Secure Messaging Chat Application

AIM

Build a secure messaging chat app using Flask in Python, where users can enter a message, choose an encryption method (like Fernet or Caesar cipher), and view the encrypted output. The app ensures message confidentiality through basic cryptographic techniques.

Project Overview

Secure Messaging Chat App is a web-based application that enables users to send encrypted messages using two classical encryption techniques:

- **Fernet Encryption** (modern symmetric encryption)
- **Caesar Cipher** (classical substitution cipher)

The platform is designed for simplicity, educational value, and basic secure communication, making it ideal for demonstrations, prototypes, or student learning tools.

Technology Stack

Python 3.x

- **Purpose**: Core programming language
- **Why chosen**: Wide community support, built-in libraries, perfect for rapid backend development with Flask.

Flask (Web Framework)

- **Purpose**: Lightweight web server and routing framework
- **Features**: Request handling, HTML templating, form processing.

Cryptography Library

- **Purpose**: Secure Fernet encryption
- **Features**: AES-based symmetric encryption with built-in key generation and HMAC validation.

HTML + CSS (Frontend UI)

- **Purpose**: User interface and layout
- **Features**: Form fields, dropdowns, tables, styled using responsive and accessible CSS.

Cipher Implementations

Fernet Cipher

- **Algorithm Type**: Symmetric (modern encryption)
- **Function**: encrypt_fernet(message: str) -> str
- **Key Features**:
  - Uses 32-byte secure key (auto-generated & saved to secret.key)
  - Ensures confidentiality and integrity with AES & HMAC
  - Suitable for real secure data

Caesar Cipher

- **Algorithm Type**: Classical substitution cipher
- **Function**: encrypt_caesar(message: str, shift: int) -> str
- **Key Features**:
    - Letter-by-letter substitution with shift
    - Supports upper/lowercase
    - Preserves punctuation & digits
    - Good for educational/demo purposes

User Interface Features

- **Input Fields**:
    - Message: Plaintext message input
    - Method: Dropdown to select Fernet or Caesar
    - Shift: Number input for Caesar cipher (only shown when Caesar is selected)
- **Encrypt Button**: Send: Submits the message and applies selected encryption
- **Result Display**
    - Encrypted messages shown in a list (in-memory)
    - Includes:
        - Original message
        - Encrypted result
        - Method used
        - Shift (if Caesar)
- **Error Handling**: Pop-up dialogs for invalid or missing inputs
    - **Missing Input:** Alert if any field is blank (User sees: “Missing message or method”)
    - **Invalid Caesar Key:** User sees: “Shift value must be an integer”, key must be digit.
    - **Empty Message:** Form input marked required
    - **Invalid Method :** Defaults to Fernet

Run: python main.py launches the GUI

Workflow:

- Enter message
- Select the Method (Fernet or Caesar)
- Click Encrypt & send
- View the encrypted results

Implementation Code:

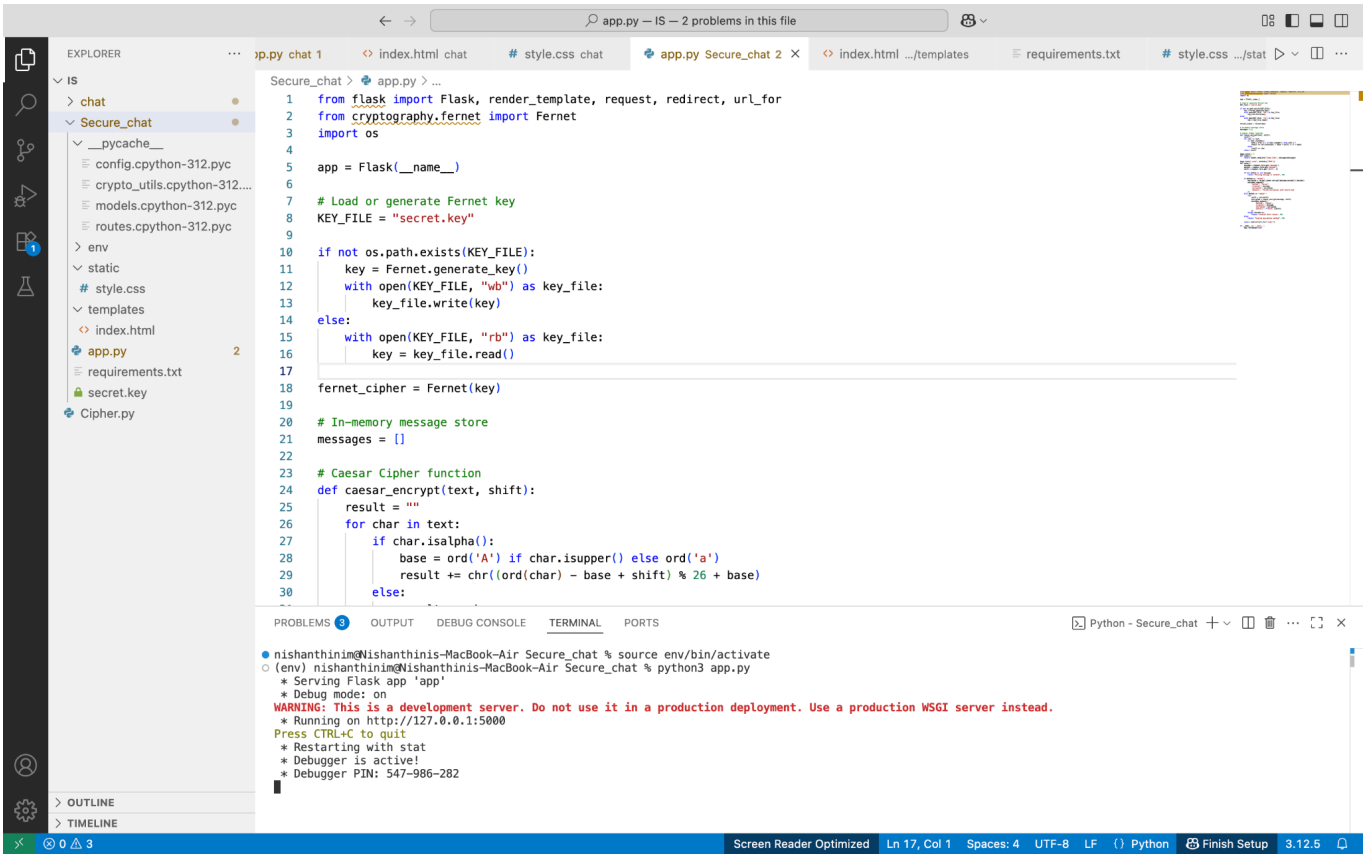
```
from flask import Flask, render_template, request, redirect, url_for

from cryptography.fernet import Fernet

import os

app = Flask(__name__)

# Load or generate Fernet key

KEY_FILE = "secret.key"

if not os.path.exists(KEY_FILE):
```

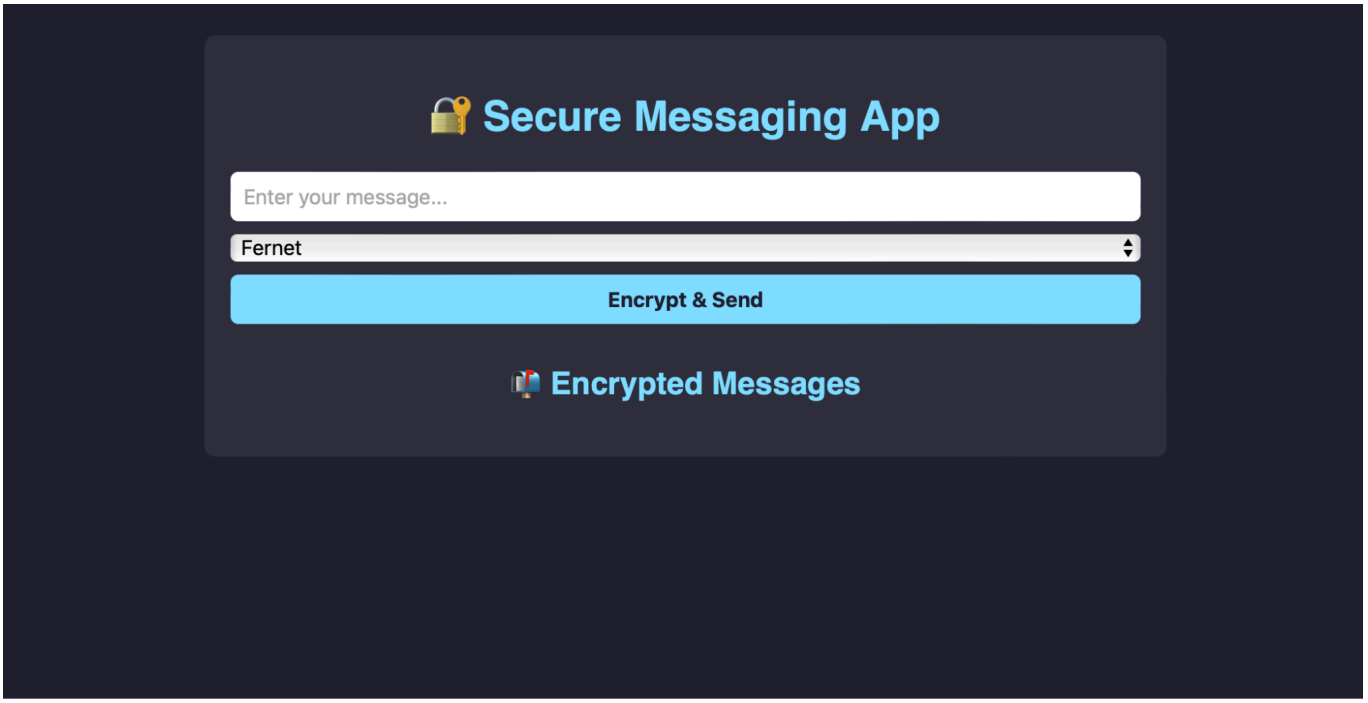
```
    key = Fernet.generate_key()
    with open(KEY_FILE, "wb") as key_file:
        key_file.write(key)
else:
    with open(KEY_FILE, "rb") as key_file:
        key = key_file.read()
fernet_cipher = Fernet(key)
messages = []  # In-memory message store
# Caesar Cipher function
def caesar_encrypt(text, shift):
    result = ""
    for char in text:
        if char.isalpha():
            base = ord('A') if char.isupper() else ord('a')
            result += chr((ord(char) - base + shift) % 26 + base)
        else:
            result += char
    return result
@app.route('/')
def index():
    return render_template("index.html", messages=messages)
@app.route('/send', methods=['POST'])
def send():
    message = request.form.get('message')
    method = request.form.get('method')
    shift = request.form.get('shift', 0)

    if not method or not message:
        return "Missing message or method", 400

    if method == 'fernet':
        encrypted = fernet_cipher.encrypt(message.encode()).decode()
        messages.append({
```

```
            'method': 'Fernet',
            'original': message,
            'encrypted': encrypted,
            'details': 'Fernet encryption with secret.key'
        })
    elif method == 'caesar':
        try:
            shift = int(shift)
            encrypted = caesar_encrypt(message, shift)
            messages.append({
                'method': 'Caesar',
                'original': message,
                'encrypted': encrypted,
                'details': f'Shift: {shift}'
            })
        except ValueError:
            return "Invalid shift value", 400
    else:
        return "Invalid encryption method", 400
    return redirect(url_for('index'))

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

Output:

Link : http://127.0.0.1:5000





🔐 Secure Messaging App

Enter your message...

Fernet

Encrypt & Send

📬 Encrypted Messages

Method: Fernet

Original: haiii

Encrypted: gAAAAABohwRTCMlBOvA83vnafGQ4tTbSZP3a9MonWyX5cA9-0rSx8u-Me8Lyz4Mw-nCmvKR-S_3fewIUt-UcDkSKsrKGTZ-TTg==

Details: Fernet encryption with secret.key

🔐 Secure Messaging App

Enter your message...

Caesar

Shift for Caesar (0-25)

Encrypt & Send

📬 Encrypted Messages

Method: Fernet

Original: haiii

Encrypted: gAAAAABohwRTCMlBOvA83vnafGQ4tTbSZP3a9MonWyX5cA9-0rSx8u-Me8Lyz4Mw-nCmvKR-S_3fewIUt-UcDkSKsrKGTZ-TTg==

Details: Fernet encryption with secret.key

Method: Caesar

Original: Hair

Encrypted: Kdlu

Details: Shift: 3