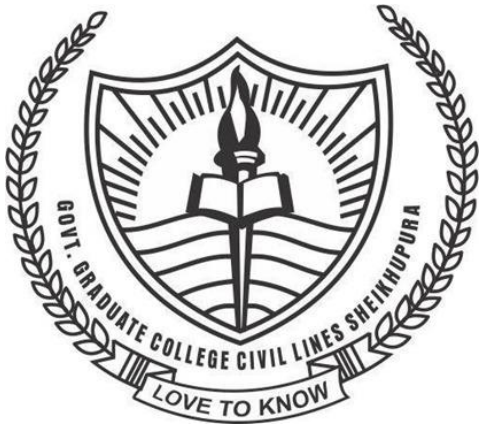



# Govt. Post Graduate College Civil Lines Sheikhupura

## Information Security Project

Ceasar Cipher GUI Web Project

## Submitted by

Muhammad Muzzamal 110837

Aakif Saleem 110840

Imran Ali 110855

## Submitted to

PROF. SEHRISH KHAN

Department of Information Technology

# Caesar Cipher Implementation Assignment

## Objective

Develop a complete Caesar Cipher encryption/decryption application while following software engineering principles including requirements analysis, design, implementation, testing, and documentation.

## Background

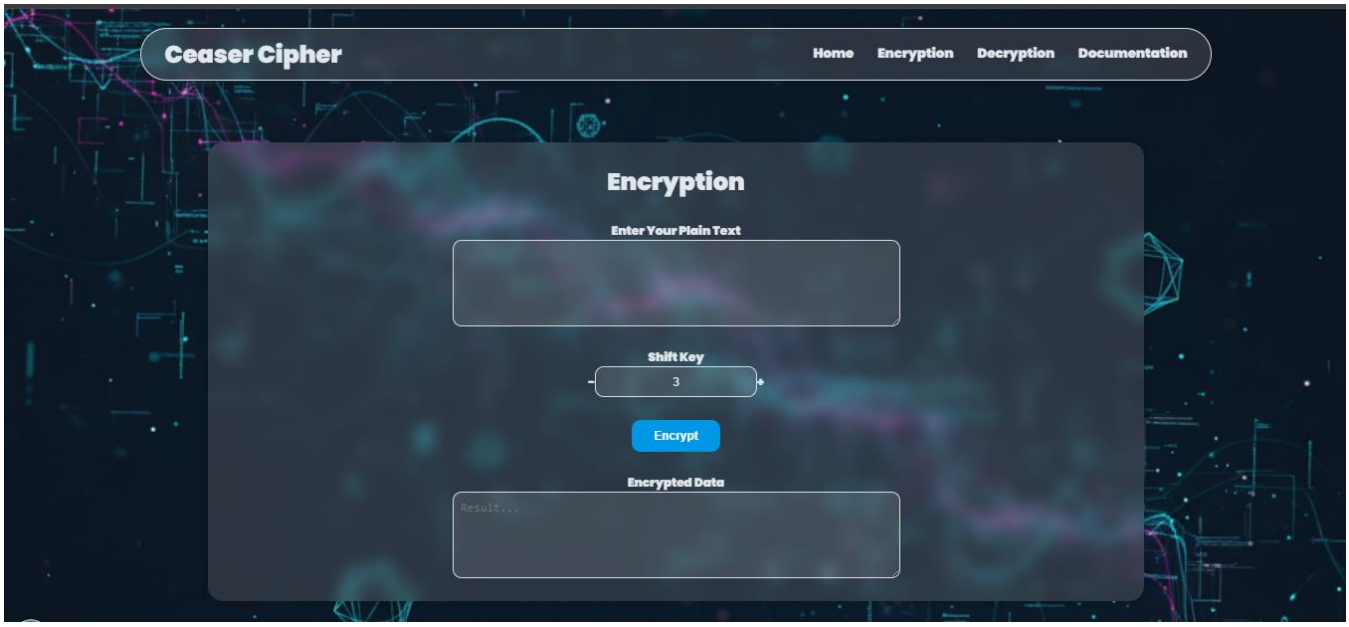
The Caesar Cipher, named after Julius Caesar, is one of the oldest encryption methods, used around 58–50 BCE to protect military messages. It works by shifting each letter of the alphabet by a fixed number of positions, with Caesar reportedly using a shift of three. Although simple and easy to break, it was effective in its time and laid the foundation for more advanced ciphers like the Vigenère Cipher and Enigma, remaining today as a fundamental example in the study of cryptography.

## Procedure

- Choose a **shift key (K)** — the number of positions each letter will be shifted in the alphabet.
- Take the **plaintext (original message)** that you want to encrypt.
- For each letter in the plaintext:
- Convert it to its **alphabetic position** (A = 0, B = 1, …, Z = 25).
- Apply the formula: **Cipher = (Plain + K) mod 26** for encryption.
- Replace each shifted position with its corresponding letter to get the **ciphertext**.
- To **decrypt**, use the reverse formula: **Plain = (Cipher − K) mod 26**.
- Keep non-alphabetic characters (like spaces or punctuation) **unchanged**.

## GUI IMPLEMENTATION

- Create the main structure using **HTML** — include sections for the **header**, **navigation bar**, and **main content area**.
- Add a **title** and description explaining the Caesar Cipher and its functionality.
- Design an **input area** where users can enter plaintext and specify the **shift key value**.
- Include **buttons** for actions such as **Encrypt**, **Decrypt**, **Increase Shift**, and **Decrease Shift**.
- Use **CSS** to style the webpage — design the layout, navigation bar, buttons, and text fields for a clean and user-friendly interface.
- Implement **JavaScript** logic to handle encryption and decryption based on user input.
- Finalize the design and functionality, ensuring the interface is responsive and easy to use.



# Algorithm of Encryption

1. **Get inputs**
    - Read the **plain text** (the message to encrypt).
    - Read the **shift key** (a small number that tells how many letters to move).
2. **Prepare the shift**
    - Make sure the shift is a valid number.
    - If the shift is negative or large, convert it to an equivalent between 0 and 25 (e.g., shift = ((shift % 26) + 26) % 26).
3. **Start an empty result string**
    - You will build the encrypted message character by character into this string.
4. **Go through each character of the plain text, one by one**
    - For each character, check what it is:
5. **If the character is an uppercase letter (A–Z)**
    - Find its position in the alphabet (A = 0, B = 1, …, Z = 25).
    - Add the shift value to that position.
    - If the result goes past Z, wrap it around to the start (use modulo 26).
    - Convert the new position back to an uppercase letter and append it to the result string.
6. **If the character is a lowercase letter (a–z)**
    - Do the same as for uppercase, but keep everything lowercase (a = 0 … z = 25).
    - Wrap with modulo 26 and append the resulting lowercase letter to the result.
7. **If the character is not a letter (space, punctuation, digit, etc.)**
    - Leave it unchanged and append it directly to the result string.
8. **Continue until all characters are processed**
    - Repeat steps 4–7 for every character in the input.
9. **Return the result string**
    - The final string is your encrypted message.

## Algorithm of Decryption

1. **Take Input**
   - Get the encrypted text (cipher text) and the shift key (same key used for encryption).
2. **Fix the Shift Value**
   - Same formula:
     ```
     shift = ((shift % 26) + 26) % 26
     ```
3. **Make an Empty Result String**
   - This will store your decrypted (plain) text.
4. **Go Through Each Character**
   - For every character in the cipher text:
5. **If it's an Uppercase Letter (A–Z)**
   - Convert to a number (A=0, B=1, …, Z=25).
   - **Subtract** the shift value.
   - If it becomes negative, add 26 to wrap it.
   - Convert back to a letter and add to result.
6. **If it's a Lowercase Letter (a–z)**
   - Same logic but with lowercase.
7. **If it's a Symbol, Space, or Number**
   - Leave it unchanged.
8. **After the Loop Ends**
   - The final string is your **decrypted (original) message**.

## Source Code:

https://github.com/Muhammad-Muzzamal/Ceasar-Cipher-Encryption-and-Decryption-GUI-Project

## Live Link:

https://ceacarcipher.netlify.app/