**Part 1**: Code Review and SSDLC Analysis

SSDLC Stages in the Code

Requirements Analysis: The script securely integrates OpenSSL for cryptographic operations.

Design: The functions are modular and perform specific security tasks such as key generation, encryption, and hashing.

Implementation: The script follows Python best practices but lacks proper input validation and secure password handling.

Verification: Logging is implemented to track execution, but security vulnerabilities exist (e.g., hardcoded passwords).

Maintenance: The script is extendable, but lacks clear documentation and user input validation.

Security Improvements

Remove hardcoded passwords: Use getpass.getpass() or a secure vault.

Validate user inputs: Implement regex or os.path checks to prevent command injection. Enhance logging security: Ensure logs do not store sensitive information.

**Part 2**: [Modified Python Script](https://docs.google.com/document/d/1hJ1WOnDBbD6sbPu3dybLh-2qnk7eHmCycqSNE-l-KKY/edit?tab=t.0)

**Part 3**: Logging and Testing

Logging Updates

Logs key generation success/failure.

Logs encryption/decryption processes.

Hash output logging.

Test Plan

| Test Case | Expected Outcome |
| --- | --- |
| Valid file encryption | Encrypted file is created |
| Valid file decryption | Decrypted file matches original |
| Invalid file path | Error logged and no operation performed |
| Secure logging | No sensitive data is stored in logs |

**Part 4**: Documentation

User Manual

Run Script

python secure\_openssl.py encrypt filename.txt --password mysecret

python secure\_openssl.py decrypt filename.txt.enc --password mysecret

python secure\_openssl.py hash filename.txt

Example Usage

python secure\_openssl.py encrypt secret.txt

Enter password: \*\*\*\*

Security Considerations

Always use secure password input.

Validate file paths to prevent injection.

Store logs securely and avoid sensitive data exposure.

**Bonus Challenge**: Secure Password Vault Integration

import keyring

keyring.set\_password("secure\_openssl", "user", "mysecurepassword")

password = keyring.get\_password("secure\_openssl", "user")