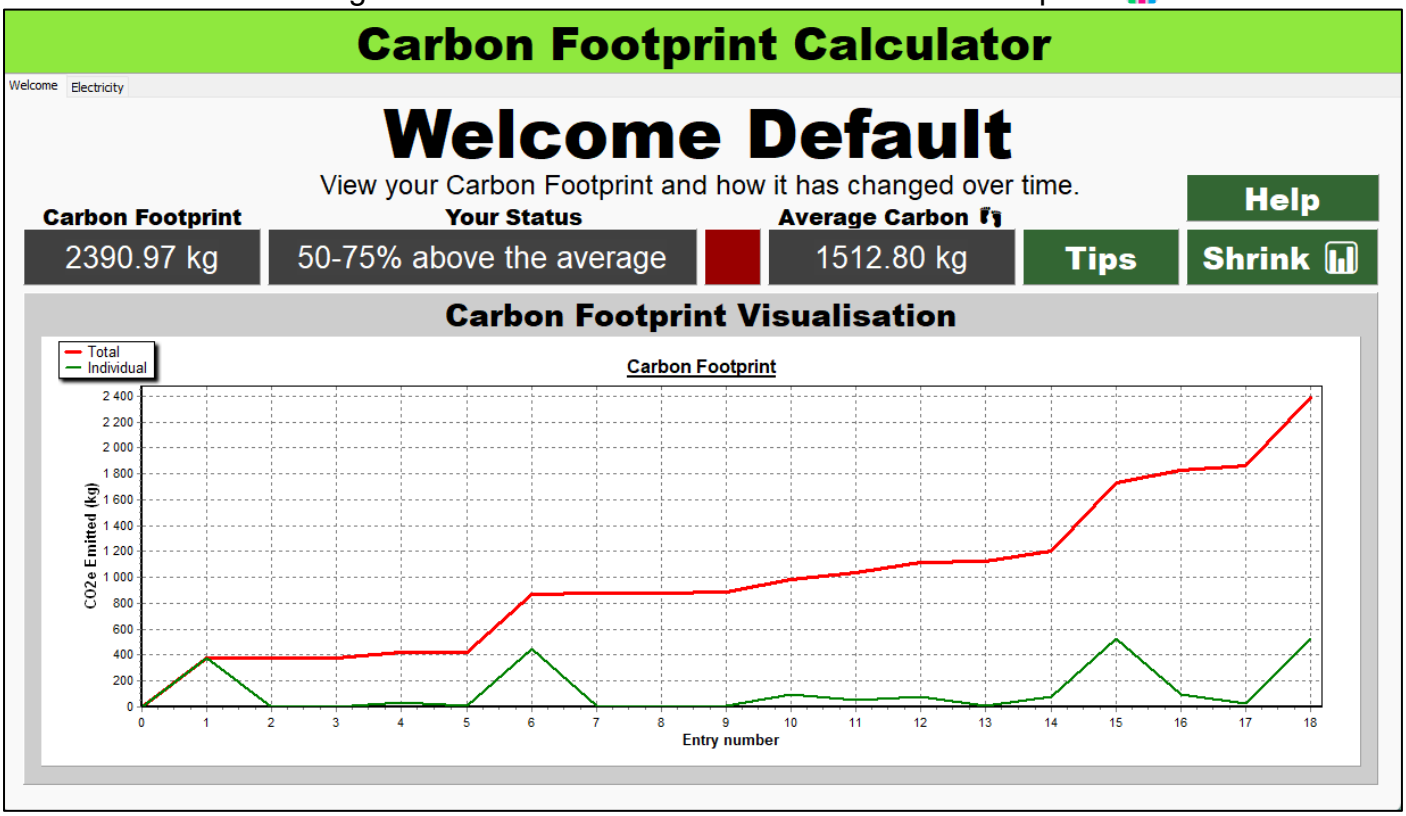
- **frmSignupLogin – pnlLogin**



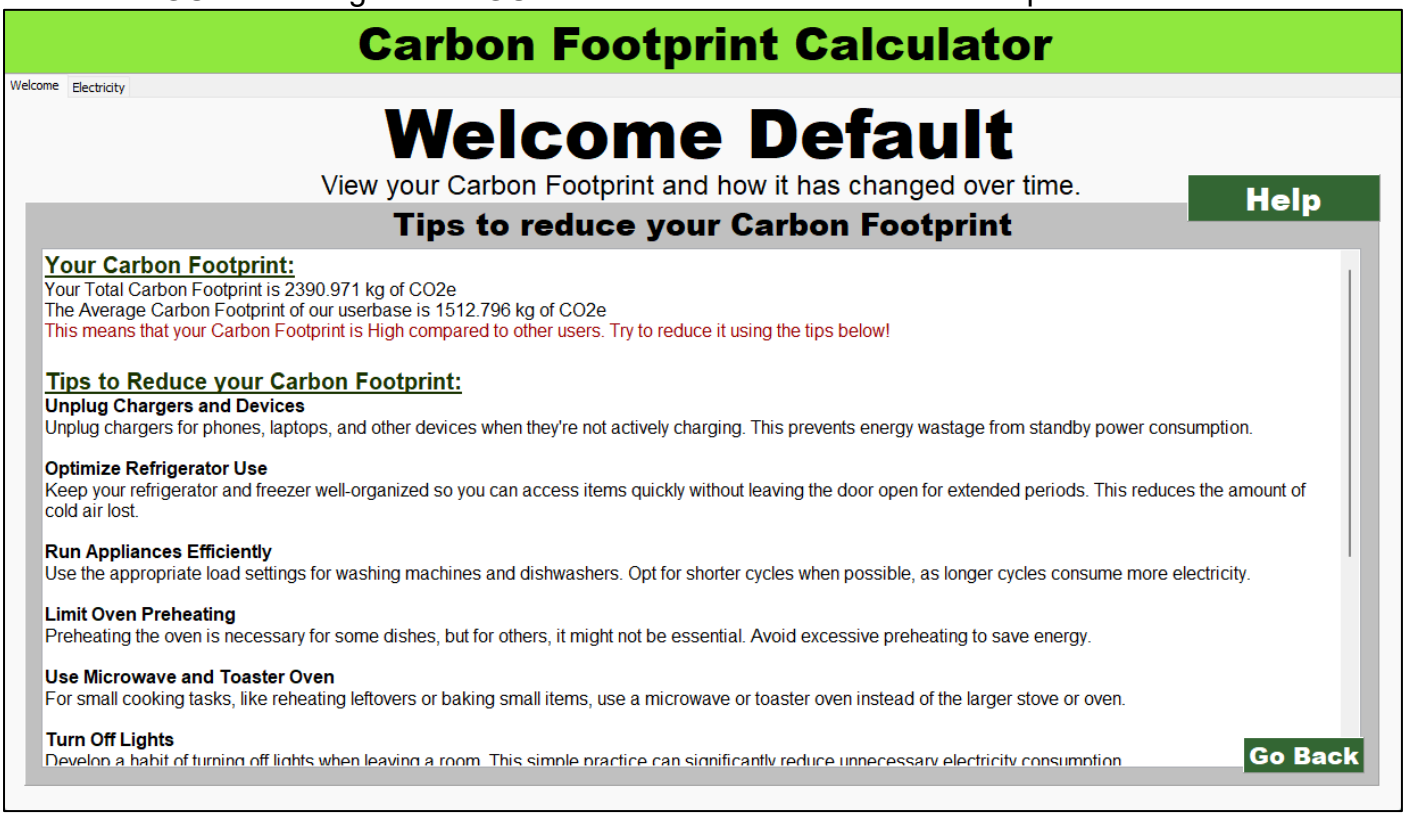
- `frmNormalUser – tbsWelcome`



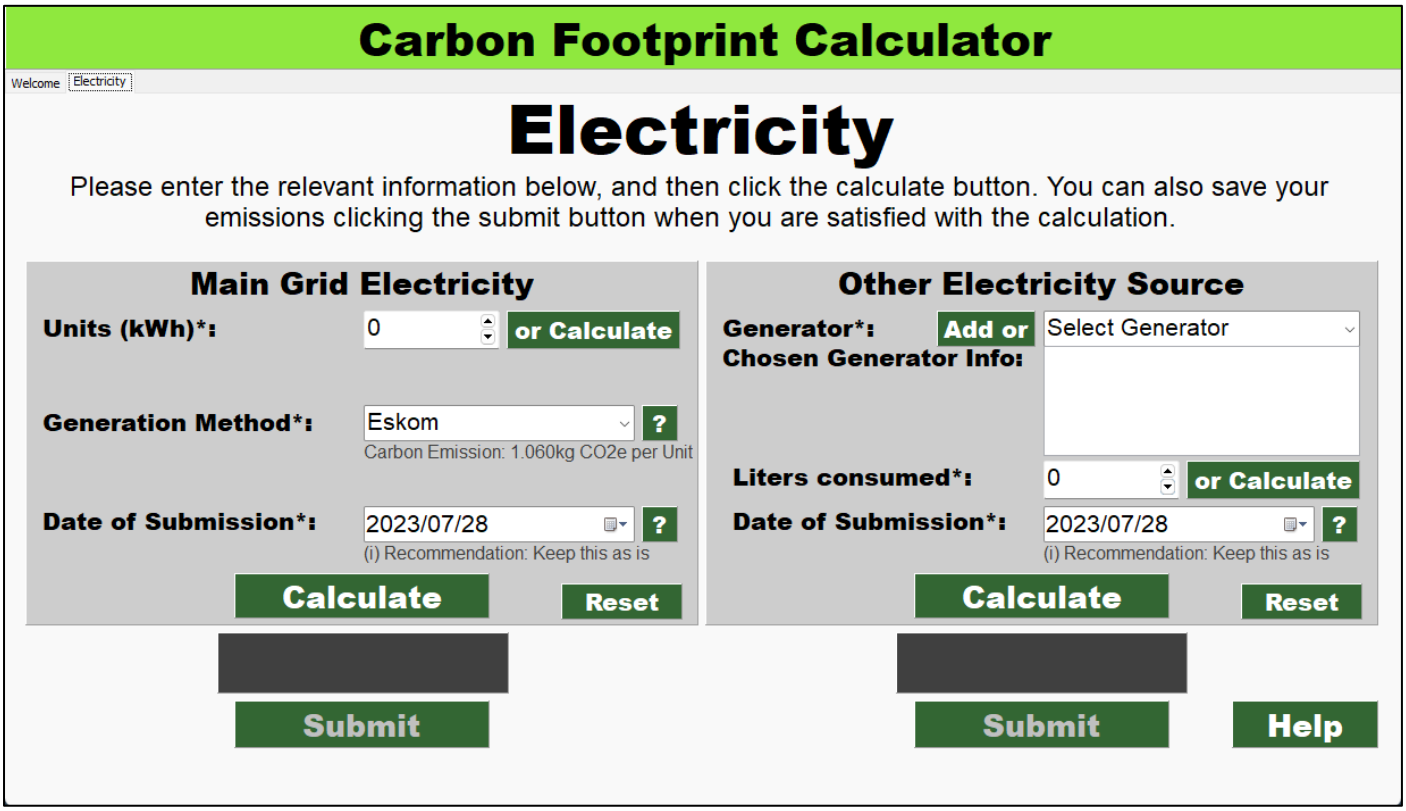
The form's GUI will change to the GUI below when the user clicks the "Expand" button:



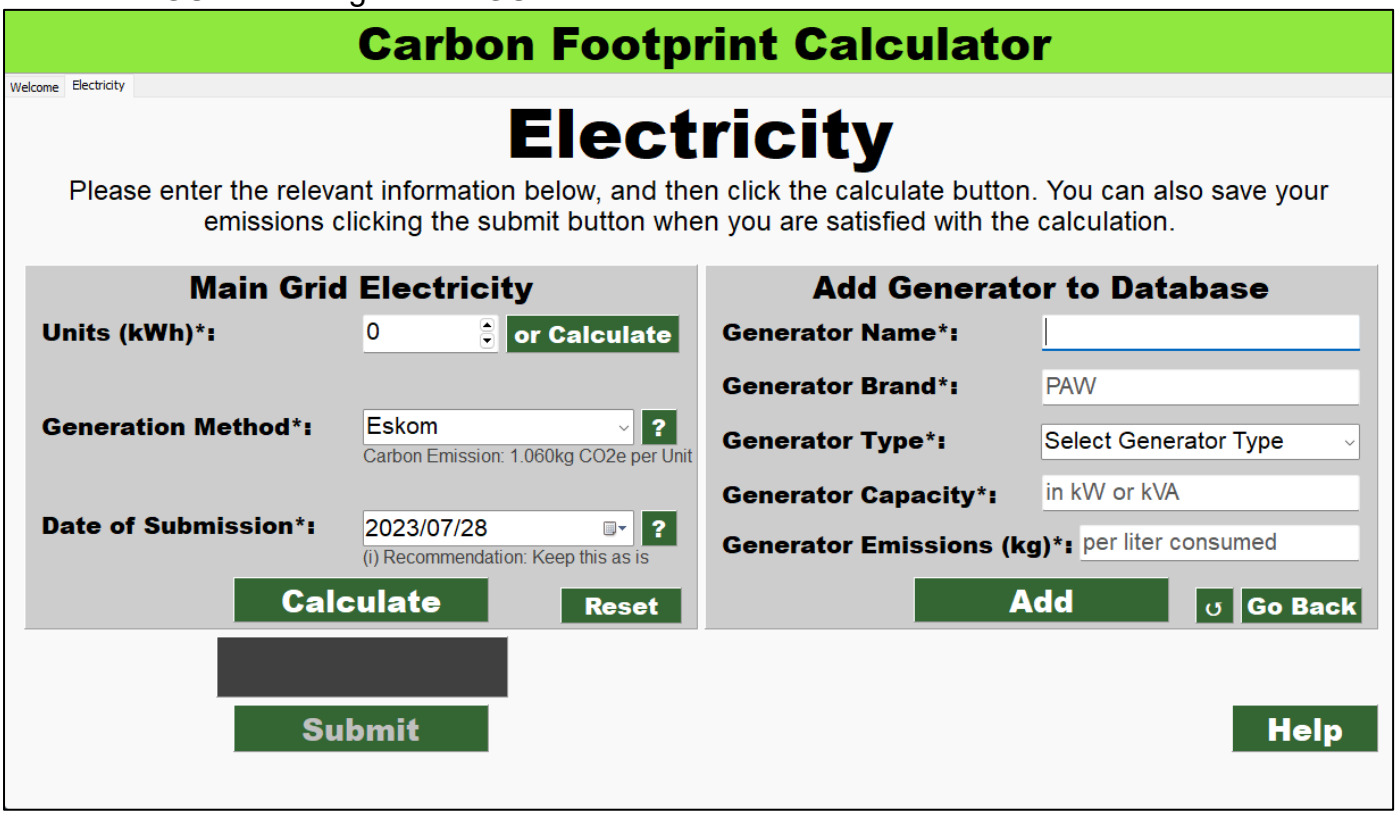
The form's GUI will change to the GUI below when the user clicks the "Tips" button:



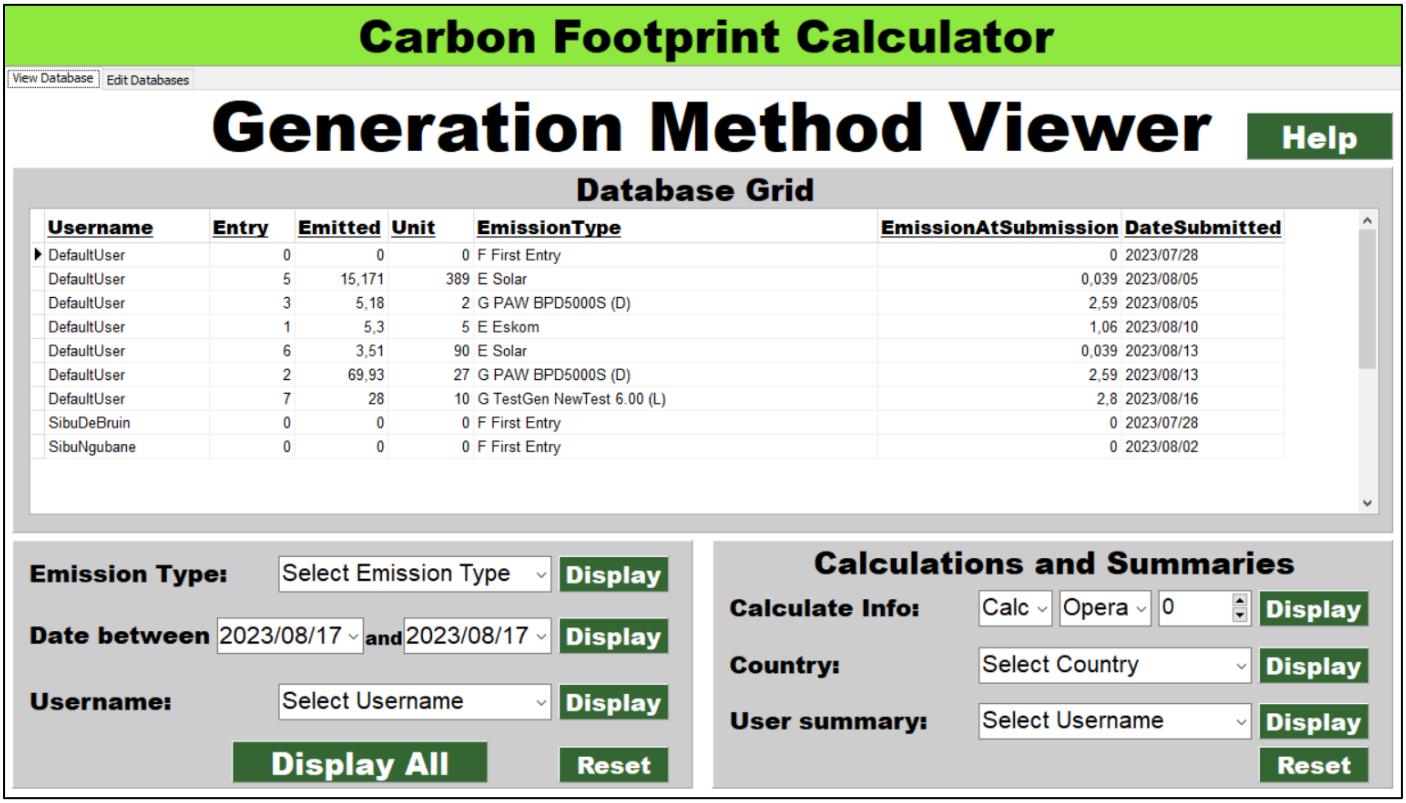
- `frmNormalUser – tbsElectricity`



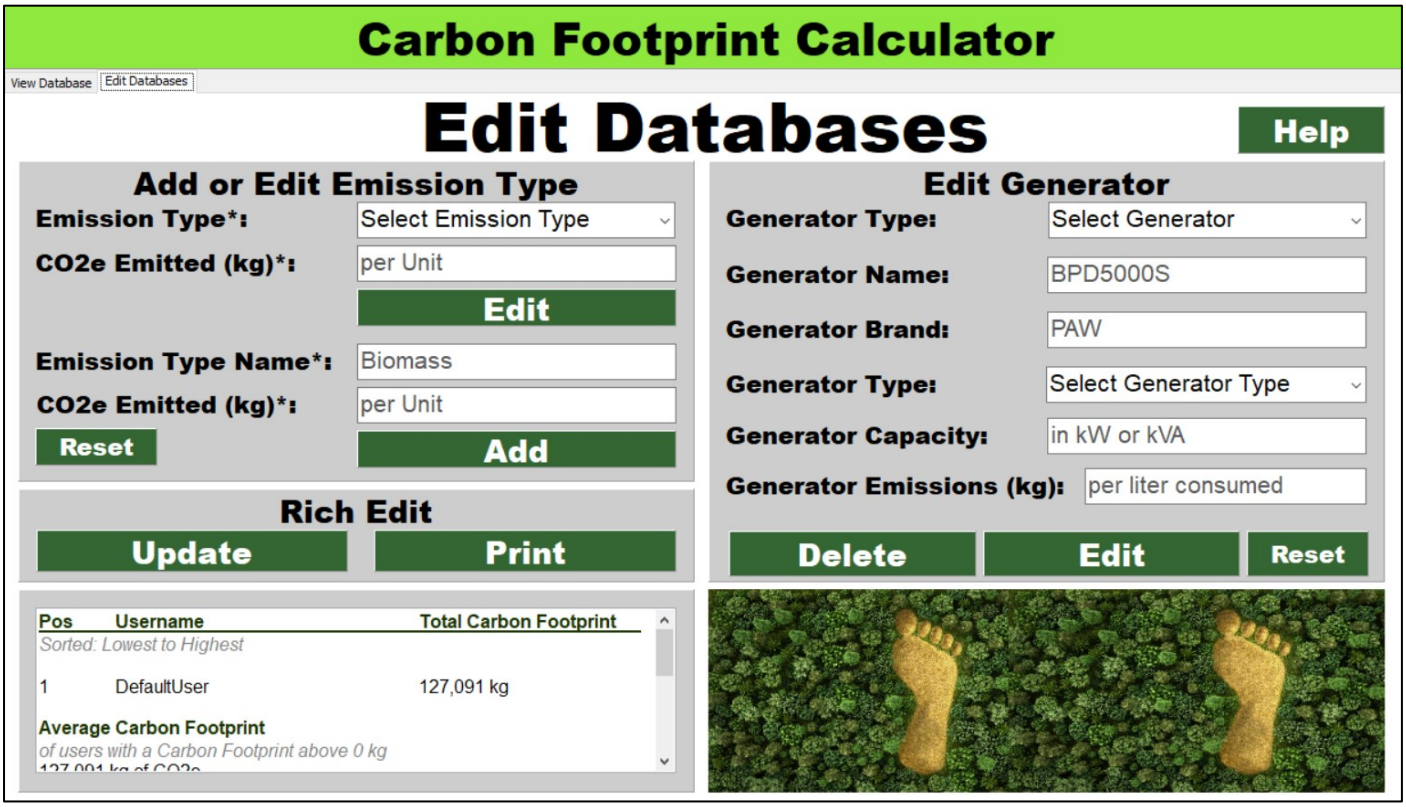
The form's GUI will change to the GUI below when the user clicks the "Add or" button:



- `frmAdmin – tbsGenerationMethod`



- **frmAdmin – tbsGeneration**



Task 5

Sign Up panel:

GUI can be found here.

- **Components and variables**

  Panel:
  pnlTitle – This panel is on all of the forms/panels/tab sheets, and shows the title of the program.

  pnlSignupOne, pnlSignupTwo – These panels house the components used for user input.

  pnlSignupFact – This panel is used to display random fun facts to the user.

  Labels:
  lblGoLogin – This blue label allows the user to go back to the login form/panel if they already have an account. It serves as a button.

  lblSignupCountries, lblSignupDOB, lblSignupGender – Used above the combo boxes and date-time pciker.

  Labelled edits:
  lbledtSignupUsername, lbledtSignupPassword, lbledtSignupPWConfirm
  lbledtSignupFirstName, lbledtSignupLastName, lbledtSignupNumber, lbledtSignupEmail, lbledtSignupEmailConfirm – The user inputs their information in these labelled edits.

  Date-time picker:
  dtpSignupDOB – The user inputs their birth date in this to ensure they input it in the correct format.

  Image:
  imgSignupEye – If the user clicks this image, it changes the password character of the password edit.

  Panel button:
  btnSignup – This panel that serves as a button executes the code that runs when the user signs up. Variables in the OnClick procedure include:

  - **String:** sUsername, sPassword, sPWConfirm, sFirstName, sLastName, sCountry, sContactNo, sEmail, sEmailConfirm, sGender, sLine, sValidError
  - **Date:** dDOB
  - **Integer:** iHigh, iLow, iNumber, iSpecial, iDot, iAt, k, iBut
  - **Boolean:** bValid, bFound, bCheckConfirm, bSpace, bTextFile
  - **TextFile:** FAdmins

- **Input and Validation – btnSignUp**

Input variable	Source	Data type	Format	GUI Component	Validation
sUsername	Keyboard	String	General text (between 6 and 20 chars)	lbledtSignup Username	If sUsername is blank. **Error:** Please enter a username. The username field is required to sign up. To avoid this error, make sure you enter a valid username.

Input variable	Source	Data type	Format	GUI Component	Validation
					If sUsername contains a space. **Error:** Please remove any spaces from the username. The entered username contains one or more spaces, which are not allowed. To avoid this error, please remove any spaces from the username (also check the last digit). If sUsername is shorter than 6 chars. **Error:** The username is too short. The entered username is shorter than the characters required. To avoid this error, please enter a username with at least 6 characters. If sUsername is longer than 20 chars. **Error:** The username is too long. The entered username is longer than the character limit. To avoid this error, please enter a username with a maximum of 20 characters or fewer.
sPassword	Keyboard	String	General text (between 8 and 30 chars)	lbledtSignup Password	If sPassword is blank. **Error:** Please enter a password. A medium or high strength password is required to sign up. To avoid this error, make sure you enter a valid password. If sPassword contains a space. **Error:** Please remove any spaces from the password. The entered password contains one or more spaces, which are not allowed. To avoid this error, please remove any spaces from the password (also check the last digit). If sPassword is shorter than 8 chars. **Error:** The password is too short. The entered password is shorter than the characters required. To avoid this error, please enter a password with at least 8 characters. If sPassword is longer than 30 chars. **Error:** The password is too long. The entered password is longer than the character limit. To avoid

Input variable	Source	Data type	Format	GUI Component	Validation
					this error, please enter a username with a maximum of 30 characters or fewer. If sPassword is low strength **Error:** The password is of low strength. The entered password can be easily guessed or cracked. To avoid this error, make sure the password is at least of medium strength. Medium strength passwords contain at least: 2 Lowercase characters, 2 Uppercase characters, 1 Number, 1 Special character, and 8 characters or longer.
sPWConfirm	Keyboard	String	General text (between 8 and 30 chars)	lbledtSignup PWConfirm	If sPassword and sPWConfirm is not the same. **Error:** The password entered in the confirmation field does not match the password in the password field. To avoid this error, please ensure that both password fields contain the same password. Retype the password in the confirmation field to match the original password exactly (The confirmation field will turn green).
sFirstName	Keyboard	String	General text (between 2 and 15 chars)	lbledtSignup FirstName	If sFirstName is blank. **Error:** Please enter a first name. The first name field is required to sign up. To avoid this error, make sure you enter a valid first name. If sFirstName contains a space. **Error:** Please remove any spaces from the first name. The entered first name contains one or more spaces, which are not allowed. To avoid this error, please remove any spaces from the first name (also check the last digit). If sFirstName is shorter than 2 chars. **Error:** The first name is too short. The entered first name is shorter than the characters required. To avoid this error, please enter a first name with at least 2 characters. If sFirstName is longer than 15 chars. **Error:** The first name is too long. The entered first name is longer

Input variable	Source	Data type	Format	GUI Component	Validation
					than the character limit. To avoid this error, please enter a first name with a maximum of 15 characters or fewer.
sLastName	Keyboard	String	General text (between 2 and 15 chars)	lbledtSignup FirstName	If sLastName is blank. **Error:** Please enter a last name. The last name field is required to sign up. To avoid this error, make sure you enter a valid last name. If sLastName is shorter than 2 chars. **Error:** The last name is too short. The entered last name is shorter than the characters required. To avoid this error, please enter a last name with at least 2 characters. If sLastName is longer than 15 chars. **Error:** The last name is too long. The entered last name is longer than the character limit. To avoid this error, please enter a last name with a maximum of 15 characters or fewer.
sCountry	Mouse and Combo Box	String	General Text (up to 30 chars)	cmbSignup Countries	If no country is selected **Error:** Please select the country you reside in. The country field is required to sign up. To avoid this error, make sure you select a country from the dropdown list.
sContactNo	Keyboard	String	General text (exactly 10 chars)	lbledtSignup Number	If sContactNo is blank. **Error:** Please enter a contact number. The contact number field is required to sign up. To avoid this error, make sure you enter a valid contact number. If sContactNo contains a space. **Error:** Please remove any spaces from the contact number. The entered contact number contains one or more spaces, which are not allowed. To avoid this error, please remove any spaces from the contact number. If sContactNo contains characters that are not between 0 to 9. **Error:** Please remove any other characters from the contact number. The entered contact

Input variable	Source	Data type	Format	GUI Component	Validation
					number contains characters that aren't numbers from 0 to 9. To avoid this error, please remove other characters from the contact number. If sContactNo is not 10 chars. **Error:** The contact number is not the correct length. The entered contact number is not 10 characters in length. To avoid this error, please enter a contact number with 10 characters.
sEmail	Keyboard	String	General text (between 6 and 25 chars)	lbledtSignup Email	If sEmail is blank. **Error:** Please enter an email address. The email address field is required to sign up. To avoid this error, make sure you enter a valid email address. If sEmail contains a space. **Error:** Please remove any spaces from the email address. The entered email address contains one or more spaces, which are not allowed. To avoid this error, please remove any spaces from the email address (also check the last digit). If sEmail is shorter than 6 chars. **Error:** The email address is too short. The entered email address is shorter than the characters required. To avoid this error, please enter an email address with at least 6 characters. If sEmail is longer than 25 chars. **Error:** The email address is too long. The entered email address is longer than the character limit. To avoid this error, please enter an email address with a maximum of 25 characters or fewer. If sEmail is does not have a full stop AND does not have an '@'. **Error:** The email address requires an "@" symbol and a "." (full stop). The entered email address does not contain an "@" symbol and does not contain a "." (full stop). To avoid this error, please enter an email address that

Input variable	Source	Data type	Format	GUI Component	Validation
					contains an "@" symbol and a "." (full stop). If sEmail is does not have a full stop. **Error:** The email address requires a "." (full stop). The entered email address does not contain a "." (full stop). To avoid this error, please enter an email address that contains a "." (full stop). If sEmail is does not have an '@'. **Error:** The email address requires an "@" symbol. The entered email address does not contain an "@" symbol. To avoid this error, please enter an email address that contains an "@" symbol.
sEmailConfirm	Keyboard	String	General text (between 6 and 25 chars)	lbledtSignup EmailConfirm	If sEmail and sEmailConfirm is not the same. **Error:** The email address entered in the confirmation field does not match the email address in the email address field. To avoid this error, please ensure that both email address fields contain the same email address. Retype the email address in the confirmation field to match the original email address exactly (The confirmation field will turn green).
dDOB	Mouse/ Keyboard and DTP	Date	Short Date: yyyy/MM/dd	dtpSignupDOB	If dDOB is the same as today's date. **Error:** Please select a date of birth from the Date-Time picker that is not today's date. Your date of birth is required to sign up. To avoid this error, make sure you select a valid date of birth. If dDOB is in the future. **Error:** Please select a past date from the Date-Time picker. The date of birth selected currently is a future date. To avoid this error, make sure you select a date of birth that has already passed. If the user is younger than 13. **Error:** You must be 13 years or older to use this program. The date of birth selected currently is shows that you are [age] years old. If you are under 13, please

Input variable	Source	Data type	Format	GUI Component	Validation
					ask a parent or gaurdian to complete sign up for you.
sGender	Mouse and Combo Box	String	General Text (1 char) M/F	cmbSignup Gender	If no gender is selected **Error:** Please select your gender. The gender field is required to sign up. To avoid this error, make sure you select a gender from the dropdown list.
sLine	Text file	String	[Name] + [Surname]	None (Text File)	If the Administrator.txt file is not found **Error:** The Administrator.txt file that is used to load the list of administrators could not be found. Unfortunately, this means you will automatically be a normal user when if signup is completed successfully. Contact a Developer to fix this Error.
iBut (Determines if the user wants to be an administrator or normal user)	Mouse clicking button	Integer	Click of a Yes or No button	MessageDlg	If the user's name is found in the Administrator.txt file, the message with Yes and No buttons will be displayed. (No error as user clicks predefined buttons)

- **Processing – btnSignup**

What processing will be done AND How will processing be done?
Calculate age `YearsBetween(dDOB, Date);`

What processing will be done AND How will processing be done?

Add user information to database tblUsers

```
tblUsers.Insert;
tblUsers['Username'] := sUsername;
tblUsers['Password'] := Encrypt(sPassword);
tblUsers['FirstName'] := sFirstName;
tblUsers['LastName'] := sLastName;
tblUsers['BirthDate'] := dDOB;
tblUsers['Email'] := sEmail;
tblUsers['ContactNo'] := sContactNo;
tblUsers['Country'] := sCountry;
tblUsers['Gender'] := sGender[1];
tblUsers['UserType'] := 'N';//Default: Normal user
tblUsers.Post;

tblElectricity.Insert;
tblElectricity['Username'] := sUsername;
tblElectricity['Entry'] := 0;
tblElectricity['Unit'] := 0;
tblElectricity['Emitted'] := 0;
tblElectricity['EmissionType'] := 'F First Entry';
tblElectricity['EmissionAtSubmission'] := 0;
tblElectricity['DateSubmitted'] := DateOf(Today);
tblElectricity.Post;
```

What processing will be done AND How will processing be done?

Test if the user is an administrator

```
bFound := False;
if bTextFile = True then
begin
    while (NOT EOF(FAdmins)) AND (bFound = False) do
    begin
        readln(FAdmins,sLine);
        if sLine = sFirstName+' '+sLastName then
        begin
            iBut := MessageDlg('Since your name was found in the Administrator.txt
            file, you have the option to become an Administrator.'+' Choosing this
            option will always take you to another form with other functions and
            advanced features. Select ''Yes'' if you wish to be an Administrator,
            or ''No'' if you prefer to as a Normal
            User.',mtConfirmation,[mbYes,mbNo],0);
        if iBut = mrYes then
        begin
            dtblUsers.Edit;
            tblUsers['UserType'] := 'A';
            tblUsers.Post;
            iBut := MessageDlg('Your User type is now
            ''Administrator''.',mtInformation,[mbOk],0);

        end
        else
            iBut := MessageDlg('Your User type is now ''Normal
            User''.',mtInformation,[mbOk],0);
        end;//if sLine = sFirstName+' '+sLastName
    end;//while (NOT EOF(FAdmins)) AND (bFound = False)
    Closefile(fAdmins);
end;//if bTextFile = True
```

What processing will be done AND How will processing be done?

Clear the Sign up form components
`tlbledtSignupUsername.Clear;`
`lbledtSignupPassword.Clear;`
`lbledtSignupPWConfirm.Clear;`
`lbledtSignupPWConfirm.Color := clWhite;`
`lbledtSignupPWConfirm.Enabled := False;`
`lbledtSignupFirstName.Clear;`
`lbledtSignupLastName.Clear;`
`cmbSignupCountries.ItemIndex := cmbSignupCountries.Items.IndexOf(DefaultCountry);`
`lbledtSignupNumber.Clear;`
`lbledtSignupEmail.Clear;`
`lbledtSignupEmailConfirm.Clear;`
`lbledtSignupEmailConfirm.Color := clWhite;`
`lbledtSignupEmailConfirm.Enabled := False;`
`dtpSignupDOB.Date := Date;`
`cmbSignupGender.ItemIndex := -1;`
 	
`lblUsernameInfo.Visible := False;`
`lblUsernameInfo.Caption := '';`
`lblPasswordInfo.Visible := False;`
`lblPasswordInfo.Caption := '';`
`lblFirstNameInfo.Visible := False;`
`lblFirstNameInfo.Caption := '';`
`lblLastNameInfo.Visible := False;`
`lblLastNameInfo.Caption := '';`
`lblContactNoInfo.Visible := False;`
`lblContactNoInfo.Caption := '';`
`lblEmailInfo.Visible := False;`
`lblEmailInfo.Caption := '';`
`lblAgeInfo.Visible := False;`
`lblAgeInfo.Caption := '';`
 	
`imgSignupEye.Picture.LoadFromFile('imgHide.jpg');`
`lbledtSignupPassword.PasswordChar := '*';`
`bSignupShow := False;`

Change Password edit char type if clicked on eye image
`if bSignupShow = False then//If Password is not visible`
`begin`
`  imgSignupEye.Picture.LoadFromFile('imgShow.jpg');`
`  lbledtSignupPassword.PasswordChar := #0;`
`  bSignupShow := True;`
`end//if bLoginShow = False`
`else`
`begin//If pPassword is visible`
`  imgSignupEye.Picture.LoadFromFile('imgHide.jpg');`
`  lbledtSignupPassword.PasswordChar := '*';`
`  bSignupShow := False;`
`end;//else of if bLoginShow = False`

- **Output - btnSignUp**

Data	Format	GUI Component

sUsername when signed up successfully. User will be taken to pnlLogin.	Text displayed in message: [FirstName], your account with the Username [Username] has been created successfully! Use this username, and the password you used upon signup to Log in. Welcome to the Carbon Footprint Calculator!	MessageDlg of type Confirmation with OK button GUI changes automatically to pnlLogin.
A message if the user's name is found in the Administrators.txt text file	Text displayed in message: Since your name was found in the Administrator.txt file, you have the option to become an Administrator. Choosing this option will always take you to another form with other functions and advanced features. Select "Yes" if you wish to be an Administrator, or "No" if you prefer to as a Normal User. (This doubles as Input)	MessageDlg of type Confirmation with yes and no buttons

Login Panel

GUI can be found here.

- **Components and variables**

  Panel:
  pnlTitle – This panel is on all of the forms/panels/tab sheets, and shows the title of the program.

  pnlFact – This panel is used to display random fun facts to the user.

  Labels:
  lblGoSignup – This blue label allows the user to go back to the sign-up form/panel if they do not have an account. It serves as a button.

  Labelled edits:
  lbledtLoginUsername, lbledtLoginPassword – The user inputs their information in these labelled edits.

  Image:
  imgLoginEye – If the user clicks this image, it changes the password character of the password edit.

  Panel button:
  btnLogin – This panel that serves as a button executes the code that runs when the user attempts to log in. Variables in the OnClick procedure include:

  - **String:** sUsername, sPassword, sValidError
  - **Integer:** iBut, k
  - **Boolean:** bFound, bValid, bUsernameSpace, bPasswordSpace

- **Input and Validation**

Input variable	Source	Data type	Format	GUI Component	Validation
sUsername	Keyboard	String	General text (between 6 and 20 chars)	lbledtLogin Username	If sUsername is blank. **Error:** Please enter a username. The username field is required to log in. To avoid this error, make sure you enter a valid username. If sUsername contains a space. **Error:** Please remove any spaces from the username. The entered username contains one or more spaces, which are not allowed. To avoid this error, please remove any spaces from the username (also check the last digit). If sUsername is shorter than 6 chars. **Error:** The username is too short. The entered username is shorter than the characters required. To avoid this error, please enter a username with at least 6 characters. If sUsername is longer than 20 chars. **Error:** The username is too long. The entered username is longer than the character limit. To avoid this error, please enter a username with a maximum of 20 characters or fewer.
sPassword	Keyboard	String	General text (between 8 and 30 chars)	lbledtLogin Password	If sPassword is blank. **Error:** Please enter a password. The password field is required to log in. To avoid this error, make sure you enter a valid password. If sPassword contains a space. **Error:** Please remove any spaces from the password. The entered password contains one or more spaces, which are not allowed. To avoid this error, please remove any spaces from the password (also check the last digit). If sPassword is shorter than 8 chars. **Error:** The password is too short. The entered password is shorter than the characters required. To avoid this error, please enter a

Input variable	Source	Data type	Format	GUI Component	Validation
					password with at least 8 characters. If sPassword is longer than 30 chars. **Error:** The password is too long. The entered password is longer than the character limit. To avoid this error, please enter a username with a maximum of 30 characters or fewer.

- **Processing – btnLogin**

What processing will be done AND How will processing be done?
Locate the user in the database, and check if passwords match

```
//Processing
  sPassword := Encrypt(sPassword); //Encrypts the password
  bFound := False;
  dmCarbonFootprint.tblUsers.First;

  while NOT(dmCarbonFootprint.tblUsers.EOF) AND (bFound = False) do //Loop through to searc
for username
  begin
    if dmCarbonFootprint.tblUsers['Username'] = sUsername then //If username matches
    begin
      bFound := True;
      if dmCarbonFootprint.tblUsers['Password'] = sPassword then //If password matches
      begin
        //Show welcome message
        iBut := MessageDlg('You have logged in successfully! Welcome
'+dmCarbonFootprint.tblUsers['FirstName']+'.',mtInformation,[mbOk],0);
        sLoggedIn := sUsername;
        //Clear login form
        lbledtLoginUsername.Clear;
        lbledtLoginPassword.Clear;
        tmrLoginFact.Enabled := False;
        imgLoginEye.Picture.LoadFromFile('imgHide.jpg');
        lbledtLoginPassword.PasswordChar := '*';
        bLoginShow := False;
```

What processing will be done AND How will processing be done?

Determine which form using user type, and go to it
`//Go to next form and Check user type`
`frmSignupLogin.Hide;`
`if  dmCarbonFootprint.tblUsers['UserType'] = 'A' then`
`  Admin_u.frmAdmin.Show`
`else`
`  NormalUser_u.frmNormalUser.Show;`
 	
`  //Stop using the FactLoader object`
`objFactLoader.Free;`
`end//if dmCarbonFootprint.tblUsers['Password'] = sPassword`
`else //If password doesn't match`
`  iBut := MessageDlg('The entered password is incorrect for the provided username. Please double-check your password and try again. If you have forgotten your password, you can use the account recovery option.',mtError,[mbAbort],0);`
`end//if dmCarbonFootprint.tblUsers['Username'] = sUsername`
`else //If username doesn't match`
`  dmCarbonFootprint.tblUsers.Next;`
`end;//while NOT(dmCarbonFootprint.tblUsers.EOF) AND (bFound = False)`
 	
`if bFound = False then //If username isn't found in the database`
`begin`
`  iBut := MessageDlg('The entered username was not found in our records. Please ensure that you have entered the correct username or consider signing up for a new account if you don''t have one.',mtError,[mbAbort],0);`
`  Exit;`
`end;//if bFound = False`

- **Output - btnLogin**

Data	Format	GUI Component
sFirstName when logged in successfully. User will be taken to frmAdmin or frmNormalUser.	Text displayed in message: You have logged in successfully! Welcome [FirstName].	MessageDlg of type Information with OK button GUI changes automatically to pnlLogin.

Admin Edit Databases form:

- **Input and Validation – btnEmiEdit**

Input variable	Source	Data type	Format	GUI Component	Validation
rEmm	Keyboard	Real	Real	edtEmiEmitted	If unable to convert rEmm to real value: **Error:** You entered an invalid CO2e Emitted. To avoid this error, Make sure to only enter the CO2e Emitted as a number. Do not use other text such as the symbols, spaces, etc.

Extra information

Credits

- **Mr Long** has helped me with my PAT significantly. A full playlist of his videos can be found here.
- **Delphi basics:** https://www.delphibasics.co.uk/ is a must-use resource for any IT learner. It clearly explains all the properties and variables of functions, and mostly everything in Delphi! Quite handy to help you learn new things. I used Delphi Basics to learn how to use the MessageDlg built-in sub-routine
- **Encrypt:** https://edn.embarcadero.com/article/28325 was used for the Encrypt function.
- **Countries text file:** https://gist.github.com/kalinchernev/486393efcca01623b18d was used for the countries text file that's used in the program.

Delphi plugins and external tools

No third-party plugins or Delphi related external tools were used for the program at the time of writing.

Program documentation

This program includes detailed documentation which can be found here:

Program Documentation.doc

It is recommended that you read through this whole document to ensure you can use the program to its full potential without issues. Most of this document's information has also been loaded into the Help buttons. The program documentation is only for normal Users.

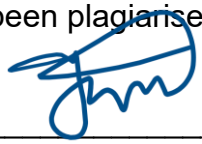
Declaration of authenticity

PRACTICAL ASSESSMENT TASK (PAT)

Learner Name: Ferdinand Johannes (Johan) Nel
Grade 12
Year: 2023
ID Number: 0511235121080

Teacher: A. Ferreira

I hereby declare that the contents of this assessment task are my own original word (except where there is clear acknowledgement and appropriate reference to the work of others) and have not been plagiarised, copied from someone else or previously submitted for assessment by anyone.



_____ 18 / 08 / 2023

Signature of learner Date