Text Encryption Using Cryptographic

To help you understand and possibly extend or modify the code, here's a complete walkthrough of what this Flask application does — from start to end — including its encryption, decryption logic, and web interface.

Purpose of the App:-

This Flask app provides a web interface to:Encrypt any text using AES (Advanced Encryption Standard) with CBC mode.Decrypt text that was encrypted using the same logic.

Libraries Used:-

from flask import Flask, render_template_string, request

Creates a web server and allows HTML rendering.request is used to get data from the user via forms.

Used for AES encryption/decryption with padding.

import os, base64

os generates secure random keys and IVs.

base64 encodes and decodes binary data for safe text transmission.

App Setup:-

```
app = Flask(_name_)

KEY = os.urandom(32)  # 32 bytes for AES-256

IV_LENGTH = 16       # 16 bytes IV for AES-CBC
```

Initializes the Flask app.

Generates a random encryption key and sets IV length.

HTML Form Template:-HTML_TEMPLATE = """..."""

Contains a simple form with:

A <textarea> for input text.

Two buttons: Encrypt and Decrypt.

Displays the result below the form.

Encryption Function:-

What It Does:Creates a random IV (Initialization Vector).

Pads the input text to make its length a multiple of 16 bytes.

Encrypts it using AES-256 in CBC mode.

Combines IV + ciphertext, encodes in base64, and returns the result.

Decryption Function:-

What It Does:Decodes the base64-encoded string.

Splits out the IV and ciphertext.Decrypts and removes padding.Returns the original plaintext.

Web Route:-

What It Does:Displays the form on GET.

On POST:Gets the user input and selected action (Encrypt/Decrypt).Calls the relevant function.Handles errors if something fails (e.g., decryption fails).

Displays the result in the web page.

App Runner:-

if _name_ == '_main_':

    app.run(debug=True)Runs the app on http://127.0.0.1:5000/.

Design Choices:

1. Technology Stack:

Backend: Python (Flask framework)

Encryption: cryptography library

Interface: HTML form served via Flask

2. Encryption Algorithm:

AES-256 with CBC mode and PKCS7 padding.

A random IV is generated for every encryption process to ensure non-deterministic outputs.

3. Key Management:

A 256-bit key (os.urandom(32)) is generated at runtime (for real deployments, store it securely).

Challenges Faced:

Correct padding and unpadding of text for AES block size.

Managing random IVs and maintaining consistent decryption.

Designing a user-friendly and secure frontend.

Project Features:

Encrypts text with strong cryptography (AES-256).

Decrypts encrypted input if key and IV match.

Simple web interface for usability.

Ensures different output for same input via random IV.

Overview: This project implements a Flask-based web application that allows users to encrypt and decrypt text using AES-256-CBC encryption. The encryption mechanism uses randomly generated Initialization Vectors (IVs), ensuring strong, secure outputs that vary even for the same input text.