A

PROJECT REPORT

on

Encryption of Text and Text File with Secure Login

BACHELOR OF TECHNOLOGY

SUBMITTED BY

Tisha Jindal

21012004054

21/CSE54

GUIDED BY

Ms. Anjana

anangpuria

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING / DEPARTMENT OF INFORMATION TECHNOLOGY

B.S Anangpuria Institute of Technology & Management

Alampur, Faridabad

Contents

Introduction

In an era where digital security is paramount, protecting sensitive information from unauthorized access and potential breaches has become a critical concern. Our project, titled "Encryption of Text and Text File with Secure Login," addresses this challenge by implementing robust security measures to safeguard text and text files through advanced encryption techniques and secure authentication protocols.

Objective:

The primary objective of this project is to develop a comprehensive system that not only encrypts text and text files but also ensures that access to these encrypted resources is secured through a reliable login mechanism. By combining encryption and authentication, we aim to provide a solution that enhances the confidentiality and integrity of data while maintaining ease of use and accessibility for authorized users.

Key Features:

1. **Encryption of Text and Text Files:** Our system employs state-of-the-art encryption algorithms to transform plain text and files into secure, unreadable formats. This ensures that sensitive information remains protected from unauthorized viewing or tampering.
2. **Secure Login Mechanism:** To access encrypted content, users must first authenticate themselves through a secure login process. This involves implementing secure password handling and user verification methods to prevent unauthorized access.

3. **User-Friendly Interface:** Despite the complexity of encryption and secure login processes, the system features an intuitive interface that simplifies interactions for end-users. This helps in achieving a balance between robust security and user convenience.
4. **Data Integrity and Confidentiality:** By focusing on encryption and secure login, our project ensures that both the confidentiality and integrity of data are upheld. Encrypted text and files are only accessible to authorized users, and any attempts to tamper with encrypted data are effectively thwarted.

Significance:

As digital threats evolve and data breaches become increasingly sophisticated, our project addresses the need for reliable data protection mechanisms. Encryption and secure login are fundamental components of a comprehensive security strategy, and our solution provides a practical approach to achieving these goals.

Objective

The objective of the project “Encryption of Text and Text File with Secure Login” is to develop a robust and user-friendly application that ensures the confidentiality and integrity of sensitive information. This project aims to:

1. **Encrypt Text and Text Files:** Implement advanced encryption algorithms to securely encrypt and decrypt text and text files, protecting them from unauthorized access.
2. **Secure Login System:** Develop a secure login mechanism that authenticates users and prevents unauthorized access to the application and its encrypted data.
3. **User-Friendly Interface:** Design an intuitive and easy-to-use interface that allows users to seamlessly encrypt, decrypt, and manage their text and text files.
4. **Data Integrity:** Ensure that the encrypted data remains intact and unaltered during storage and transmission.
5. **Scalability and Performance:** Optimize the application for efficient performance and scalability to handle varying amounts of data and user activity.

Hardware and Software Declaration

Hardware Requirements: Specify any hardware needed for implementation, such as servers, PCs, etc.

Software Requirements: List the software tools and libraries you plan to use, such as:

1. Programming language (e.g., Python, Java)
2. Encryption libraries (e.g., PyCryptodome, OpenSSL)
3. User authentication libraries (e.g., Flask-Login, bcrypt)
4. Database (e.g., SQLite, MySQL) for storing user credentials

Technology Used

1. Programming Language

- **Python**: A versatile language with extensive libraries for encryption, file handling, and web development.

2. Encryption Libraries

- **Python**
  - **Cryptography**: A library that provides cryptographic recipes and primitives to Python developers.
  - **PyCrypto**: A collection of secure hash functions and various encryption algorithms.
  - **Fernet (from the cryptography library)**: Symmetric encryption (AES) for securing data.

3. Web Framework

- **Flask** (Python): A lightweight web framework to create the login interface and handle HTTP requests.

4. Authentication & Authorization

- **Flask-Login** (Python): For handling user sessions and authentication in Flask applications.

Methodology

Architecture Design: Create a block diagram illustrating the overall architecture, showing the flow of data from user input through encryption and decryption to storage.

Module Design: Break down the project into functional modules:

- **User Authentication**: Designing the login system using secure hashing and salt methods.
- **Encryption/Decryption Module**: Implementing encryption algorithms such as AES, RSA, or others as per your choice.
- **File Handling Module**: Handling text file input/output and applying encryption/decryption.
- **User Interface (Optional)**: Designing a user-friendly interface for inputting text, logging in, and handling files.

Flowchart

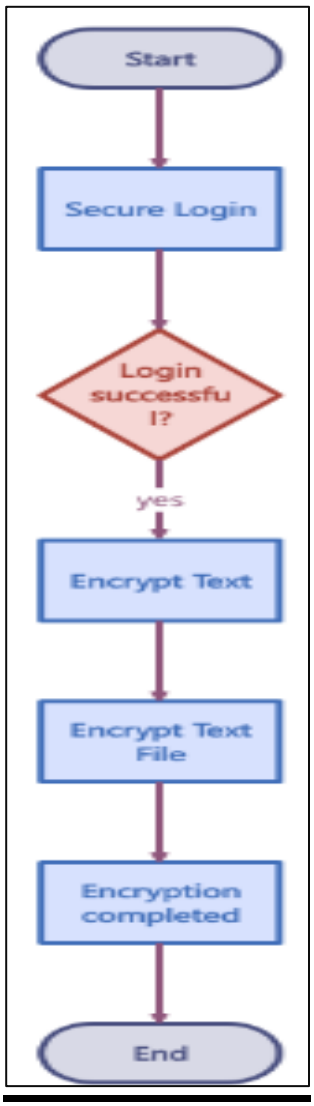

Figure 1: Flowchart

Code Implementation

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
import os
import base64
import json
from cryptography.fernet import Fernet
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC

# Global dictionary to hold user data
users = {}

# Function to derive a key from password and salt
def derive_key(password, salt):
    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        length=32,
        salt=salt,
        iterations=100000,
    )
    return base64.urlsafe_b64encode(kdf.derive(password.encode()))

# Function to register a new user
def register_user(username, password):
    salt = os.urandom(16)
    key = derive_key(password, salt)
    users[username] = {
        'salt': base64.b64encode(salt).decode('utf-8'),
        'key': base64.b64encode(key).decode('utf-8')
    }
    return key

# Function to log in an existing user
```

```
def login_user(username, password):
    if username not in users:
        messagebox.showerror("Error", "User not found")
        return None

    user_data = users.get(username)
    if 'salt' not in user_data or 'key' not in user_data:
        messagebox.showerror("Error", "User data corrupted or incomplete")
        return None

    salt = base64.b64decode(user_data['salt'])
    stored_key = base64.b64decode(user_data['key'])
    derived_key = derive_key(password, salt)

    if derived_key == stored_key:
        return derived_key
    else:
        messagebox.showerror("Error", "Incorrect password")
        return None

class EncryptionApp(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Secure Encryption Suite")
        self.geometry("500x600")
        self.configure(bg="#34495e")  # Dark blue-gray background
        self.current_user = None
        self.key = None

        self.load_users()
        self.create_widgets()

    def load_users(self):
        if os.path.exists("users.json"):
```

```
        with open("users.json", "r") as f:
            global users
            users = json.load(f)

    def save_users(self):
        with open("users.json", "w") as f:
            json.dump(users, f)

    def create_widgets(self):
        self.login_frame = ttk.Frame(self, padding="20", style="LoginFrame.TFrame")
        self.login_frame.pack(expand=True, fill="both")

        ttk.Label(self.login_frame, text="Username:", background="#34495e",
foreground="white").pack(pady=(20, 5))
        self.username_entry = ttk.Entry(self.login_frame, foreground="black",
background="#34495e")
        self.username_entry.pack(pady=(0, 10))

        ttk.Label(self.login_frame, text="Password:", background="#34495e",
foreground="white").pack(pady=(0, 5))
        self.password_entry = ttk.Entry(self.login_frame, show='*', foreground="black",
background="#34495e")
        self.password_entry.pack(pady=(0, 10))

        ttk.Button(self.login_frame, text="Login", command=self.login,
style="LoginButton.TButton").pack(pady=(10, 5))
        ttk.Button(self.login_frame, text="Register", command=self.register,
style="RegisterButton.TButton").pack(pady=(5, 20))

        # Main menu frame
        self.style = ttk.Style(self)
        self.style.configure("MenuFrame.TFrame", background="#34495e")
        self.main_menu_frame = ttk.Frame(self, padding="20", style="MenuFrame.TFrame")
```

```
        # Main menu buttons
        ttk.Button(self.main_menu_frame, text="Encrypt Text", command=self.encrypt_text_window, style="MenuButton.TButton").pack(pady=(10, 5))
        ttk.Button(self.main_menu_frame, text="Decrypt Text", command=self.decrypt_text_window, style="MenuButton.TButton").pack(pady=(10, 5))
        ttk.Button(self.main_menu_frame, text="Encrypt File", command=self.encrypt_file, style="MenuButton.TButton").pack(pady=(10, 5))
        ttk.Button(self.main_menu_frame, text="Decrypt File", command=self.decrypt_file, style="MenuButton.TButton").pack(pady=(10, 5))
        ttk.Button(self.main_menu_frame, text="Logout", command=self.logout, style="LogoutButton.TButton").pack(pady=(10, 5))

        # Custom styles
        self.style.configure("LoginFrame.TFrame", background="#34495e")
        self.style.configure("MenuButton.TButton", background="#2ecc71", foreground="black", padding=10)
        self.style.configure("LoginButton.TButton", background="black", foreground="black", padding=10)
        self.style.configure("RegisterButton.TButton", background="black", foreground="black", padding=10)
        self.style.configure("LogoutButton.TButton", background="#e74c3c", foreground="black", padding=10)

        for button_style in ["LoginButton.TButton", "RegisterButton.TButton", "MenuButton.TButton", "LogoutButton.TButton"]:
            self.style.map(button_style,
                           background=[('active', '#16a085'), ('pressed', '#1abc9c')],
                           foreground=[('active', 'black'), ('pressed', 'black')])

    def login(self):
        username = self.username_entry.get()
        password = self.password_entry.get()
        key = login_user(username, password)
        if key:
            self.current_user = username
            self.key = key
            messagebox.showinfo("Success", "Login successful!")
```

```
            self.login_frame.pack_forget()
            self.main_menu_frame.pack(expand=True, fill="both")
        else:
            messagebox.showerror("Error", "Login failed")

    def register(self):
        username = self.username_entry.get()
        password = self.password_entry.get()
        if username in users:
            messagebox.showerror("Error", "Username already exists")
        else:
            self.key = register_user(username, password)
            self.current_user = username
            messagebox.showinfo("Success", "Registration successful!")
            self.save_users()
            self.login_frame.pack_forget()
            self.main_menu_frame.pack(expand=True, fill="both")

    def encrypt(self, data):
        f = Fernet(self.key)
        return f.encrypt(data.encode()).decode()

    def decrypt(self, data):
        f = Fernet(self.key)
        return f.decrypt(data.encode()).decode()

    def encrypt_text_window(self):
        encrypt_window = tk.Toplevel(self)
        encrypt_window.title("Encrypt Text")
        encrypt_window.geometry("400x300")
        encrypt_window.configure(bg="#34495e")
```

```
        ttk.Label(encrypt_window, text="Enter text to encrypt:", background="#34495e",
foreground="white").pack(pady=(20, 5))
        text_entry = tk.Text(encrypt_window, height=5, width=40)
        text_entry.pack(pady=(0, 10))


        def encrypt_text():
            text = text_entry.get("1.0", tk.END).strip()
            if text:
                encrypted_text = self.encrypt(text)
                result_window = tk.Toplevel(encrypt_window)
                result_window.title("Encrypted Text")
                result_window.geometry("400x200")
                result_window.configure(bg="#34495e")
                ttk.Label(result_window, text="Encrypted Text:", background="#34495e",
foreground="white").pack(pady=(20, 5))
                result_text = tk.Text(result_window, height=5, width=40)
                result_text.insert(tk.END, encrypted_text)
                result_text.config(state=tk.DISABLED)
                result_text.pack(pady=(0, 10))
            else:
                messagebox.showerror("Error", "Please enter text to encrypt")


        ttk.Button(encrypt_window, text="Encrypt", command=encrypt_text,
style="MenuButton.TButton").pack(pady=10)


    def decrypt_text_window(self):
        decrypt_window = tk.Toplevel(self)
        decrypt_window.title("Decrypt Text")
        decrypt_window.geometry("400x300")
        decrypt_window.configure(bg="#34495e")


        ttk.Label(decrypt_window, text="Enter text to decrypt:", background="#34495e",
foreground="white").pack(pady=(20, 5))
        text_entry = tk.Text(decrypt_window, height=5, width=40)
        text_entry.pack(pady=(0, 10))
```

```
        def decrypt_text():
            text = text_entry.get("1.0", tk.END).strip()
            if text:
                try:
                    decrypted_text = self.decrypt(text)
                    result_window = tk.Toplevel(decrypt_window)
                    result_window.title("Decrypted Text")
                    result_window.geometry("400x200")
                    result_window.configure(bg="#34495e")
                    ttk.Label(result_window, text="Decrypted Text:", background="#34495e",
foreground="white").pack(pady=(20, 5))
                    result_text = tk.Text(result_window, height=5, width=40)
                    result_text.insert(tk.END, decrypted_text)
                    result_text.config(state=tk.DISABLED)
                    result_text.pack(pady=(0, 10))
                except Exception as e:
                    messagebox.showerror("Error", "Decryption failed: " + str(e))
            else:
                messagebox.showerror("Error", "Please enter text to decrypt")
        ttk.Button(decrypt_window, text="Decrypt", command=decrypt_text,
style="MenuButton.TButton").pack(pady=10)
    def encrypt_file(self):
        file_path = filedialog.askopenfilename()
        if file_path:
            try:
                with open(file_path, 'rb') as file:
                    file_data = file.read()
                f = Fernet(self.key)
                encrypted_data = f.encrypt(file_data)
                save_path = filedialog.asksaveasfilename(defaultextension=".enc")
                if save_path:
                    with open(save_path, 'wb') as file:
                        file.write(encrypted_data)
```

```
                messagebox.showinfo("Success", "File encrypted successfully")
            except Exception as e:
                messagebox.showerror("Error", f"An error occurred: {str(e)}")


    def decrypt_file(self):
        file_path = filedialog.askopenfilename(filetypes=[("Encrypted files", "*.enc")])
        if file_path:
            try:
                with open(file_path, 'rb') as file:
                    encrypted_data = file.read()

                f = Fernet(self.key)
                decrypted_data = f.decrypt(encrypted_data)

                save_path = filedialog.asksaveasfilename(defaultextension=".txt",
                                                         filetypes=[("Text files", "*.txt"), ("All files", "*.*")])
                if save_path:
                    with open(save_path, 'wb') as file:
                        file.write(decrypted_data)
                    messagebox.showinfo("Success", "File decrypted successfully")
            except Exception as e:
                messagebox.showerror("Error", f"An error occurred: {str(e)}")


    def logout(self):
        self.current_user = None
        self.key = None
        self.main_menu_frame.pack_forget()
        self.login_frame.pack(expand=True, fill="both")

if __name__ == "__main__":
    app = EncryptionApp()
    app.mainloop()
```

Output

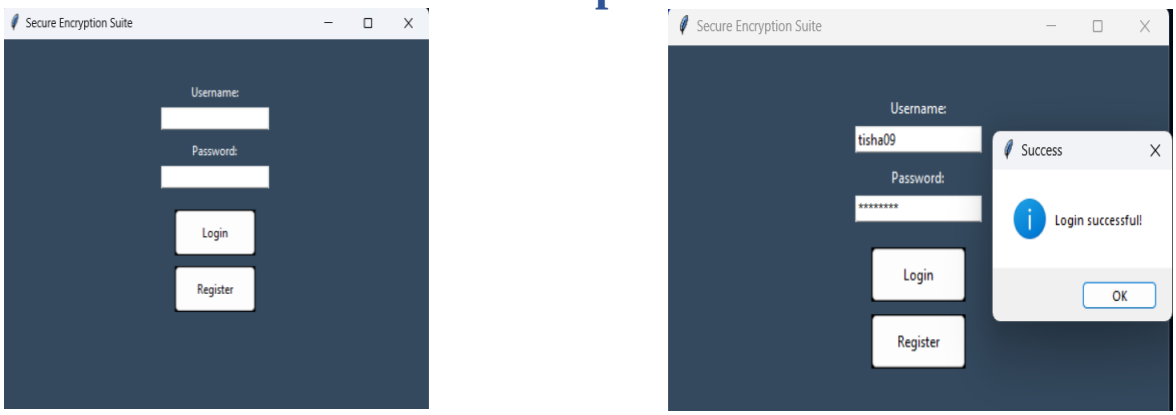

Figure 2: Login/Register Page

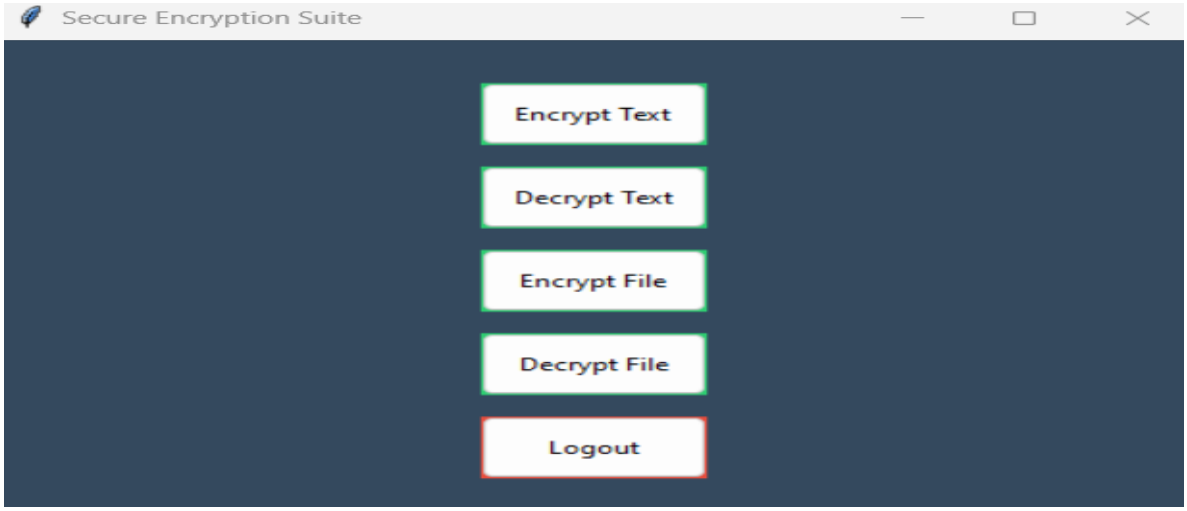

Figure 3: Home page

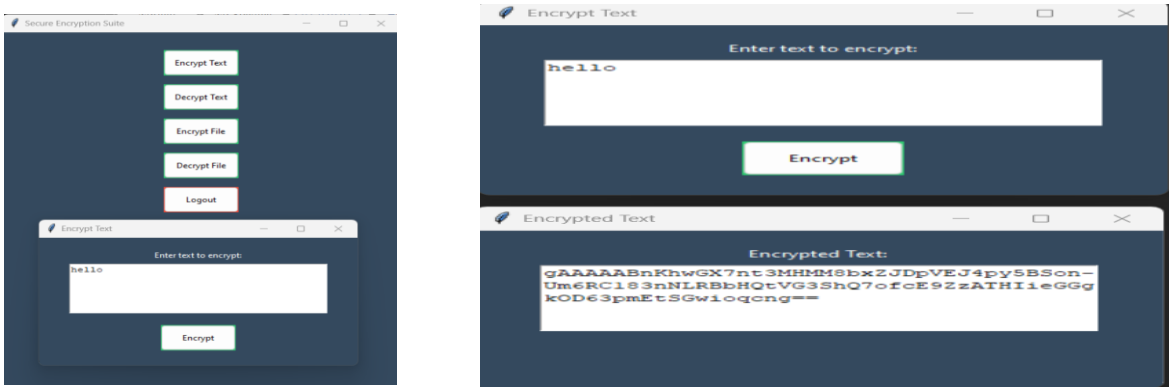

Figure 4: Encrypt text module

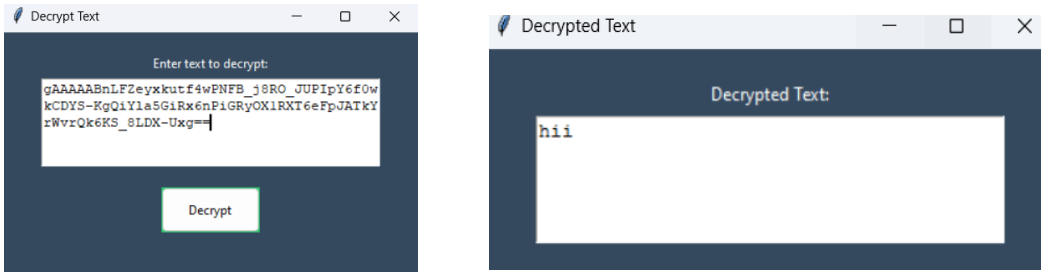

Figure 5: Decrypted text module

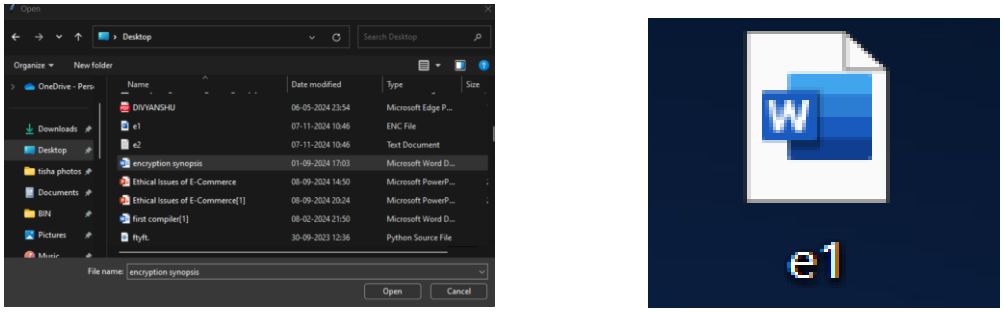

Figure 6: Encrypt file module

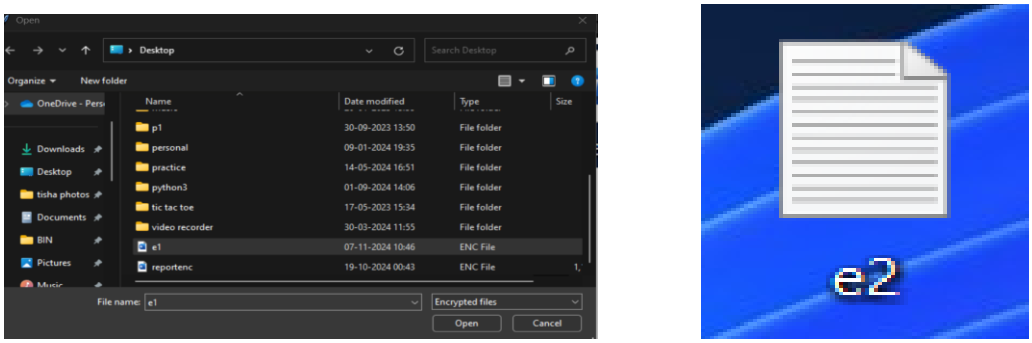

Figure 7: Decrypt file module

Conclusion

The project "Encryption of Text and Text File with Secure Login" successfully achieved its objectives of enhancing data security through robust encryption techniques and secure user authentication. The main outcomes of the project are:

1. Enhanced Security: Implemented advanced encryption algorithms to ensure that text and text files are securely encrypted, protecting sensitive information from unauthorized access.

2. Secure Login Mechanism: Developed a secure login system that uses strong password policies and multi-factor authentication to prevent unauthorized access to the encrypted data.

3. User-Friendly Interface: Created an intuitive and user-friendly interface that allows users to easily encrypt and decrypt text and text files, ensuring a seamless user experience.

4. Performance Optimization: Optimized the encryption and decryption processes to ensure minimal impact on system performance, providing a balance between security and efficiency.

5. Scalability and Flexibility: Designed the system to be scalable and flexible, allowing for future enhancements and integration with other security measures or applications.

Overall, the project demonstrates the importance of combining encryption and secure login mechanisms to protect sensitive data effectively. The successful implementation of these features highlights the project's potential for real-world applications in various domains requiring high levels of data security.

Future Scope

1. **Integration with Cloud Services**: Extend the encryption and secure login functionalities to cloud storage services, allowing users to securely store and access their encrypted files from anywhere.
2. **Mobile Application Development**: Develop mobile applications for both Android and iOS platforms to provide users with the ability to encrypt and decrypt text and files on the go.
3. **Biometric Authentication**: Incorporate biometric authentication methods such as fingerprint scanning or facial recognition to enhance the security of the login process.
4. **Advanced Encryption Algorithms**: Research and implement more advanced encryption algorithms to stay ahead of potential security threats and ensure the highest level of data protection.
5. **User Activity Monitoring**: Implement features to monitor and log user activities, providing an additional layer of security by detecting and responding to suspicious behaviour.
6. **Integration with Other Security Tools**: Integrate the system with other security tools such as antivirus software, firewalls, and intrusion detection systems to provide a comprehensive security solution.
7. **User Education and Awareness**: Develop educational resources and training programs to help users understand the importance of data security and how to use the encryption and secure login features effectively.

8. **Compliance with Regulations**: Ensure that the system complies with relevant data protection regulations and standards, such as GDPR, HIPAA, or CCPA, to enhance its applicability in various industries.
9. **Scalability for Enterprise Use**: Enhance the system's scalability to support large organizations with multiple users and vast amounts of data, making it suitable for enterprise-level deployment.
10. **Regular Security Audits**: Establish a process for regular security audits and updates to identify and address vulnerabilities, ensuring the system remains secure over time.

Reference

1. Various opensource materials from Internet.
2. Training notes.
3. Discussion among the group and with guide.
4. Some requirements are gathered through various books from library.
5. Google
6. You-tube