# Documentation for VBA Encryption-Decryption Project

**Introduction**
The VBA Encryption-Decryption project is designed to perform various encryption and decryption methods, including Caesar cipher, Playfair cipher, and Vigenere cipher. The project is structured around a module and six forms, facilitating user interaction and input for text and encryption preferences.

**Module Overview**
The main module orchestrates the entire encryption/decryption process. It contains functions for each encryption method and manages the user interface through form interactions.

**Variables**
- `**processType:** Stores the type of process - encryption or decryption.
- `**userText:** Holds the text provided by the user.
- `**chosenMethod:** Represents the selected encryption method.
- `**resultText:** Stores the resulting encrypted or decrypted text.
- `**userKeyword:** Stores the keyword used for certain encryption methods.
- `**shiftValue:** Holds the value for the shift in the Caesar cipher.
- `**cancelStatus:** Tracks if the user cancels any step in the process.

**Subroutines and Functions**
- **StartApplication():** Acts as the main controller, displaying forms and guiding the user through the encryption/decryption process based on the chosen method.

The `StartApplication()` subroutine is the core controller that initiates the encryption/decryption process. Here's a breakdown of its essential parts:

### *Initialization and Form Display*

- `cancelStatus = False`: Sets the cancellation status to false initially.
- `Enc_Dec_form.Show`: Displays the form for encryption or decryption selection.
- `InputText_form.Show`: Shows the form for inputting text data.
- `Encryption_form.Show`: Displays the form for choosing the encryption method.

### *Conditional Actions based on Chosen Method*

- **Encryption Method Check**: Determines the chosen encryption method:
  - `If chosenMethod = "CAESAR" Then ... End If`: Checks if the method is Caesar cipher.
  - `If chosenMethod = "FAIRPLAY" Then ... End If`: Checks if the method is Playfair cipher.
  - `If chosenMethod = "VIGENERE" Then ... End If`: Checks if the method is Vigenere cipher.

### *Interaction with User Input Forms*

- **Forms Show & Cancel Status Check**: Each method involves displaying specific forms and checking for cancellation:
  - `Shift_form.Show`: Displays the shift form for Caesar cipher if chosen.
  - `Keyword_form.Show`: Displays the keyword form for Playfair and Vigenere ciphers if chosen.
  - `If cancelStatus = True Then Exit Sub`: Checks if the user canceled the process and exits if true.

### *Encryption or Decryption Process*

- **Execution of Encryption/Decryption Functions**: Based on the selected method and process type, executes the appropriate encryption or decryption function.

### *Result Storage and Display*

- `Sheets("Sheet1").Range("A1").Value = resultText`: Stores the resulting text in cell A1 of "Sheet1".
- `Result_form.Show`: Displays the form to show the resulting encrypted/decrypted text.

### *Comments and Message Boxes (Optional)*

- **Comments**: Commentary lines explaining the logic flow (commented out for execution).
- **Message Boxes**: Debugging or informative message boxes (commented out for execution).

- **EncryptText():** Implements the Caesar cipher encryption.

***Purpose:***

- **Encrypts Text using Caesar Cipher**: This function is designed to perform text encryption using the Caesar cipher method.

***Parameters:***

- `inputText` (String): Represents the text to be encrypted.
- `shift` (Integer): Indicates the shift value for the Caesar cipher.

***Breakdown of the Function:***

1. **Initialization**:
    - Declares `Result` as an empty string to store the encrypted text.
    - Initializes a loop to iterate through each character of the `inputText`.
2. **Character Processing**:
    - Retrieves each character from the `inputText` using `Mid` function based on the iteration.
    - Checks if the character is a letter:
        - If uppercase (`A` to `Z`), shifts its ASCII value by the given `shift` and wraps it within the range of alphabets using modulo operation.
        - If lowercase (`a` to `z`), performs a similar shift operation for lowercase characters.
3. **Concatenation**:
    - Appends the encrypted character to the `Result` string.
4. **Loop Conclusion**:
    - Continues this process for each character in the `inputText`.
5. **Return Value**:
    - Returns the `Result` string containing the fully encrypted text.

- **DecryptText():** Implements the Caesar cipher decryption.

***Purpose:***

- **Decrypts Text Encrypted with Caesar Cipher**: This function facilitates the decryption process by inversely shifting the encrypted text.

### *Parameters:*

- `inputText` (String): Represents the text to be decrypted.
- `shift` (Integer): Signifies the original shift value used for encryption.

### *Breakdown of the Function:*

1. **Inverse Shift Calculation**:
    - Calculates the inverse shift required for decryption by subtracting the original `shift` from 26 (total number of alphabets used in the Caesar cipher).
2. **Calling the Encryption Function**:
    - Utilizes the `EncryptText` function, passing `inputText` and the calculated inverse shift value as parameters.
    - The `EncryptText` function, with the inverse shift, essentially performs the decryption operation.
3. **Return Value**:
    - Returns the result obtained from the `EncryptText` function, which contains the decrypted text.

- **`PlayfairEncrypt()`**: Performs encryption using the Playfair cipher.

### *Purpose:*

- **Encrypts Text using Playfair Cipher**: This function encrypts plaintext using the Playfair cipher technique.

### *Parameters:*

- `inputText` (String): Represents the text to be encrypted.
- `key` (String): Denotes the keyword used to generate the Playfair matrix.

### *Breakdown of the Function:*

1. **Initialization**:
    - Initializes `resultText` as an empty string to store the encrypted text.
2. **Key Processing**:
    - Standardizes the `key` by converting it to uppercase, replacing 'J' with 'I', and appending the remaining alphabet characters.

- Removes duplicate characters from the standardized key to form the basis of the Playfair matrix.

3. **Matrix Generation**:
   - Constructs a 5x5 matrix for the Playfair cipher using the processed `key`.
   - Assigns characters from the standardized key to the matrix in a row-major order.
4. **Input Text Preparation**:
   - Converts `inputText` to uppercase, replaces 'J' with 'I', and removes non-letter characters.
   - Adjusts the input text for Playfair cipher by pairing consecutive characters and adding an 'X' if necessary.
5. **Pair Processing and Encryption**:
   - Iterates through the prepared input text in pairs.
   - Determines the positions (rows and columns) of the character pairs in the Playfair matrix using `FindPosition` function.
   - Applies Playfair rules for encryption based on row and column positions of the character pairs.
     - If the characters are in the same row, replaces them with the character to their right (circular).
     - If the characters are in the same column, replaces them with the character below (circular).
     - Otherwise, substitutes them with characters from the same row but with different columns.
6. **Result Assembly**:
   - Concatenates the encrypted characters to the `resultText`.
7. **Return Value**:
   - Returns the `resultText` containing the fully encrypted text using the Playfair cipher.

- **PlayfairDecrypt():** Performs decryption using the Playfair cipher.

***Purpose:***

- **Decrypts Text Encrypted with Playfair Cipher**: This function decrypts ciphertext encrypted with the Playfair cipher method.

### *Parameters:*

- `inputText` (String): Represents the text to be decrypted.
- `key` (String): Represents the keyword used for generating the Playfair matrix.

### *Breakdown of the Function:*

1. **Initialization**:
    - Initializes `resultText` as an empty string to store the decrypted text.
2. **Key Processing and Matrix Generation**:
    - Standardizes the `key` by converting it to uppercase, replacing 'J' with 'I', and appending the remaining alphabet characters.
    - Removes duplicate characters to form the basis for the Playfair matrix.
    - Constructs a 5x5 matrix for the Playfair cipher using the processed `key` in a row-major order.
3. **Input Text Preparation**:
    - Converts `inputText` to uppercase and removes non-letter characters.
    - Adjusts the input text for Playfair decryption by pairing consecutive characters and preparing them for decryption.
4. **Pair Processing and Decryption**:
    - Iterates through the prepared input text in pairs.
    - Determines the positions (rows and columns) of the character pairs in the Playfair matrix using the `FindPosition` function.
    - Applies Playfair decryption rules based on the row and column positions of the character pairs:
        - If the characters are in the same row, replaces them with the character to their left (circular).
        - If the characters are in the same column, replaces them with the character above (circular).
        - Otherwise, substitutes them with characters from the same row but with different columns.
5. **Result Assembly**:
    - Concatenates the decrypted characters to the `resultText`.
6. **Return Value**:

- Returns the `resultText` containing the fully decrypted text using the Playfair cipher.

### - RemoveDuplicates()

***Purpose:***

- **Removes Duplicate Characters**: This function eliminates duplicate characters from a given input text.

***Parameter:***

- `inputText` (String): Represents the text containing potential duplicate characters.

***Breakdown of the Function:***

1. **Initialization**:
   - Initializes `outputText` as an empty string to store the text without duplicates.
2. **Duplicate Removal Logic**:
   - Iterates through each character of the `inputText`.
   - Checks if the character at the current position (using `Mid` function) is already present in the `outputText`.
   - If the character is not found in `outputText` (using `InStr`), it is appended to `outputText`.
   - This process effectively eliminates duplicate characters from the `inputText`.
3. **Return Value**:
   - Returns the `outputText` string devoid of duplicate characters.

### , RemoveNonLetters()

***Purpose:***

- **Removes Non-Alphabetic Characters**: This function eliminates characters that are not within the range of alphabetic letters (A-Z) from a provided text.

***Parameter:***

- `inputText` (String): Represents the text containing characters to be filtered.

***Breakdown of the Function:***

1. **Initialization**:

- Initializes `outputText` as an empty string to store the filtered text.

2. **Non-Alphabetic Character Removal**:
   - Iterates through each character of the `inputText`.
   - Checks if the character at the current position (using `Mid` function) is within the ASCII range of uppercase letters (A-Z).
   - If the character's ASCII value falls within the range of uppercase letters, it is appended to the `outputText`.
   - This process effectively filters out characters that are not letters from the `inputText`.
3. **Return Value**:
   - Returns the `outputText` string containing only alphabetic characters.

**, **`AdjustInputText()`**:**

***Purpose:***

- **Pairs Characters and Inserts 'X' if Necessary**: This function processes the input text by pairing characters and adding an 'X' between consecutive identical characters when needed for the Playfair cipher.

***Parameter:***

- `inputText` (String): Represents the text to be adjusted for Playfair cipher processing.

***Breakdown of the Function:***

1. **Initialization**:
   - Initializes `adjustedText` as an empty string to store the processed text.
2. **Character Pairing and 'X' Insertion**:
   - Iterates through the `inputText` in pairs of characters.
   - Checks each pair of characters and assigns them to `pair1` and `pair2`.
   - If the second character in the pair is not available (i.e., end of text), assigns 'X' to `pair2`.
   - Compares `pair1` and `pair2`:

- If the characters in the pair are identical, appends `pair1` and 'X' to `adjustedText`.
- Otherwise, appends both `pair1` and `pair2` to `adjustedText`.

3. **Return Value**:
    - Returns the `adjustedText` string containing paired characters with 'X' added as required for the Playfair cipher.

Helper functions for processing text for the Playfair cipher.

**- FindPosition()**

***Purpose:***

- **Locates Position in Matrix**: This subroutine finds the row and column indices of a specified letter within a given matrix.

***Parameters:***

- `matrix` (Variant): Represents the matrix structure containing characters.
- `letter` (String): Represents the character to be located within the matrix.
- `rowIdx` (ByRef Integer): Stores the row index of the found letter.
- `colIdx` (ByRef Integer): Stores the column index of the found letter.

***Breakdown of the Subroutine:***

1. **Searches Matrix**:
    - Iterates through the rows and columns of the `matrix`.
    - Checks if each element (`matrix(i, j)`) matches the specified `letter`.
    - If a match is found:
        - Records the current `i` (row) and `j` (column) indices.
        - Exits the loop.

Locates positions in the Playfair cipher matrix.

**EncryptVigenere()**

***Purpose:***

- **Encrypts Text Using Vigenere Cipher**: This function performs encryption using the Vigenere cipher.

### *Parameters:*

- `inputText` (String): Represents the text to be encrypted.
- `Keyword` (String): Represents the keyword used for encryption.

### *Breakdown of the Function:*

1. **Initialization**:
    - Initializes variables for tracking the encryption process.
2. **Text Encryption**:
    - Converts `inputText` and `Keyword` to uppercase.
    - Calculates the length of the `Keyword`.
    - Iterates through each character in `inputText`.
    - For alphabetic characters:
        - Retrieves the current character's shift value based on the keyword.
        - Applies the Vigenere encryption formula to each character.
        - Builds the encrypted `Result` string.
3. **Return Value**:
    - Returns the encrypted `Result` text.

## DecryptVigenere()

### *Purpose:*

- **Decrypts Text Encrypted by Vigenere Cipher**: This function performs decryption using the Vigenere cipher.

### *Parameters:*

- `inputText` (String): Represents the text to be decrypted.
- `Keyword` (String): Represents the keyword used for decryption.

### *Breakdown of the Function:*

1. **Initialization**:
    - Initializes variables for tracking the decryption process.
2. **Text Decryption**:
    - Follows a similar process to encryption but applies the inverse shift to decrypt the text.
    - Converts `inputText` and `Keyword` to uppercase.
    - Calculates the length of the `Keyword`.

- Iterates through each character in `inputText`.
- For alphabetic characters:
    - Retrieves the current character's shift value based on the keyword.
    - Applies the Vigenere decryption formula to each character.
    - Builds the decrypted `Result` string.

3. **Return Value**:
    - Returns the decrypted `Result` text.

**Implement Vigenere cipher encryption and decryption.**

**Forms Overview**
The project employs six forms to interact with users at various stages of the encryption/decryption process.

**Enc-Dec Form**
- Allows users to select encryption or decryption.
- Contains buttons to proceed or cancel.

**Encryption Form**
- Enables users to choose the encryption method (Caesar, Playfair, or Vigenere).
- Provides options to proceed or cancel.

**Input Text Form**
- Displays the text retrieved from the worksheet for user input.
- Offers an input field and cancel/confirm options.

**Keyword Form**
- Asks for the keyword input necessary for Playfair and Vigenere ciphers.
- Validates user input and allows cancellation.

**Shift Form**
- Specific to the Caesar cipher, allowing users to input the shift value or cancel.

**Code Analysis**

- **Structured Process:** The project is well-structured, guiding users through the encryption/decryption process step-by-step.
- **Modular Approach**: Functions are divided based on specific encryption methods, enhancing code readability and maintenance.
- **User Input Handling**: The forms ensure user inputs are captured, validated, and processed effectively.

**Code Highlights**

1. **Dynamic UI**: The forms manage user interactions seamlessly, displaying relevant information and guiding users through each step.
2. **Encryption Algorithms**: The implementation of Caesar, Playfair, and Vigenere ciphers demonstrates a clear understanding of cryptographic principles.
3. **Error Handling**: The code includes validation checks to ensure inputs meet specific requirements, enhancing user experience and preventing errors.

**Potential Improvements**

1. **Enhanced User Feedback**: Providing more detailed feedback to users about input errors or successful encryption/decryption could improve the user experience.
2. **Optimization**: The code could be optimized for efficiency, especially within the Playfair cipher functions where matrix creation and manipulation occur.
3. **Additional Encryption Methods**: Expanding the project to include more encryption methods could enhance its versatility.

**Conclusion**

The VBA Encryption-Decryption project demonstrates a structured approach to implement various encryption methods. Through intuitive forms and well-defined code, it offers users a practical tool for encrypting and decrypting text data. With potential enhancements and optimizations, it can further improve its functionality and user experience.