**Project Title: Secure File Encryption and Decryption System**

**Objective:**

Develop a Python application that allows users to encrypt and decrypt files using symmetric and asymmetric cryptography techniques. This system will ensure file confidentiality, integrity, and user authentication.

**Features:**

1. **User Authentication**:
   * Add a login system to ensure that only authorized users can encrypt/decrypt files.
   * Use hashed passwords for authentication (e.g., bcrypt or argon2).
2. **Symmetric Encryption**:
   * Encrypt and decrypt files using algorithms like AES (Advanced Encryption Standard) from the cryptography library.
3. **Asymmetric Encryption**:
   * Encrypt and share files using RSA for secure key exchange.
   * Allow users to share encrypted files securely with others by encrypting the symmetric key with the recipient's public key.
4. **File Integrity**:
   * Generate and verify file integrity using cryptographic hash functions (e.g., SHA-256).
5. **Graphical User Interface (Optional)**:
   * Create a user-friendly interface using tkinter or PyQt.

**Technologies and Libraries:**

* **Python Libraries**:
  + cryptography for cryptographic operations.
  + os and sys for file handling.
  + hashlib for hashing.
  + tkinter or PyQt for GUI (optional).
  + sqlite3 for user authentication database (optional).

**Steps to Implement:**

1. **Setup Environment**:
   * Install required libraries: pip install cryptography bcrypt.
2. **User Authentication**:
   * Create a user registration and login system.
   * Hash passwords before storing them in the database.
3. **Symmetric Encryption**:
   * Generate a random symmetric key using AES.
   * Encrypt and decrypt files using this key.
4. **Asymmetric Encryption**:
   * Generate RSA key pairs for users.
   * Encrypt the symmetric key with the recipient's public key.
   * Decrypt the symmetric key with the recipient’s private key.
5. **File Integrity**:
   * Generate a SHA-256 hash of the file before encryption.
   * Verify the hash after decryption to ensure integrity.
6. **GUI (Optional)**:
   * Create a graphical interface to upload, encrypt, decrypt, and verify files.