**Information Security and Assurance**
**Final Project Documentation**

| | |
|---|---|
| **Project Title:** | **Final Project in Building a Cipher Tool** |
| **Group Leader:** | **Dharel Flores** |
| **Members:** | **Claudette Jan A. Mercado** |
| | **Jhunell Juan** |
| | **Jorge R. Baniaga, Jr.** |

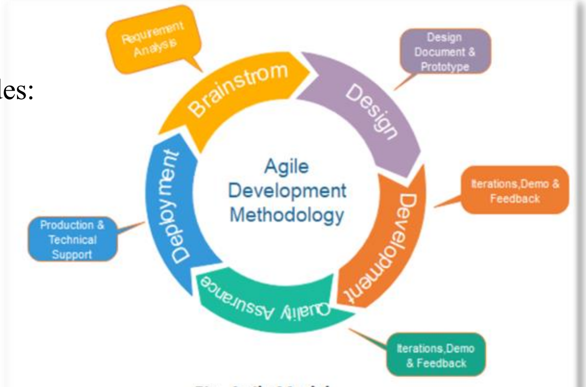
## I. Project Context

This project is all about a tool that can encrypt and decrypt a plaintext/ciphered text. Our team chose the Caesar cipher as our tool in encryption and decryption of plaintext/ciphered text. Our tool supports in reading a file which can be ciphered and decrypt the text file. Also, it can store the output to another text file. The tool also supports mod 36 which are the alphanumeric values. The programming language that we chose is C++ for this implementation.

## II. Methodology

Our team anticipates that in some aspects of our program code there will be changes or errors along the road, because "there is no perfect system and all system have flaws"[Contillo, 2023]. So, we chose Agile Methodology for implementing our cipher tool.

The Agile Methodology phases includes:
1. Requirements gathering
2. Designing requirements
3. Construction/iteration
4. Testing/Quality assurance
5. Deployment
6. Feedback


Requirement Analysis
Brainstrom
Design Document & Prototype
Design
Agile Development Methodology
Iterations,Demo & Feedback
Development
Production & Technical Support
Deployment
Quality Assurance
Iterations,Demo & Feedback
*Fig. Agile Model*

1. **BRAINSTORM:**
   For the first part of the gathering requirements our team researched on what type of cipher tool to be used in our encryption and decryption tool. There are two options we discussed, which are the Vigenere Cipher and, Caesar Cipher.

2. **DESIGN AND DEVELOPMENT:**
   The design DEVELOPMENT Initially, our team decided to push through Vigenère Cipher. The code is already done and we thought we are ready to present our final project. But upon reviewing the requirements given, the cipher code must be able to hash the file. Here, we faced a problem so we switched to Caesar cipher

which is our other option of cipher tool and since it's quite hard for us to hash file in Vigenère, Caesar is the other resort. We discussed and started working on Caesar cipher. The code is working with our desired functions or behavior as we expected. It can encrypt and decrypt a plain text as well as a file.

3. **QUALITY ASSURANCE:**

   As a group who needs constant reassurance of a quality of the program, we approached our Laboratory Instructor- Sir Franco for initial checking and to hear his feedback. There is just one thing that he corrected and that is the encryption process. Originally, our code loops for A-Z and 0-9 but Sir Franco suggested that it should be A-9 (alphanumeric). After hearing his feedback, we worked on our code for final checking.

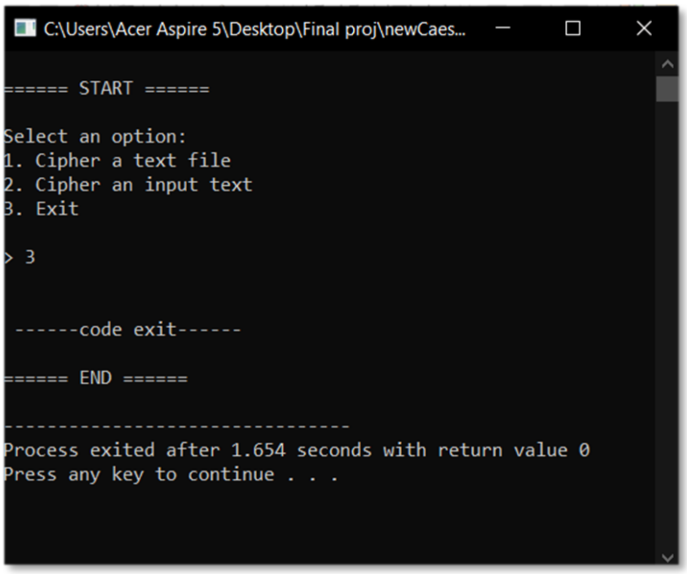
4. **DEPLOYMENT:**

   After finalizing our project, from Vigenere cipher tool switching to Caesar cipher tool and, modifying our program little did we know that we are left with one more obstacle. That is having trouble with our device during final presentation. Upon encrypting a huge amount of file, our device keeps overloading. As a solution. Sir Franco let's us use his laptop. Finally the program runs and ciphered the desired requirements in presenting our final project.

   Overall, the functions and behaviors of our cipher tool delivers and satisfies the requirement that our team assigned. We faced troubles along the process but we resolved and can say that we made a cipher tool.

## III. System Design

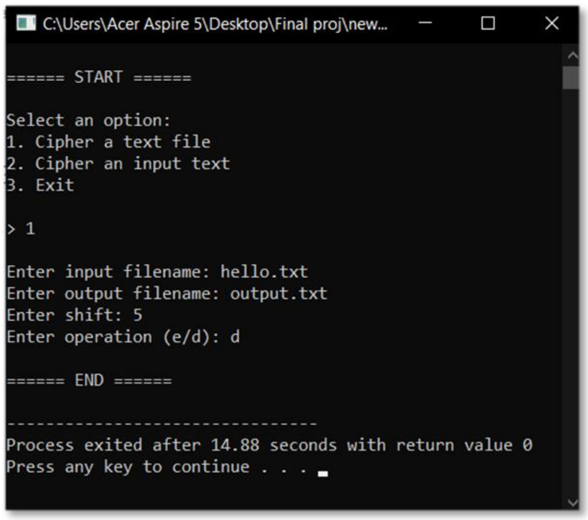
Our system design can be used in the command line interface. The user will be greeted with a selection, as you can see in fig.1. We input '3' as a sample that the code is terminated, if the user inputs or selects the 3rd option.



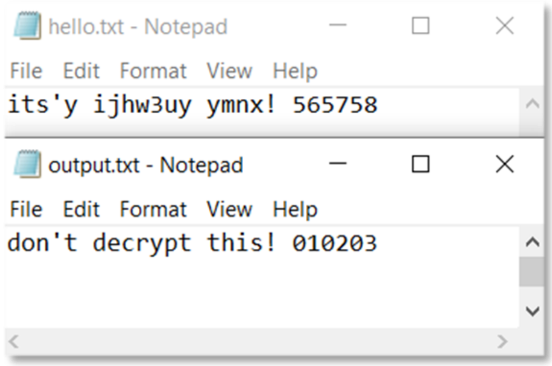
```
====== START ======

Select an option:
1. Cipher a text file
2. Cipher an input text
3. Exit

> 3


 ------code exit------

====== END ======

--------------------------------
Process exited after 1.654 seconds with return value 0
Press any key to continue . . .
```

**Fig.1**

In fig.2, if the user inputs '1' to the terminal, the program will cipher/decrypt a text file. Also, in fig.2 we can see that the 'hello.txt' file is decrypted and saved in 'output.txt', we can check the output/texts of the file in the Fig 2. And the code works perfectly fine.



```
====== START ======

Select an option:
1. Cipher a text file
2. Cipher an input text
3. Exit

> 1

Enter input filename: hello.txt
Enter output filename: output.txt
Enter shift: 5
Enter operation (e/d): d

====== END ======

--------------------------------
Process exited after 14.88 seconds with return value 0
Press any key to continue . . .
```

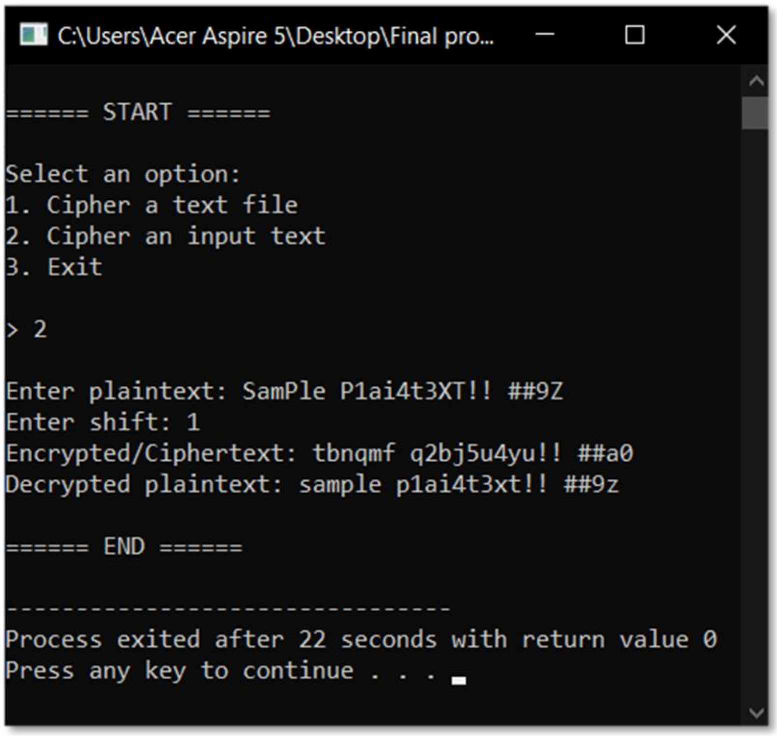


```
its'y ijhw3uy ymnx! 565758
```

```
don't decrypt this! 010203
```

**Fig.2**

For the remaining option, if user opt to enter '2' in the terminal it will ask the user to input a plaintext that he wants to cipher and the number of key shift. After entering the inputs, it will automatically display the ciphered and decrypted text in the terminal.



```
====== START ======

Select an option:
1. Cipher a text file
2. Cipher an input text
3. Exit

> 2

Enter plaintext: SamPle P1ai4t3XT!! ##9Z
Enter shift: 1
Encrypted/Ciphertext: tbnqmf q2bj5u4yu!! ##a0
Decrypted plaintext: sample p1ai4t3xt!! ##9z

====== END ======

--------------------------------
Process exited after 22 seconds with return value 0
Press any key to continue . . .
```

**Fig. 3**

## IV. Source Code

```
#include <iostream>
#include <string>
#include <fstream>
```

```
using namespace std;

string alphanum = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
int index, shiftedIndex, j;

string toUpperCase(string plaintext)
{
    /*
    **  this function convert a lowercase letter to uppercase...
    */

    string plaintextUpper;
    for (int i = 0; i < plaintext.length(); i++)
    {
        if (plaintext[i] >= 'A' && plaintext[i] <= 'Z')
            // if character is uppercase
            plaintextUpper += plaintext[i];

        else if (plaintext[i] >= 'a' && plaintext[i] <= 'z')
            // if character is lowercase
            plaintextUpper += plaintext[i] + 'A' - 'a';
        else
            // check if it's a special char or num
            plaintextUpper += plaintext[i];
    }
    return plaintextUpper;
}

string sanitize(string text)
{
    /*
    **this function converts a character from uppercase to lowercase
    */

    string sanitized;
    for (int i = 0; i < text.length(); i++)
    {
        if (text[i] >= 'a' && text[i] <= 'z')
            // if the input is a small character
            sanitized += text[i];
        else if (text[i] >= 'A' && text[i] <= 'Z')
            // if the current char is uppercase then store the smaller char.
            sanitized += text[i] + ('a' - 'A');
        else if ((text[i] >= '0' && text[i] <= '9') || (text[i] < 'A' || text[i] >
'Z'))
            // if current char is digit or special characters
            sanitized += text[i];
    }
    return sanitized;
}

string encrypt(string plaintext, int shift)
{
    /*
    ** Function in encryption; Formula: Ei = Pi + K
    */

    string plaintextUpper = toUpperCase(plaintext);

    string ciphertext;

    for (int i = 0; i < plaintextUpper.length(); i++)
    {
        char currentChar = plaintextUpper[i];
        if   ((plaintextUpper[i]   >=   'A'   &&   plaintextUpper[i]   <=   'Z')   ||
(plaintextUpper[i] >= '0' && plaintextUpper[i] <= '9'))
        {
            j = 0;
            while (true)
            {
                // this loop checks the index of a current character in plaintext,
to alphanum string variable
                if (alphanum[j] == currentChar)
                {
                    index = j;
                    break;
                }
                j++;
            }

            shiftedIndex = (index + shift) % 36; // Ei = Pi + K

            ciphertext += alphanum[shiftedIndex];
        }
        else

            ciphertext += plaintextUpper[i];
    }

    return ciphertext;
}

string decrypt(string ciphertext, int shift)
{
    /*
    ** Function in decryption; Formula: Ei = Pi + K
    */

    string plaintext;

    for (int i = 0; i < ciphertext.length(); i++)
    {
        char currentChar = ciphertext[i];
        if ((ciphertext[i] >= 'A' && ciphertext[i] <= 'Z') || (ciphertext[i] >= '0'
&& ciphertext[i] <= '9'))
        {
            j = 0;
            while (true)
            {
                // this loop checks the index of a current character in plaintext,
to alphanum string variable
                if (alphanum[j] == currentChar)
                {
                    index = j;
                    break;
                }
                j++;
            }

            shiftedIndex = (index - shift) % 36; // Ei = Pi - K

            if (shiftedIndex < 0)
                shiftedIndex = (36 + shiftedIndex) % 36;

            plaintext += alphanum[shiftedIndex];
        }
        else

            plaintext += ciphertext[i];
    }

    return plaintext;
}

void textCipher()
{
    /*
    **  driver function, for cipher an input from the user
    */

    // variables
    string plaintext;
    string ciphertext, decrypted;
    int shift;

    // input
    cout << "Enter plaintext: ";
    getline(cin, plaintext);

    cout << "Enter shift: ";
    cin >> shift;

    // function call and storing
    ciphertext = encrypt(plaintext, shift);
    decrypted = decrypt(ciphertext, shift);

    // printing
    cout << "Encrypted/Ciphertext: " << sanitize(ciphertext) << endl;
    cout << "Decrypted plaintext: " << sanitize(decrypted) << endl;
}

void fileCipher()
{
    /*
    **  driver function, for cipher a txt file from the user
    */

    //  variables
    string filename, output_filename;
    string line, output;
    int shift;
    char operation;

    // inputs
    cout << "Enter input filename: ";
    cin >> filename;

    cout << "Enter output filename: ";
    cin >> output_filename;

    cout << "Enter shift: ";
    cin >> shift;

    cout << "Enter operation (e/d): ";
    cin >> operation;

    // obj call
    ifstream input_file(filename.c_str());
    ofstream output_file(output_filename.c_str());

    if (!input_file)
    {
        // if file doesn't exists or something is wrong with the file
        cerr << "Error opening file " << filename << endl;
        return;
    }

    while (getline(input_file, line))
    {
        if (operation == 'e')
            output += sanitize(encrypt(line, shift)) + "\n";

        else if (operation == 'd')
            output += sanitize(decrypt(toUpperCase(line), shift)) + "\n";
    }
    output_file << output << endl; // save the output

    input_file.close();
    output_file.close();
}

int main()
{
    // main driver..

    int option;
    cout << "\n====== START ======\n"
         << endl;

    cout << "Select an option:\n1. Cipher a text file\n2. Cipher an input text\n3.
Exit\n\n> ";
    cin >> option;
    cout << endl;
    cin.ignore();

    switch (option)
    {
    case 1:
        fileCipher();
        break;
    case 2:
        textCipher();
        break;
    case 3:
        cout << "\n ------code exit------" << endl;
        break;
    default:
        cout << "\nInvalid input!\n -----code terminated----" << endl;
        break;
    }

    cout << "\n====== END ======" << endl;
    return 0;
}
```