National College of Ireland

SafeCryption

Cyber Security

2022/2023

Robertas Kuchtovas

X20124287

X20124287@student.ncirl.ie

SafeCryption

Technical Report

Contents

Executive Summary

This project provides an overview of a file encryption application which is developed using Python and Tkinter.

The application allows users to generate encryption keys which are used to select files to encrypt or decrypt, in addition these keys can be exported for safe keeping and later imported when needed to decrypt a file. There is a GUI that is developed using Tkinter library which is built into python, alongside the ttk Notebook widget which allows for multiple tabs to be created.

This application uses AES encryption which was implemented with Fernet module from the cryptography library. The Encryptor was tested using various file types and sizes and results were found to be satisfactory.

This encryptor is a useful tool for anyone who wants to protect any sort of sensitive data. The application has a friendly UI which helps navigate around the whole application easily. It is a valuable addition to any security-conscious user.

1.0 Introduction

1.1. Background

As a cybersecurity student I wanted to create an application that relates to this topic and having studied secure application programming and penetration testing I feel like this application is very current. Because of those topics I developed an interest in cryptography and wanting to learn more about how encryption and decryption algorithms work I decided that it would be a good project to do so. I've also wanted to learn improve my skills in Python programming as I've only touched base with it in third year, but not fully diving into depts of it. Always hearing that Python is one of the biggest coding languages out there I wanted to improve on this aspect as I've mostly used java throughout the whole college cycle when there were any applications that I could develop. This project allowed me to combine my interests and improve my technical skills, while creating something that could be beneficial to others.

1.2. Aims

The encryption application aims to allow users to encrypt and decrypt files using an encryption key. Whilst being easy to use and provide a secure method for protecting sensitive files from unauthorized access. With database breaches happening more often and hackers getting into sensitive information this application aims to eliminate that.

1.3. Technology

This project utilizes Python as the main programming language and implements Tkinter from the Python library for building the GUI around the application. The cryptography is also implemented from the Python library which uses the Cipher encryption technique for encryption and decryption of files. It uses 128-bit AES keys and once the file has been encrypted with a specific key it cannot be decrypted without it. PyInstaller library creates a standalone executable for the program. GUI is built with object-oriented programming in mind with Python and each tab being its own separate class that inherits from the ttk.Frame class. Adding in Notebook from the Python library help's the application to display different tabs which will be used for home page, encryption, decryption etc. Buttons and Labels are also used to help guide the user and perform necessary actions. Application also includes error handling to catch any exception that may occur during encryption and decryption so that the application doesn't shut down unexpectedly and doesn't work.

1.4. Structure

Section 1 – This section outlines what the application will be and some background on why I chose to do develop this application, it will also touch on the aims, and technology used.

Section 2 – Will have the system requirements to run this application, the functional requirements, some use cases and use case diagrams to outline how the application works. User requirements Environmental requirements which are what is needed to run this application. Usability requirements which state how easy or complex the application will be to use. Design & architecture of the application, how all the features were

implemented along with a few screenshots. Graphical user interface which has pointers on how it works and how it all comes together along with a few screenshots. How the testing was done. And finally, the evaluation.
Section 3 – A conclusion to bring all the documentation together.
Section 4- Further development or research, any future plans for the application and the research that was done in order to complete this project.

2.0 System

2.1. Requirements

The application was developed, ran, and tested on a computer with the current specifications.

OS: Windows 10

CPU: AMD 8 core CPU

RAM: 16GB

Python Version: Python 3.10.11

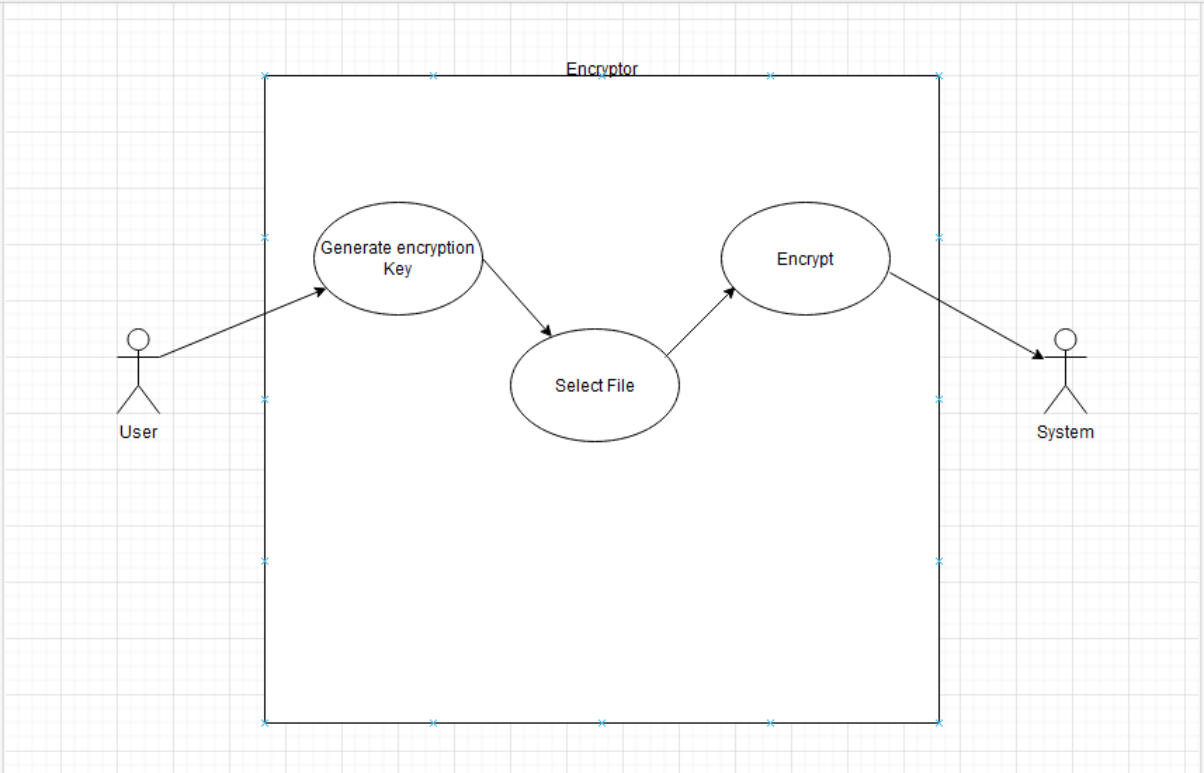
Libraries: tkinter 8.6, Pillow (PIL) 9.5.0

1. Application is easy to use and navigate, having clear labels and instructions.
2. Encrypt and Decrypt functions can handle files of various sizes and formats.
3. Experienced users should be able to use all system functions after a brief period of training, numbers of errors should not exceed two per day.
4. Application is secure with files which are encrypted are protected by a cryptographic algorithm and keys.

2.1.1. Functional Requirements

1. Encrypt file using an encryption algorithm.
2. Users can generate and manage their own encryption keys being able to export and import them.
3. The application provides support for different file types.
4. Offers the ability to decrypt encrypted data using the specific encryption/decryption key.
5. Provide a user-friendly interface for encryption and decryption operations.

2.1.1.1. Use Case Diagram



2.1.1.2. Requirement 1: Encrypt File

This requirement discusses one of the main functions of the application which is the encryption.

Description & Priority

It is one the main functions of the system and it's responsible for encrypting user's file to ensure confidentiality and security. Without this use case the application would not provide its primary service which would not be useful for its intended purpose. Meaning this use case is essential to the overall system. Which means it's a high priority.

2.1.1.3. Use Case

Scope

The scope of this use case is to describe the functionality of the system related to the encryption of files.

Description

This use case describes the process a user will take to encrypt a file using the application.

Flow Description

Precondition

User has launched the application.

User has a file to encrypt.

Activation

This use case starts when the user clicks into the encryption tab.

Main flow

1. User selects the Encrypt option from the application menu.
2. Application prompts the user to select the file to be encrypted.
3. User selects the file to be encrypted.
4. Application prompts the user to generate encryption key.
5. User generates a random encryption key.
6. Application encrypts the file using algorithm and key.
7. Application saves the encrypted file in specified user's location.
8. Application displays a message indicating that the file has been encrypted successfully.

Alternate flow

A1 : <No encryption key>

1. The system tries to encrypt user's file.
2. System prompts an error as no key has been generated.
3. Continue main flow 5.

Exceptional flow

E1 : <File already encrypted>

4. User selects a file to encrypt.
5. User generates a random encryption key.
6. Application tries to encrypt the file using algorithm and key.
7. Application prompts an error due to the file already being encrypted.
8. Continue main flow 3.

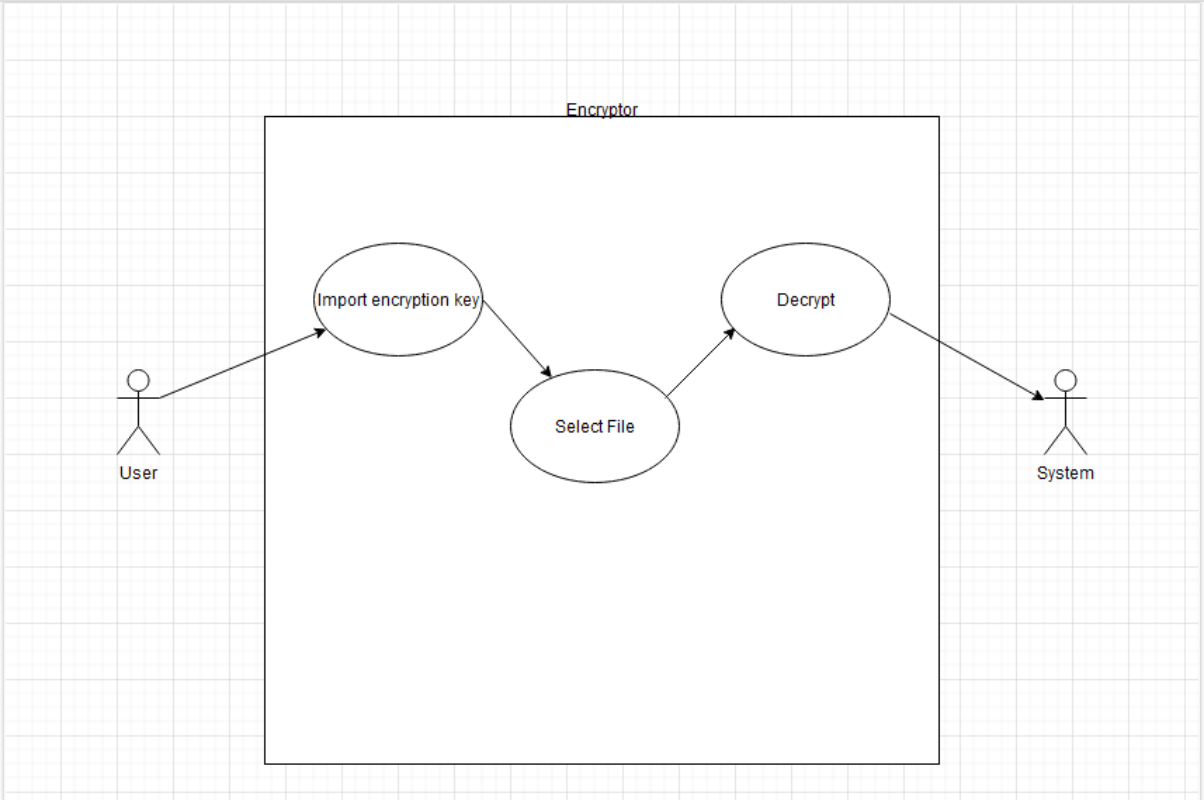
Termination

The system presents the message indicating the file has been encrypted.

Post condition

Selected file is encrypted and saved with a new encrypted file name.

2.1.1.4. Use Case Diagram



2.1.1.5. Requirement 4: Decrypt File

This requirement discusses the decryption of a file.

Description & Priority

Another one of the main functions of the application to decrypt the file that has been encrypted previously with the same key which was used to encrypt the application. High priority.

2.1.1.6. Use Case

Scope

Decryption of files.

Description

Allows user to select a file which he would like to decrypt.

Flow Description

Precondition

A file has been encrypted.
User has generated/has a decryption key.

Activation

This use case starts when the user clicks into the decryption tab.

Main flow

1. User click “Decrypt” Button.
2. User selects an encrypted file to decrypt.
3. Application verifies that the selected file is a valid encrypted file.
4. Application asks user to enter decryption key.
5. User import decryption key.
6. Application verifies it is the correct decryption key.
7. User clicks “Decrypt file” button.
8. Application decrypts selected file using the decryption key.
9. Application prompts user to select location to save the decrypted file.
10. Users selects location to save the file.
11. Application saves the file to selected location.
12. Application displays a success message to the user.

Alternate flow

A1 : <Not an encrypted file >

1. User clicks a file to decrypt which is not encrypted.
2. Application displays an error and prompts user to select another file.
3. Continue main flow 2.

A2 : <User doesn’t specify save location >

1. User decrypt a file.
2. Application asks user for location to save.
3. User exits out of save location.
4. Application does not complete decryption.
5. Continue main flow 9.

Exceptional flow

E1 : <Decryption key not valid>

1. User selects file to decrypt.
2. Application prompts for a decryption key.
3. User selects wrong decryption key.
4. Application tries to decrypt.
5. Application throws an error due to wrong decryption key.
6. Continue main flow 5.

Termination

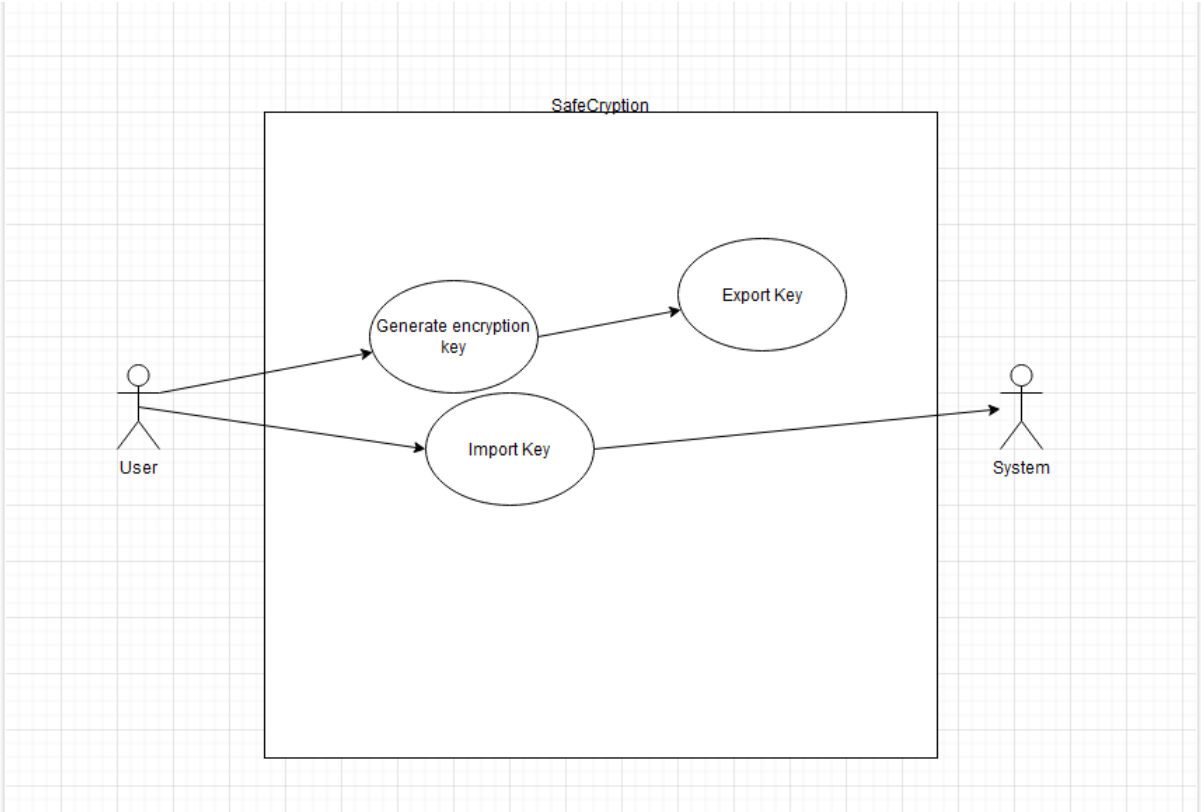
Encrypted file is successfully decrypted and saved to user specified location.

Post condition

Encrypted file is decrypted using key the user provided.

User has generated a valid decryption key using the application.

2.1.1.7. Use Case Diagram



Requirement 2: Decrypt File

This requirement discusses encryption/decryption keys

Description & Priority

This use case describes how a use can generate and manage their own encryptions and decryption keys and be able to export/import them. It has a high priority because it involves sensitive information which the user must store securely.

2.1.1.8. Use Case

Scope

This applies to all users who use the system.

Description

Allows users to export and import encryption keys.

Flow Description

Precondition

The user must have access to the encryption feature in the system and understand how to use it.

Activation

The user opens the encryption tab and chooses to generate a new encryption key and import it.

Main flow

1. The user selects the “Generate encryption key” option in the encryption tab and the system generates the user a new encryption key for them.
2. User selects to export the key.
3. The system prompts the user to for a path to save the key.
4. The user can choose to import a key in the decryption tab and the user prompts the user to select the file path where the key is stored.

Alternate flow

A1 : < Key Format >

1. If the user chooses to export the key, they can choose the format of the exported key. Continue at 3

Exceptional flow

E1 : <No valid key>

1. If user has no existing key, the system prompts them to generate a new key. Continue at 1

Termination

User successfully exports/imports a key.

Post condition

The user has generated a new encryption key, and has an option to export or import them as needed.

2.1.2. Data Requirements

User Files

A user will need to upload a file to the application to get it encrypted. These files can be of any type/size, and they are the data that the application manipulates. The application may require some metadata about the file being encrypted/decrypted such as name, size, and type this is to validate user inputs and be able to save the files when they are done being encrypted/decrypted with appropriate names and extensions.

Encryption Key

In order for the application to be able to encrypt and decrypt files the user must either provide a valid encryption key or generate a random encryption key using the applications features. The key must be safe and secure and only authorized by the user being able to withstand attacks. In order for this to withstand attacks it must be complex and lengthy. Before properly starting to use the application, the user must understand the importance of what the encryption keys do and how they are stored. As if an encryption key is lost the data that has been encrypted with it will also be lost.

2.1.3. User Requirements

User requirements are what a user will need from the system.

- User should be able to select files to be encrypted/decrypted from their computer and be able to save resulting files to a location of their choice.
- User should be able to generate new encryption keys and be able to save it in a safe location.
- User should be able to import a pre-existing encryption key to decrypt a file and be warned if an incorrect key is used.
- User should be able to select multiple files encryption/decryption.
- Application should be secure and reliable with there being no risk or data loss or corruption during encryption/decryption process.

2.1.4. Environmental Requirements

- Up to date operating system, Windows, Mac, Linux
- Python version 3.11 in order to run the application.
- Python libraries Used Pillow(PIL), cryptography, tkinter, random, time, OS.

2.1.5. Usability Requirements

Usability requirements state how easy the application should be for the user to use.

- User friendly interface to help guise user's around the application.
- Clear instructions must be provided to guide the user through the encryption/decryption process.
- Application should display clear error messages and a user makes a mistake or when something goes wrong within the application.
- Application should give the user feedback during the encryption/decryption process.
- Application should be compatible with different operating systems.

2.2. Design & Architecture

The application was designed with the python library tkinter which allows for a simple and clean GUI to be created with the code. Before creating the GUI I had sketched up prototypes of what I want the application to look like in order to get a better understanding in my head what I am aiming for.

This is the main home page prototype that I had set out to achieve. It is basic and helps navigate the user around the whole application with ease. It has all the features in an image encrypt, decrypt, and help section.

Next, I developed the encryption tab which the user can pick from a file that he wishes to encrypt, the user clicks the generate secret key button and is given an encryption key. In order to keep the application, secure I made it, so the key is generated in a .key file instead of appearing on the application. The user can export this key if wish to and store it in a safe place.

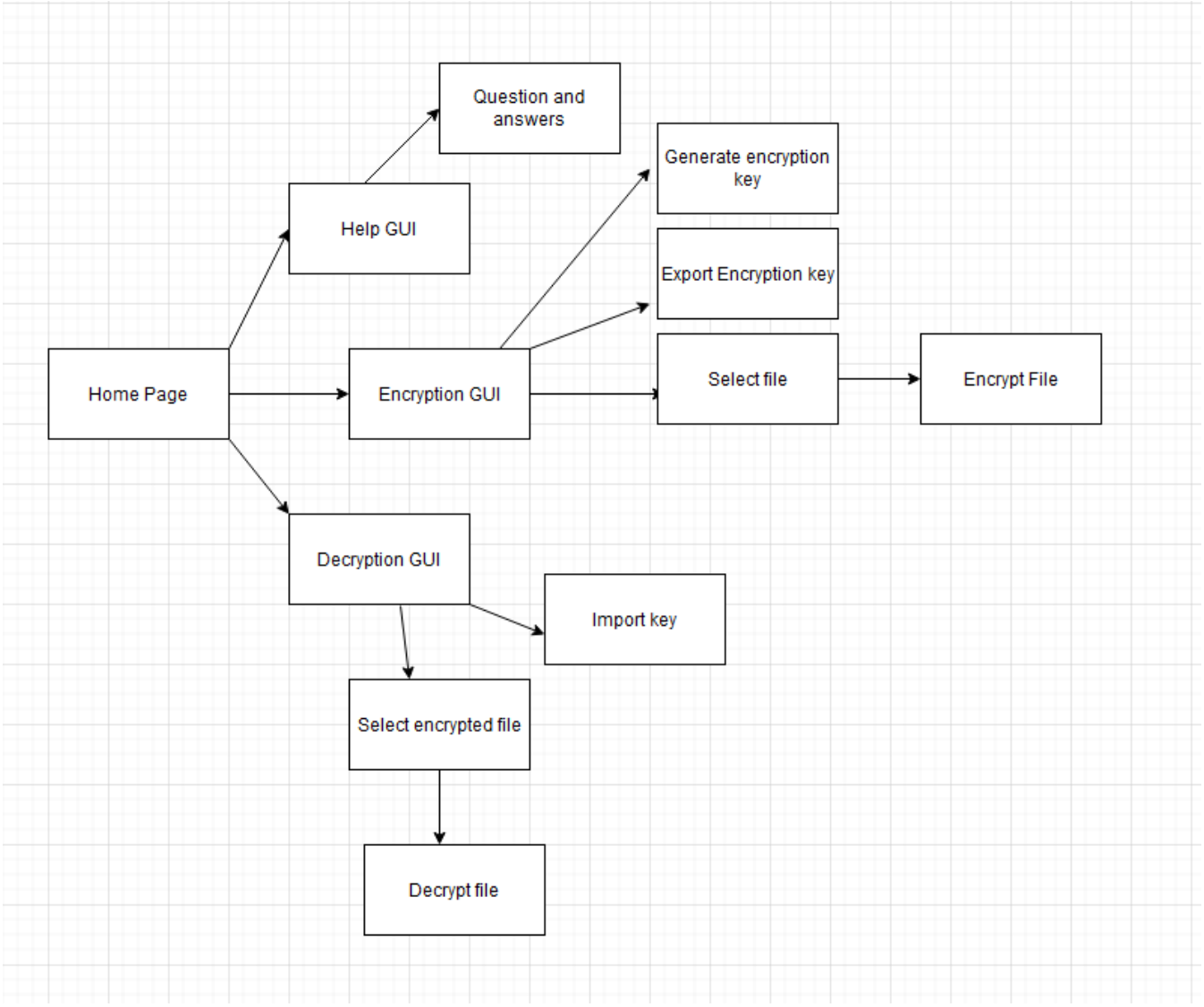
In the decryption that you can see the file selection dialog to choose the encrypted file that you would like to decrypt and the unique key which was changed in the final product. The key must be imported via a .key file which the application checks when the decryption process happens.

The help tab has simple FAQ which user's might be confused about. It's just simple question and answers. In the final application it is very similar although the user can choose from what question they are searching for and receive the answer to it.

System architecture of this application is a client-server model which means it's operated on the user's machine and doesn't connect to a database. Although it has Python's

built in modules which are third-party libraries which help the functionality of the application, using the cryptography module it generates encryption keys which perform the encryption process without transmitting anything to the third-party library. The only purpose that the libraries serve is performing cryptographic operations.

The whole code is written with Python implementing safe and secure ways to encrypt and decrypt your files. This is done with the cryptography Fernet third-party library which is imported the application encrypts and decrypts the files using symmetric encryption.



2.3. Implementation

In order to implement a GUI into the application it was done using the notebook widget from one of the python imported libraries called tkinter. In the screenshot below you can see a snippet of code which was used in order to implement the home tab. Starting off with creating an instance of the ttk.Frame widget which is then assigned to the home tab attribute, this is done in so the frame can hold the widgets will be used on the home tab. Followed by adding the Home tab at the top of the application for use navigation around the

page application. Then you can see just a simple label which is a welcome to safecryption and some aesthetics such as font, size, and background colour. Then the same is done for the encryption, decryption and help buttons which appear on the home page this again is for the user to help navigate around the application with ease.

```
#Home Tab
self.home_tab = ttk.Frame(self.notebook, padding=10)
self.notebook.add(self.home_tab, text="Home")
self.home_label = Label(self.home_tab, text= "Welcome to SafeCryption", font= ("Helvetica", 16, "bold"),background='#ADD8E6')
self.home_label.pack(pady=10)
#Encrypt and Decrypt buttons at home tab
self.encrypt_button = Button(self.home_tab, text="Encrypt", font=("Helvetica", 10, "bold"),height=70, width=100, image=self.encrypt_photo, compound="top", command=lambda:
self.encrypt_button.pack(pady=10, padx=25, side="left", anchor="n")
self.decrypt_button = Button(self.home_tab, text="Decrypt", font=("Helvetica", 10, "bold"),height=70, width=100, image=self.decryption_photo, compound="top", command=lamb
self.decrypt_button.pack(pady=10, padx=25, side="left", anchor="n")
self.help_button = Button(self.home_tab, text="Help me", font=("Helvetica", 10, "bold"),height=70, width=100, image=self.help_photo, compound="top", command=lambda: self.
self.help_button.pack(pady=10, padx=25, side="left", anchor="n")

#Background colour of tabs
new_style = ttk.Style()
new_style.configure('TFrame', background='#ADD8E6')
self.home_tab.style = new_style
self.home_tab.configure(style='TFrame')
```

The application contains four tabs which are home, encrypt, decrypt, and help which can be seen in the GUI section of the documentation. With the home page providing guidance around the application helping you choose for what you would like to do. Encryption tab has all the tools to encrypt a file or your choosing including generating an encryption key, exporting a generated encryption key, selecting a file to encrypt, and encrypting it. When a user encrypts a file there is also a progress bar that pops up to let the user know the progress of the encryption status in order to give them more feed. The decrypt follows a similar style to the encrypt tab. With the help tab there are a few common questions that are asked about the application which will help user get a better understanding for the application and what it can do, what files it can encrypt and what happens if an encryption key is lost. The GUI is designed on allowing the user ease of access with having icon beside all the buttons, so it is more appealing to the eye, having clear and concise instructions for each step of encryption and decryption. With implementation of error handling the user will receive error messages when they miss a step or incorrect input has been entered. As you can see in the screenshot below.

```
13  def encrypt_file(self, filenames):
14          ENCRYPTED_FILE_SIGNATURE = b"ENCRYPTED:"
15          #Error Handling
16          if self.key is None:
17              messagebox.showerror("Error", "Please generate a key.")
18              return
19          print(self.filenames)
20          if not filenames:
21              messagebox.showerror("Error", "Please one or more files.")
22              return
23
```

Encryption is done using the Fernet symmetric encryption algorithm which is from the cryptography library. Fernet is a symmetric encryption algorithm which is based on

the AES encryption algorithm in CBC mode which combines the plaintext of a block and combines it with a ciphertext of the previous block resulting in encryption, with a 128-bit key size. This process involves generating a key which is a random sequence of bytes which is then used to encrypt the file. As you can see in screenshot below.

```
4   def generate_key(self):
5           # Generate a random key for encryption
6           self.key = Fernet.generate_key()
7           messagebox.showinfo("Success", "A random key has been generated.")
8
```

Then the key can be exported and be stored for safe keeping in order to later decrypt that file. The Below screenshot is what the exported encryption key looks like.

test.key

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000   58 5F 39 4A 46 62 43 44 37 43 4A 35 44 38 2D 6C  X_9JFbCD7CJ5D8-l
00000010   48 46 75 48 37 54 6A 66 5F 59 61 39 4B 54 37 43  HFuH7Tjf_Ya9KT7C
00000020   63 70 32 57 57 51 76 33 52 4A 59 3D              cp2WWQv3RJY=
```

When a file is encrypted the encryption application adds a signature at the begging of the file in order to know which files have been encrypted so the application doesn't allow for encryption of the same file twice. This is also important for the decryption process so the application knows which files are encrypted so it can decrypt them with the given key, this decrypts the data and write's it back to the original file overwriting the encrypted data. If a user tries to open a file that has been encrypted, they will either be unable to do so, or they will just get a lot of nonsense and not be able to make out what that file was about. The screenshot below shows the line of code which is responsible for creating an instance of the Fernet class from the cryptography.Fernet module it uses the self.key which is the encryption key that was generated in order to initialize the encryption and decryption.

```
23
24      cipher_suite = Fernet(self.key)
25
```

Once the application creates the cipher_suite object it can be used to encrypt data by calling the encrypt method. In the screenshot below you can see the code which is responsible for encrypting the files. With the application starting with opening the file and reading it in binary mode this done when dealing with non-text files such as images or audio files it is opened and read in a sequence of bytes which can be done by anyone if you open a file with a simple digital forensics' application such as hex editor, when encrypting the data is modified in the byte level. When the file has been read in binary mode it is assigned the variable plaintext which the bytes of that file are then encrypted using cipher_suite object using the encryption key. Then in the file a signature is written so that the program can recognize this later in order to know that it has been encrypted. F.write(encrypted_text) writes the encrypted data with the signature onto the file and the next two lines ensure that this is done properly and not just buffered in the memory and actually written into the file.

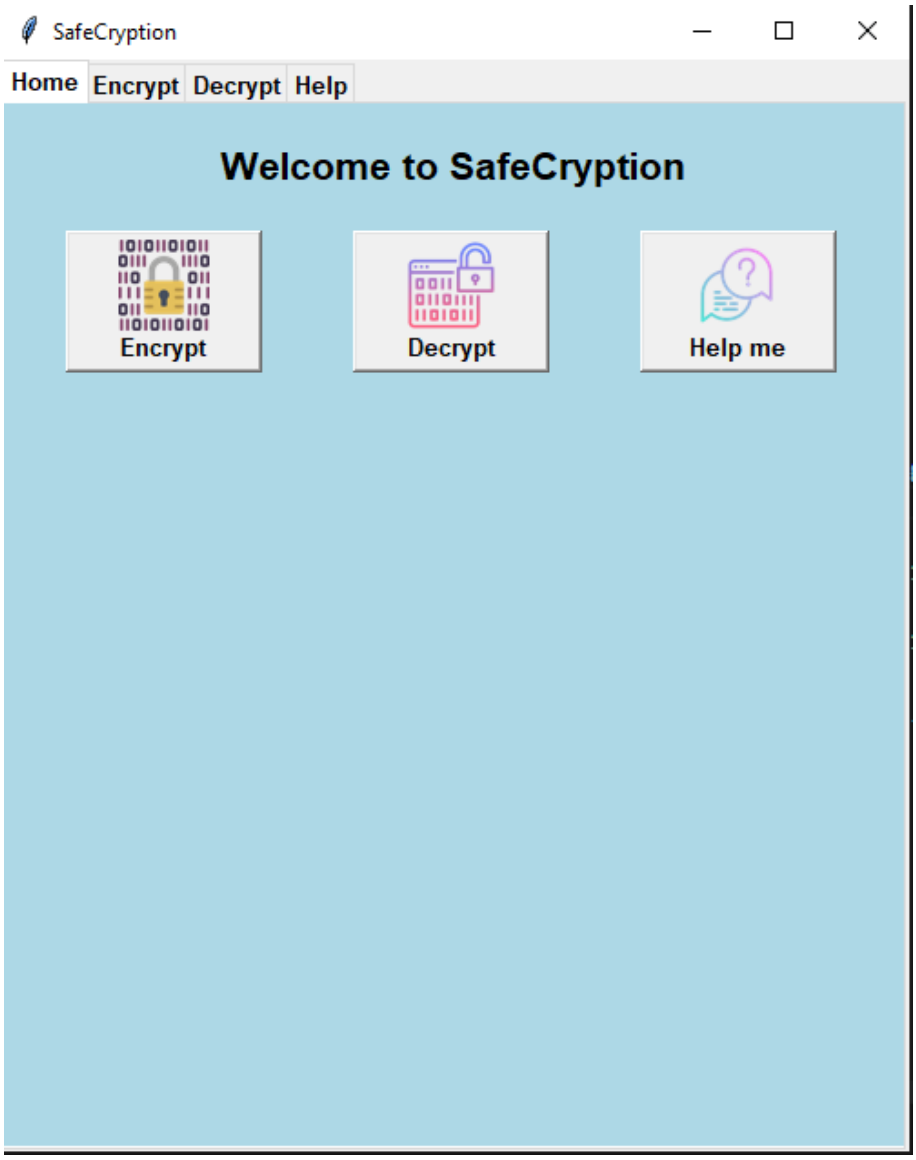
```
36          # Encrypt the file using the key
37          with open(filename, "rb") as f:
38              plaintext = f.read()
39          encrypted_text = cipher_suite.encrypt(plaintext)
40          encrypted_text = ENCRYPTED_FILE_SIGNATURE + encrypted_text
41          with open(filename, "wb") as f:
42              f.write(encrypted_text)
43              f.flush()
44              os.fsync(f.fileno())
```

As mentioned previously when a file is encrypted there is a line added to the signature of the file to let the application know that it has been encrypted. In the screenshot below I opened a file which is encrypted in a simple digital forensics' application called hex editor. This allows you to see what is written in the encrypted file and the file signature which will be ENCYRPTED. The highlighted part is what the application writes into the file signature and then the rest is what is included in the file. This was previously just a random Lorem Ipsum generated word document, but it had writing that you could make out. When the file is encrypted, everything is scrambled, and you cannot make out any part of it.

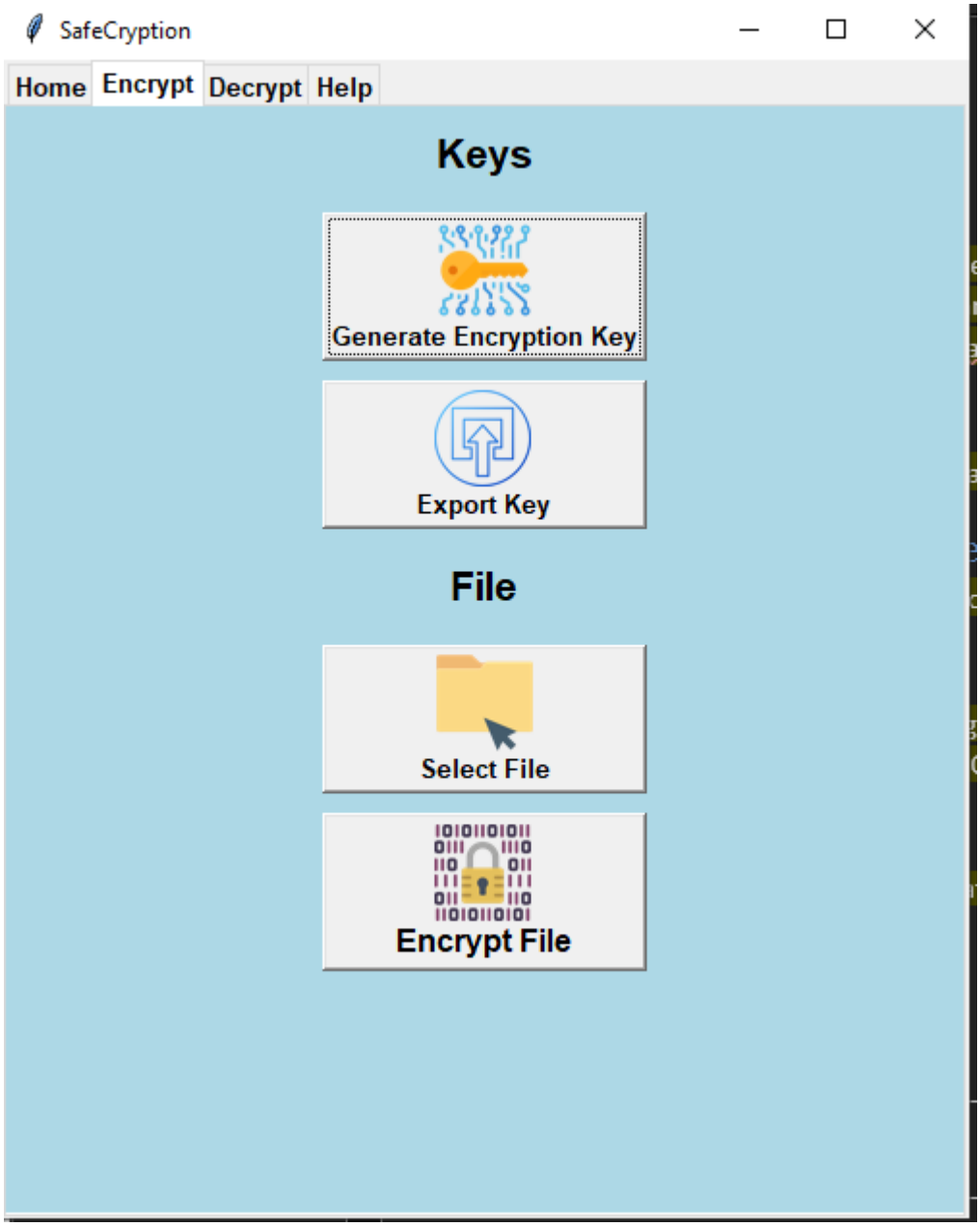
Test file.docx

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	45	4E	43	52	59	50	54	45	44	3A	67	41	41	41	41	41	ENCRYPTED:gAAAAA
00000010	42	6B	57	72	50	54	61	44	56	52	50	6D	76	52	7A	50	BkWrPTaDVRPmvRzP
00000020	41	62	33	35	41	71	35	46	4D	64	46	4B	69	70	6E	44	Ab35Aq5FMdFKipnD
00000030	37	53	44	61	33	4C	56	4F	38	74	48	70	56	71	78	54	7SDa3LVO8tHpVqxT
00000040	53	47	62	31	77	32	47	37	6B	2D	38	66	5A	6E	38	33	SGb1w2G7k-8fZn83
00000050	78	78	7A	58	57	53	37	4A	59	72	44	57	4C	66	4A	7A	xxzXWS7JYrDWLfJz
00000060	6E	6C	6F	51	6C	4F	53	68	72	2D	32	7A	61	66	4F	62	nloQlOShr-2zafOb
00000070	62	34	48	57	61	33	71	66	37	37	53	4E	70	50	4B	56	b4HWa3qf77SNpPKV
00000080	5A	4F	74	7A	38	75	44	55	75	4F	30	59	75	6B	78	54	ZOtz8uDUuO0YukxT
00000090	68	42	30	78	67	50	51	56	4A	6D	49	4D	6B	34	7A	57	hB0xgPQVJmIMk4zW
000000A0	68	32	73	2D	48	49	51	48	79	4C	74	36	76	49	34	63	h2s-HIQHyLt6vI4c
000000B0	61	51	73	44	49	53	38	6D	56	56	50	2D	34	51	39	7A	aQsDIS8mVVP-4Q9z
000000C0	62	48	47	47	51	64	30	69	59	44	79	5A	56	73	61	7A	bHGGQd0iYDyZVsaz
000000D0	77	53	49	34	71	6E	48	45	35	41	59	47	2D	76	4A	6C	wSI4qnHE5AYG-vJl
000000E0	4C	4C	77	30	69	42	78	4C	62	41	74	35	69	39	54	4A	LLw0iBxLbAt5i9TJ
000000F0	53	69	4B	41	65	58	45	77	58	69	6D	5F	7A	6A	76	53	SiKAeXEwXim_zjvS
00000100	62	2D	41	59	73	79	6B	54	5A	53	6D	34	6C	6A	52	64	b-AYsykTZSm4ljRd
00000110	5F	78	47	50	4F	65	4C	53	71	46	33	2D	45	79	68	41	_xGPOeLSqF3-EyhA
00000120	41	61	4D	53	61	74	69	79	38	49	43	4C	51	48	41	37	AaMSatiy8ICLQHA7
00000130	32	4C	34	44	59	66	42	6B	42	58	50	79	69	50	54	55	2L4DYfBkBXPyiPTU
00000140	71	4C	56	4B	49	61	2D	52	45	31	4A	79	4D	69	69	67	qLVKIa-RE1JyMiig
00000150	52	74	4E	48	52	35	6B	48	47	61	4C	48	32	4B	59	6E	RtNHR5kHGaLH2KYn
00000160	52	57	56	67	46	33	74	54	2D	30	70	41	41	70	59	48	RWVgF3tT-0pAApYH
00000170	69	41	57	39	57	5F	33	74	31	32	30	6C	6C	6A	54	75	iAW9W_3t120lljTu
00000180	35	39	6B	65	77	4E	78	67	6E	75	43	79	70	55	6E	45	59kewNxgnuCypUnE
00000190	74	41	50	4D	33	5F	42	30	39	43	75	39	4B	6F	41	30	tAPM3_B09Cu9KoA0
000001A0	58	72	41	35	75	47	43	46	33	4D	55	61	55	5F	51	67	XrA5uGCF3MUaU_Qg
000001B0	31	6A	57	65	51	78	71	68	35	59	56	33	6A	78	4A	39	1jWeQxqh5YV3jxJ9
000001C0	4C	78	53	47	75	57	34	33	5F	72	52	6C	77	6B	39	6E	LxSGuW43_rRlwk9n
000001D0	61	65	4B	36	4C	7A	70	57	6E	48	79	76	63	75	75	68	aeK6LzpWnHyvcuuh
000001E0	44	53	6F	44	79	59	51	55	66	62	5F	4F	59	35	6C	54	DSoDyYQUfb_OY5lT
000001F0	51	77	4C	44	6C	32	77	4A	5A	58	61	72	56	4A	66	54	QwLDl2wJZXarVJfT
00000200	54	4C	71	66	6F	41	33	59	63	4A	39	36	72	2D	4C	74	TLqfoA3YcJ96r-Lt
00000210	74	2D	32	77	49	64	63	66	37	62	41	4D	6C	58	66	37	t-2wIdcf7bAMlXf7
00000220	48	34	6C	6F	4B	2D	4E	75	4D	6C	30	51	31	7A	65	6E	H4loK-NuMl0Q1zen
00000230	45	4D	4A	38	6C	43	58	52	59	50	50	74	46	5F	4B	51	EMJ8lCXRYPPtF_KQ
00000240	32	49	52	6C	68	77	63	45	74	63	30	30	53	5A	72	46	2IRlhwcEtc00SZrF

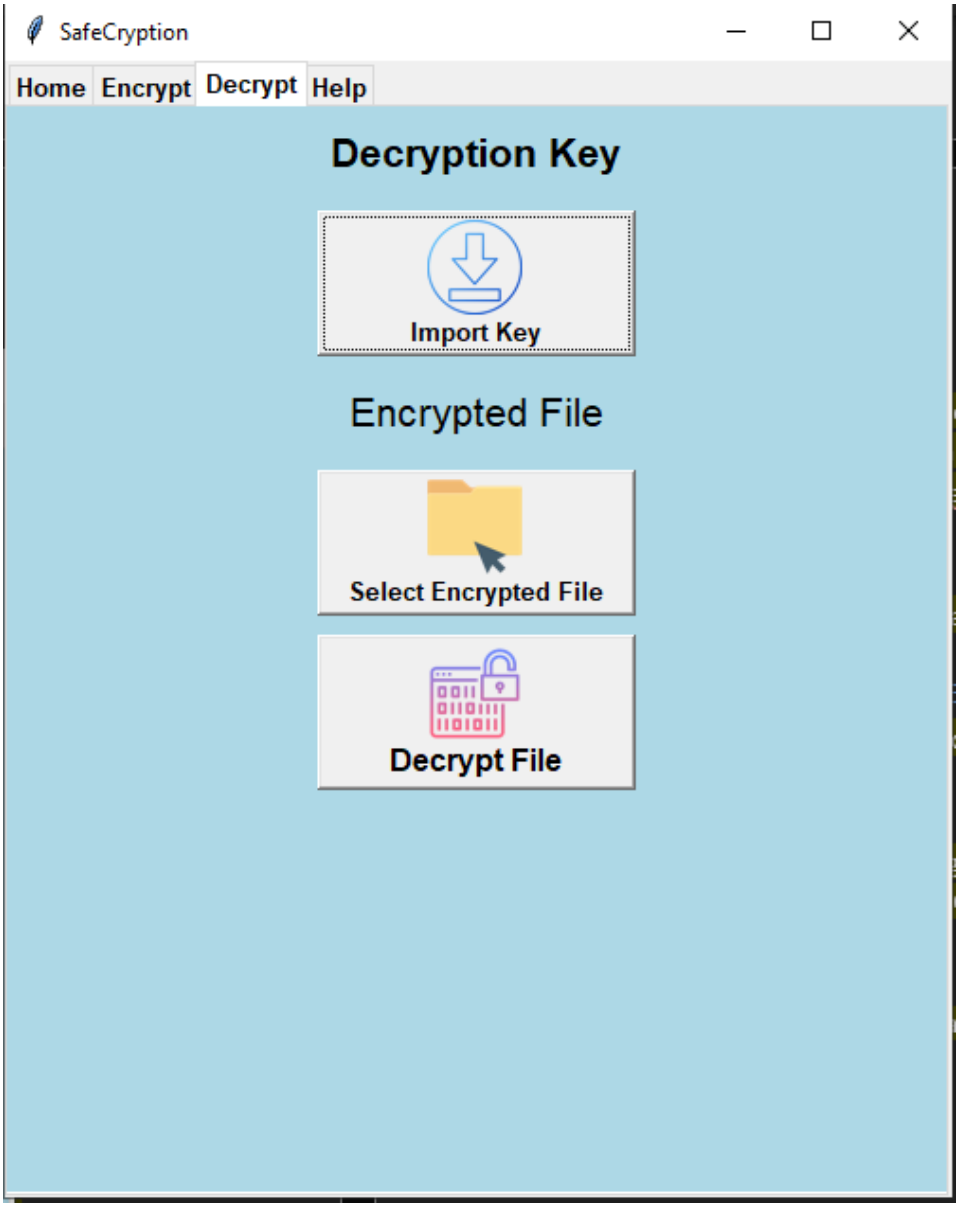
2.4. Graphical User Interface (GUI)



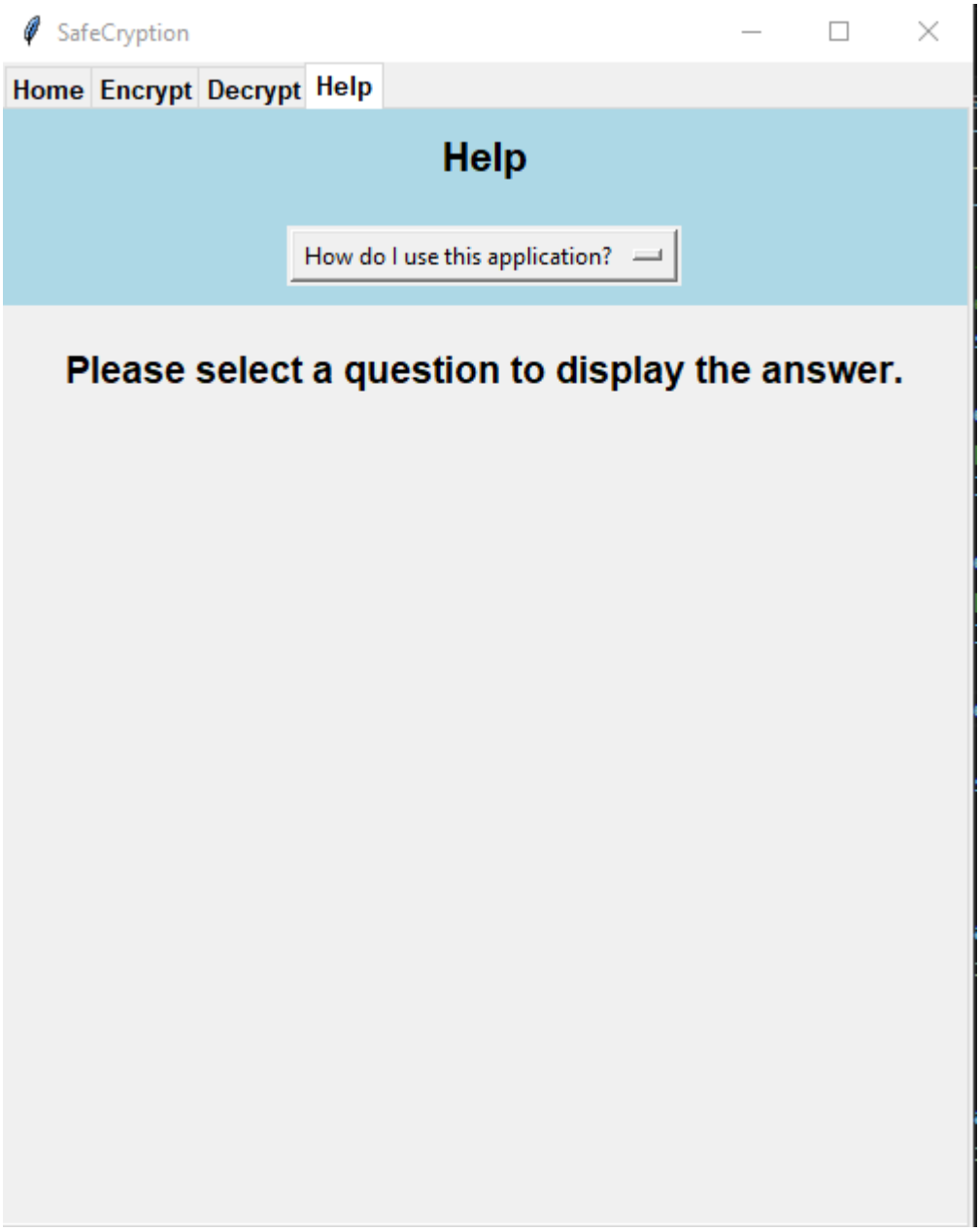
This is the first screen that the user will see when they run the application. It is a basic home page with buttons which lead to each of the tabs. The user can choose from encrypt, decrypt, and help me which is the help page.



If the user decides to click the Encrypt in the home page or go into it themselves using the tabs above, they will be brough to this page which they have all the tools they need to encrypt their file. At the top you will see keys which the user can generate a random encryption key and export it if they wish to do so for safe keeping. Under the keys is the file section which the user can select a file they would like to encrypt and once done so they can proceed to encrypting said file.



The decryption tab is very similar to the encryption tab it has the decryption keys which can be imported or if a random key was generated before and the app wasn't closed you will not need to import a decryption key as one will already be in use. The select encrypted file dialog is also there and then finally when the user is ready, they can click decrypt file.



This is the help page which has Frequently asked question that user can choose from in order to get a better understanding of the application or the limitations. Once a user selects one of the following questions the answer will appear in the text box under the question answering their questions.

2.5. Testing

End user testing

I done some user testing where I invited some colleagues to test out my application and see what they think of it. Most of them only knew the basics of technology but do

not know how programs and applications are coded and worked on so it was a perfectly opportunity to get some opinions of people who aren't tech savvy. The test was conducted without giving them any prior training of the application and just letting them navigate through and figure it out for themselves. I had asked them to encrypt and decrypt a given file and seen how they got on. At first, they didn't understand what to do with the application but after clicking through the tabs and going into the help page to they start to get the hang of it. I made them fill out a questioner afterwards which I have posted the results of this below.

How did you find design of the user interface?

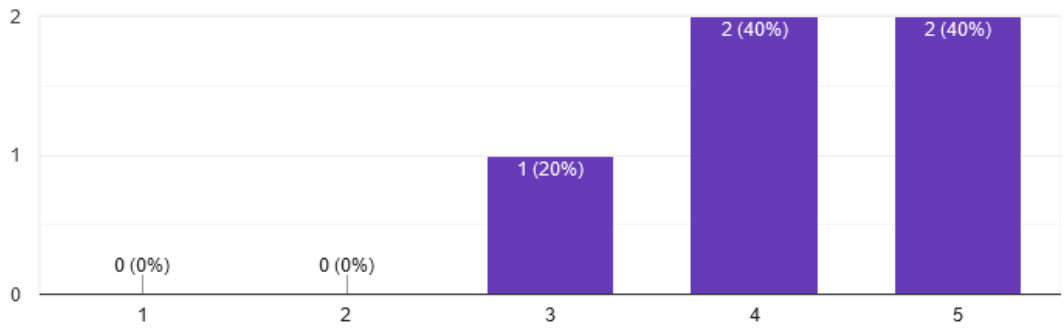
5 responses

- It was good
- Great
- Good
- User Friendly and straight to the point
- amazing

How easy was the app to navigate?

Copy

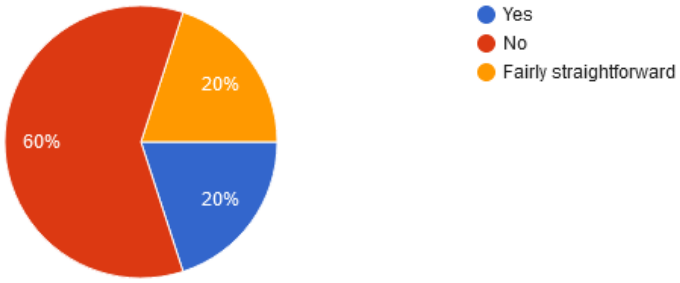
5 responses



Do you think training is needed before using this app?

Copy

5 responses



Is there any improvements that you think we could add?

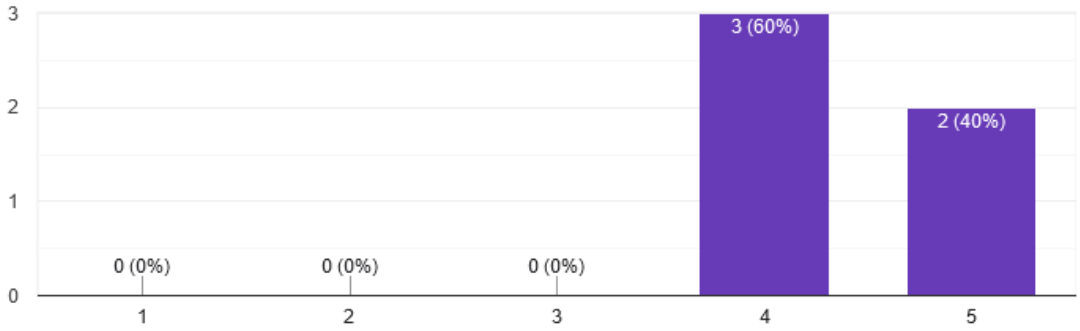
4 responses

Nothing
No
More colorful layout
more design

How easy is it to use?

Copy

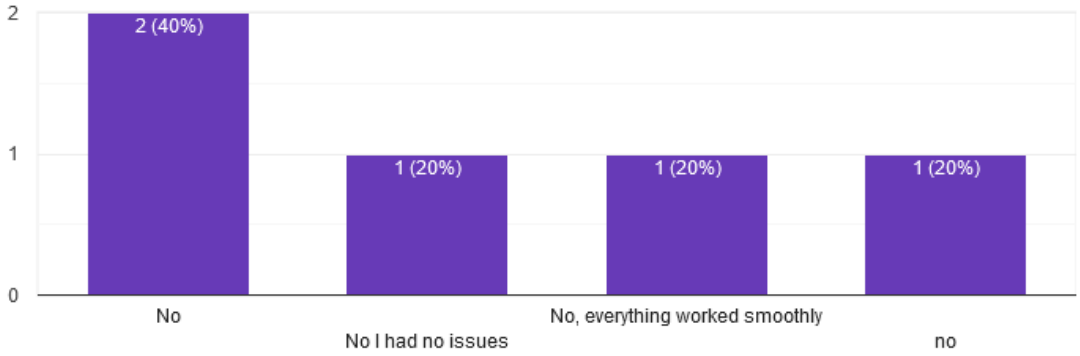
5 responses

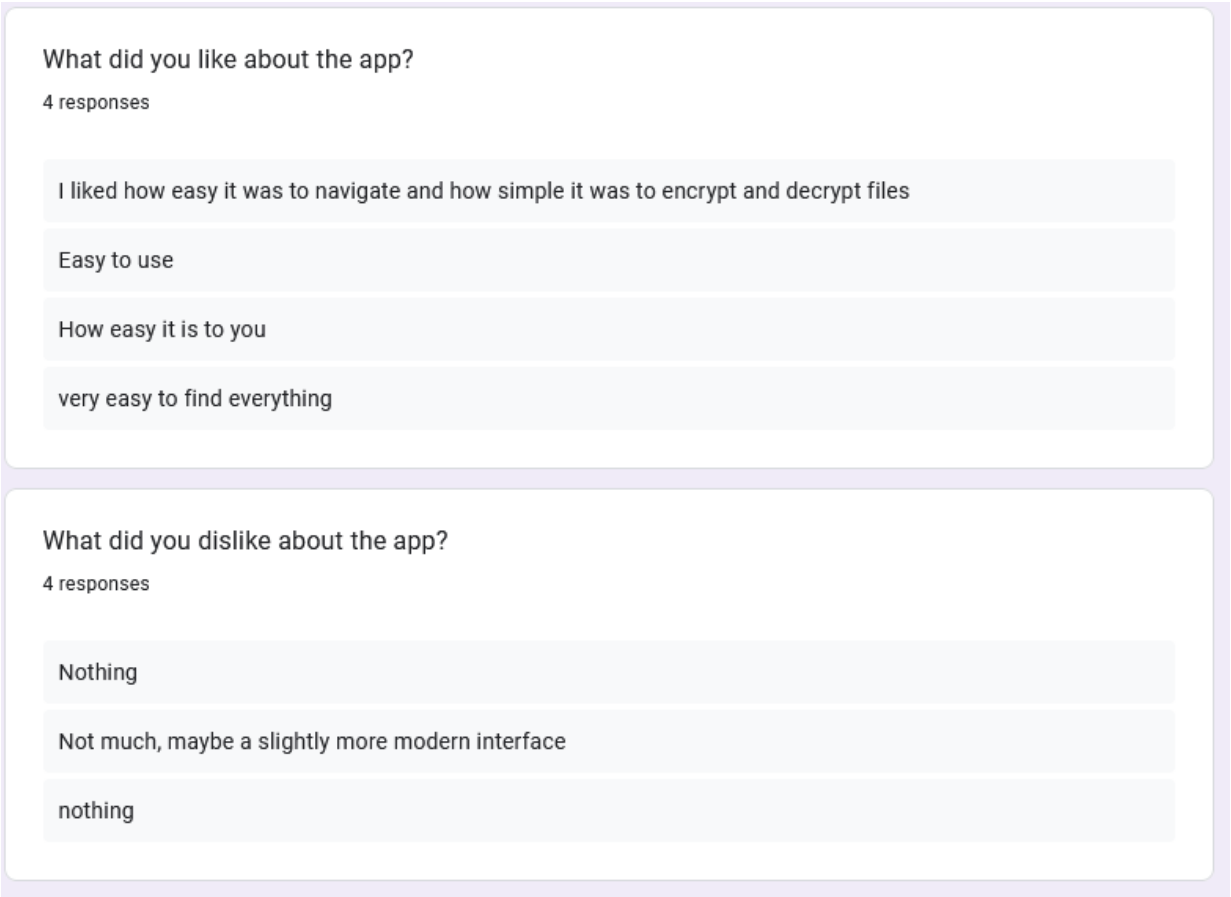


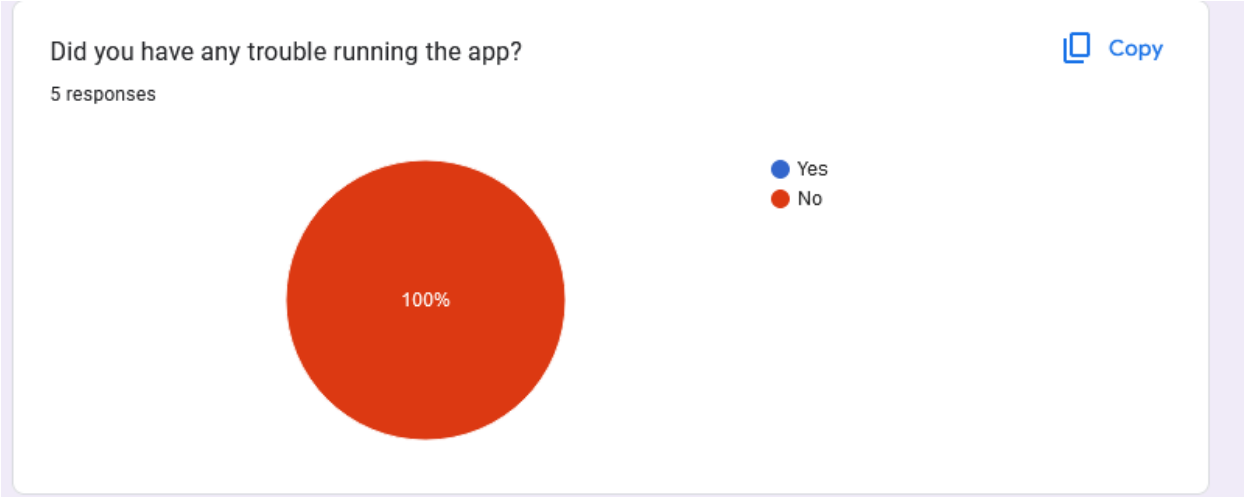
Did you have any issues with the app?

Copy

5 responses







Unit Testing

Test case: Generate encryption key.

Description: Before encrypting a file, a user must first generate an encryption key which will be used to scramble the file in order to make it unreadable.

Expected result: Pop up will appear stating an encryption key has been successfully generated.

Actual Result: As expected.

```python
def generate_key(self):
        # Generate a random key for encryption
        self.key = Fernet.generate_key()
        messagebox.showinfo("Success", "A random key has been generated.")

```

Test Case: Exporting encryption key.

Description: User must be able to export their encryption key for safe keeping in order to decrypt files at a later time.

Expected result: User is prompted with a directory where they wish to export the encryption key.

Actual Result: As expected.

```python
def export_key(self):
        # Export the key to a file
        if self.key is None:
            messagebox.showerror("Error", "Please generate a key.")
        else:
            filename = filedialog.asksaveasfilename(defaultextension=".key")
            if not filename:
                return
            with open(filename, "wb") as f:
                f.write(self.key)
                messagebox.showinfo("Success", "Key has been exported to {}".format(filename))

def import_key(self):
```

Test Case: File selection for encryption/decryption

Description: When a user has generated an encryption key, they will need to select a file, by clicking the select file button.

Expected result: Use is prompted with a directory, and they must choose which file to encrypt.

Actual Result: As expected.

```
def select_file(self):
    # Show a file selection dialog
    self.filenames = filedialog.askopenfilenames()
```

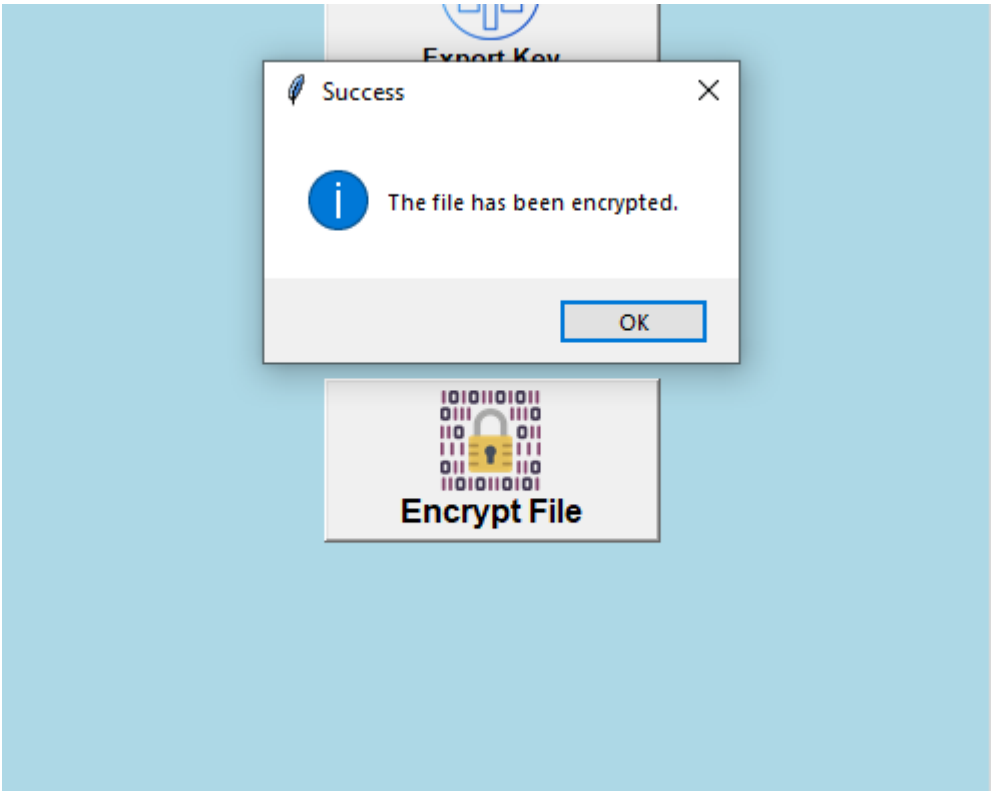
Test Case: Encrypt File.

Description: When a user has generated an encryption key and selected a file they can then encrypt that said file by clicking the button encrypt in the encryption tab.

Expected result: Pop up appears letting the user know that the file has been successfully encrypted and the file becomes encrypted.

Actual Result: As expected.

```
13  def encrypt_file(self, filenames):
14          ENCRYPTED_FILE_SIGNATURE = b"ENCRYPTED:"
15          #Error Handling
16          if self.key is None:
17              messagebox.showerror("Error", "Please generate a key.")
18              return
19          if not filenames:
20              messagebox.showerror("Error", "Please one or more files.")
21              return
22
23          cipher_suite = Fernet(self.key)
24
25          for idx, filename in enumerate(self.filenames):
26              if not os.path.isfile(filename):
27                  messagebox.showerror("Error", f"{filename} does not exist.")
28                  continue
29              elif filename.startswith(ENCRYPTED_FILE_SIGNATURE.decode()):
30                  messagebox.showerror("Error", f"{filename} is already encrypted")
31                  continue
32              # Encrypt the file using the key
33              with open(filename, "rb") as f:
34                  plaintext = f.read()
35              encrypted_text = cipher_suite.encrypt(plaintext)
36              encrypted_text = ENCRYPTED_FILE_SIGNATURE + encrypted_text
37              with open(filename, "wb") as f:
38                  f.write(encrypted_text)
39                  f.flush()
40                  os.fsync(f.fileno())
```



Test Case: Importing decryption key.

Description: If a user has previously encrypted a file with a key and then reset the application, they must import that previous key in again to decrypt said file.

Expected result: User is prompted with a directory where they must point to where the encryption key is stored. Once opened a pop up will appear to let the user know that the encryption key has been imported so that they can decrypt.

Actual Result: As expected.

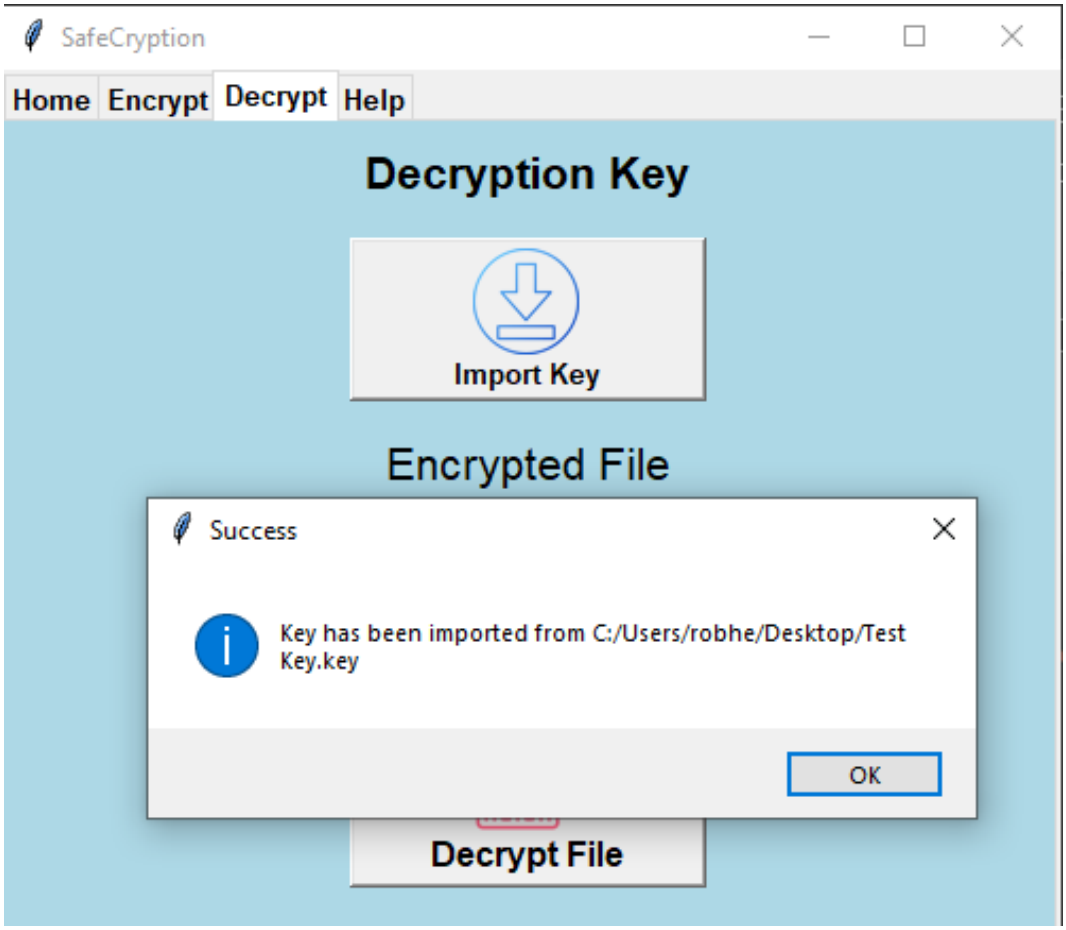
```python
def import_key(self):
        #File selection
        filename = filedialog.askopenfilename()

        #If user exits dialog do nothing
        if not filename:
            return

        #Read the key
        with open(filename, "rb") as f:
            key = f.read()

        #check if the key is valid
        try:
            Fernet(key)
        except:
            messagebox.showerror("Error", "Invalid key file")
            return

        #Key variable to imported key
        self.key = key
        messagebox.showinfo("Success", "Key has been imported from {}".format(filename))
```



Test Case: Decrypt File

Description: When a user has encrypted a file and is now ready to decrypt, they can go into the decryption tab and select a file which they would like to decrypt. The application reads this encrypted file to make sure that it is really encrypted and proceeds to decrypt it with either the key already generated or an imported key.

Expected result: The file that the user selects are decrypted, and a pop up appears letting the user know that it was successful.

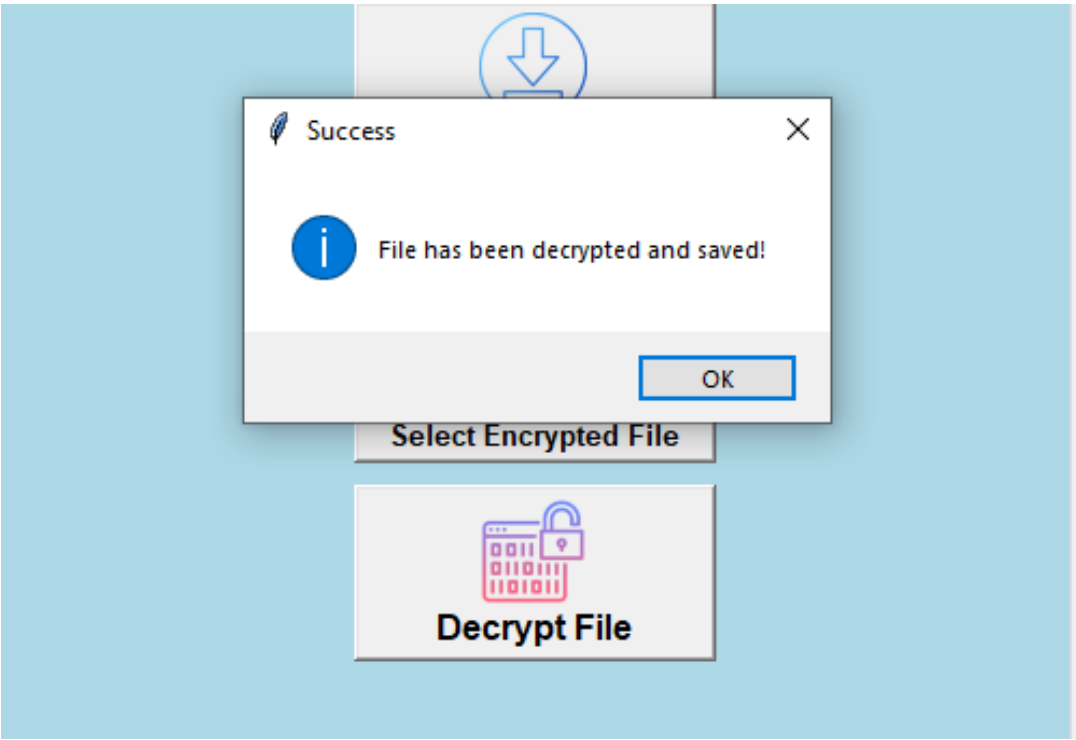
Actual Result: As expected.

```
def decrypt_file(self, filenames):
        ENCRYPTED_FILE_SIGNATURE = b"ENCRYPTED:"
        #Error Handling
        if self.key is None:
            messagebox.showerror("Error", "Please generate a key.")
        if not self.filenames:
            messagebox.showerror("Error", "Please one or more files.")
            return

        cipher_suite = Fernet(self.key)

        for idx,filename in enumerate(self.filenames):
            if not os.path.isfile(filename):
                messagebox.showerror("Error", f"{filenames} does not exist")
                continue
            with open(filename, "rb") as f:
                signature = f.read(len(ENCRYPTED_FILE_SIGNATURE))
                if signature != ENCRYPTED_FILE_SIGNATURE:
                    messagebox.showerror("Error", f"{filenames} is not encrypted")
                    continue

            # Decrypt the file using the key
                encrypted_text = f.read()
            try:
                decrypted_text = cipher_suite.decrypt(encrypted_text)
            except:
                messagebox.showerror("Error", "Incorrect Decryption key, please try again!")
                return
            with open(filename, "wb") as f:
                f.write(decrypted_text)
                f.flush()
                os.fsync(f.fileno())
```



2.6. Evaluation

The application was evaluated through a combination of user testing and unit testing. For unit testing it was done throughout the application while it was being developed to make sure all the implemented features are working properly alongside the error handling. For user testing I got a group of colleagues together who tested the final product of the application, and I presented the results which they have filled out. User feedback was good for the application as I got an insight into how the application works and what needs to be changed. To evaluate the encryption and decryption algorithms I downloaded some test files which helped me encrypt and decrypt various file types and sizes to see how the application would respond. Resulting in the application being able to encrypt various sizes and types, although with some files which are bigger in size the application slowed down a little bit and it took a bit longer than expected.

Using Nielsen's Heuristics, the GUI of the application can be evaluated which will help identify any issues with usability or user interface design. Using the 10 points I will evaluate my application's GUI.

1. Visibility of system status: The GUI has a bar to show progress of any file that is being encrypted/decrypted in order to enhance the user experience.
2. Match between System and real world: Application uses images that correspond to real-world objects, such as the lock for encryption and a key for generating encryption keys. Making it easier for the user to understand functionality of the application.

3. User Control and freedom: User's can navigate freely between different tabs and the home tab provides access to each individual tab with a picture or they can use the tabs at the top.
4. Consistency and standards: The GUI has consistent button styles and layouts throughout the whole application even matching up the images so that they are all coloured and similar format.
5. Error prevention: When a user makes a mistake in the application and tries to encrypt a file before choosing one or before generating a key, they are presented with an error letting them know what they missed.
6. Recognition rather than recall: Having clear and visible navigation buttons help's user know where each part of the application is located.
7. Flexibility and efficiency of use: User's have quick access to frequently used features such as file select and encryption.
8. Aesthetic and minimalist design: The GUI uses a minimalist design with a simple and clean layout. With all the fonts matching up together.
9. Help users recognize, diagnose, and recover from errors: The application incorporates error checking when a user makes a mistake or tries to move a step forward too quick.
10. Help and documentation: There is a help tab with frequently asked questions which help's the user to learn more about the application and it's features.

3.0 Conclusions

The advantage of this project is that it has a secure and easy way of encrypting and decrypting files which can be used to help users protect their sensitive data from unauthorized access by using a strong encryption algorithm, whilst having a simple and user-friendly interface which guides the user through the whole application with ease. Designed with object-oriented programming principles in mind any additions to the application can be done so easily as the application has all the code commented and easy to read.

Some of the limitations of the applications is when encrypting larger files, it could take a little bit longer than expected meaning it may not be suited for them. Only having the AES encryption algorithm may not meet the requirements of some of the users and it is only developed for the desktop and not deployed online or on mobile devices.

4.0 Further Development or Research

With additional time and resources there are several changes and additions that I would make to this project to further improve it. I would like to enhance the security of the encryption algorithm by implementing different algorithms which the user can encrypt with. It would provide the user with options while making it more difficult for attackers to decrypt them. Adding cloud support is another functionality that I would further develop this project into which would allow users to access their encrypted files from anywhere. This would be done using either Amazon S3 or google drive. And also deploying the application online so that it can be used by anyone without the limitations of having it on your computer,

creating the application online rather than desktop. With all these features combined it could be a very valuable tool for user's who need to encrypt their files securely on the go.

5.0 References

Videos and websites I used for research of the project.

Encryption program in Python - https://www.youtube.com/watch?v=vsLBErLWBhA

Python Beginner Project: Build a Caesar Cipher Encryption App - https://www.youtube.com/watch?v=x71kJyNvB5o

How to encrypt and decrypt data using Cryptography Library Python - https://www.youtube.com/watch?v=QbdxLvecflM

Cryptography with Python - Quick Guide - https://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_quick_guide.htm

Python GUI Development With PySimpleGUI - https://www.youtube.com/watch?v=-_z2RPAH0Qk

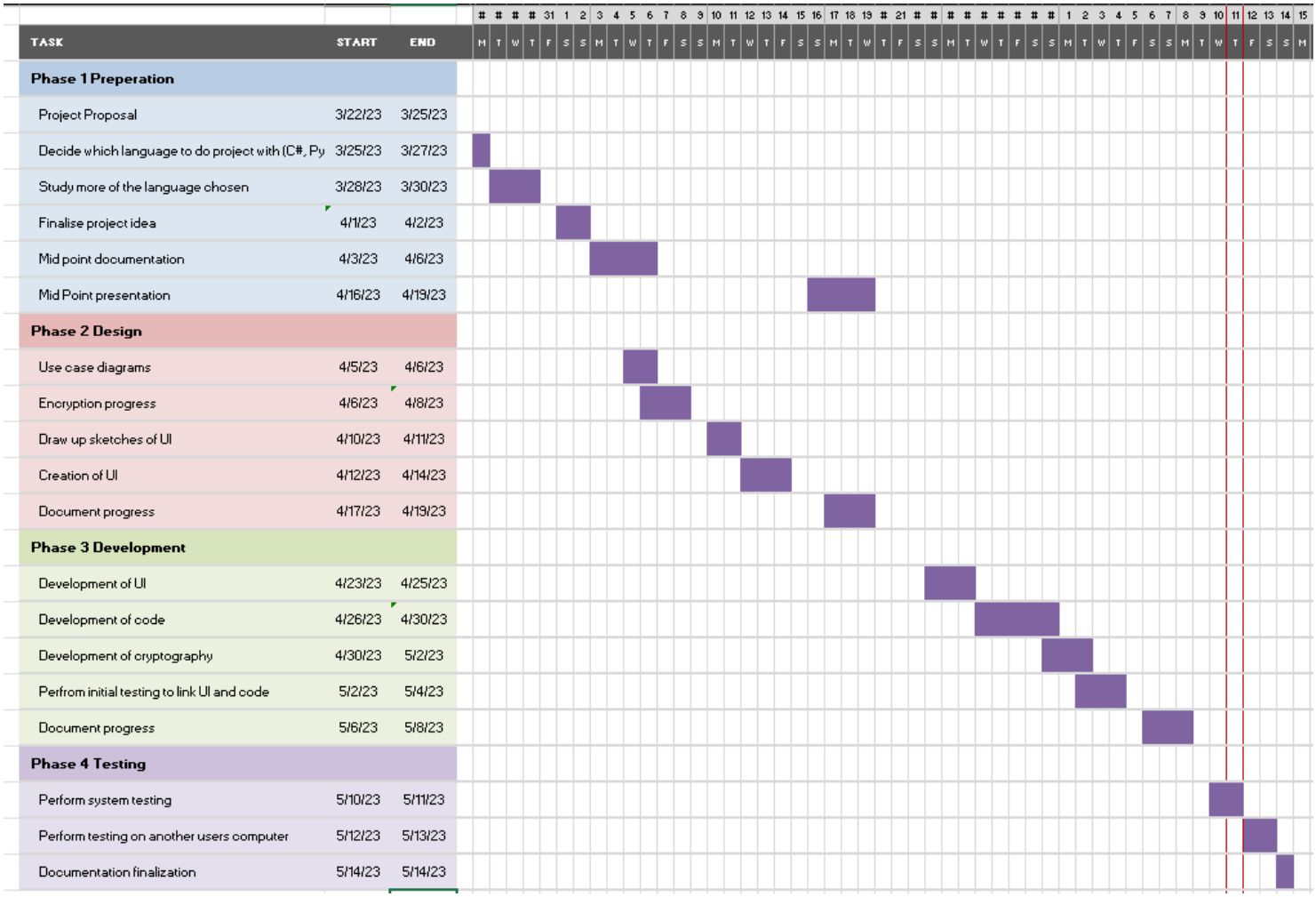
Python GUI: Build Your First Application Using Tkinter - https://www.simplilearn.com/tutorials/python-tutorial/python-graphical-user-interface-gui

Learn Python - Free Interactive Python Tutorial - https://www.learnpython.org/

10 Usability Heuristics for User Interface Design - https://www.nngroup.com/articles/ten-usability-heuristics/

6.0 Appendices

I developed a gantt chat in order to keep me on track with the project and so I know what I need to be working on next. I referenced back to this every so often and looked at the progress which needed to be made.



TASK	START	END
Phase 1 Preperation		
Project Proposal	3/22/23	3/25/23
Decide which language to do project with (C#, Py	3/25/23	3/27/23
Study more of the language chosen	3/28/23	3/30/23
Finalise project idea	4/1/23	4/2/23
Mid point documentation	4/3/23	4/6/23
Mid Point presentation	4/16/23	4/19/23
Phase 2 Design		
Use case diagrams	4/5/23	4/6/23
Encryption progress	4/6/23	4/8/23
Draw up sketches of UI	4/10/23	4/11/23
Creation of UI	4/12/23	4/14/23
Document progress	4/17/23	4/19/23
Phase 3 Development		
Development of UI	4/23/23	4/25/23
Development of code	4/26/23	4/30/23
Development of cryptography	4/30/23	5/2/23
Perfrom initial testing to link UI and code	5/2/23	5/4/23
Document progress	5/6/23	5/8/23
Phase 4 Testing		
Perform system testing	5/10/23	5/11/23
Perform testing on another users computer	5/12/23	5/13/23
Documentation finalization	5/14/23	5/14/23

6.1. Project Proposal

The original plan for my project proposal was a keylogging application which I was going to develop using python. Although after talking to my supervisor he said there is ethical issues in regard to that in that I should think of a different application to develop. This discussion was after I had already submitted my project proposal, so it is completely different to the final application that I developed.

National College of Ireland

Project Proposal

Computing Project

2/11/22

Keylogging program

Cyber Security

2022/2023

Robertas Kuchtovas

X20124287

X20124287@student.ncirl.ie

1.0 Objectives

This project is going to be developing a keylogging program this is to help me get a better understanding of what a keylogging program does and how it is developed. A keylogging program is a type of malicious program that is installed in people's computers and when ran it records every key stroke that is typed. Although it is used in malicious ways most of the time it can also be used to keep an eye on a user's laptop and make sure there is no suspicious activity happening. With most people working from home and not coming into the office some businesses choose to install keyloggers to keep an eye on technical problems on their systems and know how to fix them if a user runs into an issue.

My plan is to build this program using python and or C . These are the best languages to use to build a keylogger, although I haven't learned much of Python or C I will have to do some research and get a basic understanding of the languages.

2.0 Background

I have always had an interest in keyloggers since I was a kid and used to browse on different forums wanting to know how they work and try to use them. Although I never did this would be the perfect opportunity to get a good understanding of them and also how they make them hidden away so that the basic antivirus doesn't catch onto them. As stated previously I will need to learn how to code in Python and C and will start to research them languages in my spare time. While researching I found that they are the best languages to use although any language can be used that I know, I didn't want to develop in java as that's what all my projects have been so far apart from last years computing project. I will try to make a UI that will go along with it also instead of just having it as big piles of code that just execute and show nothing in return.

This keylogger could also be a useful tool to install on my laptop in case it does every get lost and I can see if it has been accessed or if someone tried to log into it. This project will not be used in any malicious ways and will only be used on my own personal laptop or a laptop of a person who volunteers to help me.

3.0 State of the Art

There are many keyloggers that are already out there as it is such a useful tool for business and blackhat hackers. My project will have a neat UI and will be undetectable by common antivirus programs so that when installed it doesn't get picked up straight away and deleted. Although it will be difficult to make it a one off keylogger in the case that it hasn't been seen before and something that very new.

4.0 Technical Approach

When programming this project I will be using Python and or C as when doing research it seems that most of them are developed in these languages. I will have to get myself familiar with C as I have never used it before and I have seen that Python is very similar to java but just a more basic approach, I have touched base with Python last year but didn't dive too deep into it so I will need to refresh myself on how it works.

Starting off I will need to get a base done for the keylogger so work out what I want the UI to look like and draw up sketches of this before I start implementing anything. After I have done that I will start to work on the actual UI and start implementing code when this is done.

Breaking down all the project tasks will be done using a Gantt as I have used this before last year and I am familiar with it. It's an easy way to track progress and see if you are falling behind on any tasks. It will give me a time frame of how much time I have to develop each part of the project and make sure I stay on top of it.

5.0 Technical Details

Keyloggers can be coded in any language that you feel most comfortable with that's what makes them so diverse but since I have been coding in java for most of my projects I have decided to switch from java and use python which will be more challenging for myself, I will also look into the idea of how it's made in C and C# as they are two other popular languages that are used to make keyloggers.

Keylogger applications are designed by implementing the Exact String Matching algorithm which can record all user activities related to keyboard strokes. These results are then stored automatically in a database which only the owner of the keylogging software can access. Antivirus companies work hard to make sure that keylogging software is detected and people do not get infected so my goal is to get this keylogger to be undetectable by antivirus software also. String matching algorithms will also be used in order to record user activity making it easier to interpret. In order to monitor the results of the keyboard stokers I will set up an email which will send the recordings to me periodically.

6.0 Project Plan

The project plan has been set out in a Gantt chart this is in order to help me keep track of the progress I have made and the deadlines I have set out for myself. I can adjust it depending if I am running behind on time or moving ahead too quickly.

The Gantt chart has been included in this document.

Phase 1 Preperation		
Project Proposal	11/3/22	11/6/22
Decide which language to do project with (C#, Python)	11/6/22	11/11/22
Study more of the language chosen	11/12/22	11/19/22
Ethics form	11/21/22	11/24/22

Finalise project idea	11/25/22	11/30/22
Phase 2 Design		
Flow charts	11/28/22	12/3/22
Draw up sketches of UI	12/3/22	12/7/22
Database creation	12/9/22	12/16/22
Creation of UI	12/17/22	12/24/22
Document progress	12/27/22	12/30/22
Phase 3 Development		
Development of UI	1/3/23	1/9/23
Development of code	1/10/23	1/20/23
Development of database	1/21/23	1/28/23
Perfrom initial testing to link UI, code and dabatase	1/28/23	2/5/23
Implemntation of bypassing antivirus	2/5/23	2/12/23
Document progress	2/14/23	2/19/23
Phase 4 Testing		
Perform system testing	2/21/23	2/25/23
Perform testing on another users computer	2/26/23	3/1/23
Scan program with antivirus	3/4/23	3/7/23
Documentation finalization	3/8/23	3/15/23

7.0 Testing

I will start the testing by using it on my own system since it is a keylogger it will record every keyboard stroke that will be inputted and then recorded and sent back to me, so testing that it fully works first is important before I get anyone else to test it. When I have

tested it making sure it works I will ask some of my friends if they are willing to run the application on their system with, I will explain how the application works and only with their consent I will send the program to them to test if I do not receive any consent from them then I will not use them as a test subject as this is program can be used in malicious ways it is important that I receive consent from them before using them as a test subject.

Making sure all the strokes are recorded are sent back to me are the main testing points that will need to be addressed. Also making sure that the antivirus system does not pick it up is also very important.

7.1. Reflective Journals

Student Name	Robertas Kuchtovas
Student Number	X20124287
Course	Cyber Security
Supervisor	Vikas Sahni

Month: October

What?

- Researched Idea for project
- Talked to people in my class what their ideas were.
- Decided to make the project in Python still don't have a concrete idea what to do.
- Talked with my supervisor about ideas.
- Submitted project proposal.

So What?

- Don't have my heart set on the project that I proposed to do may be ethical issues.
- Have a few ideas what I can do just need to pick something that relates to my course.
- Have a few projects that I need to do for other subjects.

Now What?

- Do more research into python projects.

• Waiting feedback on project proposal.	
Student Signature	Robertas Kuchtovas

Month: November

What? • Had a lot of project work and assignments to do • Had a meeting with my supervisor again to discuss the idea I submitted he said there is ethical issues • Not much progress to report • Once assignments are finished I will have more time to focus on project • Still doing some research on python
So What? • Talking to a few people in class to see how their progress is getting on everyone seems to be on the same boat.
Now What? • Do more research on what application I could develop and need to do more research in to how Python works

Student Signature	Robertas Kuchtovas

Month: December/January

What? • Was very busy with assignments and projects. • Didn't have much time to focus on final year project. • Very stressed about cloud application project.

• Put together a plan for the application I will be developing and submitted the mid-point presentation.
So What? • Really like the idea of creating an encryption application just need to research it more and if Python supports it.
Now What? • Research how encryption application works. • Make sure Python supports it. • Learn more Python.

Student Signature	Robertas Kuchtovas

Month: February

What? • Have a lot more time to focus on the project having less classes. • Done some research on how encryption works with python. • Got results on mid-point presentation not very happy with.
So What? • Feel like this project could really work.
Now What? • Make a plan for meeting deadlines and making sure I have a decent start to the project soon. • Still have to focus on other modules and hand in assignments.

Student Signature	Robertas Kuchtovas

Month: March

What? • Worked on the assignments for other modules. • Looked up tutorials on how other people have built encryptors. • Installed the necessary modules in order to get a start on the project
So What? • Tested a random application I downloaded to make sure I have everything installed correctly
Now What? • Start creating mock up of the application on how I want it to look like. • Get a rough idea of how I want to lay out the code.

Student Signature	Robertas Kuchtovas

Month: April

What? • Continued to work on the project • Got a good idea of how to code in Python • Still have some CA's and TABA's coming up which I want to do well in • Had meeting with my supervisor
Now What? • Continue to work on my project • Start working on the documentation alongside the code • Start to implement a GUI

Student Signature	Robertas Kuchtovas

1. Title of Invention

2. Inventors

Name	School/Research Institute	Affiliation with Institute (i.e. department, student, staff, visitor)	Address, contact phone no., e-mail	% Contribution to the Invention

3. Contribution to the Invention

Each contributor/potential inventor should write a paragraph relating to his/her contribution and include a signature and date at the end of the paragraph.

4. Description of Invention

(Please highlight the novelty/patentable aspect. Attach extra sheets if necessary including diagrams where appropriate). What is novel, the 'inventive step'? For more information on patents, please look at http://www.patentsoffice.ie/en/patents.aspx

5. Why is this invention more advantageous than present technology?

What is its novel or unusual features? What problems does it solve? What are the problems associated with these technologies, products or processes? Explain how this invention overcomes these problems (*i.e*. what are its advantages).

6. What is the current stage of development / testing of the invention?

7. List the names of companies which you think would be interested in using, developing or marketing this invention

8. Funding Partner(s)

Government Agency & Department	
% Support	
Contract/Grant No.	
Contact Name	
Phone No.	
Address	

Industry or other Sponsor	
% Support	
Contract/Grant No.	
Contact Name	
Phone No.	
Address	

9. Where was the research carried out?

10. What is the potential commercial application of this invention?

11. Was there transfer of any materials/information to or from other institutions regarding this invention?

If so please give details and provide signed agreements where relevant.

12. Have any third parties any rights to this invention?

If yes, give names and addresses and a brief explanation of involvement.

13. Are there any existing or planned disclosures regarding this invention?

Please give details.

14. Has any patent application been made? Yes/No

If yes, give date: _____ Application No.: _____

Name of patent agent: _____

Please supply copy of specification.

15. Is a model or prototype available? Has the invention been demonstrated practically?

I/we acknowledge that I/we have read, understood and agree with this form and the Institute's *Intellectual Property and Procedures* and that all the information provided in this disclosure is complete and correct.

I/we shall take all reasonable precautions to protect the integrity and confidentiality of the IP in question.

Inventor: _____ _____

Signature Date

_____

Signature

7.2. Other materials used

Any other reference material used in the project for example evaluation surveys etc.