Phase 2: Privacy and Security Implementation – Documentation Report

Objective

The purpose of this phase is to implement essential privacy policies and protection procedures to ensure the confidentiality, integrity, and regulatory compliance of the dataset. This implementation focuses on data hashing, encryption, access control, and demonstrating alignment with data protection regulations (GDPR, CCPA, HIPAA).

1. Hashing Sensitive Information

Library Used: hashlib

Purpose: To irreversibly hash sensitive user data such as passwords and phone numbers using SHA-256.

Code Functionality:

```
def hash_data(data: str) -> str:
   return hashlib.sha256(data.encode()).hexdigest()
```

Justification: Hashing protects sensitive data from being stored or transferred in plaintext and ensures integrity.

2. Substitution Encryption (Caesar Cipher)

Custom Functions: caesar_encrypt, caesar_decrypt

Purpose: To encrypt and decrypt textual data (e.g., notes, records) using Caesar Cipher.

Example Output:

```
Original: Confidential Notes
Encrypted: Frqilghqwldo 1rwhv
Decrypted: Confidential Notes
```

Justification: While Caesar Cipher is basic, it demonstrates the concept of substitution-based encryption clearly.

3. Role-Based Access Control (RBAC)

Class: User

Purpose: To ensure that only authorized roles (admin, analyst, viewer) have access to read/write/delete operations.

Access Levels:

```
permissions = {
   "admin": ["all"],
   "analyst": ["read", "write"],
   "viewer": ["read"]
}
```

Justification: RBAC restricts access based on roles, which is essential in any secure system.

4. Regulatory Compliance Demonstrations

a. GDPR (General Data Protection Regulation)
Class: GDPRHandler
- Methods: request_consent(), erase_user_data()

b. CCPA (California Consumer Privacy Act)
Class: CCPAHandler
- Method: process_access_request()

c. HIPAA (Health Insurance Portability and Accountability Act)
Class: HIPAAHandler
- Method: scan_system()

Purpose: These handlers simulate real-world regulatory actions to demonstrate awareness of international data laws.

Example Execution Snapshot

```
Hashed Password: ...
Hashed Phone Number: ...
Encrypted: Frqilghqwldo 1rwhv
Decrypted: Confidential Notes
Can 'analyst' write?: True
Consent requested from user user123 under GDPR.
Data for user user123 erased as per GDPR.
CCPA access request processed for user@example.com
Running HIPAA compliance scan using pyTenable (mock).
```

Code File Distribution

File Name: privacy_protection.py

Structure:
- Hashing Function
- Caesar Cipher Encryption/Decryption
- RBAC Class
- Compliance Handler Classes
- Example Usage Block

Conclusion

This implementation satisfies all requirements of Phase 2 by demonstrating key privacy, encryption, and compliance mechanisms using Python. The project shows readiness for real-world privacy applications, especially when dealing with sensitive or regulated data.