UNIVERSITY OF IBADAN

Arithmetic Arranger

CSC 231 (Scientific Programming) Course Project

COMPUTER SCIENCE CLASS OF 2018/2019

Submitted to :

DR. J. D. AKINYEMI

MAY 31, 2021

Students Name and Matric Number

FIRSTNAME	SURNAME	MATRIC NUMBER
Emmanuel	Osanebi	214910
Warith	Adeoti	214851
Christiana	Adisa	214853
Oluwaseun	Alatise	214860
Ifeoluwa	Akinwusi	214858
Stephen	Ivuelekwa	214882
Daniel	Brai	214868
Timothy	Adeleke	214849
Victoria	Matthews	214886
Opeoluwa	Ojewale	214899
Toluwanimi	Osuolale	214912
Prince	Ajayi	215221
Yusuff	Afenifere	222456
Godwin	Daniel	214871
Adedapo	Anjorin	214864
Ayomide	Arowolo	214865
David	Eziagulu	214872
Nuh	Ibraheem	214879
Precious	Ogbolu	214894
Daniel	Oghie	214895
Oluwanifise	Oluwayelu	215257
Joshua	Okoro	214902
Odi Glory	Ebube	186355
James	Emiade	214874
Uchechukwu	Lawal	214885
Lateefat	Salami	214915
Mustapha	Kareem	214883
Joshua	Farayola	214878
Michael	Olatunji	214903
Amazing-Grace	Ubaka	214918
Blessing	Akinrinola	214857
Khadijat	Olalere	222502
Omowunmi	Alao	222461
Joshua	Okoro	214902
Ramadan	Mikail	214887
Promise	Okafor	213930
Uchechukwu	Ahunanya	214854
Kehinde	Soyoye	214916
Olamide	Akinade	215049
Ahmad	Animasaun	214863
Jamiu	Jimoh	222478
Eniola	Olawale	214904
Mustapha	Kareem	214883
Iyanuoluwa	Oluwatade	214907
Oghenetega	Denedo	214873
Chidinma	Nwatu	214890
Adedapo	Anjorin	214864
Prince	Okunade	215248
Oluwapelumi	Onasoga	214909
Michael	Olatunji	214903
Sherifdeen	Adeleke	214848
Rukevwe	Okumagba	222498
Oluwatobi	Odulate	214893
Peter	Sadiq	214914
Ebikabowei Caleb	Olorogun	214906
Samuel	Adelowo	214850
Faith Eniola	Akinade	222459
Maluchukwu	Nebeolisa	214889
Joshua	Olagidi	222500
SOLOMON	ADIM	222455
Emmanuel	Daniel	214870
PETER	KAYODE	208077
Joseph	Adebowale	214846
Adenike	Ojo	222494
Olamilekan	Cole	223942
Olamide	Akinade	215049
Eniola	Oyekanmi	214913
Temidayo	Ogunyemi	214898
Kip	Emeka	215061
Laura	Kubiat	214884
Rhoda	Ogunesan	214897
Lisa	Okafor	214901
Wisdom	Oyor	215206
Michael	Ikwuegbu	214881
Onyinyechukwu	Chiatula	214869

Arithmetic Arranger

Project Description:

Students in primary school often arrange arithmetic problems vertically to make them easier to solve. For example, "235 + 52" becomes:

```
  235
+  52
-----
```

Create a function that receives a list of strings that are arithmetic problems and returns the problems arranged vertically and side-by-side. The function should optionally take a second argument. When the second argument is set to “`True`”, the answers should be displayed.

For example

Function Call:

```
arithmetic_arranger(["32 + 698", "3801 - 2", "45 + 43", "123 + 49"])
```

Output:

```
   32      3801      45      123
+ 698    -    2    + 43    +  49
-----    ------    ----    -----
```

Function Call:

```
arithmetic_arranger(["32 + 8", "1 - 3801", "9999 + 9999", "523 - 49"], True)
```

Output:

```
  32         1      9999      523
+  8    - 3801    + 9999    -  49
----    ------    ------    -----
  40     -3800     19998      474
----    ------    ------    -----
```

Rules

The function will return the correct conversion if the supplied problems are properly formatted, otherwise, it will **return** a **string** that describes an error that is meaningful to the user.

- Situations that will return an error:

  - If there are **too many problems** supplied to the function. The limit is **seven**, anything more will return: "***Error: Too many problems.***"
  - The appropriate operators the function will accept are **addition** and **subtraction**. **Multiplication** and **division** will return an error. Other operators not mentioned in this bullet point will not need to be tested. The error returned will be: "***Error: Operator must be '+' or '-'.***"
  - Each number (operand) should only contain digits (0 - 9). Otherwise, the function will return: "***Error: Numbers must only contain digits.***"
  - Each operand (a.k.a number on each side of the operator) has a max of six digits in width. Otherwise, the error string returned will be: "***Error: Numbers cannot be more than six digits.***"

- If the user supplied the correct format of problems, the conversion you return will follow these rules:
  - There should be a single space between the operator and the longest of the two operands, the operator will be on the same line as the second operand, both operands will be in the same order as provided (the first will be the top one and the second will be the bottom).
  - Numbers should be right-aligned.
  - There should be four spaces between each problem.
  - There should be dashes at the bottom of each problem and at the bottom of each answer. The dashes should run along the entire length of each problem individually. (The example above shows what this should look like.)

Project’s Execution

The project’s execution is done using a very simple algorithm. In the algorithm, the idea is to first take care of all the expected errors (successively as directed in the project instruction) and then the formatting of expressions as required.

The implementation of the algorithm is as follows:

- Function’s Definition: The “*arithmetic_arranger*” function is defined with precisely two arguments (the first being a required argument of a ***list type*** named “*expression*” and the second, an optional ***boolean*** argument named “*result*” defaulted to “`False`”).

```
def arithmetic_arranger(expression, result = False):
```

- First Error Handling (Function must only accept a maximum of seven expressions): Taking care of the first error by checking the length of list of expression argument using the “***len()***” function.

.

```
    if len(expression) > 7:
        return "Error: Too many problems."
```

- String objects instantiations: The function is going to use a loop as part of the algorithm implementation, but before entering the loop construct, predefinition of these four string objects will be required, as these objects are used in the body of loop. It can be seen that there are (maximum of) four lines (if result argument is “`True`”) in the arranged expressions which the function is expected to return (if correct arguments are supplied) with the first line containing the first operands from each of the expressions, the second, containing the operators together with the second operands of each of the expression, the third line containing a number of dashes (as underline) required for each expressions and the fourth being the line that contain the results of evaluation of these expressions, These defined strings will be used to store/hold all these lines of each expression.

```
    line1 = “”          # To hold the first line of each of the expressions
    line2 = “”          # To hold the second line of each of the expressions
    line3 = “”          # The third line for underlining
    line4 = “”          # Fourth line of each expression for solution of a problem
```

- Loop: Looping through the ***list*** argument(with a ***for*** loop) in order to parse each string contained in the expression. The “***exp***” variable will hold an individual string contained in ***expression*** on each iteration

```
    for exp in expression:
```

- Removing whitespaces from the string of expressions: The first statement in the body of the “*for loop”* is for removing all whitespaces from the string of expression. The aim of this is to allow the function to accept the cases of expressions supplied by the user, regardless of whether the individual strings are separated by spaces or not (e.g. case “`345 + 56`” or “`345+56`”) and also in the next few statements it will be required that the strings are free of whitespaces. Using string’s “***`replace()`***” method, all whitespaces are replaced with empty string.

```
        exp = exp.replace(" ", "")
```

- Parsing string for the required Operators: In these next few statements, the required/supported operators (“+” or “-“) are checked in the string of expression (by membership operation), and with ***if/else*** statements. If any of the supported operators is found in the string, the string is then split using the same operator found in the string as the split delimiter, so as to have the two operands of the expression in the string separated. So, using the “***split()***” string method the string is split, this will return a list object which is expected to contain the two operands of the expression. If none of the supported operators are found in the string, then the function will return the error message, thereby handling the second error.

```
        if "+" in exp:
            exp = exp.split("+")
            operator = "+"
        elif "-" in exp:
            exp = exp.split("-")
            operator = "-"
        else:
            return "Error: Operator must be '+' or '-'."
```

- Third Error Handling (Expressions must be digits): To take care of the third error, by checking for operands with non-digit characters. From the previous statements the “***exp***” variable is expected to be a list containing two elements(operands). Both elements in the list are checked if they have non-digit characters using a **NAND** Boolean expression. By indexing, the two operands in the “***exp***” list are accessed and the “***isdigit()***” method is called on them to check if they are digits, returning the error message if non-digit character are found.

```
        if not(exp[0].isdigit() and exp[1].isdigit()):
            return "Error: Numbers must only contain digits."
```

- Fourth Error Handling (Operands can only contain maximum digits of six): Taking care of the fourth error, by checking for the number of digits contained in each operand. We use “***len()***” function to check the length of each string operand in the “*exp*” ***list*** variable, returning the error message if lengths are greater than six.

```
        if len(exp[0]) > 6 or len(exp[1])> 6:
            return "Error: Numbers cannot be more than six digits."
```

And with these, all the possible errors are taken care of.

Next is evaluation of the expressions and arranging the expression in string.

- Calculating the maximum spanning of characters in each line of arranged expression, required for alignment: Each string on each line of the arranged expression is required to be rightly justified. For this reason, the maximum number of characters that can be on a line will need to be computed, and this is done by finding the number of characters in the operand that has the largest number of digits, using the “***len()***” and the “***max()***” functions and then, adding 2 to this result since the instruction required that a whitespace must be between the largest operand and the operator (i.e two more characters). The resulting value is stored in the variable “***align***”.

```
        align = max([len(exp[0]), len(exp[1])]) + 2
```

- Evaluation of expression: If the “*result*” argument is set to “`True`”, the function is expected to return an arranged expression containing the result of evaluation of the expression. We use the “***eval()***” function to evaluate the string containing the concatenation of the first operand, the operator and the second operand. After evaluation, the result is justified rightly using the “***rjust()***” string method by the alignment length calculated above. We then concatenate the rightly justified result with the required four spaces between each problem and then, add them up to whatever value we already have in the ***line4*** string variable.

```
        if result:
            res = str(eval(exp[0] + operator + exp[1]))
            line4 += res.rjust(align) + “    “
```

- Adding up appropriate values to the appropriate ***line string variable***: The remaining three string line variables (***line1, line2, lin3***) are populated with the appropriate values, justified rightly. Each value is concatenated with the required four spaces between each problem.

```
        line1 += exp[0].rjust(align) + “    “
        # Adding into line1, first Operand with four spaces, rightly-justified
        line2 += operator + exp[1].rjust(align - 1) + “    “
        # Adding into line2, operator and the second operand with four
spaces, rightly-justified

        line3 += "-" * (align) + “    “
        # Adding into line3, the underlining set of dashes with four spaces
```

This marks the end of the “***for loop***”, and now to arrange the expressions in a string

- Taking off the whitespaces at the end of each line: Using the “***rstrip()***” string method, we remove the four spaces at the end of each line since they are not required.

```
    line1 = line1.rstrip()
    line2 = line2.rstrip()
    line3 = line3.rstrip()
```

- Joining the lines together in a single string: The lines generated from above statements are then further joined together, separated by a new line character. The first three are joined and the stored in a variable “***arranged_string***”.

```
    arranged_string = "\n".join([line1, line2, line3])
```

- Adding the result line if the “***result***” argument is set to “True”: Concatenating the result line with “*arranged_string*” as well as the “underlining line” (***line3***) together with newline characters at appropriate positions.

```
    if result:
        line4 = line4.rstrip()
        arranged_string += "\n" + line4 + "\n" + line3
```

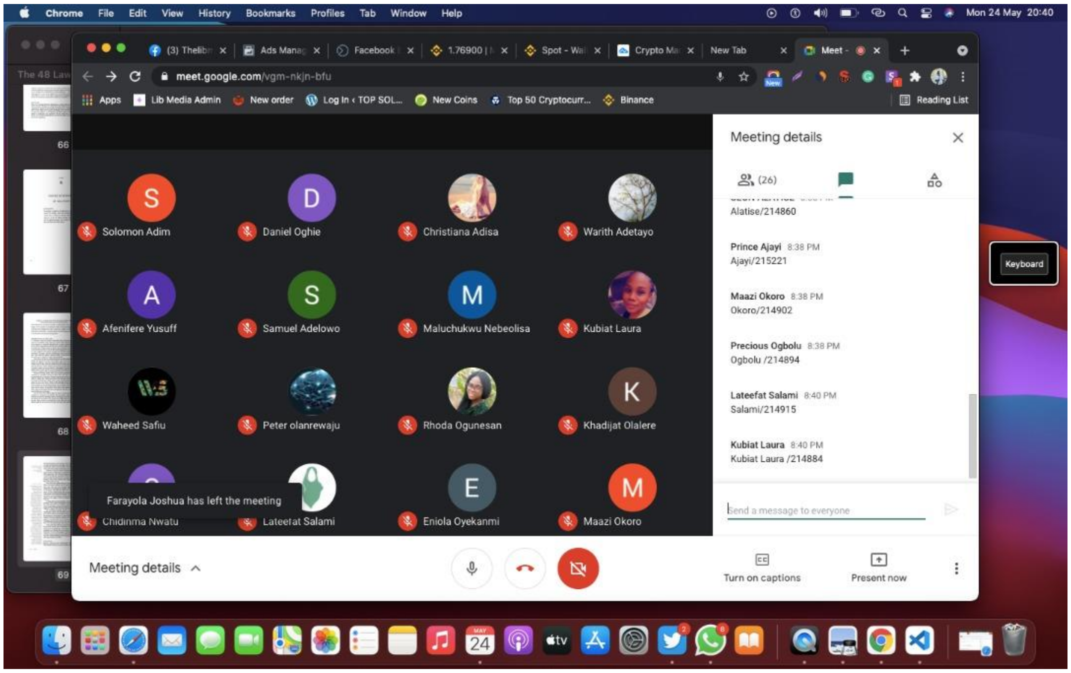
- The “*arranged_string*” is finally returned.

```
    return arranged_string
```

And this marks the end of the function.

END OF PROJECT!!!

SCREENSHOTS OF OUR MEETINGS



Alatise/214860

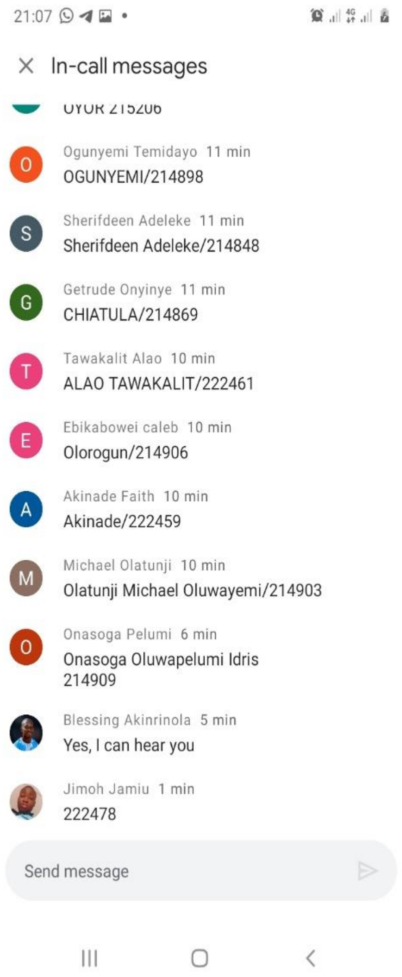
Prince Ajayi 8:38 PM
Ajayi/215221

Maazi Okoro 8:38 PM
Okoro/214902

Precious Ogbolu 8:38 PM
Ogbolu /214894

Lateefat Salami 8:40 PM
Salami/214915

Kubiat Laura 8:40 PM
Kubiat Laura /214884



OYOR 215206

Ogunyemi Temidayo 11 min
OGUNYEMI/214898

Sherifdeen Adeleke 11 min
Sherifdeen Adeleke/214848

Getrude Onyinye 11 min
CHIATULA/214869

Tawakalit Alao 10 min
ALAO TAWAKALIT/222461

Ebikabowei caleb 10 min
Olorogun/214906

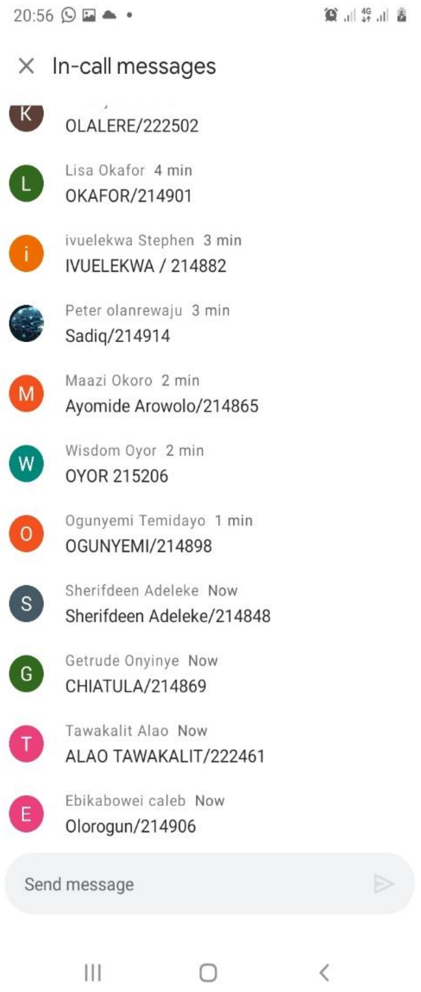
Akinade Faith 10 min
Akinade/222459

Michael Olatunji 10 min
Olatunji Michael Oluwayemi/214903

Onasoga Pelumi 6 min
Onasoga Oluwapelumi Idris
214909

Blessing Akinrinola 5 min
Yes, I can hear you

Jimoh Jamiu 1 min
222478



OLALERE/222502

Lisa Okafor 4 min
OKAFOR/214901

ivuelekwa Stephen 3 min
IVUELEKWA / 214882

Peter olanrewaju 3 min
Sadiq/214914

Maazi Okoro 2 min
Ayomide Arowolo/214865

Wisdom Oyor 2 min
OYOR 215206

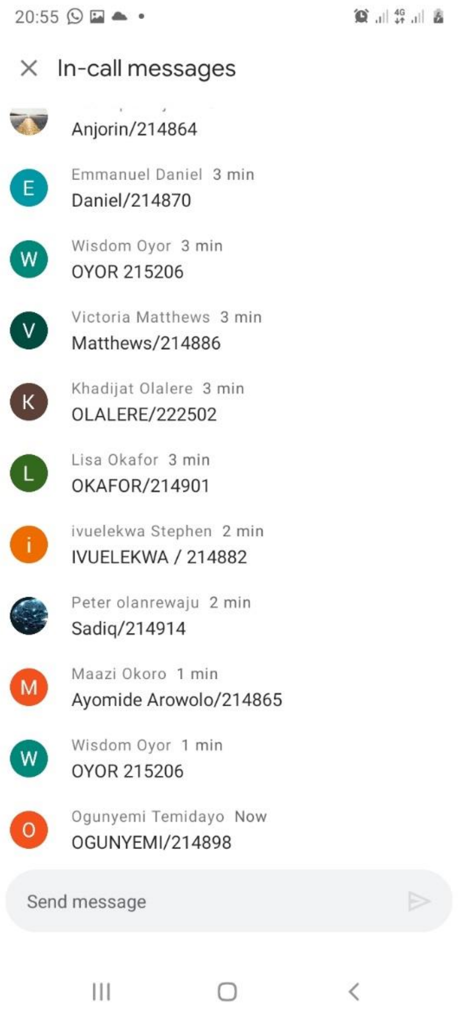
Ogunyemi Temidayo 1 min
OGUNYEMI/214898

Sherifdeen Adeleke Now
Sherifdeen Adeleke/214848

Getrude Onyinye Now
CHIATULA/214869

Tawakalit Alao Now
ALAO TAWAKALIT/222461

Ebikabowei caleb Now
Olorogun/214906



Anjorin/214864

Emmanuel Daniel 3 min
Daniel/214870

Wisdom Oyor 3 min
OYOR 215206

Victoria Matthews 3 min
Matthews/214886

Khadijat Olalere 3 min
OLALERE/222502

Lisa Okafor 3 min
OKAFOR/214901

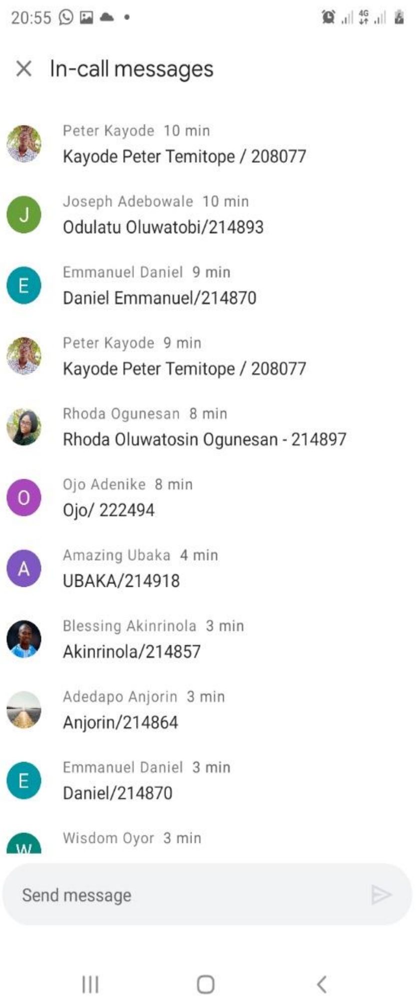
ivuelekwa Stephen 2 min
IVUELEKWA / 214882

Peter olanrewaju 2 min
Sadiq/214914

Maazi Okoro 1 min
Ayomide Arowolo/214865

Wisdom Oyor 1 min
OYOR 215206

Ogunyemi Temidayo Now
OGUNYEMI/214898



Peter Kayode 10 min
Kayode Peter Temitope / 208077

Joseph Adebowale 10 min
Odulatu Oluwatobi/214893

Emmanuel Daniel 9 min
Daniel Emmanuel/214870

Peter Kayode 9 min
Kayode Peter Temitope / 208077

Rhoda Ogunesan 8 min
Rhoda Oluwatosin Ogunesan - 214897

Ojo Adenike 8 min
Ojo/ 222494

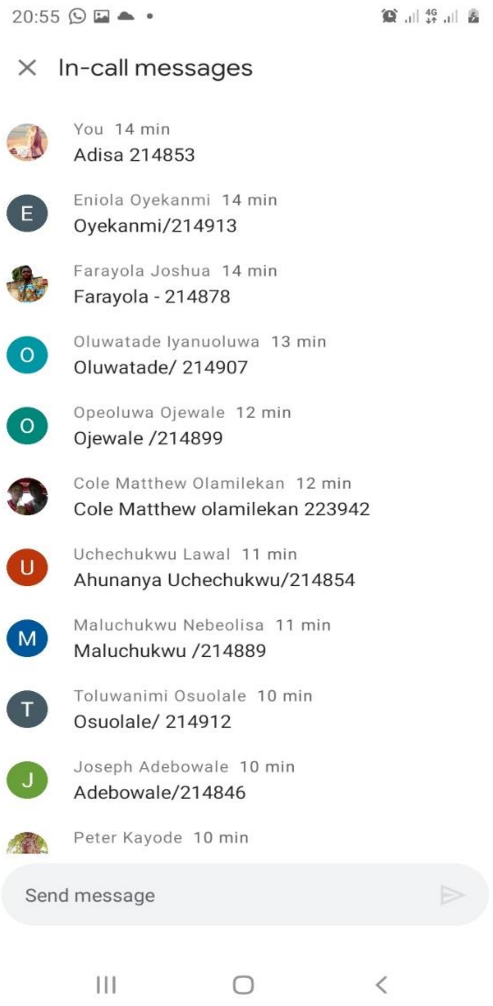
Amazing Ubaka 4 min
UBAKA/214918

Blessing Akinrinola 3 min
Akinrinola/214857

Adedapo Anjorin 3 min
Anjorin/214864

Emmanuel Daniel 3 min
Daniel/214870

Wisdom Oyor 3 min



You 14 min
Adisa 214853

Eniola Oyekanmi 14 min
Oyekanmi/214913

Farayola Joshua 14 min
Farayola - 214878

Oluwatade Iyanuoluwa 13 min
Oluwatade/ 214907

Opeoluwa Ojewale 12 min
Ojewale /214899

Cole Matthew Olamilekan 12 min
Cole Matthew olamilekan 223942

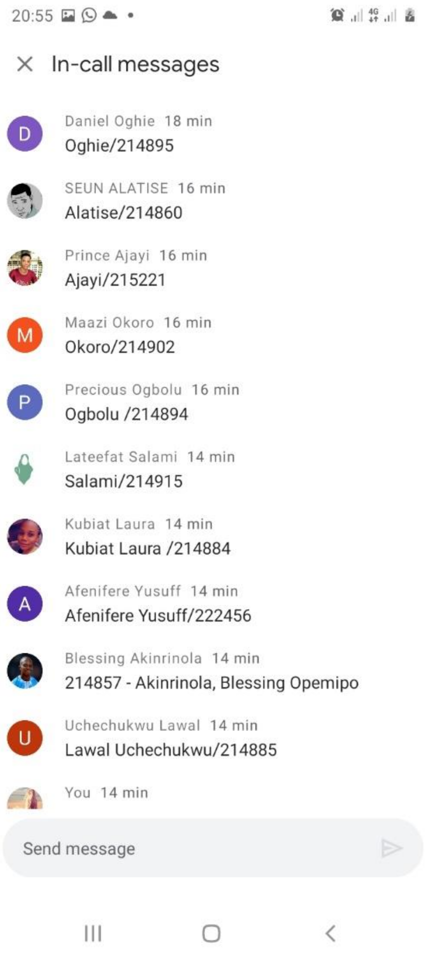
Uchechukwu Lawal 11 min
Ahunanya Uchechukwu/214854

Maluchukwu Nebeolisa 11 min
Maluchukwu /214889

Toluwanimi Osuolale 10 min
Osuolale/ 214912

Joseph Adebowale 10 min
Adebowale/214846

Peter Kayode 10 min



Daniel Oghie 18 min
Oghie/214895

SEUN ALATISE 16 min
Alatise/214860

Prince Ajayi 16 min
Ajayi/215221

Maazi Okoro 16 min
Okoro/214902

Precious Ogbolu 16 min
Ogbolu /214894

Lateefat Salami 14 min
Salami/214915

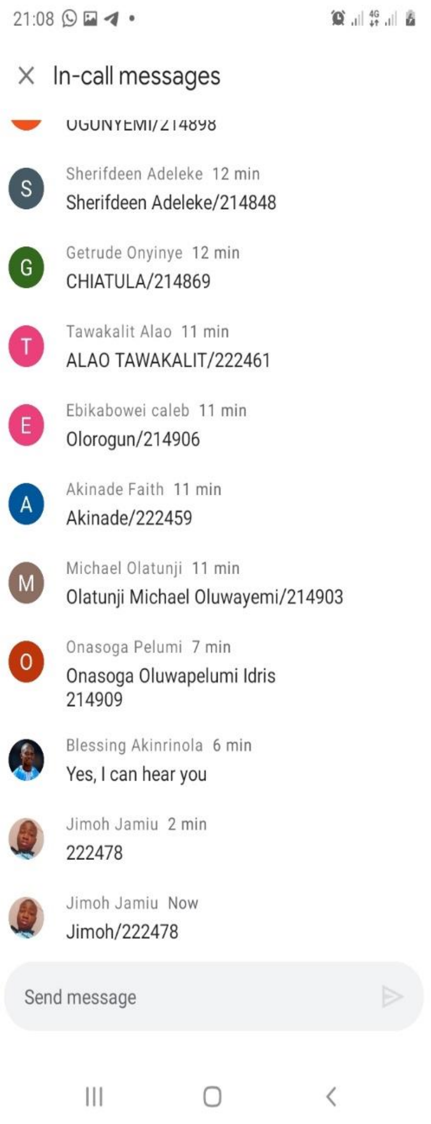
Kubiat Laura 14 min
Kubiat Laura /214884

Afenifere Yusuff 14 min
Afenifere Yusuff/222456

Blessing Akinrinola 14 min
214857 - Akinrinola, Blessing Opemipo

Uchechukwu Lawal 14 min
Lawal Uchechukwu/214885

You 14 min



21:08

In-call messages

OGUNYEMI/214898

Sherifdeen Adeleke 12 min
Sherifdeen Adeleke/214848

Getrude Onyinye 12 min
CHIATULA/214869

Tawakalit Alao 11 min
ALAO TAWAKALIT/222461

Ebikabowei caleb 11 min
Olorogun/214906

Akinade Faith 11 min
Akinade/222459

Michael Olatunji 11 min
Olatunji Michael Oluwayemi/214903

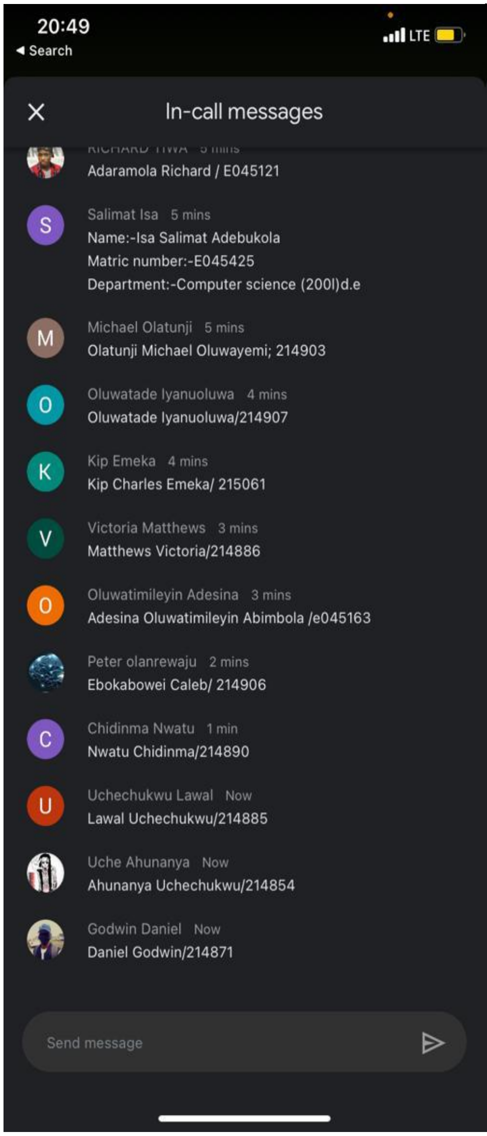
Onasoga Pelumi 7 min
Onasoga Oluwapelumi Idris 214909

Blessing Akinrinola 6 min
Yes, I can hear you

Jimoh Jamiu 2 min
222478

Jimoh Jamiu Now
Jimoh/222478

Send message



20:49
◀ Search

In-call messages

RICHARD TIWA 5 mins
Adaramola Richard / E045121

Salimat Isa 5 mins
Name:-Isa Salimat Adebukola
Matric number:-E045425
Department:-Computer science (200l)d.e

Michael Olatunji 5 mins
Olatunji Michael Oluwayemi; 214903

Oluwatade Iyanuoluwa 4 mins
Oluwatade Iyanuoluwa/214907

Kip Emeka 4 mins
Kip Charles Emeka/ 215061

Victoria Matthews 3 mins
Matthews Victoria/214886

Oluwatimileyin Adesina 3 mins
Adesina Oluwatimileyin Abimbola /e045163

Peter olanrewaju 2 mins
Ebokabowei Caleb/ 214906

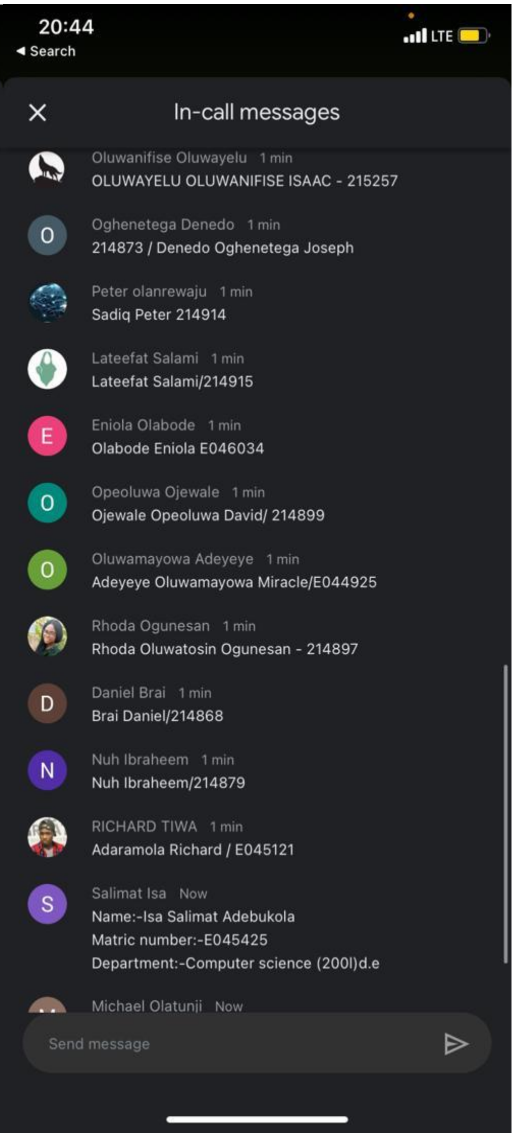
Chidinma Nwatu 1 min
Nwatu Chidinma/214890

Uchechukwu Lawal Now
Lawal Uchechukwu/214885

Uche Ahunanya Now
Ahunanya Uchechukwu/214854

Godwin Daniel Now
Daniel Godwin/214871

Send message



20:44
◀ Search

In-call messages

Oluwanifise Oluwayelu 1 min
OLUWAYELU OLUWANIFISE ISAAC - 215257

Oghenetega Denedo 1 min
214873 / Denedo Oghenetega Joseph

Peter olanrewaju 1 min
Sadiq Peter 214914

Lateefat Salami 1 min
Lateefat Salami/214915

Eniola Olabode 1 min
Olabode Eniola E046034

Opeoluwa Ojewale 1 min
Ojewale Opeoluwa David/ 214899

Oluwamayowa Adeyeye 1 min
Adeyeye Oluwamayowa Miracle/E044925

Rhoda Ogunesan 1 min
Rhoda Oluwatosin Ogunesan - 214897

Daniel Brai 1 min
Brai Daniel/214868

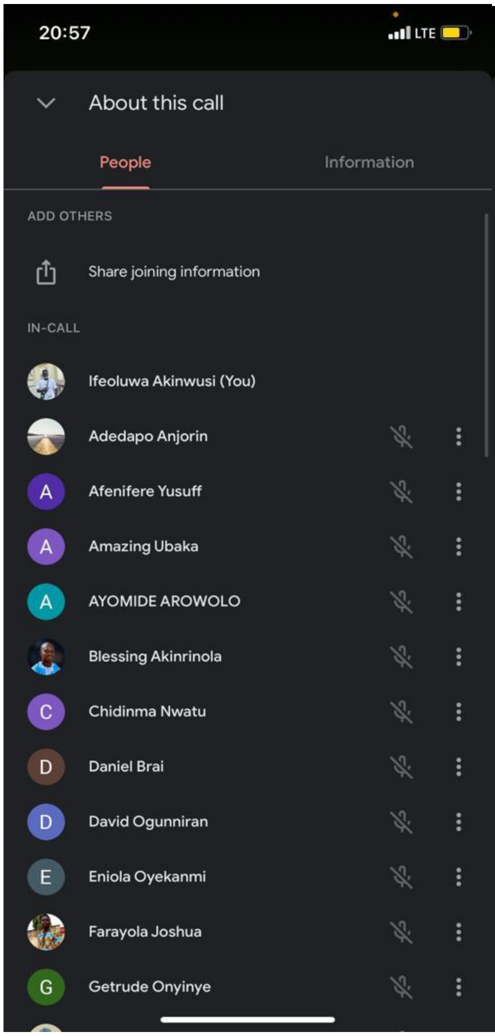
Nuh Ibraheem 1 min
Nuh Ibraheem/214879

RICHARD TIWA 1 min
Adaramola Richard / E045121

Salimat Isa Now
Name:-Isa Salimat Adebukola
Matric number:-E045425
Department:-Computer science (200l)d.e

Michael Olatunji Now

Send message



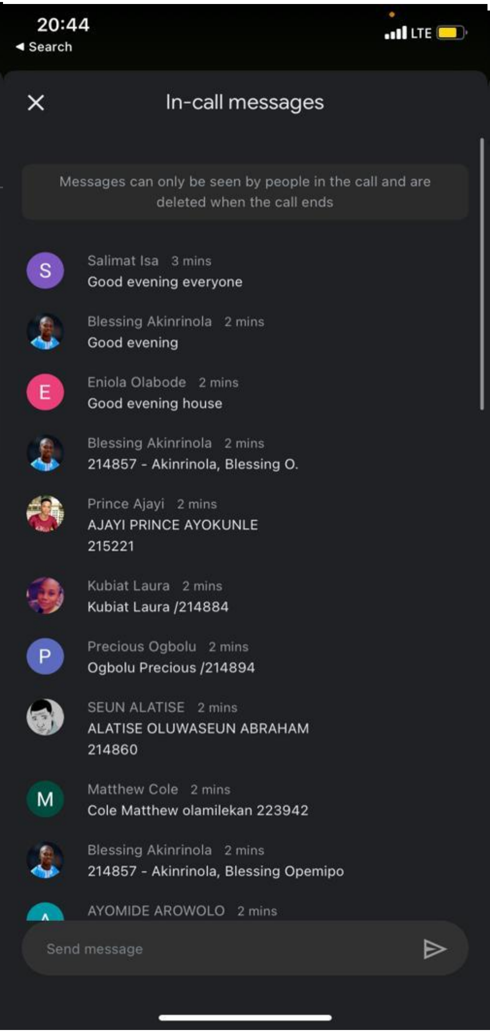
About this call

People | Information

ADD OTHERS

IN-CALL

- Ifeoluwa Akinwusi (You)
- Adedapo Anjorin
- Afenifere Yusuff
- Amazing Ubaka
- AYOMIDE AROWOLO
- Blessing Akinrinola
- Chidinma Nwatu
- Daniel Brai
- David Ogunniran
- Eniola Oyekanmi
- Farayola Joshua
- Getrude Onyinye



In-call messages

Messages can only be seen by people in the call and are deleted when the call ends

Salimat Isa 3 mins
Good evening everyone

Blessing Akinrinola 2 mins
Good evening

Eniola Olabode 2 mins
Good evening house

Blessing Akinrinola 2 mins
214857 - Akinrinola, Blessing O.

Prince Ajayi 2 mins
AJAYI PRINCE AYOKUNLE
215221

Kubiat Laura 2 mins
Kubiat Laura /214884

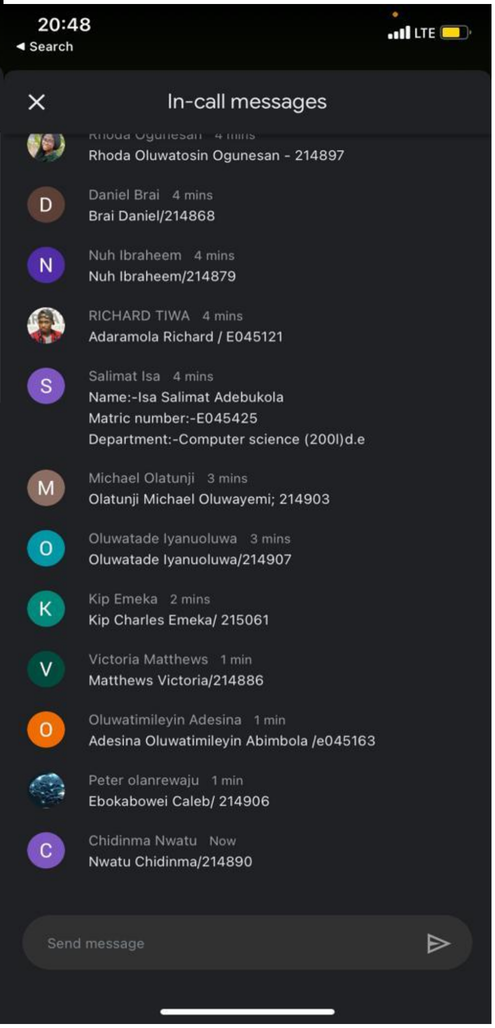
Precious Ogbolu 2 mins
Ogbolu Precious /214894

SEUN ALATISE 2 mins
ALATISE OLUWASEUN ABRAHAM
214860

Matthew Cole 2 mins
Cole Matthew olamilekan 223942

Blessing Akinrinola 2 mins
214857 - Akinrinola, Blessing Opemipo

AYOMIDE AROWOLO 2 mins



In-call messages

Rhoda Ogunesan 4 mins
Rhoda Oluwatosin Ogunesan - 214897

Daniel Brai 4 mins
Brai Daniel/214868

Nuh Ibraheem 4 mins
Nuh Ibraheem/214879

RICHARD TIWA 4 mins
Adaramola Richard / E045121

Salimat Isa 4 mins
Name:-Isa Salimat Adebukola
Matric number:-E045425
Department:-Computer science (200l)d.e

Michael Olatunji 3 mins
Olatunji Michael Oluwayemi; 214903

Oluwatade Iyanuoluwa 3 mins
Oluwatade Iyanuoluwa/214907

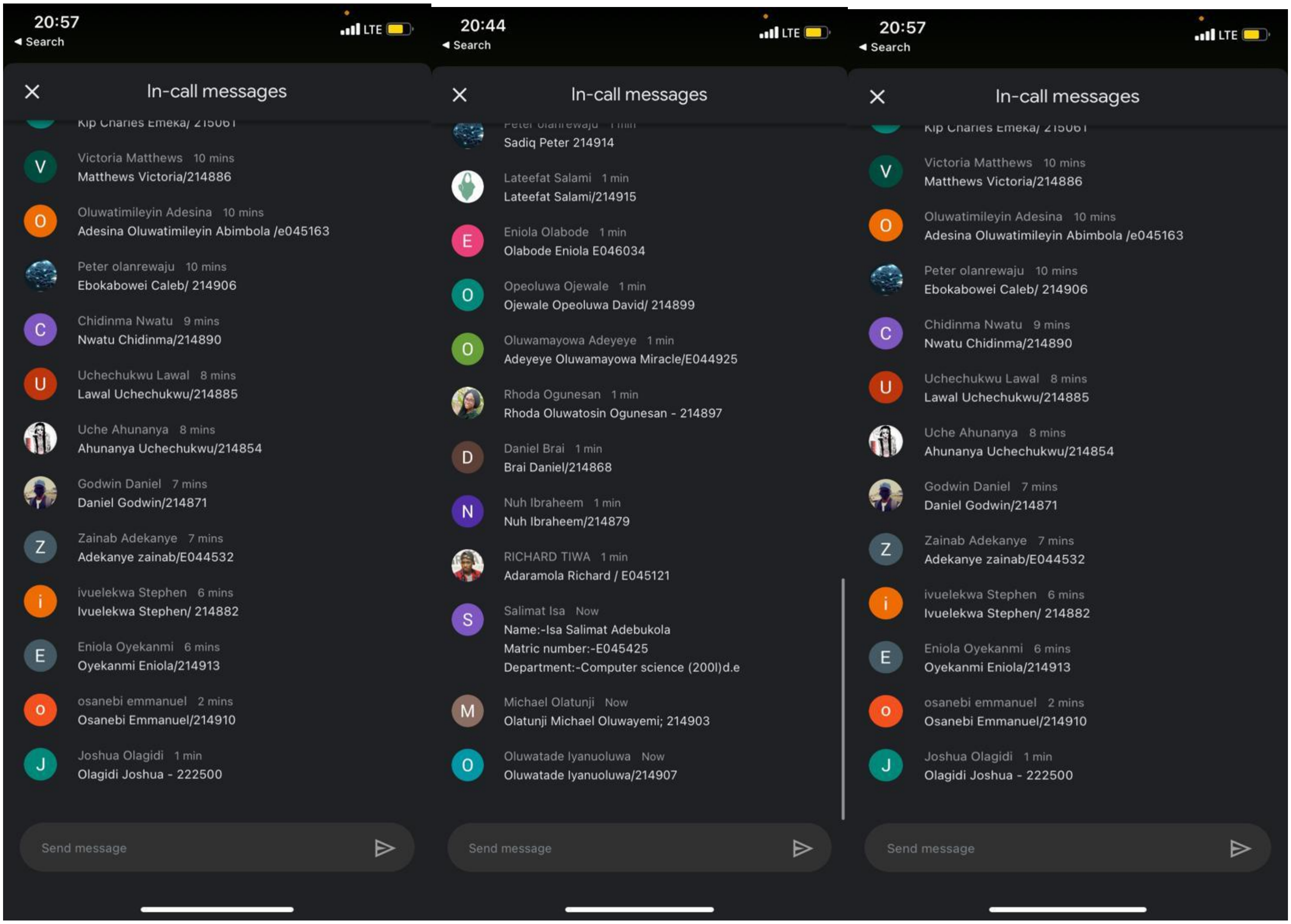
Kip Emeka 2 mins
Kip Charles Emeka/ 215061

Victoria Matthews 1 min
Matthews Victoria/214886

Oluwatimileyin Adesina 1 min
Adesina Oluwatimileyin Abimbola /e045163

Peter olanrewaju 1 min
Ebokabowei Caleb/ 214906

Chidinma Nwatu Now
Nwatu Chidinma/214890



Kip Charles Emeka/ 215061

Victoria Matthews 10 mins
Matthews Victoria/214886

Oluwatimileyin Adesina 10 mins
Adesina Oluwatimileyin Abimbola /e045163

Peter olanrewaju 10 mins
Ebokabowei Caleb/ 214906

Chidinma Nwatu 9 mins
Nwatu Chidinma/214890

Uchechukwu Lawal 8 mins
Lawal Uchechukwu/214885

Uche Ahunanya 8 mins
Ahunanya Uchechukwu/214854

Godwin Daniel 7 mins
Daniel Godwin/214871

Zainab Adekanye 7 mins
Adekanye zainab/E044532

ivuelekwa Stephen 6 mins
Ivuelekwa Stephen/ 214882

Eniola Oyekanmi 6 mins
Oyekanmi Eniola/214913

osanebi emmanuel 2 mins
Osanebi Emmanuel/214910

Joshua Olagidi 1 min
Olagidi Joshua - 222500

Peter olanrewaju 1 min
Sadiq Peter 214914

Lateefat Salami 1 min
Lateefat Salami/214915

Eniola Olabode 1 min
Olabode Eniola E046034

Opeoluwa Ojewale 1 min
Ojewale Opeoluwa David/ 214899

Oluwamayowa Adeyeye 1 min
Adeyeye Oluwamayowa Miracle/E044925

Rhoda Ogunesan 1 min
Rhoda Oluwatosin Ogunesan - 214897

Daniel Brai 1 min
Brai Daniel/214868

Nuh Ibraheem 1 min
Nuh Ibraheem/214879

RICHARD TIWA 1 min
Adaramola Richard / E045121

Salimat Isa Now
Name:-Isa Salimat Adebukola
Matric number:-E045425
Department:-Computer science (200l)d.e

Michael Olatunji Now
Olatunji Michael Oluwayemi; 214903

Oluwatade Iyanuoluwa Now
Oluwatade Iyanuoluwa/214907

Kip Charles Emeka/ 215061

Victoria Matthews 10 mins
Matthews Victoria/214886

Oluwatimileyin Adesina 10 mins
Adesina Oluwatimileyin Abimbola /e045163

Peter olanrewaju 10 mins
Ebokabowei Caleb/ 214906

Chidinma Nwatu 9 mins
Nwatu Chidinma/214890

Uchechukwu Lawal 8 mins
Lawal Uchechukwu/214885

Uche Ahunanya 8 mins
Ahunanya Uchechukwu/214854

Godwin Daniel 7 mins
Daniel Godwin/214871

Zainab Adekanye 7 mins
Adekanye zainab/E044532

ivuelekwa Stephen 6 mins
Ivuelekwa Stephen/ 214882

Eniola Oyekanmi 6 mins
Oyekanmi Eniola/214913

osanebi emmanuel 2 mins
Osanebi Emmanuel/214910

Joshua Olagidi 1 min
Olagidi Joshua - 222500

Ending Notes:

Upon completion of the project, and as it was stated that the project will be subjected to a unit testing, we decided to write a script to unit-test the program to make sure it is working as expected. All codes and scripts concerning the project are posted on the repository - https://github.com/holynation/Python-Arranger.