Introduction

Encryption plays a crucial role in cybersecurity. This project demonstrates a browser-based encryption tool where users can enter text, select a cipher method, provide a key, and receive the encrypted output. It is designed to introduce core encryption concepts in a hands-on manner.

Encryption Algorithms

1. Caesar Cipher

- **Concept**: Each letter in the plaintext is shifted by a fixed number (key) along the alphabet.
- **Example**: With a shift of `3`, `A` becomes `D`, `B` becomes `E`, etc.
- **Key Type**: Numeric (e.g., `3`)

Code:

```
function encryptCaesar(str, shift) {
  return str.toUpperCase().replace(/[A-Z]/g, char =>
    String.fromCharCode((char.charCodeAt(0) - 65 + shift) % 26 + 65)
  );
}
```

Explanation:

- Converts input to uppercase.
- Replaces each letter with its shifted version using ASCII.
- Uses modulo 26 to wrap around the alphabet (`A–Z`).

---

2. Vigenère Cipher

- **Concept**: A polyalphabetic cipher where each letter in the plaintext is shifted based on a repeating keyword.
- **Example**: `HELLO` with key `KEY` becomes encrypted with varying shifts depending on letters in `KEY`.
- **Key Type**: Alphabetic string (e.g., `KEY`)

Code:

```
function encryptVigenere(text, key) {
  text = text.toUpperCase();
  key = key.toUpperCase();
  let result = '';
  let j = 0;

  for (let i = 0; i < text.length; i++) {
    const char = text[i];
    if (/[A-Z]/.test(char)) {
      const shift = key.charCodeAt(j % key.length) - 65;
      result += String.fromCharCode((char.charCodeAt(0) - 65 + shift)
% 26 + 65);
      j++;
    } else {
      result += char;
    }
  }
  return result;
}
```

Explanation:

- Converts both text and key to uppercase.
- Iterates through each character in the plaintext.
- Shifts each letter based on the corresponding key letter.
- Skips non-alphabetic characters.

---

3. Rail Fence Cipher

- **Concept**: A transposition cipher that writes letters in a zigzag pattern across multiple "rails", then reads them line by line.
- **Example** (with 3 rails):

```
H . . . O . . . R
. E . L . W . D .
. . L . . . O . .
```

Result: `HORELWDLO`

- **Key Type**: Numeric (e.g., number of rails)

Code:

```
function encryptRailFence(text, rails) {
  if (rails < 2) return text;
  let fence = Array.from({ length: rails }, () => []);
  let rail = 0, dir = 1;

  for (let char of text) {
    fence[rail].push(char);
    rail += dir;
    if (rail === 0 || rail === rails - 1) dir *= -1;
  }

  return fence.flat().join('');
}
```

Explanation:

- Initializes an array for each rail.

- Zigzags across the rails, placing each character accordingly.
- Combines all rails to form the encrypted text.

UI Components

The user interface is designed with **HTML** and styled using **CSS**, featuring a modern dark-themed layout.

- **Text Area**: For user plaintext input.
- **Dropdown**: To choose the desired encryption algorithm.
- **Input Field**: For key input (number or keyword).
- **Encrypt Button**: Executes the encryption process.
- **Output Box**: Displays the final encrypted result.

JavaScript Code Explanation

Main Logic:

```
function encryptText() {
  const text = document.getElementById('inputText').value;
  const algorithm = document.getElementById('algorithm').value;
  const key = document.getElementById('key').value;
  let encrypted = '';

  if (algorithm === 'caesar') {
    const shift = parseInt(key);
    if (isNaN(shift)) return alert('Please enter a numeric key for
Caesar Cipher');
    encrypted = encryptCaesar(text, shift);
```

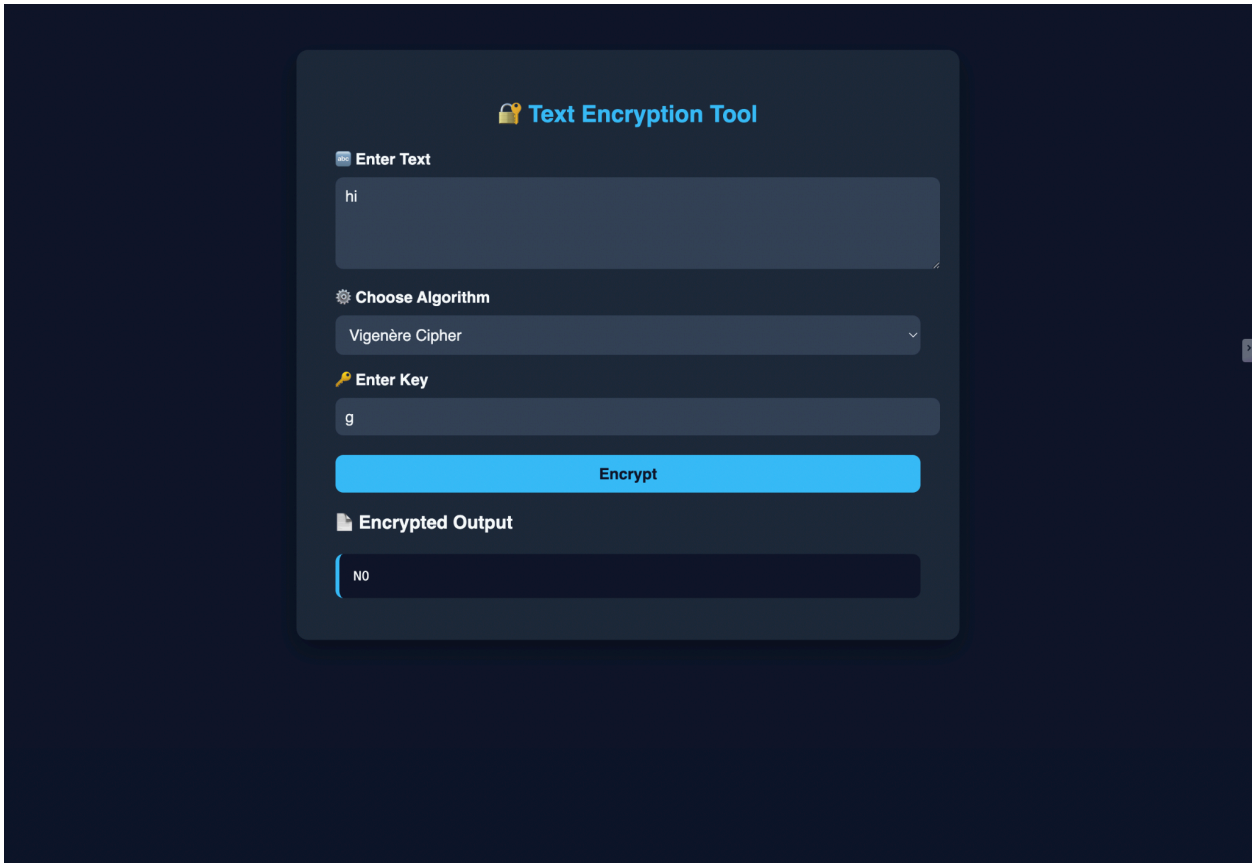
```
  } else if (algorithm === 'vigenere') {
    if (!key.match(/^[A-Za-z]+$/)) return alert('Keyword must contain
only letters for Vigenère Cipher');
    encrypted = encryptVigenere(text, key);

  } else if (algorithm === 'railfence') {
    const rails = parseInt(key);
    if (isNaN(rails) || rails < 2) return alert('Please enter a number
greater than 1 for Rail Fence Cipher');
    encrypted = encryptRailFence(text, rails);
  }

  document.getElementById('outputText').textContent = encrypted || 'No
text encrypted.';
}
```

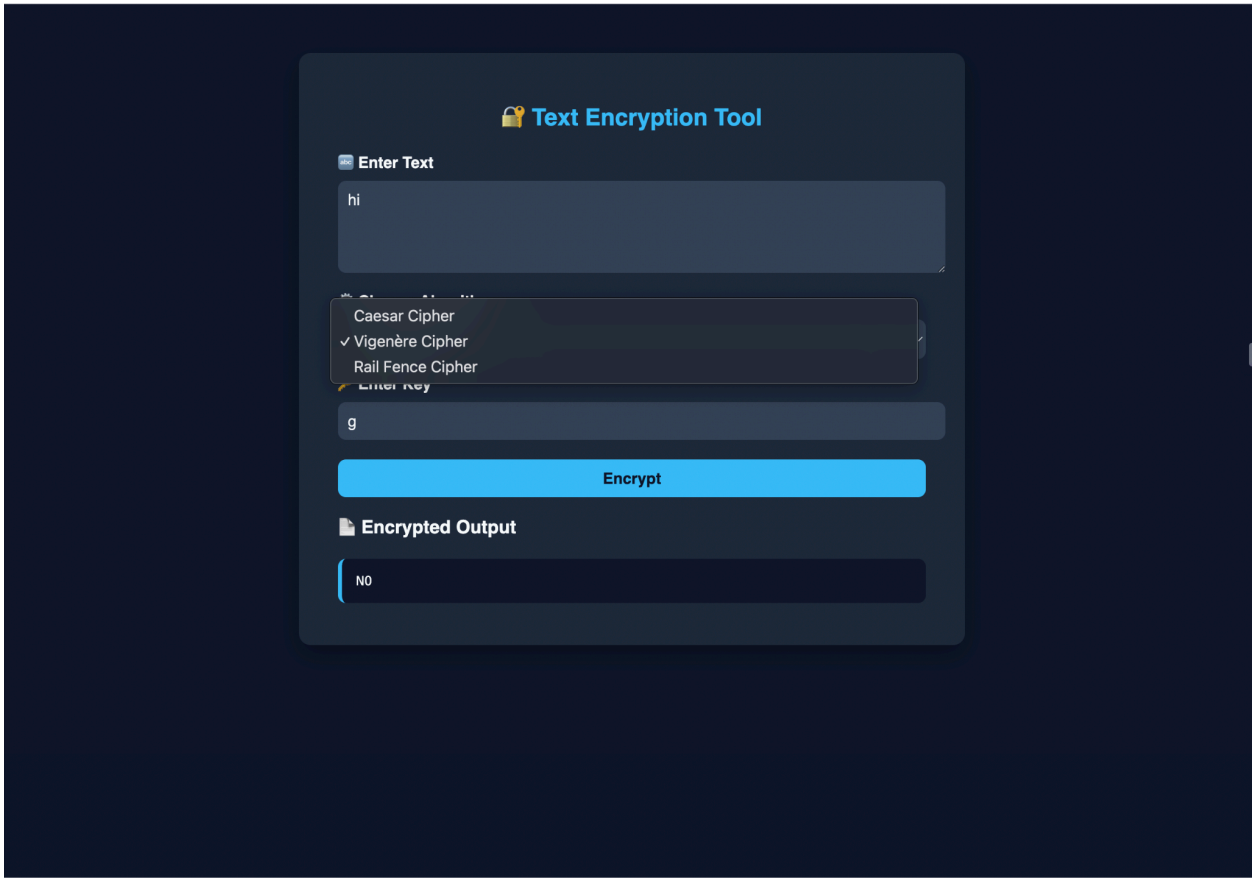
Explanation:

- Validates key input based on the chosen algorithm.
- Calls the corresponding encryption function.
- Displays the encrypted output in a styled container.

Git Hub Link:https://github.com/brahmaputraS/text_encryption.git

Screenshots:




Text Encryption Tool
Enter Text
hi
Caesar Cipher
Vigenère Cipher
Rail Fence Cipher
g
Encrypt
Encrypted Output
NO