Complete this "test" as assignment 4 (advance to page 6)

Save our Earth

We need the Earth; the Earth doesn't need us!

You are tasked to write a small C++ application to make your friends aware and to control your electricity usage better. A small **E**lectricity **B**illing (EB-)Tracking System for *multiple* appliances was prepared for you. You must complete the code for each event/ method provided.

The prepared system starts with the following interface:

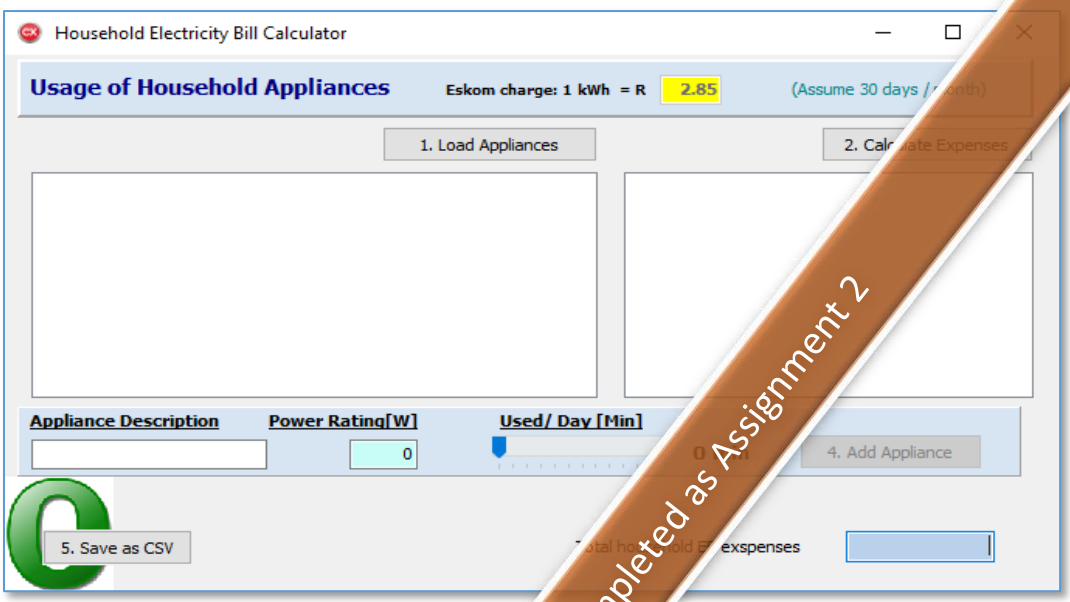


Figure 1: Starting application

Completed as Assignment 2

The system provides you with test data copied into the subfolder **myData**. The file is called: **`House_EBill.txt`**. The file is TAB delimited. A screen print displays the content (Figure 2).

House_EBill.txt - Notepad

File Edit Format View Help

PowerRate[W]	Usage/Day[Min]	Appliance Description
1800	150	Oven
110	120	HiFi
320	180	PC/Laptop
230	240	Lights
2000	30	Kettle
900	20	Microwave

Figure 2: Contents of text file provided viewed through Notepad

URL: https://businesstech.co.za/news/property/346852/how-to-calculate-your-own-electricity-bill-in-south-africa/

Overview

The system tracks **multiple appliances** from your residence. You must complete the following 7 tasks:

1. Simulate counting from "0" to "9".
2. Load the test data into a list box.
3. Click on a listed appliance to display the details on the bottom panel.
4. Create and define a *class method* to be used.
5. Calculate some details for each appliance and sum up the total cost.
6. Add a new appliance with usage to your existing data.
7. Save all data in a CSV file.

Developing the project

1. *Simulate counting from "0" to "9"* *(This challenge can also be answered **any time later**!)* *[8]*

1.1 Define a suitable variable to help count the numbers appearing at the bottom of your interface. Initialize this variable as part of an early firing event. (1)

1.2 The numbers do appear in picture format running from "0" on the far left to "9" on the far right. Compare with Figure 1 displaying the first number "0", and Figure 3 displaying the 8th digit. The component ***imgNum*** is made available to be used for this purpose. Supply the code for the OnTimer-event to simulate the counting/ movement of the single digits in incremental order.

The simulation from 0 to 9 only happens twice (2 * 10 = 20 iterations), then the image component *imgNum* must be hidden and the timer's action de-activated. The bitmap images to be loaded are saved in the subfolder **myData**. (7)

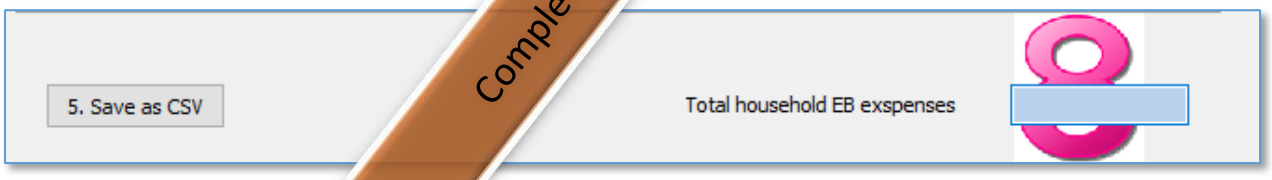


Figure 3: Movement of single digits - showing the 8th digit for only 0.15 seconds

2. *Load the test data into a list box* *[2]*

2.1 Supply the code for the OnClick-event of button [1. Load Appliances] to load the file **`House_EBill.txt`** into the provided list box *lstAppliances*. Enable the button [4. Add Appliance]. (2)

3. *Click on a listed appliance to display the details on the bottom panel* *[7]*

3.1 An OnClick-event for the list box *lstAppliances* is prepared for you.

Supply the code to extract the high-lighted line's information and display the details further down – see Figure 4. The selected line's information must be analysed.

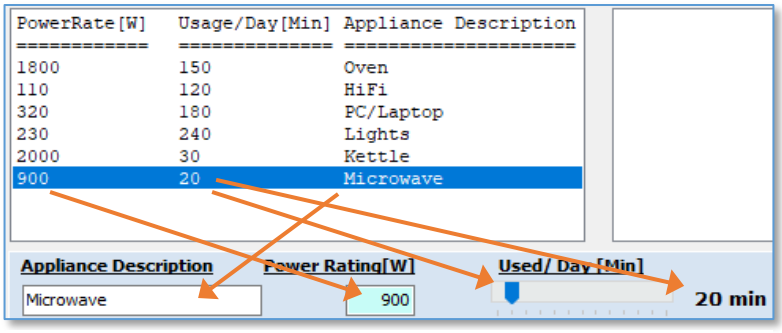


Figure 4: Selected line's information analysed and displayed

Analyse the line ***only*** if it contains data from an appliance! Don't allow the headings to be analysed, if they would have been selected. (7)

4. *Create and define a class method to be used* *[10]*

4.1 A self-defined *class* method must be created for further repetitive use. Name the method: ***CalculateKWexpenses()***. - (KW = kilo Watt) - The method has two parameters:

1st parameter: The average use of the appliance per day in **MINUTES** (as integer).

2nd parameter: The *power rating* of the appliance in WATT (as integer).

The method returns a complete *TAB-delimited* string with typical values as indicated in Figure 5.

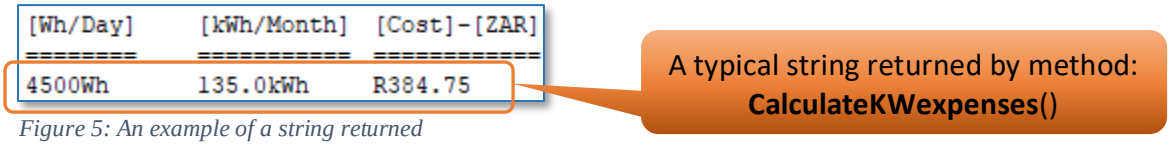


Figure 5: An example of a string returned (see framed line)

Provide the coding for the method and calculate the following values as displayed in Figure 5 (with examples):

- **Watt-hours per day** (how many Watt(s) are consumed by the appliance, if it was used for the indicated time span measured in hours – the value is an average estimate for one day) *Example:* The Oven has a power rating of 1800 Watt. If the oven is used 150 minutes (= 2.5 hours) per day in average, then the [**Wh/Day**] value can be calculated as: 1800 x 2.5 = 4500Wh – compare with the first value in Figure 5.
- **Kilo-Watthours per Month**. Based on the first calculated value, the total average kilo-Watthours can be calculated per month. To simplify our calculations, accept 30 days for every month.
- Based on the indicated charged rate in the edit box *edtKWhCost* (2.85) the total price for electricity usage for that appliance can be calculated now. Our example in Figure 5 indicates R384.75. Make provision that calculations are still correct if the charged rate increases.

Guarantee the number formatting as seen above in Figure 5.

(10)

5. *Calculate some detail for each appliance and sum up the total cost* *[9]*

5.1 Code the contents of the OnClick-event button [2. Calculate Expenses]. It must supply the headings for the list box *lstUsage* and calculate the data for each line correlating with list box *lstAppliances*. Make use of the completed class method in question 4.

Also calculate the total amount for all listed appliances. A possible result is displayed in Figure 6.

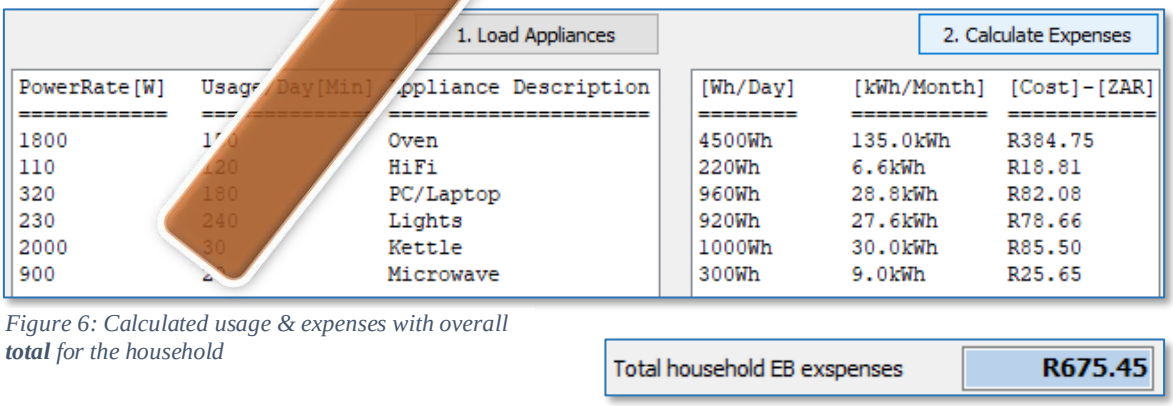


*Figure 6: Calculated usage & expenses with overall **total** for the household*

(9)

Completed as Assignment 2

6. *Add a new appliance to your existing data* *[8]*

6.1 To add another appliance to our list box on the left, the user must complete the expected values as displayed in Figure 7.

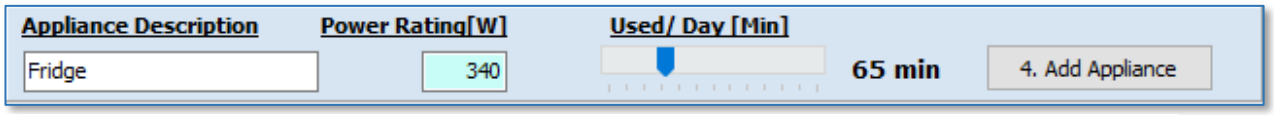
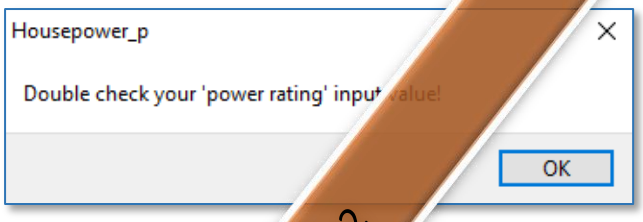


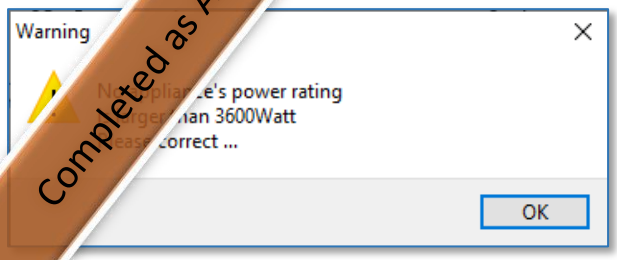
Figure 7: a new appliance is added

6.2 Supply the coding for the OnChange-event of the track bar, for adjusting the value of the label *lblUsageM* accordingly. (1)

6.3 Add the code for the OnClick-event from button [4. Add Appliance]. This event must make use of exception handling. If the field "Power Rating[W]" has any other value than a true integer, like for example the value: **340W**, then a general exception must be thrown and a simple message box must display the following:



6.4 The input value for the field "Power Rating[W]" must also be validated and may not be larger than **3600**. If it is larger than 3600, then respond with a message dialog box as follows:



If no problems are encountered, add the resulting line to the list box on the left. See Figure 8.

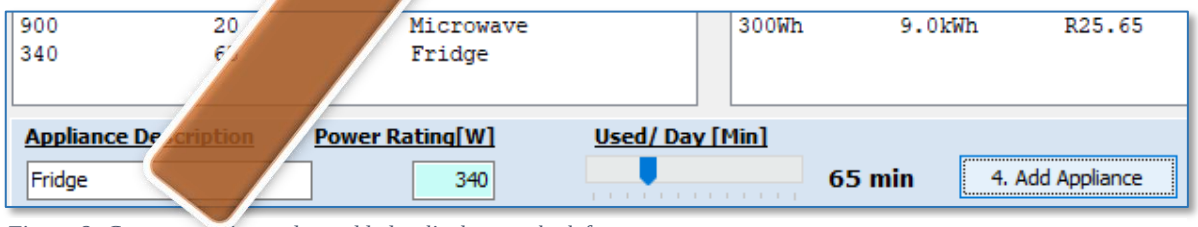


Figure 8: Correct appliance data added to list box on the left

(7)

6.5 If the button [2. Calculate Expenses] is pressed *again* (after adding a new appliance), the necessary calculations must be performed: calculated data must be added on the RHS list box and the overall total must be adapted. – If necessary, go back to the answer of question 5 and update your code to accommodate such actions. Our example's result should reflect something like Figure 9.

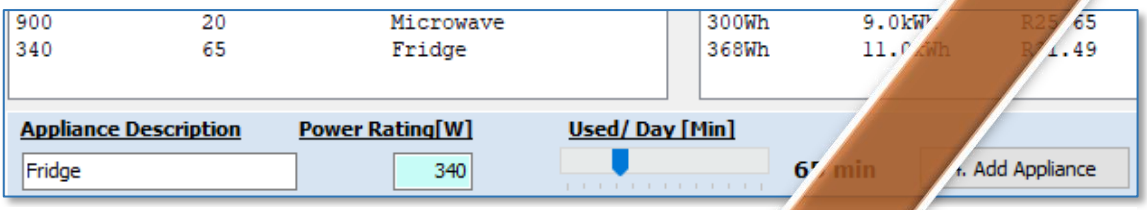


Figure 9: The button [2. Calculate Expenses] pressed again - resulting in a new Total of R 706.94

7. *Save all data in a CSV file* *[10]*

7.1 Supply the code for the OnClick-event of the button [5. Save as CSV]. Any saving method may be used for this purpose. The use of a ***TStreamWriter-object*** is recommended.

7.2 The file name must include todays date in the following format: "**EBill_2020Mar13.csv**". Copy the content of both list boxes – first the contents of the second list box and then the first list box. Add an empty line underneath and add the overall total amount. A screen print from the saved file is provided as seen in Notepad++ and Excel. See Figure 10 and 11 respectively.

```
[Wh/Day],[kWh/Month],[Cost]-[ZAR],PowerRate[W],Usage/Day[Min],Appliance Description
========,==========,==========,============,==============,====================
4500Wh,135.0kWh,R384.75,1800,150,Oven
220Wh,6.6kWh,R18.81,110,120,HiFi
960Wh,28.8kWh,R82.08,320,180,PC/Laptop
920Wh,27.6kWh,R78.66,230,240,Lights
1000Wh,30.0kWh,R85.50,2000,30,Kettle
300Wh,9.0kWh,R25.65,900,20,Microwave
368Wh,11.0kWh,R31.49,340,65,Fridge

,TOTAL,R706.94
```

Figure 10: Saved file content as seen through Notepad++

	A	B	C	D	E	F
1	[Wh/Day]	[kWh/Month]	[Cost]-[ZAR]	PowerRate[W]	Usage/Day[Min]	Appliance Description
2	========	==========	==========	==========	==========	==========
3	4500Wh	135.0kWh	R384.75	1800	150	Oven
4	220Wh	6.6kWh	R18.81	110	120	HiFi
5	960Wh	28.8kWh	R82.08	320	180	PC/Laptop
6	920Wh	27.6kWh	R78.66	230	240	Lights
7	1000Wh	30.0kWh	R85.50	2000	30	Kettle
8	300Wh	9.0kWh	R25.65	900	20	Microwave
9	368Wh	11.0kWh	R31.49	340	65	Fridge
10						
11		TOTAL	R706.94			

Figure 11:Saved CSV-file opened in Excel

(10)

Assignment 4 takes off ….

The source code till here is provided to you with minimal changes to assignment 2. Make sure you understand all the pre-existing code with changes. A page control component was added with a second- and third-tab sheet!

TAB SHEET TWO →

1. **Prepare the String grid**

1.1 Add a self-defined local method that should ease the task to retrieve specific fields from a character-delimited structured data line. The header could be defined as follows:

```
AnsiString getFieldNr(AnsiString aLine, int fldNum, char delim)
```

1.2 Use the OnShow-event of the second Tab Sheet. Supply the coding to assign the hard-coded column headings as seen in **Figure 13** and define correct sized column widths.

2. **Load Selected data into the String Grid**

2.1 Supply the code for the Button **[Load Data]**. In question 7 it was expected to generate a "current-date-marked" file name for the data to be saved. The file name with path could have been stored inside the provided Class data member: **`outFile`**, or your own declared variable.

It is your task here to retrieve the selected information from this saved file. But, add code for the possibility that no data file was saved. If your previous programming did not produce a data file, make use of your additional coding that will retrieve the predefined file name: "**EBill_2021Feb20.dat**" to be found inside the subfolder myData.

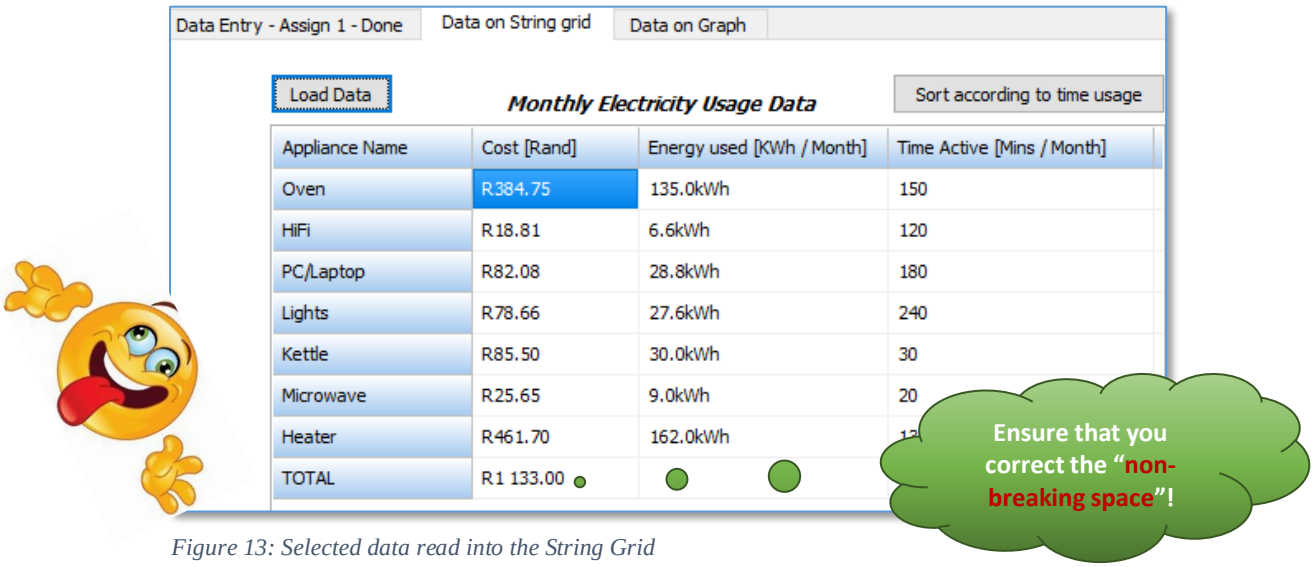


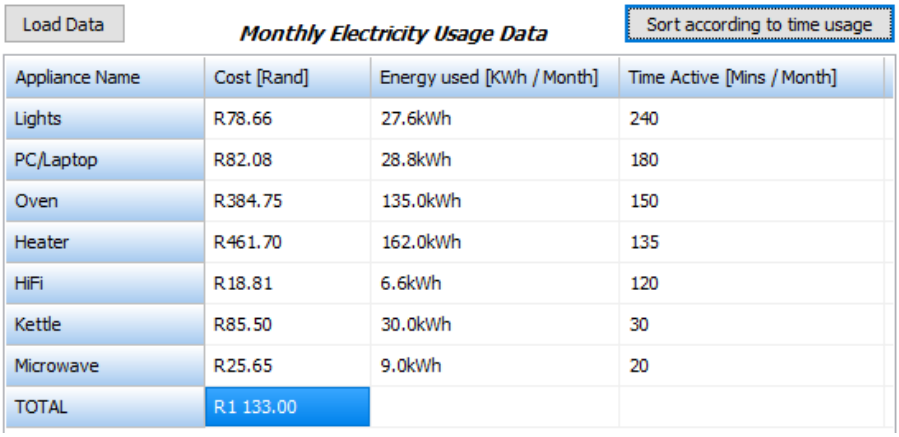
Figure 13: Selected data read into the String Grid

A String List component called **aList** is declared as global object and instantiated as part of the default constructor. Read the data from the file into the object aList. From aList retrieve only selected fields as displayed in figure 13.

Make use of the local function defined in 1.1 (assignment 4) to retrieve values from specific fields/ columns.

3. Sort the String Grid according to the time used by each appliance

2.1 Code the OnClick-event of the button **[Sort according to time usage]**. It must sort the string grid data section according to the last populated column. Make sure you do not include the headers nor the final summary row. A sorting result for the predefined data file look as follows:



Appliance Name	Cost [Rand]	Energy used [KWh / Month]	Time Active [Mins / Month]
Lights	R78.66	27.6kWh	240
PC/Laptop	R82.08	28.8kWh	180
Oven	R384.75	135.0kWh	150
Heater	R461.70	162.0kWh	135
HiFi	R18.81	6.6kWh	120
Kettle	R85.50	30.0kWh	30
Microwave	R25.65	9.0kWh	20
TOTAL	R1 133.00		

Figure 124: String grid sorted to last column's data

4. Display data retrieved in a graphical format

4.1 Turn to tab sheet three and find an "empty" graph as displayed in Figure 15.

The OnClick-event of the button **[Display Graph]** must be coded first.

Make sure that data is available, otherwise quit the event with a short message.

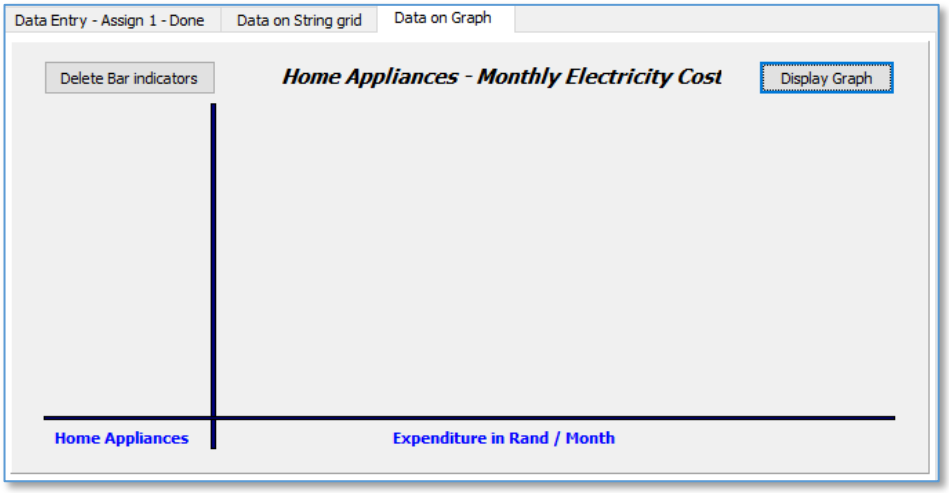


Figure 15: Empty graph

The graph has a horizontal axis which indicates the name of the appliance with Rand expenditure per month on the LHS and the cost per appliance per month graphically demonstrated with progress bars on the RHS.

The data of selected columns from the previous string grid are carried over to be displayed as part of this "graph". All data and components you add to this graph must be instantiated dynamically. A fully populated graph could look like Figure 16.

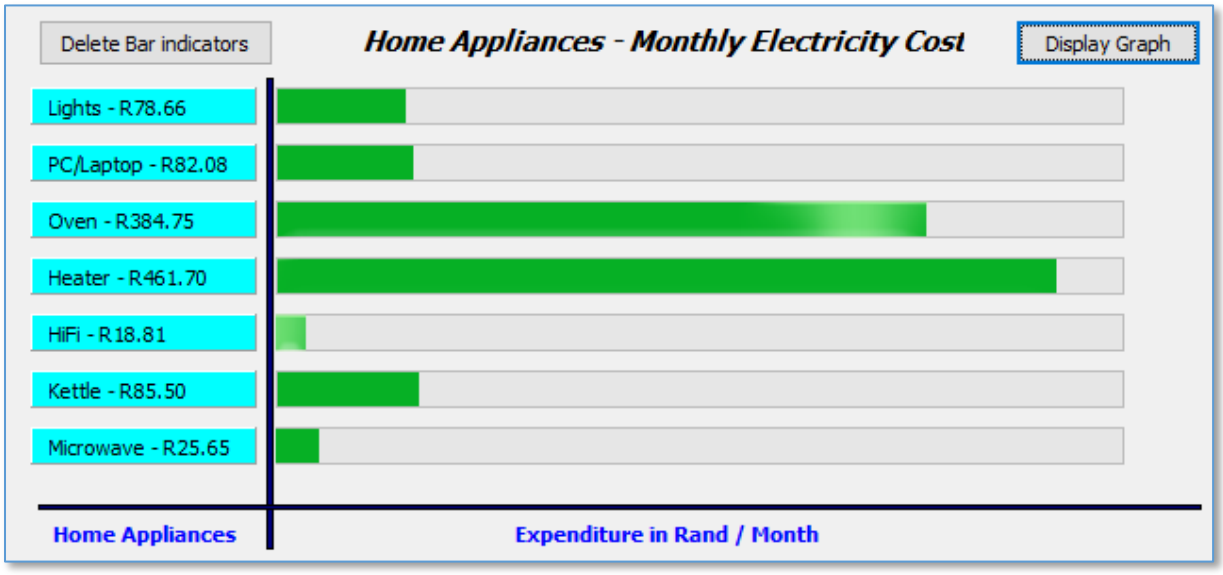


Figure 136: Fully populated graph

Three groups of dynamically instantiated components can be noticed:

- The ***appliance name merged with expenditure amount*** is captured on ***labels***. These labels are then "pasted" on a ***panel*** which are neatly aligned on the far left. Each appliance is spaced evenly to the next.
- The ***Expenditure in Rand / Month*** -values are extracted from the second column. Each value is indicated and demonstrated by using a dynamically instantiated progress bar. Find the approximate correct *scaling* for the amount to be displayed appropriately.

5. Delete bars on the RHS

5.1 Add coding to the OnClick-event of the button **[Delete Bar indicators]**. This button is responsible to delete only the progress bars on the RHS. After deletion, your "graph" should look similar to this:

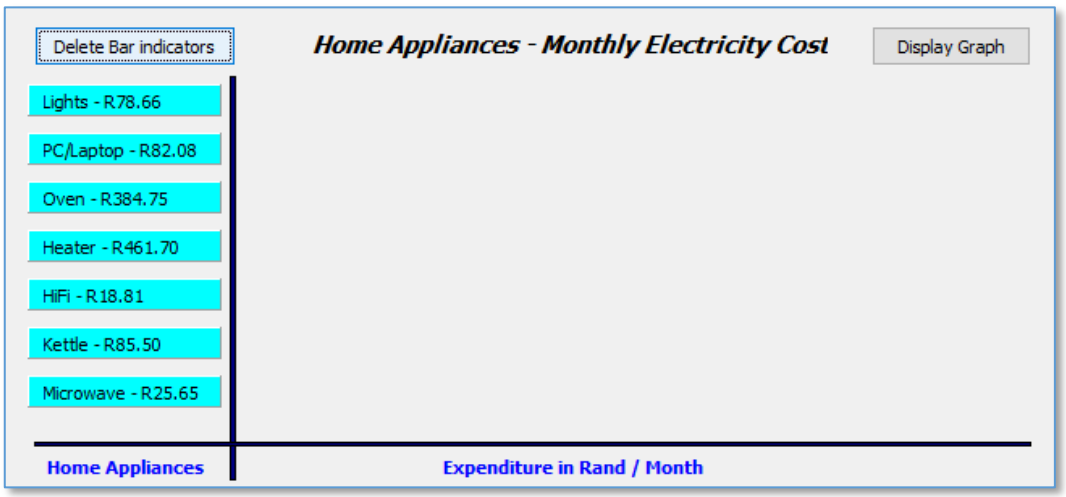


Figure 147 Progress bars deleted

Happy Coding – Run the Demo-App!