|  |
| --- |
|  |

|  |
| --- |
| **PAT Gr 10 Phase 1** |
| Planning of Ciphet/Decipher application |
|  |
| Cipher and deciphering are discussed. The user requirement is explained by a Ucase diagram and a table. A flow diagram is used to demonstrate the flow of the application. A GUI is designed for the application and the data structures are described by using Input, Processing, Output tables. |
|  |
| **Naam Van** |
| **[Pick the date]** |
|  |

INDEX

[1 Introduction 1](#_Toc44163017)

[2 Scenario: Encryption/Decryption 1](#_Toc44163018)

[3 User requirements 2](#_Toc44163019)

[3.1 User-Case diagram 2](#_Toc44163020)

[3.2 Table 2](#_Toc44163021)

[4 Flow Diagram 2](#_Toc44163022)

[5 Graphical User Interface 3](#_Toc44163023)

[6 Data Dictionary 4](#_Toc44163024)

[7 Data Structures 4](#_Toc44163025)

[7.1 Button: Generate Mono Alphabet 4](#_Toc44163026)

[7.2 Button: Cipher Text 4](#_Toc44163027)

[7.3 Button: Decipher Text 5](#_Toc44163028)

[7.4 Button: Reset 5](#_Toc44163029)

PHASE 1

# Introduction

This is an example of good planning document that you can use to create an Encryption/Decryption application. You can use this example, or you can another Cipher/Decipher code that you can find in literature, or you can create your own Cipher/Decipher code. Keep in mind, the more creative you are in creating your project, will result in better marks.

# Scenario: Encryption/Decryption

Sometimes one wants to ensure that other people cannot read one’s correspondence or messages or that personal information remains confidential. One way of keeping written information confidential or secret is to encrypt the message using a cipher. Some people, on the other hand, may like the art of cryptanalysis or the challenge to crack ciphers.

The purpose of this project is to develop a program that encrypts messages using a cipher and decrypts messages using the same cipher. For example, if someone sends an encrypted message to another person, that person must be able to use the program to decrypt the message.

This example uses the Mono-alphabetic Cipher method to develop this project. The Mono-alphabetic Cipher is classified under substitution. Substitutions is when one character is substituted with another character according to the cipher- and mono-alphabet. During the duration of the message both the alphabets must stay unchanged.

In the example described below each alphabet letter corresponds with a different mono-alphabet

For example

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Alphabet | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | Q | r | s | t | u | v | w | x | y | z |
| Mono-alphabet | J | A | S | E | F | C | H | B | X | W | Y | P | N | Q | I | D | 0 | G | K | U | L | R | M | T | V | Z |

Plaintext: Ciphertext:

BFPPI  
MIGPE

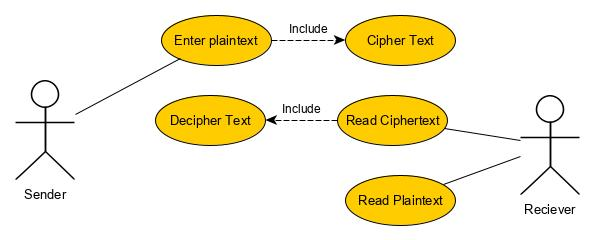
hello  
world

Just imagine, by rearranging the alphabet we know with different combinations there are over 400,000,000,000,000,000,000,000,000 possibilities. By using cryptoanalysis (code breaking) it is possible to crack the code.

# User requirements

## User-Case diagram

Cipher/Decipher App



## Table

|  |  |  |
| --- | --- | --- |
|  | **User 1: Sender** | **User 2: Receiver** |
| Role | The role of the sender is to create a message that he/she wants to send to the receiver | The role of the receiver is to read the message the sender sent to him/her |
| Activity | Enter the message  Send the message | Receive the message  Read the message |
| Limitations | The sender does not know how to cipher the plaintext | The receiver does not know how to decipher the ciphertext |

# Flow Diagram

The sender of the message will enter a text that he/she wants to send to someone. When the sender sends the message, the message must be ciphered. This is to ensure that when someone intercepts the message that they cannot read it. When the message is received it will be deciphered and the receiver will be able to read the message.

# Graphical User Interface

Form: Mono-Alphabet

A

L

P

H

A

B

E

T

Label

a

b

c

d

e

f

ghijklmnopqrstuvwxyz

Memo Box

Group Box

Scramble Alphabet

Button

Enter Plain Text:

Cipher Text

Enter Plain Text:

Cipher Text

Group Box

Button

Edit box

Rich Edit

Group Box

Button

Edit box

Rich Edit

Reset

Close

BitButton

M

O

N

O

A

L

P

H

A

B

E

T

Figure 1: Mono-Alphabetic Cipher

# Data Dictionary

A text file is used to store a Mono-Alphabet.

# Data Structures

## Button: Generate Mono Alphabet

|  |  |  |
| --- | --- | --- |
| **Input** | **Processing** | **Output** |
| Text File | If file exists  load from file  If file do not exist  create error message | Display the content of the text file  Display error message |

## Button: Cipher Text

|  |  |  |
| --- | --- | --- |
| **Variable** | **DataType** | **Description** |
| sPlainText | String | Stores the plain text entered by the user |
| cLetter | Char | Stores a letter of the plain text |
| iPos | Integer | Determine the position of a letter of the plain text in the alphabet |
| sCipherText | String | Stores the CipherText that was ciphered |

**Input, processing and output IPO**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Source** | **Data Type** | **Format of the Input** | **GUI component** | |
| Plain Text | Keyboard | String | Sentence in English | Edit box | |
| **Data Validation** | Range check  sPlainText[i] IN [‘A’..’Z, ‘a..z’, ‘ ‘]  Error message 1  “Only Alphabet letters and spaces allowed” | | | | |
|  | Null Check  sPlainText = ’’  Error message 2  “Edit cannot be empty, provide a message to encrypt” | | | | |
| **Processing** | **Action** | | | | **GUI component** |
| Generate Cipher text | WHAT:  Capture plain text and encrypt to Cypher text  HOW:  Read plain text  Do null check if invalid show Error message 1  else  Do range check if invalid show error message 2 else //encrypt  FOR 1 to the length of the Plain text  Letter 🡨 a letter from the plain text  Determine the position of the letter in the alphabet  Add each corresponding letter to the Cipher text | | | | Button |
| **Output** | **Format** | | | | **GUI component** |
| Cipher Text | String | | | | RichEdit |

## Button: Decipher Text

|  |  |  |
| --- | --- | --- |
| **Variable** | **DataType** | **Description** |
| sCipherText | String | Stores the cipher text entered by the user |
| cLetter | Char | Stores a letter of the plain text |
| iPos | Integer | Determine the position of a letter of the plain text in the alphabet |
| sPlainText | String | Stores the Plain Text that was deciphered |

**Input, processing and output**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Source** | **Data Type** | **Format of the Input** | **GUI component** |
| Cipher Text | Keyboard | String | Sentence in Cipher Text | Edit box |
| **Data Validation** | Range check  sPlainText[i] IN [‘A’..’Z, ‘a..z’, ‘ ‘]  Error message 1  “Only Alphabet letters and spaces allowed” | | | |
|  | Null Check  sPlainText = ’’  Error message 2  “Edit cannot be empty, provide a message to encrypt” | | | |
| **Processing** | **Action** | | | **GUI component** |
| Decipher Plain Text | WHAT:  Capture cipher text and decrypt to plain text  HOW:  Read plain text  Do null check if invalid show Error message 1  else  Do range check if invalid show error message 2 else //decrypt  FOR 1 to the length of the Plain text  Letter 🡨 a letter from the cipher text  Determine the position of the letter in the mono alphabet  Add each corresponding letter to the plain text | | | Button |
| **Output** | **Format** | | | **GUI component** |
| Decipher Text | String | | | RichEdit |

## Button: Reset

**Processing**

|  |  |  |
| --- | --- | --- |
| **Processing** | **Action** | **GUI component** |
| Clear relevant objects | Clear the text of the mono alphabet  Clear the plain text  Clear the Cipher Text | bitButton |