- **BASIC PROJECT – CYBERSECURITY DOMAIN**

  Task 1 - Text Encryption Using Cryptographic Algorithms

  Description: You can build a simple web application to encrypt and decrypt textual information that the user keys in. Remember that strong encryption should produce different outputs even given the same input.

  Overview:

  This web application allows users to encrypt and decrypt textual information using strong cryptographic algorithms. The application is built using Flask for the backend and uses Fernet encryption from the cryptography library to ensure that the same input will produce different outputs each time it is encrypted.

  Project Structure:
  The project is structured as follows:

```
encryption_app/
├── app.py
├── templates/
│   └── index.html
├── static/
│   ├── css/
│   │   └── styles.css
│   └── js/
│       └── script.js
```

  app.py: Main Flask application file.
  templates/index.html: HTML template for the user interface.
  static/css/styles.css: CSS file for styling the web page.
  static/js/script.js: JavaScript file for handling user interactions and AJAX requests.

  Installation:
  Prerequisites
  Python pycharm
  Pip (Python package installer)

  Steps:
  1. Clone the repository or download the source code.
  2. Navigate to the project directory:
     -cd encryption_app
  3. Install the required Python packages:
     pip install Flask cryptography

Usage:
Running the Application
To start the Flask application, run the following command in your terminal:

- python app.py

This will start the server, and the application will be accessible at http://127.0.0.1:5000/.

Features:
Encrypt Text: Users can enter text in the text area and click the "Encrypt" button to encrypt the text.
Decrypt Text: Users can enter encrypted text in the text area and click the "Decrypt" button to decrypt the text.

File Details
app.py
The main Flask application file. It sets up the routes for rendering the page and processing encryption and decryption requests.

```
from flask import Flask, render_template, request, jsonify
from cryptography.fernet import Fernet

app = Flask(__name__)

# Generate a key for Fernet encryption
key = Fernet.generate_key()
cipher_suite = Fernet(key)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/encrypt', methods=['POST'])
def encrypt():
    plaintext = request.form['plaintext']
    encrypted_text = cipher_suite.encrypt(plaintext.encode())
    return jsonify({'encrypted_text': encrypted_text.decode()})

@app.route('/decrypt', methods=['POST'])
def decrypt():
    encrypted_text = request.form['encrypted_text']
    decrypted_text = cipher_suite.decrypt(encrypted_text.encode())
    return jsonify({'decrypted_text': decrypted_text.decode()})

if __name__ == '__main__':
    app.run(debug=True)
```

templates/index.html

The HTML template for the user interface.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Encryption App</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}">
</head>
<body>
  <h1>Text Encryption and Decryption</h1>
  <div>
    <textarea id="inputText" placeholder="Enter text here..."></textarea>
  </div>
  <div>
    <button onclick="encryptText()">Encrypt</button>
    <button onclick="decryptText()">Decrypt</button>
  </div>
  <div>
    <h2>Output</h2>
    <p id="outputText"></p>
  </div>
  <script src="{{ url_for('static', filename='js/script.js') }}"></script>
</body>
</html>
```

static/css/styles.css

The CSS file for styling the web page.

```
body {
  font-family: Arial, sans-serif;
  margin: 20px;
}

textarea {
  width: 100%;
  height: 100px;
  margin-bottom: 10px;
}

button {
  margin-right: 10px;
}

#outputText {
  white-space: pre-wrap;
```

```
    word-wrap: break-word;
}
```

static/js/script.js

The JavaScript file for handling user interactions and AJAX requests.

```
function encryptText() {
    const plaintext = document.getElementById('inputText').value;
    fetch('/encrypt', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/x-www-form-urlencoded',
        },
        body: `plaintext=${encodeURIComponent(plaintext)}`,
    })
    .then(response => response.json())
    .then(data => {
        document.getElementById('outputText').innerText = data.encrypted_text;
    })
    .catch(error => console.error('Error:', error));
}

function decryptText() {
    const encryptedText = document.getElementById('inputText').value;
    fetch('/decrypt', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/x-www-form-urlencoded',
        },
        body: `encrypted_text=${encodeURIComponent(encryptedText)}`,
    })
    .then(response => response.json())
    .then(data => {
        document.getElementById('outputText').innerText = data.decrypted_text;
    })
    .catch(error => console.error('Error:', error));
}
```

Conclusion

This web application provides a simple interface for encrypting and decrypting text using strong cryptographic algorithms. The use of Flask for the backend and Fernet from the cryptography library ensures that the encryption is secure and the same plaintext will result in different ciphertexts each time it is encrypted.