```
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.primitives import padding
from cryptography.hazmat.backends import default_backend
import base64
import unittest
import sys


def log_to_html(message):
    print(message)  # Вывод в консоль
    with open("log.html", "a", encoding="utf-8-sig") as log_file:
        with open("log.html", "a", encoding="utf-8") as log_file:
            log_file.write(f"<p>{message}</p>\n")


def get_user_input():
    key = input("Введите ключ (32 байта, например, 12345678901234567890123456789012): ").encode()
    iv = input("Введите IV (16 байт, например, 1234567890123456): ").encode()
    data = input("Введите данные для шифрования: ").encode()
    mode = input("Выберите режим (CBC, CFB, OFB): ").upper()

    if len(key) != 32:
        raise ValueError("Ключ должен быть 32 байта!")
    if len(iv) != 16:
        raise ValueError("IV должен быть 16 байт!")
    if mode not in ["CBC", "CFB", "OFB"]:
        raise ValueError("Неверный режим шифрования!")

    return key, iv, data, mode


def get_cipher(mode, key, iv):
    if mode == "CBC":
        return Cipher(algorithms.AES(key), modes.CBC(iv), backend=default_backend())
    elif mode == "CFB":
        return Cipher(algorithms.AES(key), modes.CFB(iv), backend=default_backend())
    elif mode == "OFB":
        return Cipher(algorithms.AES(key), modes.OFB(iv), backend=default_backend())


def encrypt_aes(key, iv, data, mode):
    cipher = get_cipher(mode, key, iv)
    encryptor = cipher.encryptor()
    padder = padding.PKCS7(algorithms.AES.block_size).padder()
    padded_data = padder.update(data) + padder.finalize()
    encrypted_data = encryptor.update(padded_data) + encryptor.finalize()
    return base64.b64encode(encrypted_data).decode()


def decrypt_aes(key, iv, encrypted_data, mode):
    cipher = get_cipher(mode, key, iv)
    decryptor = cipher.decryptor()
    decrypted_padded_data = decryptor.update(base64.b64decode(encrypted_data)) + decryptor.finalize()
    unpadder = padding.PKCS7(algorithms.AES.block_size).unpadder()
    decrypted_data = unpadder.update(decrypted_padded_data) + unpadder.finalize()
    return decrypted_data.decode()


class TestAESMethods(unittest.TestCase):
    def setUp(self):
        self.valid_key = b"12345678901234567890123456789012"
        self.invalid_keys = [b"short_key", b"too_long_key_which_is_not_valid_for_aes_32_bytes"]
        self.iv = b"1234567890123456"
        self.data_cases = [b"Test data for AES", b"Another test case", b"Short", b"Long data " * 10]
        self.modes = ["CBC", "CFB", "OFB"]

    def test_encryption_decryption(self):
        for mode in self.modes:
            for data in self.data_cases:
                with self.subTest(mode=mode, data=data):
                    encrypted = encrypt_aes(self.valid_key, self.iv, data, mode)
                    decrypted = decrypt_aes(self.valid_key, self.iv, encrypted, mode)
                    self.assertEqual(decrypted, data.decode())
                    log_to_html(
                        f"<strong>Тест {mode}</strong>: Исходные данные: '{data.decode()}' | Зашифровано: '{encrypted}' | P

    def test_invalid_keys(self):
        for mode in self.modes:
            for key in self.invalid_keys:
                with self.subTest(mode=mode, key=key):
                    try:
                        encrypt_aes(key, self.iv, b"Test", mode)
                    except ValueError as e:
                        log_to_html(
                            f"<strong>Тест {mode}</strong>: Неверный ключ: '{key}' | Ожидаемая ошибка: {e} | <span style='c


if __name__ == "__main__":
    with open("log.html", "w", encoding="utf-8-sig") as log_file:
        log_file.write("""
        <html>
        <head>
        <div class='header'>by verlliann prod.</div>
            <title>Лог тестирования</title>
            <style>
                body { font-family: Arial, sans-serif; margin: 20px; padding: 20px; background-color: #f4f4f4; }
```

```
            p { background: white; padding: 10px; border-radius: 5px; box-shadow: 0px 0px 5px rgba(0,0,0,0.1); }
            .footer { text-align: center; margin-top: 20px; font-weight: bold; }
        </style>
    </head>
    <body>
    <h2>Отчёт о тестировании</h2>
    """)

if len(sys.argv) > 1 and sys.argv[1] == "test":
    print("Запуск системных тестов...")
    unittest.main(argv=[sys.argv[0]])
else:
    try:
        key, iv, data, mode = get_user_input()
        encrypted_text = encrypt_aes(key, iv, data, mode)
        print(f"Зашифрованные данные ({mode}): {encrypted_text}")
        decrypted_text = decrypt_aes(key, iv, encrypted_text, mode)
        print(f"Расшифрованные данные: {decrypted_text}")
        log_to_html(
            f"<p><strong>Ручной ввод</strong>: Данные: '{data.decode()}' | Зашифровано: '{encrypted_text}' | Расшифрова
    except ValueError as e:
        log_to_html(f"<p><strong>Ошибка</strong>: {e}</p>")

with open("log.html", "a", encoding="utf-8") as log_file:
    log_file.write("<div class='footer'>by verlliann prod.</div></body></html>")
```