



EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

DEPT. OF PROGRAMMING LANGUAGES AND COMPILERS

Secure Digital Vault

Supervisor:

Gregory Morse

PhD Student

Author:

Yousef Al-Akhali

Computer Science BSc

Budapest, 2024

Contents

1	Introduction	5
I	User Documentation	6
2	Guide	7
2.1	Installation	7
2.1.1	Requirements	7
2.1.2	Run	7
2.2	Interface	8
2.2.1	Vault Creation	8
2.2.2	Vault Search	9
2.2.3	Vault Recovery	10
2.2.4	Any Button Details	12
2.2.5	Plain Vault View	12
2.2.6	Add File View	14
2.2.7	Regular Vault View	15
2.2.8	Item View	16
2.2.9	Encrypt a File	17
2.2.10	Decrypt a File	17
2.2.11	Add a Note for a File	18
2.2.12	Get an attached Note of a File	18
2.2.13	Extracting Items	19
2.2.14	Finding Files	20
2.2.15	Vault Settings	20
2.2.16	Change Vault Details	21
2.2.17	Change Vault Password	22
2.2.18	Log Usage	22

2.2.19	Log Extraction	23
2.2.20	True Encryption	24
2.2.21	Possible Corruption	24
2.2.22	Portability	25
2.2.23	Trust	26
II	Developer Documentation	27
3	Design	28
3.1	Architecture	28
3.1.1	Header Decomposition	30
3.1.2	File Decomposition	32
3.1.3	Directory Decomposition	33
3.1.4	Note Decomposition	34
3.1.5	Footer Decomposition	34
3.1.6	Logging	35
3.1.7	Vault View Design	35
3.1.8	File Explorer	37
3.1.9	View File Information	37
3.1.10	File Search	39
3.1.11	Vault Settings	40
3.1.12	Interaction Diagram	40
3.2	Limitations	41
4	Implementation	43
4.1	Software Requirements	43
4.1.1	Chosen Software limitations	43
4.2	Class Diagrams	45
4.2.1	Main UML Diagram of the Qt Windows	46
4.2.2	Package Management	46
4.3	Backend implementation	47
4.3.1	Creating a Vault	48
4.3.2	Utilities	49
4.3.3	Main Override Function	49

4.3.4	Encryption	51
4.3.5	Token Generation	51
4.3.6	Magic Values	52
4.3.7	Decryption	53
4.3.8	Logging	54
4.3.9	Signaling	55
4.3.10	Threading and Mutable Values	56
4.4	Frontend implementation	57
4.4.1	Frontend challenges	57
4.4.2	Managers	58
4.4.3	VaultView	58
4.4.4	Dialogs and Windows	59
4.5	Technical limitations	61
4.5.1	Interaction Based	61
4.5.2	Security	62
4.6	Improvements	62
4.6.1	GUI	63
4.6.2	Storage Optimization	63
5	Testing	65
5.1	Unit Tests	65
5.2	Manual Tests	67
5.2.1	Vault Creation Window Test	67
5.2.2	Welcome Window Test	68
5.2.3	Vault Search Window Test	68
5.2.4	Vault View Window Test	70
5.2.5	Add Items Test	71
5.2.6	Extract Items Test	71
5.2.7	View Items Test	72
5.2.8	Find Items Test	73
5.2.9	Note Test	74
5.2.10	Settings Test	75
5.3	Additional Testing	76
5.3.1	Security Testing	76

5.3.2 Performance Testing	76
6 Conclusion	77
Bibliography	78
List of Figures	80
List of Tables	82
List of Codes	83

Chapter 1

Introduction

The Digital Vault aims to provide a solution for preserving critical data in a secure environment; thus making it similar to an actual vault but on a computer where the data is encrypted. As this is a digital scenario, it maintains friendly user access, provides the possibility to interact with the inner contents of the vault in a file explorer like interface, and records any important vault events such as new addition, individual encryption, and settings manipulation. It is important to differentiate the project between TrueCrypt ¹ and VeraCrypt ², as these two make it possible to create a virtual encrypted disk within a file, or encrypt a partition or the whole storage device, whereas the Vault creates an encrypted File-Table which with its use is able to locate every Directory or File existing in the concatenation of bytes, the Table is unreadable by any system or program unless its decrypted.

The software is designed for all users. The efficiency focus is in hard-disk [1] usage being its main advantage over other available software. It is not resource-heavy in terms of RAM [2] consumption and does not create additional temporary data blocks on the storage unit while running.

¹TrueCrypt is a discontinued source-available freeware utility used for on-the-fly encryption

²Successor of TrueCrypt, with more reliability and functionality

Part I

User Documentation

Chapter 2

Guide

2.1 Installation

2.1.1 Requirements

The requirements to run the Vault are narrowed down to:

- Windows 10 or 11 Machine
- 8GB of RAM or more
- Python version greater than 3.9, while Python 3.12¹ is recommended, pip² Python package installer, Graphviz³, and git⁴.

The application must be built to an executable. A powershell script shall be used:

```
git clone https://github.com/Yousef-1022/Secure-Digital-Vault.git
./build.ps1
```

2.1.2 Run

Double click on "*Vault.exe*", or use the following commands if running it without an executable is needed, however, the icons will not be visible:

```
pip install -r requirements.txt
py main.py
```

¹<https://www.python.org/downloads/release/python-3120>

²<https://pip.pypa.io/en/stable>

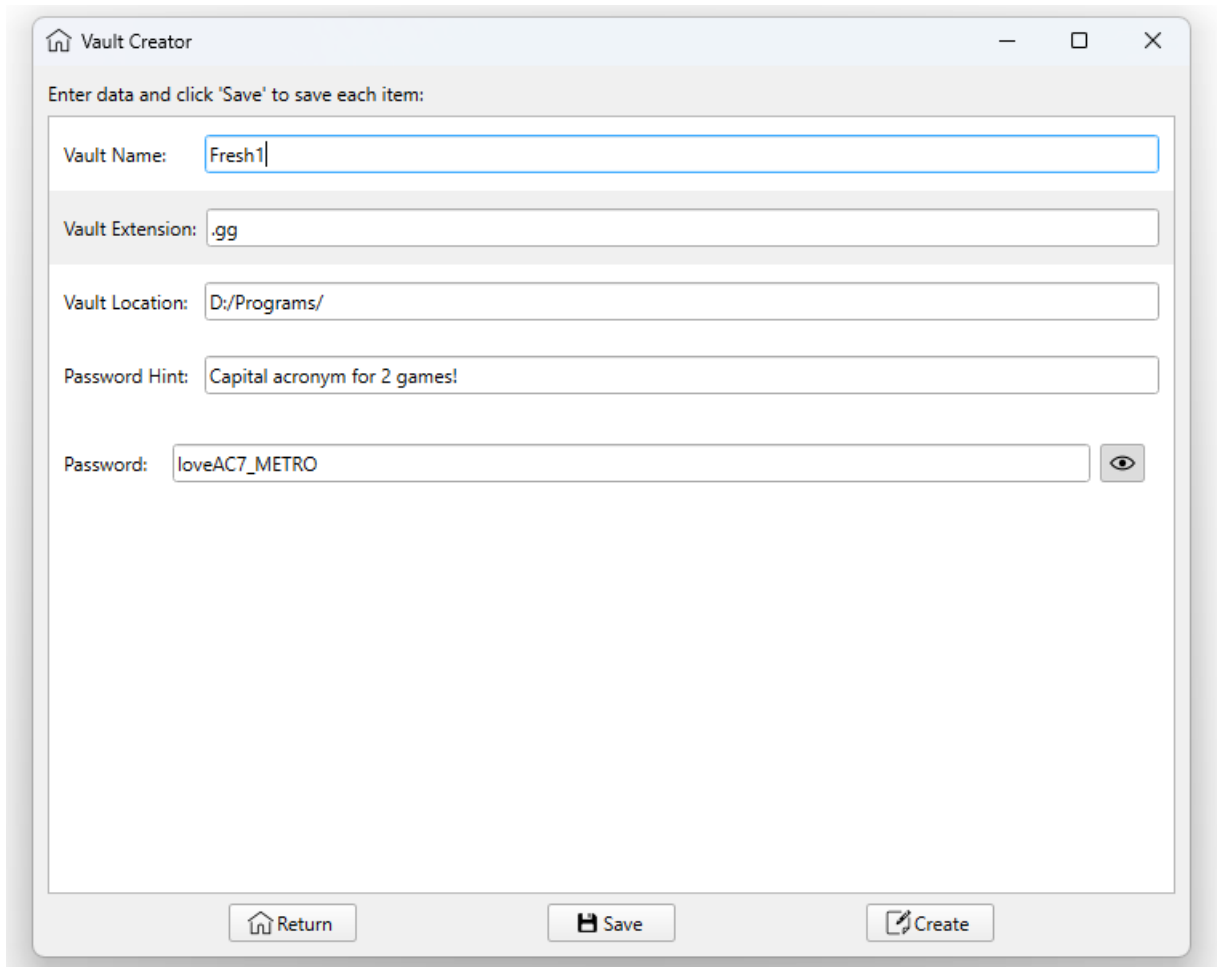
³<https://graphviz.org/download>

⁴<https://git-scm.com/downloads>

2.2 Interface

2.2.1 Vault Creation

For the creation of the Vault, at least a minimum of **5KB** shall be available in the chosen drive for successful creation.



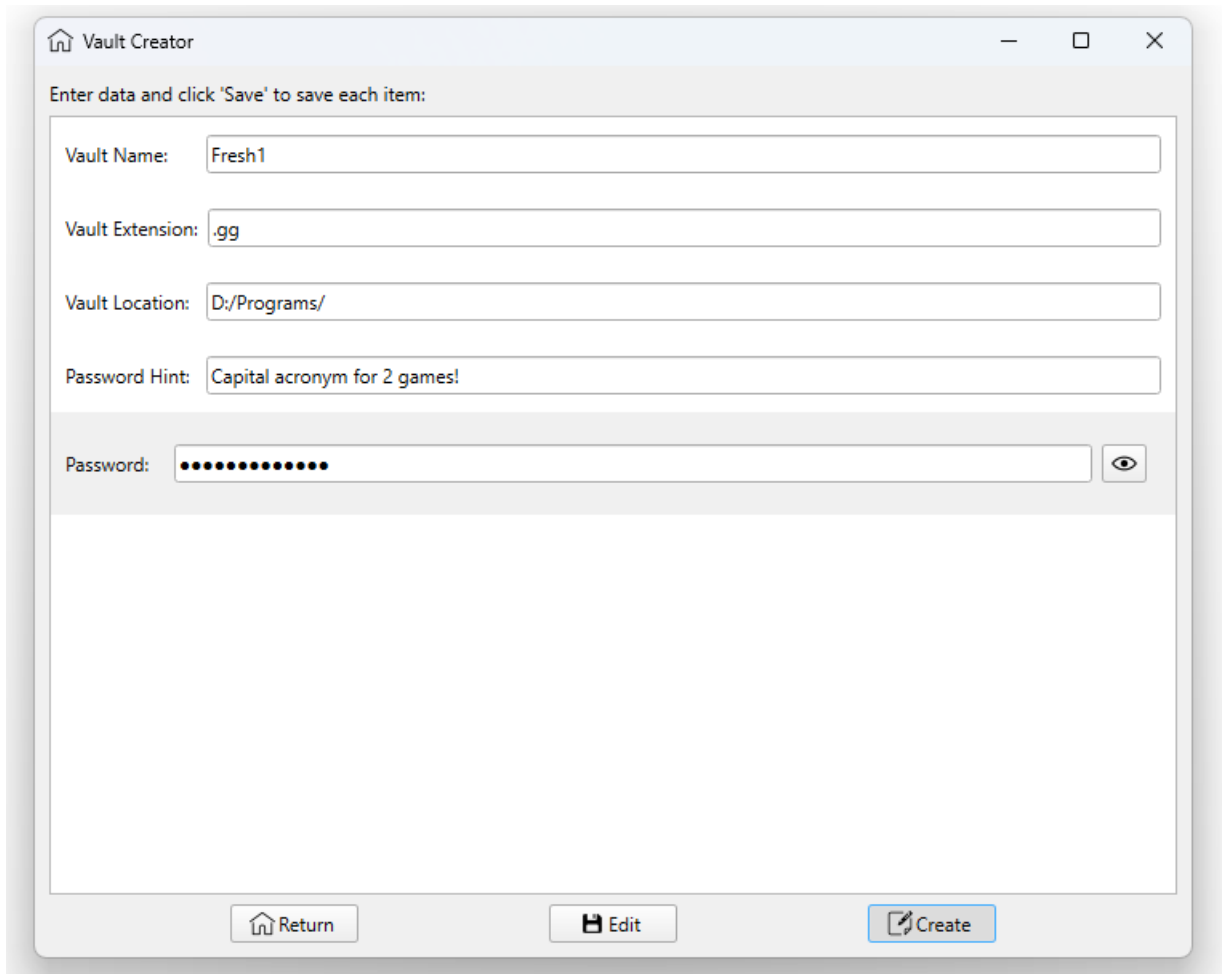
The image shows a window titled "Vault Creator" with standard Windows window controls (minimize, maximize, close). Below the title bar, there is a instruction: "Enter data and click 'Save' to save each item:". The main area contains five input fields: "Vault Name:" with the text "Fresh1", "Vault Extension:" with ".gg", "Vault Location:" with "D:/Programs/", "Password Hint:" with "Capital acronym for 2 games!", and "Password:" with "loveAC7_METRO". To the right of the password field is an eye icon. At the bottom of the window, there are three buttons: "Return" (with a house icon), "Save" (with a floppy disk icon), and "Create" (with a checkmark icon).

Figure 2.1: Vault Creation Window

Key items to consider:

- The extension can be of any type, the more ambiguous, the better.
- The hint should be less than 32 characters.
- The password should be strong, a detailed instruction list will be shown incase its not.
- The details must be saved first, and then click create in order for the Vault to show up.

In order to be able to create the Vault, the save button must be first clicked in order to confirm the data before execution.



The image shows a window titled "Vault Creator" with standard Windows window controls (minimize, maximize, close). Below the title bar, there is a instruction: "Enter data and click 'Save' to save each item:". The main area contains five input fields: "Vault Name:" with the text "Fresh1", "Vault Extension:" with ".gg", "Vault Location:" with "D:/Programs/", "Password Hint:" with "Capital acronym for 2 games!", and "Password:" which is masked with dots. To the right of the password field is an eye icon for toggling visibility. Below these fields is a large empty rectangular area. At the bottom of the window, there are three buttons: "Return" (with a house icon), "Edit" (with a document icon), and "Create" (with a checkmark icon and highlighted with a blue border).

Figure 2.2: Ready Vault Creation Window

After creation, the user will see the extra passwords called "**tokens**" in the file: *tokens.txt* in the same directory the Vault exists in. It is advised to store these tokens in a secure place, or write them down on a piece of paper incase the original password is forgotten.

2.2.2 Vault Search

The default interaction Window for finding a Vault. The user may click the "*Detect*" button to search in the current drive for the first occurrence of that specific extension. The Vault location and the extension of the Vault must be correct.

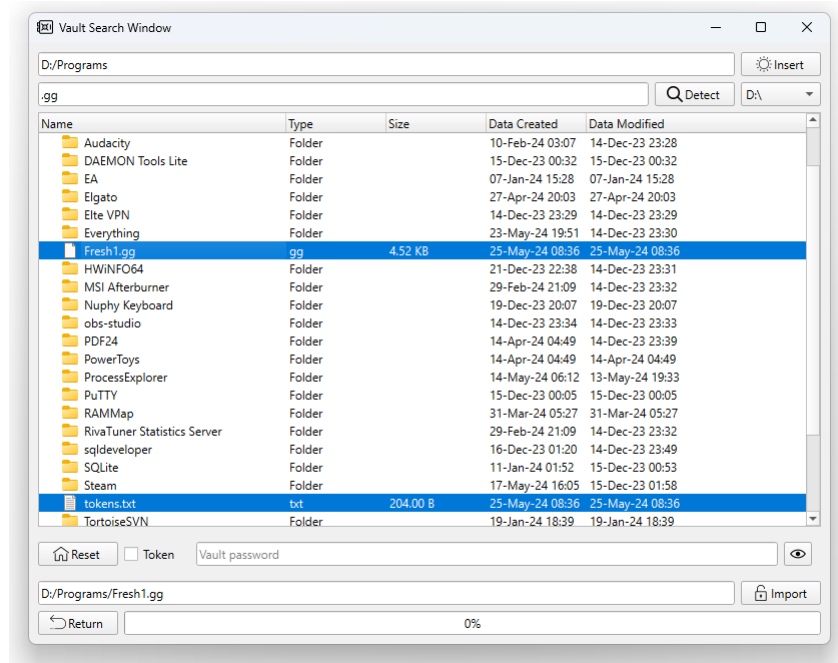


Figure 2.3: Vault Search Hint

If the user is not able to open the Vault, the special Hint set at the beginning will show up in the password section as a placeholder after 3 unsuccessful tries.

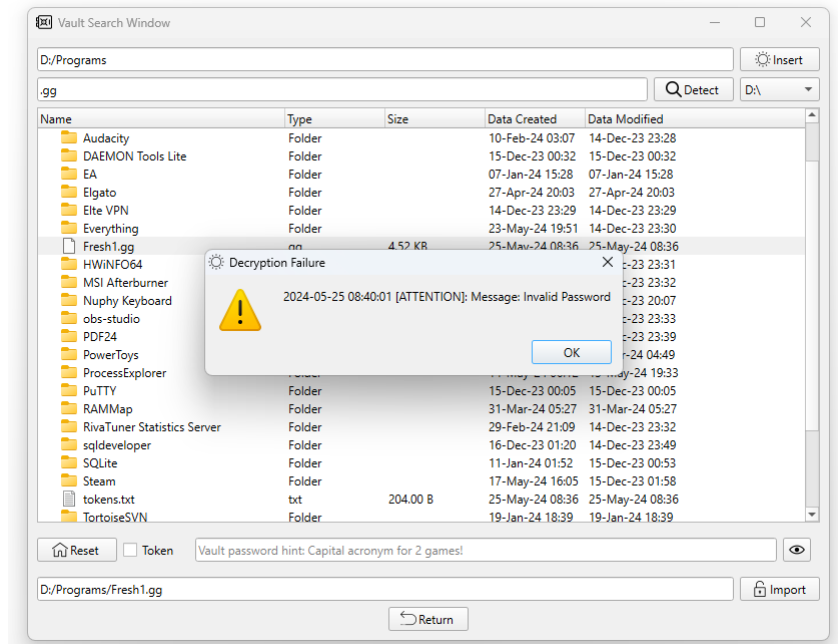


Figure 2.4: Vault Search Hint

2.2.3 Vault Recovery

If the original Vault password is forgotten, the user is able to login by using any of the 3 tokens provided after creation the Vault, these do not expire throughout the

lifetime of the Vault as long as the old password is not changed. If a new password is created, then these tokens become obsolete.

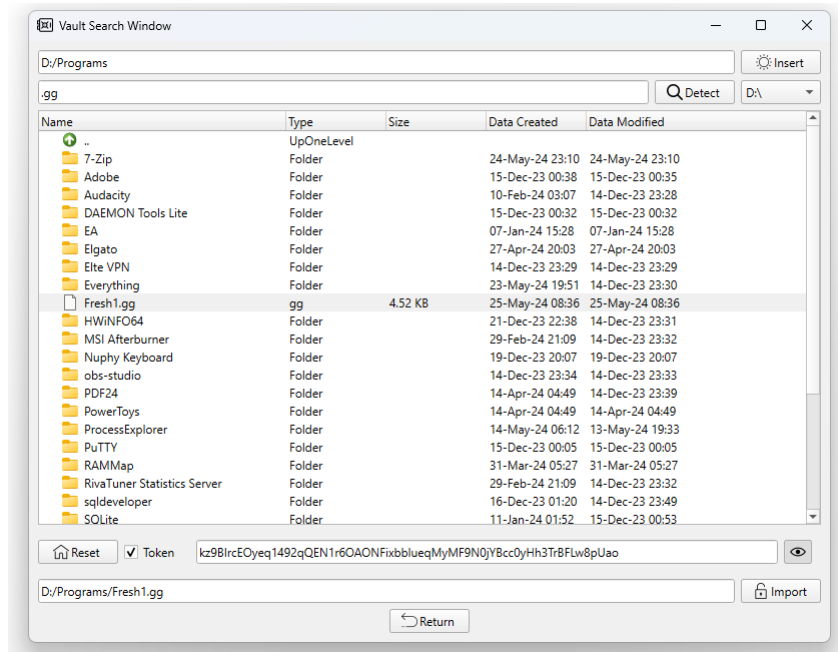


Figure 2.5: Vault Search Window With Token

However, a regular password can be used without ticking the Token checkbox.

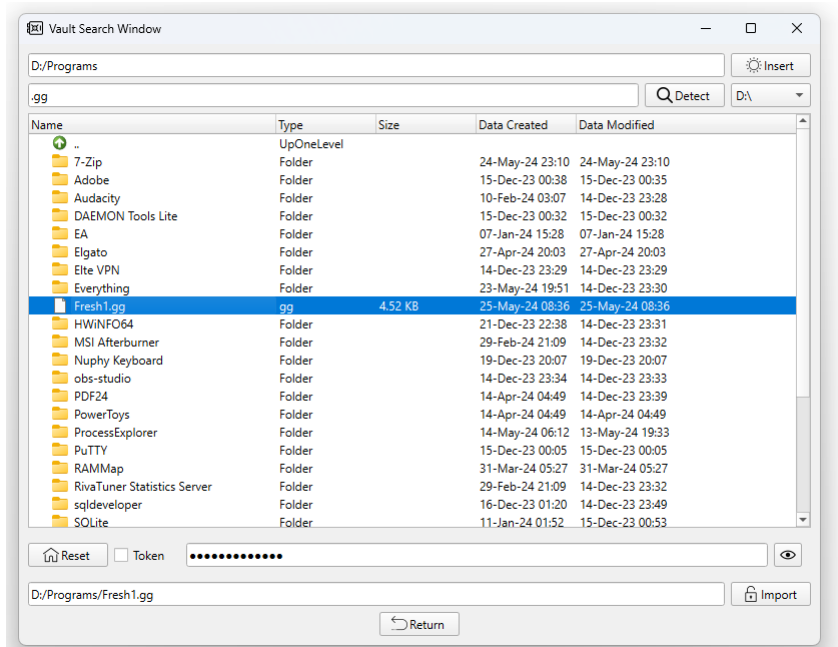


Figure 2.6: Vault Search Window Without Token

In either Figure 2.6 or Figure 2.5, the user has to click on **Import** in order to unlock the Vault and start using it.

2.2.4 Any Button Details

In order to understand what does a button do, the user must *"Right Click"* on it to see more details about what sort of action does it.

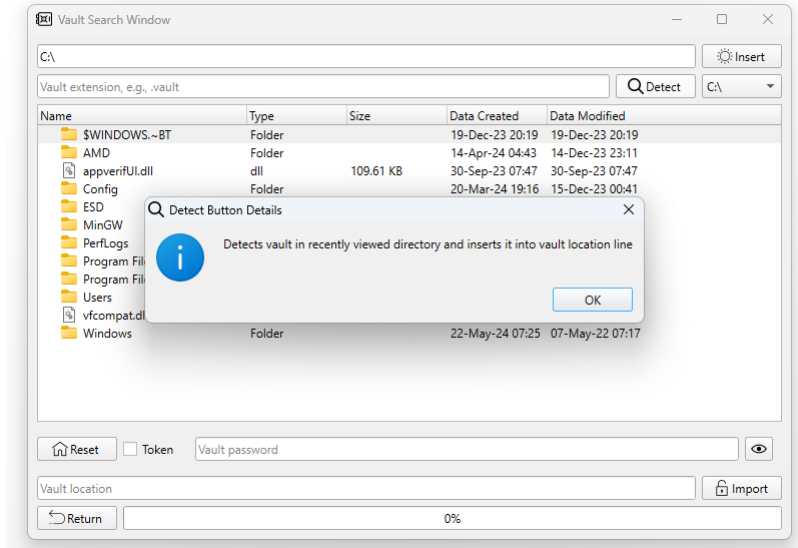


Figure 2.7: Right Click on Any Button

2.2.5 Plain Vault View

The Vault View is the main interface which will be used to interact with the Vault itself, it also possible to press **TAB** if there is a need to highlight of what can be pressed. Its possible to resize the Vault by clicking the resize button in the top right corner or by hovering over one of the edges and dragging to adjust the size.

By default, the Vault uses a Window Light Theme⁵. In future releases, it may be possible to switch to a Dark Theme or a White Theme. However, the Light Theme closely resembles what Windows provides.

⁵Windows allows to personalize the color or accent color for window title bars and borders, the Start button, or the taskbar. The default theme is the Light Theme in most Windows releases.

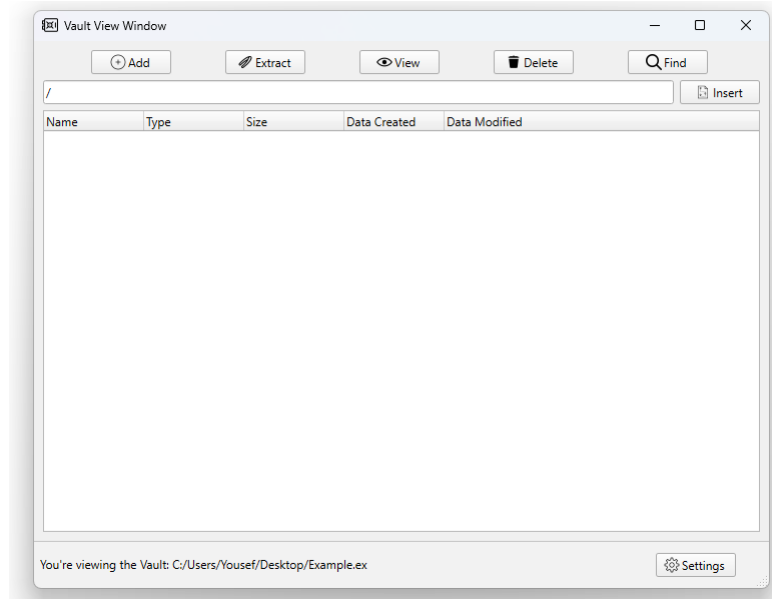


Figure 2.8: Plain Empty Vault

No files or folders can be visible, this is because nothing was added yet. In order to go up one folder, either click on the *UpOneLevel* item twice, it has the name " .. ", as shown in Figure 2.9, or simply press the *Backspace* key instead.

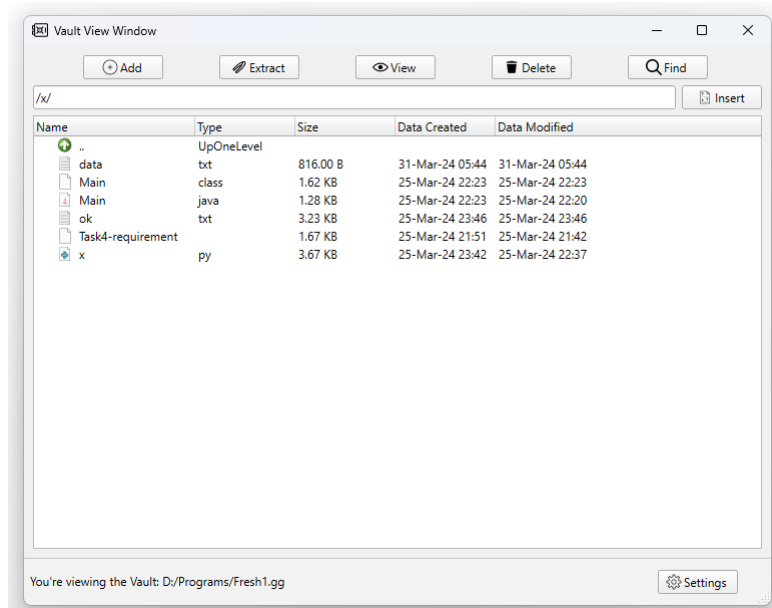


Figure 2.9: Vault With Items

At this time of release, the user is presented with 6 main intractable buttons. Insert button is used for navigation to a specific folder. "/" is the representation of the *Rootfolder* which essentially means the main Parent Folder.

2.2.6 Add File View

In Figure 2.10, the user is able to add any type of files and folders into the Vault. The addition of symbolic⁶ links is disabled, that is why it will not be possible to see them on the interface.

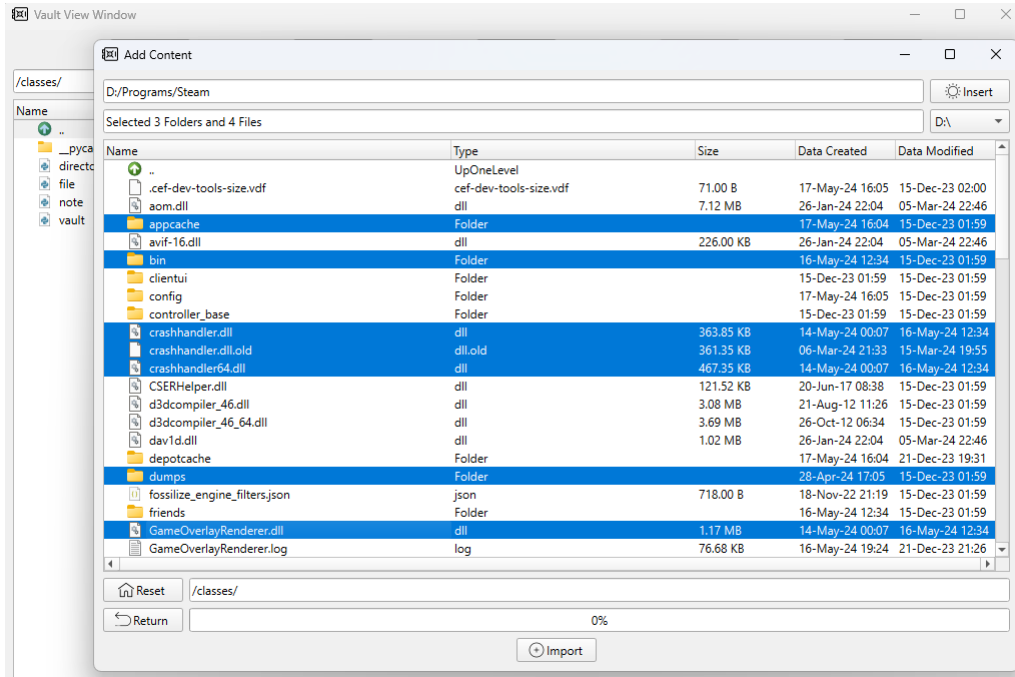


Figure 2.10: Add Content Window

The larger the overall size of the items currently held, the more time it takes to insert them successfully into the Vault. The abort button will appear once the import process begins, as shown in Figure 2.11. This abort operation safely stops the import but does not remove any items added during the import.

The progress bar⁷ will start functioning once the operation begins. It provides an approximate indication of the remaining time to successfully complete the operation.

⁶Symbolic link is a file whose purpose is to point to a file or directory by specifying a path thereto.

⁷A progress bar is a graphical control element used to visualize the progression of an extended computer operation.

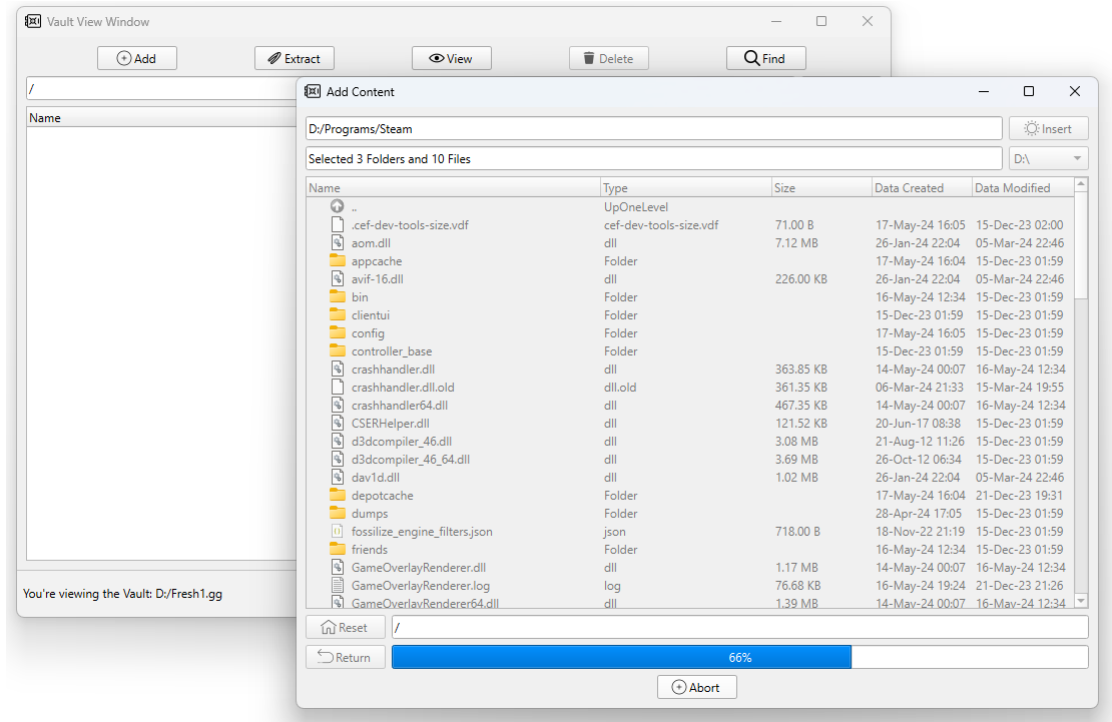


Figure 2.11: Add File Window

2.2.7 Regular Vault View

After the addition of files and folders, the user is able to see the *"UpOneLevel"* interactable item, it simply goes Up one folder. It disappears if the user is in the *RootFolder*.

Be aware, that the way the path is written inside the Vault does not follow the Windows path convention, but rather the Unix path convention. For example:

```
/documents/secret/codes/
```

And Not Windows Default:

```
\documents\secret\codes\
```

There are forbidden characters for both Folders and Files. The list is as follows:

```
["<", ">", ":", "\"", "/", "\\", "|", "?", "*", "\0", ".", ","]
```

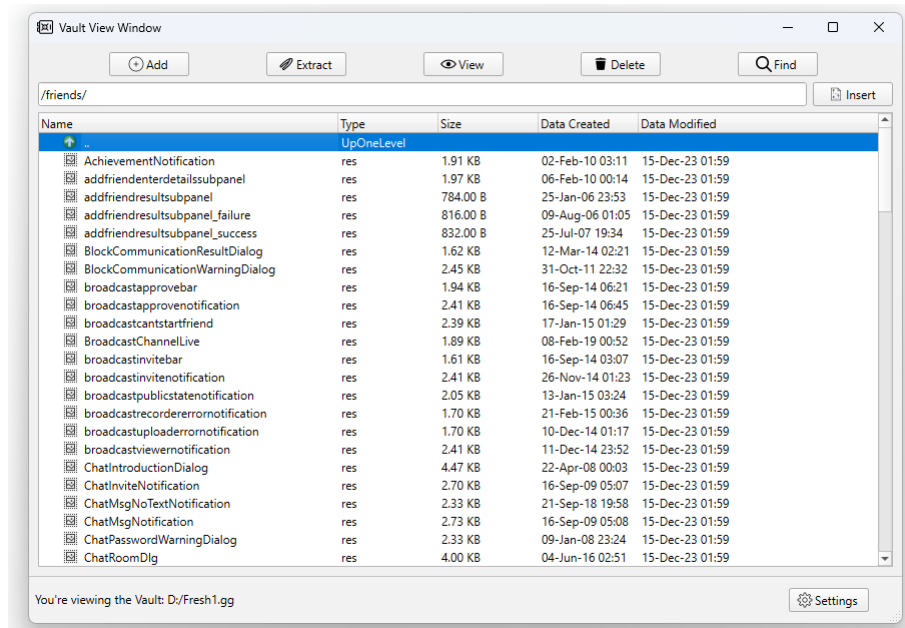



Figure 2.12: Regular Vault View After Addition

2.2.8 Item View

To view the details of a specific item. If a Folder is chosen, then generic information about the Folder will be shown, and no interaction modification is enabled at the initial release, but for files, these buttons are enabled as shown in Figure 2.13.

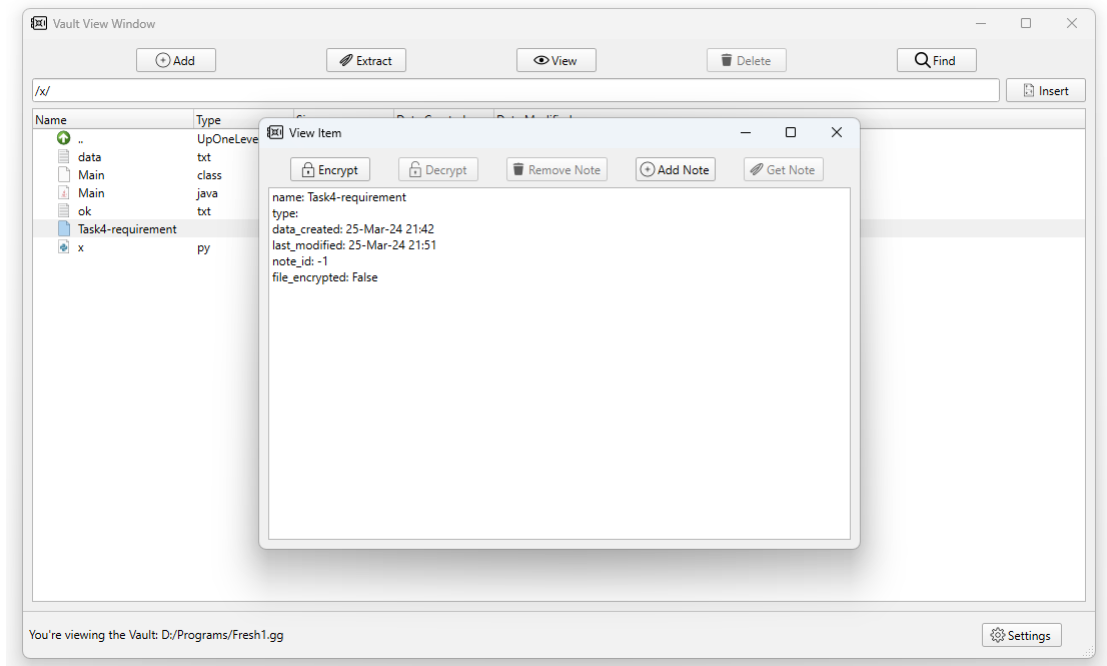


Figure 2.13: Viewing a File

2.2.9 Encrypt a File

To add extra encryption to a file, click **Encrypt**. This will provide an additional layer of security on top of the existing Vault encryption. Its called **Individual Encryption**.

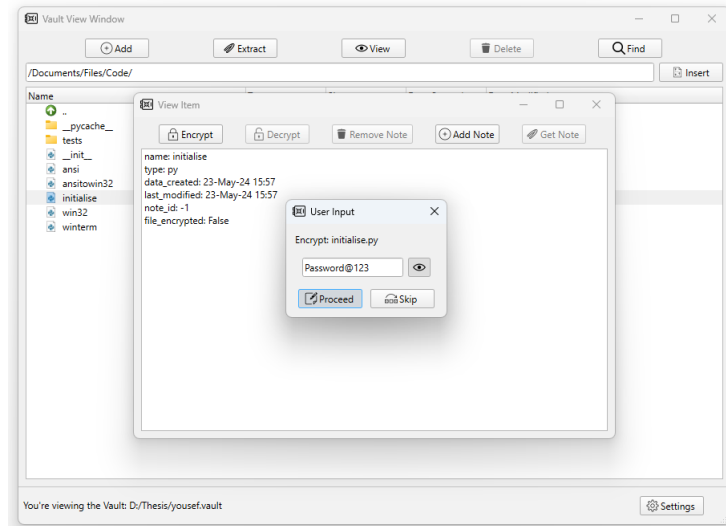


Figure 2.14: Encrypting a File

2.2.10 Decrypt a File

To remove the extra encryption from a file, click **Decrypt**. This will remove the additional layer of security, leaving the file with only the Vault's encryption. Its called **Individual Decryption** and the original password used to individually encrypt it must be used.

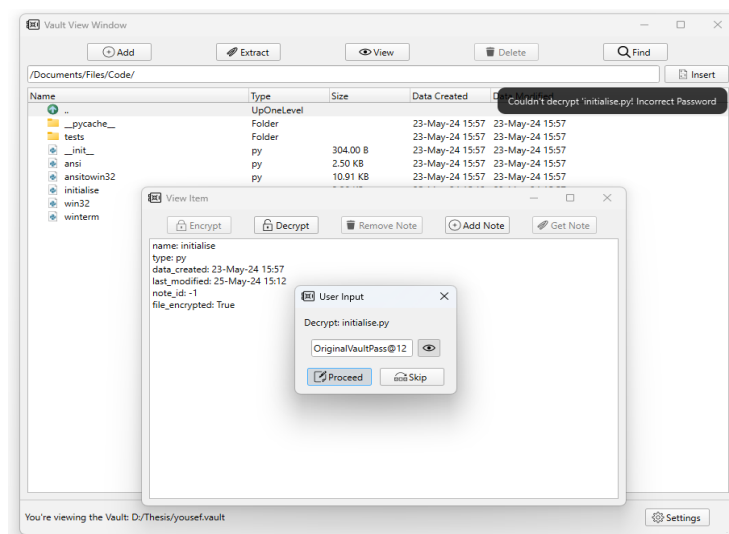


Figure 2.15: Decrypting an Encrypted File

2.2.11 Add a Note for a File

For a File, it is also able to add a Note, but the size limit is 7 MB.

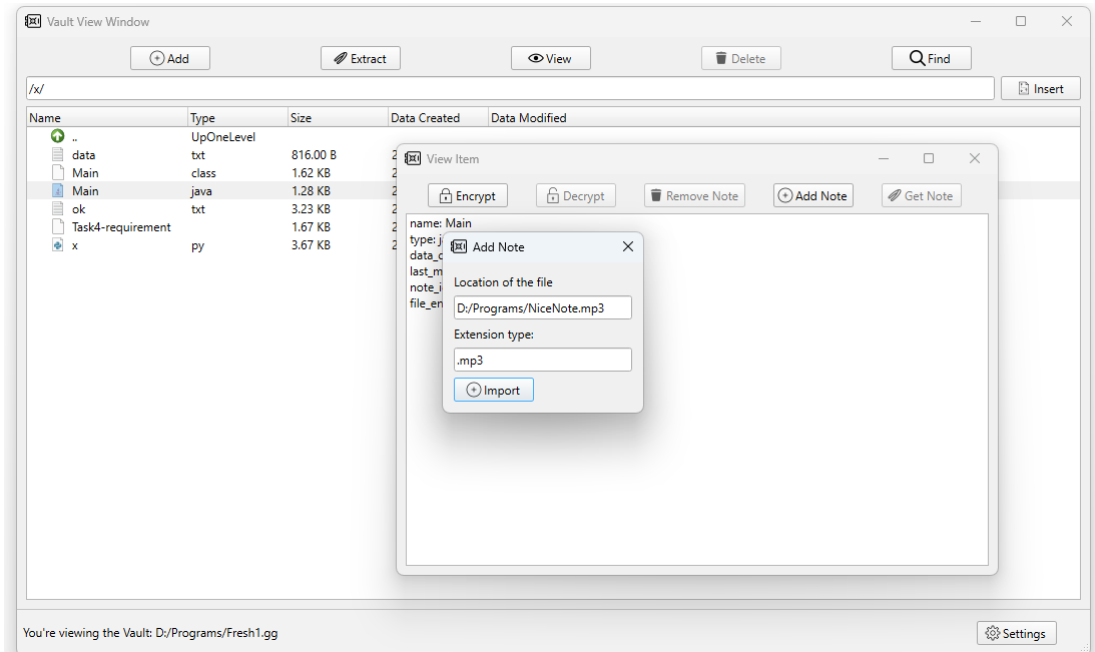


Figure 2.16: Add a Note

2.2.12 Get an attached Note of a File

For a File with an attached Note, it is possible to extract it without deleting it.

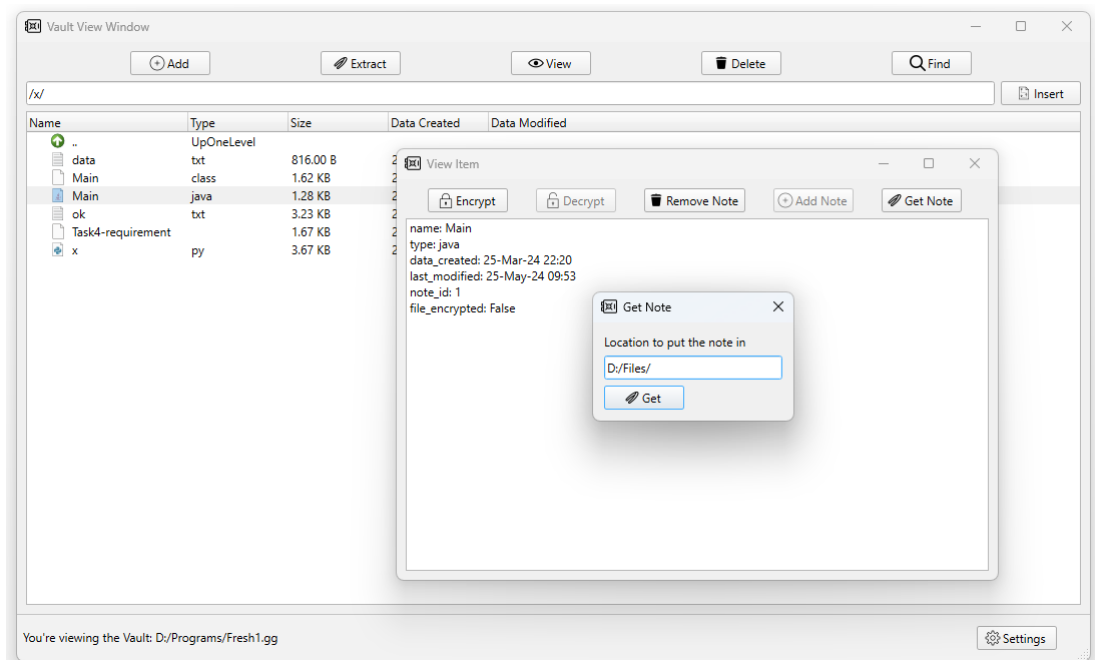


Figure 2.17: Get a Note

2.2.13 Extracting Items

In order to extract items out of the Vault, the extraction section shall be entered. Do note, the extraction does not delete the data from the Vault.

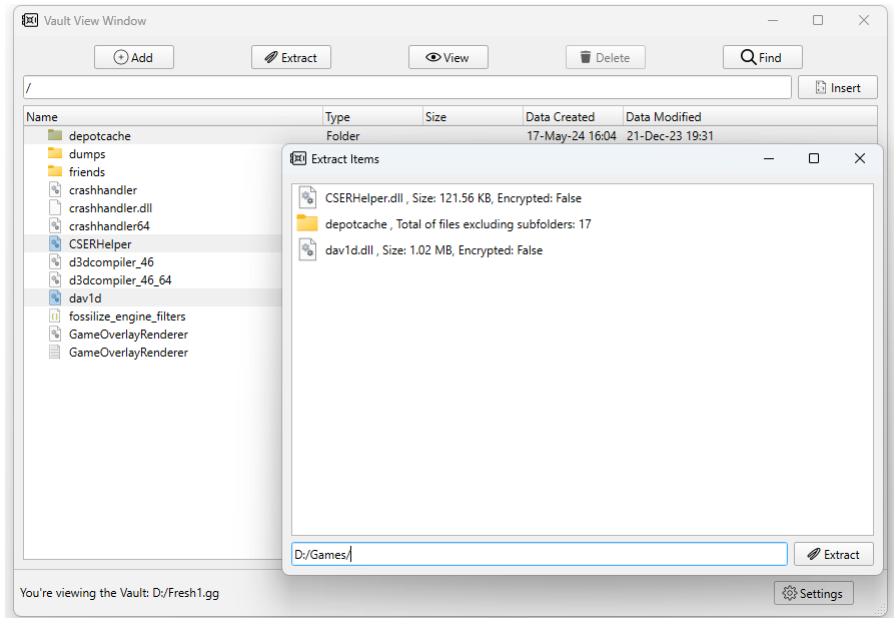


Figure 2.18: Extracting Files or Folder

During extraction, if an encrypted File is included, then the progress shall halt until it reaches this File, next action is either to click on the skip button or to provide the password of the Encrypted File as shown in Figure 2.19 to extract successfully.

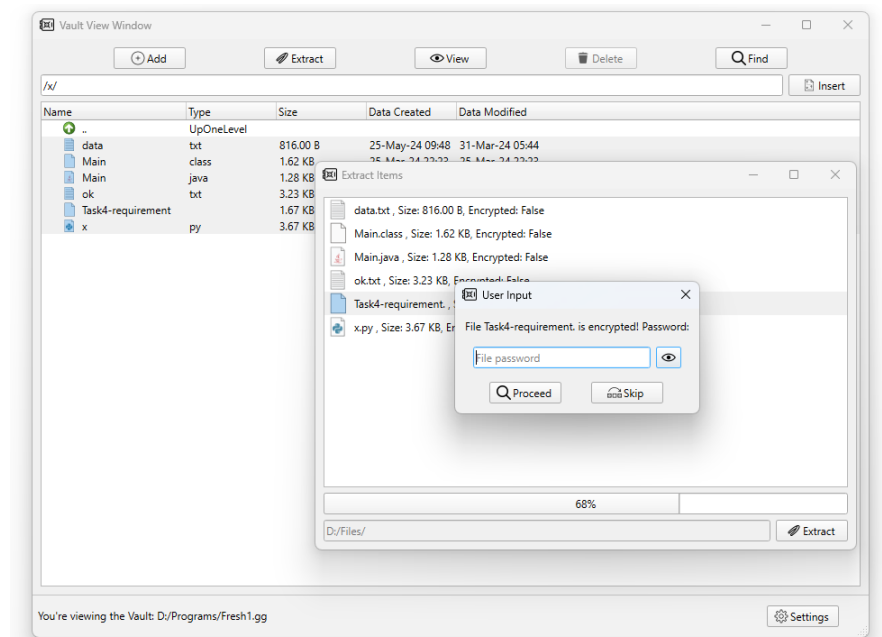


Figure 2.19: Extracting Files or Folder

2.2.14 Finding Files

To search for Files with specific details, this section must be entered. The user will be presented with the following data presented on Figure 2.20.

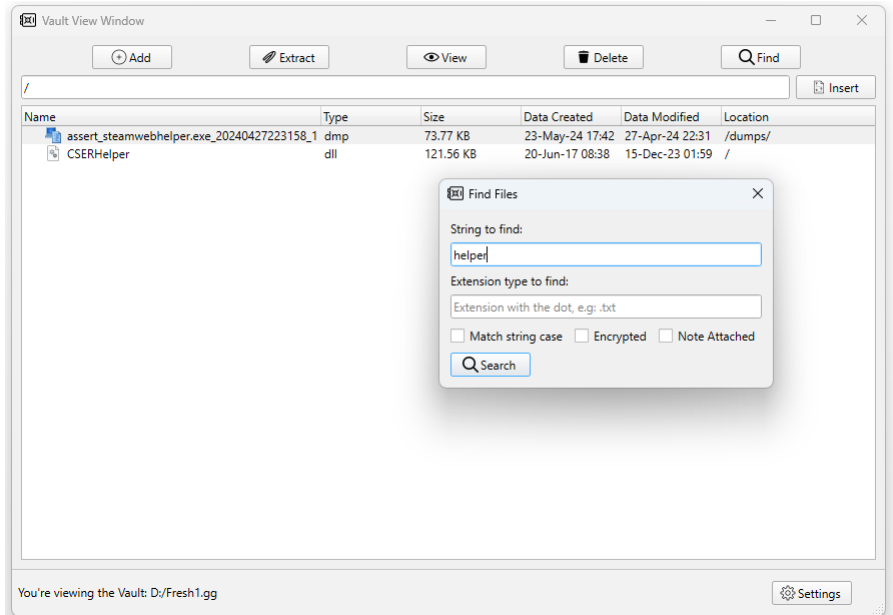


Figure 2.20: Find File View

As for exact usability, refer to the following:

- **Extension:** Look for files with this extension.
- **Match String:** Look for file names that contain this string.
- **Is Encrypted:** Look for files that are encrypted.
- **Note Attached:** Look for files that have a note attached to them.

2.2.15 Vault Settings

In order to change the Vault Name, Extension, Password, View Vault Internal Data, or Extract logs, this is the section that provides this sort of functionality. For "*Close*" and "*Maximize/Restore*", simply click on them like in any other executable application, they're fully functional.

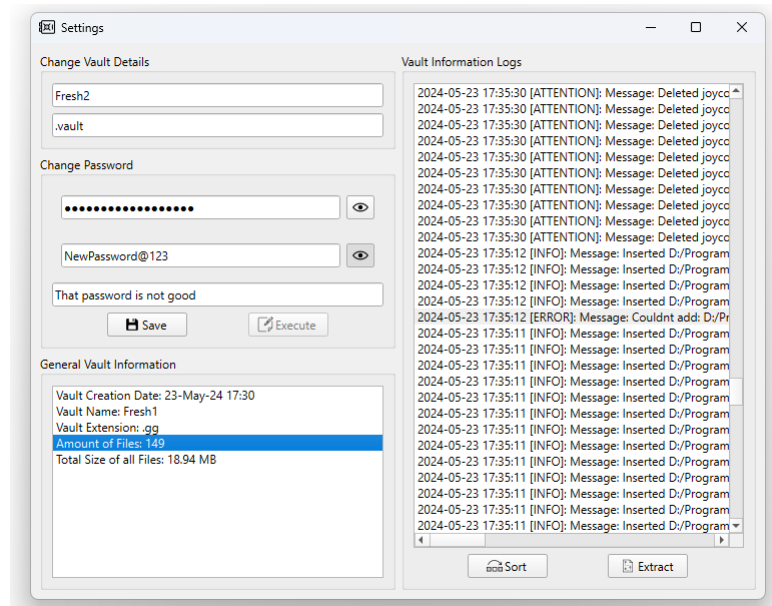


Figure 2.21: Vault Settings

2.2.16 Change Vault Details

The user can modify the Vault Name or Vault Extension without a password. However, the system may also modify these. The original name will not appear on the Vault's status bar, but the previous name can be viewed in the General Vault Information section.

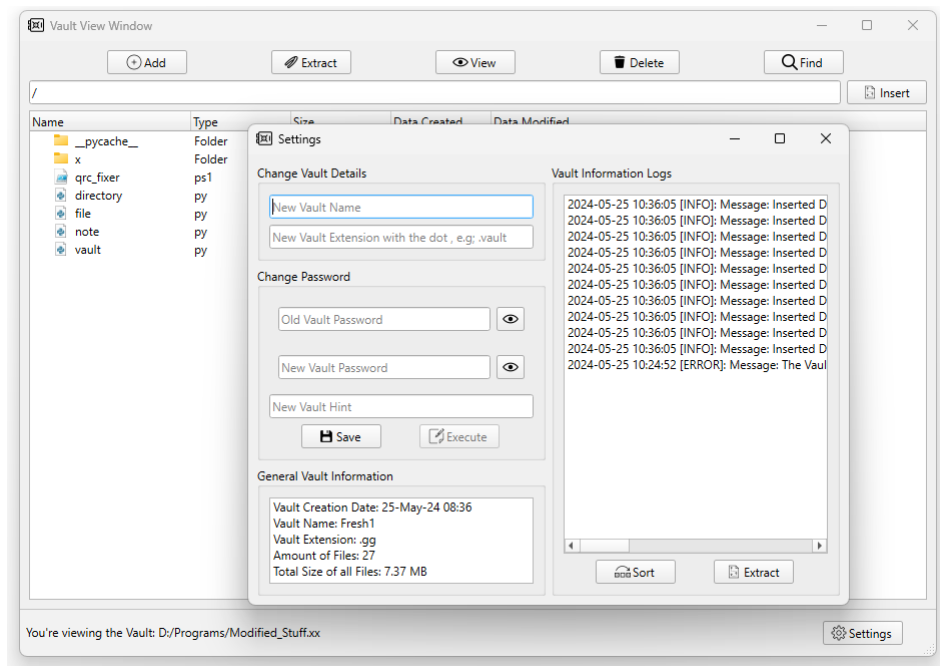


Figure 2.22: Vault General Information

2.2.17 Change Vault Password

When changing a password, click save before proceeding. After clicking execute, it will not be possible to interact with the Vault. A certain amount of time shall be waited until the change is successful, as no abort button is provided in this scenario. The encryption or decryption process must not be interrupted. Furthermore, the user will receive three new tokens, which can be used to decrypt the Vault if the password is forgotten, the older previous will automatically be unusable.

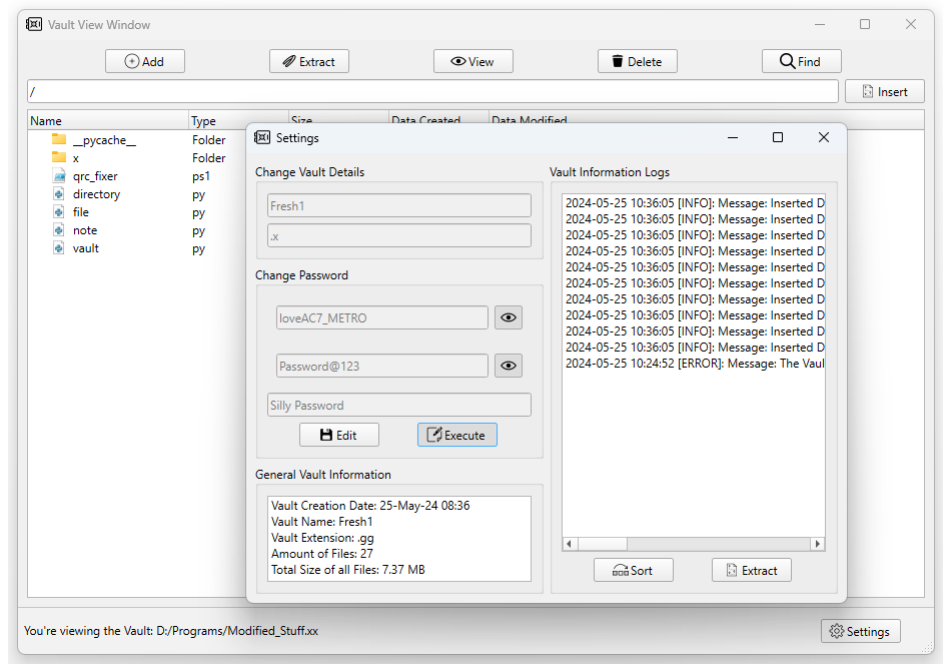


Figure 2.23: Vault Change Password

2.2.18 Log Usage

As for logging⁸, it is possible to sort them by either newly generated or older entries. By default, the new entries are shown. They show the time, the level of importance for the log, and the message.

⁸Logging is the act of keeping a log of events that occur in a computer system, such as problems, errors or just information on current operations.

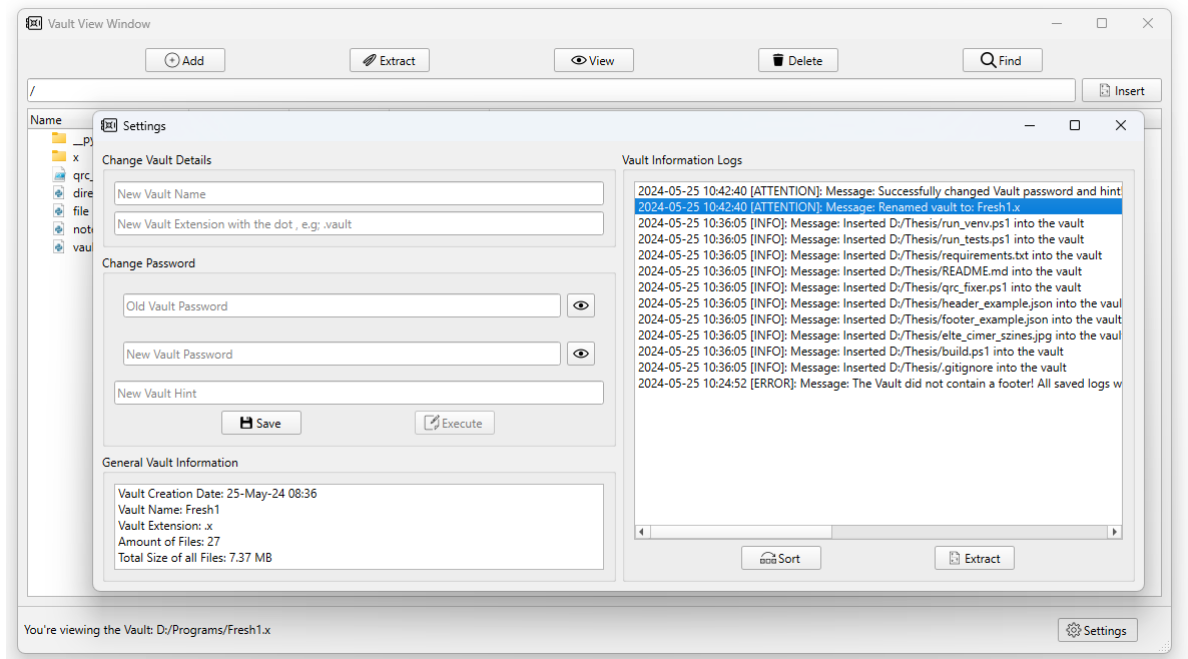


Figure 2.24: Vault Log Sorting

2.2.19 Log Extraction

It's also possible to extract the logs without deleting them to a specific path on the system.

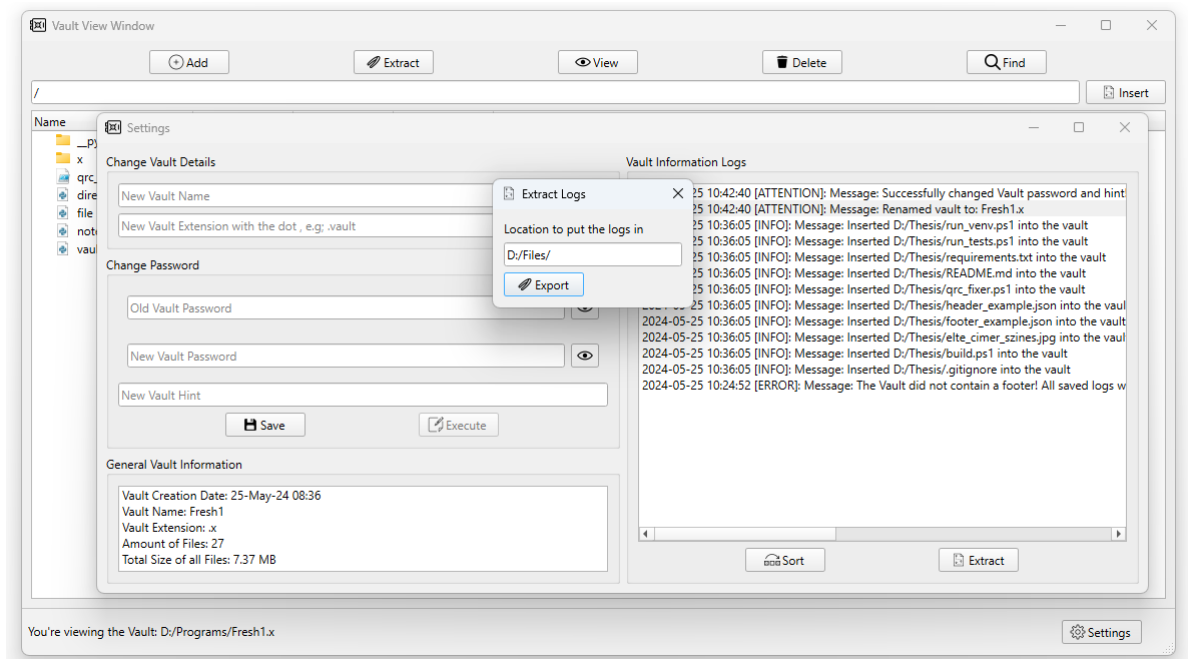


Figure 2.25: Vault Logs Extraction

2.2.20 True Encryption

The user can confirm the Vault's encryption by inspecting the bytes with a hex editor⁹, as shown in Figure 2.26.

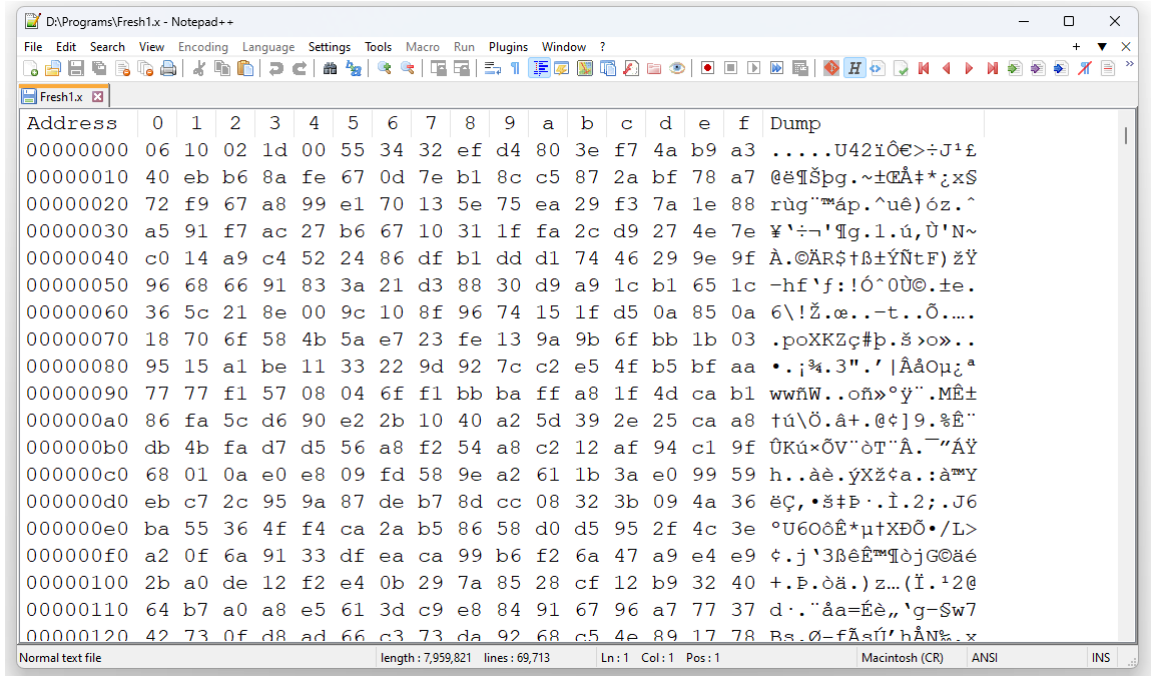


Figure 2.26: Vault Byte Checks

Even during the Vault usage, all items inside remain encrypted. There is no method to know the contents of the Vault unless the application itself is used to do so. Therefore, if an unsafe shutdown happens such as a force close, the contents remain encrypted.

2.2.21 Possible Corruption

It is important to note that no external modifications should be made to the Vault file. If external modifications occur, the Vault is at risk of corruption. The Vault file should be treated like any other system file, and no changes should be made to it unless done by the Vault itself. However, even with such external forceful modification, its possible to detect corruption as the user will be notified with a popup message as shown in Figure 2.27.

⁹A hex editor (or binary file editor or byte editor) is a computer program that allows for manipulation of the fundamental binary data that constitutes a computer file.

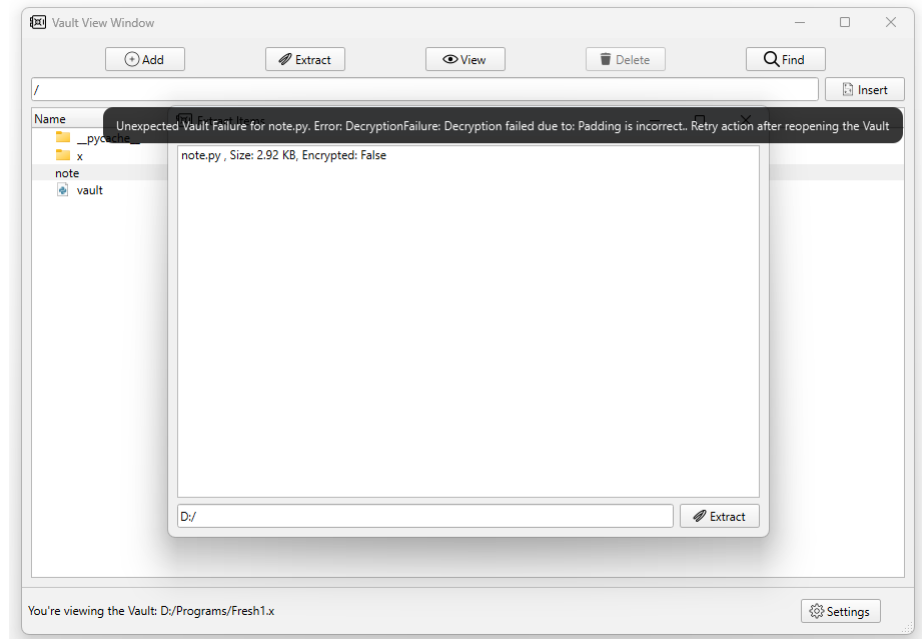


Figure 2.27: Vault Popup Notification

These notifications are logs, to investigate the trial or if needed to extract them, refer back to Figure 2.25 for more information.

2.2.22 Portability

It is possible to move or copy the Vault File into different folders, different disks, or even a different system as long as its the same operating system¹⁰. The Vault executable shall be used to view it.

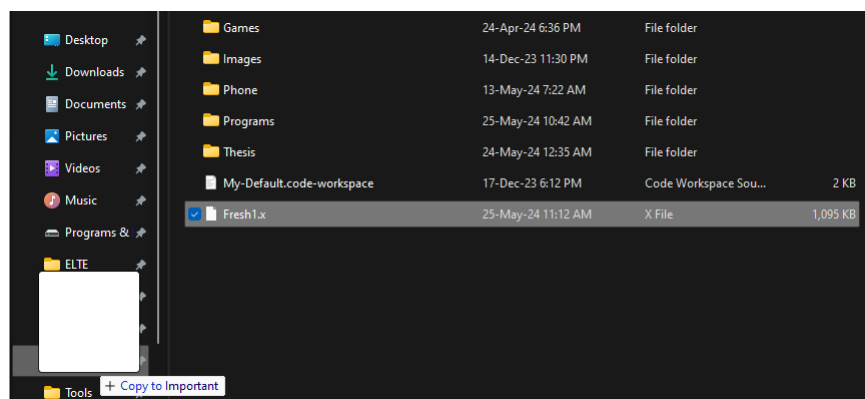


Figure 2.28: Moving or Copying a Vault

¹⁰An OS such as Windows manages the computer's memory and processes and all of its software and hardware. It enables the user to communicate with the computer without knowing how to speak the computer's language.

2.2.23 Trust

It's important to note that some certain AntiVirus¹¹ software may detect that the Vault executable as a virus and try to terminate it.

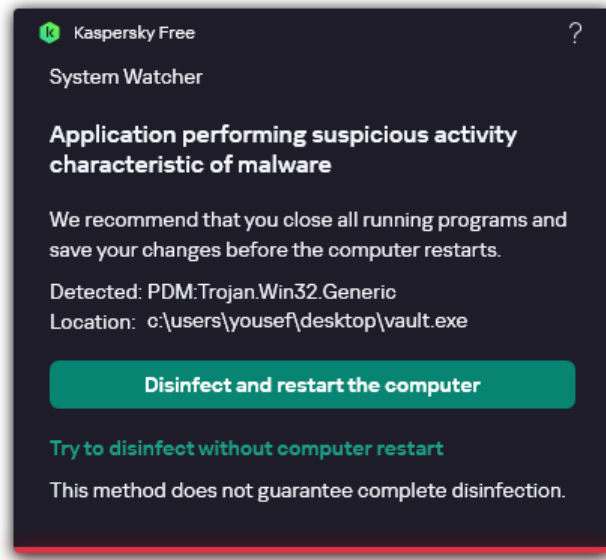


Figure 2.29: AntiVirus marking the Vault

In such scenario, the best two options are as follows:

- **Exclude the Vault executable from scanning:** This option is available in almost all AntiVirus software, the user must refer to the guide of the respective software in order to be able to exclude it.
- **Disable the AntiVirus:** Simply disabling the AntiVirus would suffice during Vault operation. Do not forget to enable it after closing the Vault.
- **Keep an extra Vault File:** The AntiVirus could directly modify the Vault File itself or immediately delete the Executable. Therefore, it is worth to keep an extra copy of both if an aggressive AntiVirus is installed on the machine.

¹¹Antivirus software, also known as anti-malware, is a computer program used to prevent, detect, and remove malware.

Part II

Developer Documentation

Chapter 3

Design

The architectural design will be split into two large sections, Header and Footer internal detailed design, this will illustrate the idea of how the tables work and how it locates all the files or folders within the Vault. Furthermore, the other section will contain information about what sort of possible interactions the Vault offers in regards to user accessibility, and how the internal design differs from regular Windows APIs ¹ which can be utilized to render the Window.

3.1 Architecture

The original design idea centers on the usage of dictionaries, as they provide quick access to the required value by just giving the correct key. This approach is useful because it achieves $O(1)$ complexity on average for reaching or displaying the designated files. The Main Vault structure consists of three parts:

- **Header**; the table that is used to locate all files in the Vault
- **File Blocks**; all concatenated files in the Vault
- **Footer**; save data for error or session logs

This structure provides a clear and understandable way to handle data management in the Vault. There can only be one Header and one Footer per Vault. Between these two, all the files saved in the Vault exist and are concatenated with each other.

¹An application programming interface acts as intermediary that allows for two or more computer programs or components to communicate or share data with each other

The dictionary is represented as a serialized JSON² file, from here on out, a dictionary is a representation of a JSON.

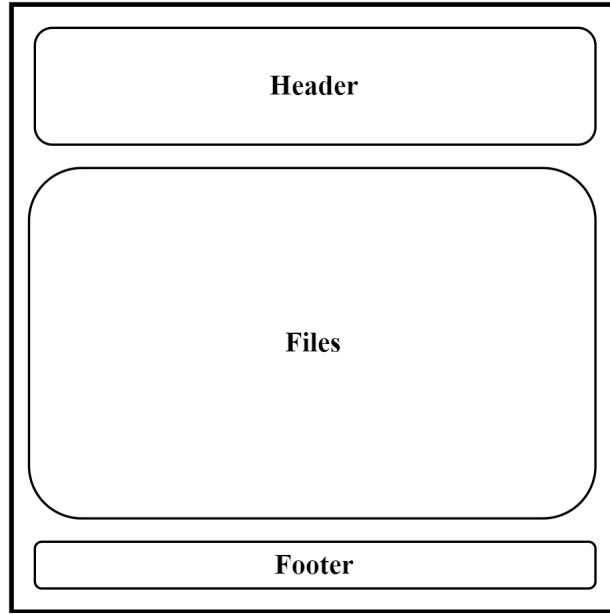


Figure 3.1: Main Structure

Header³: Core area which contains the map or table necessary to identify the location of the files and their relevant information. This is always put in the beginning of the Vault, and is put in between three magic identifier values:

- Begin Header: Starting point where the dictionary representing the full Header begins.
- Begin Pad: Starting point put after the end of the Header, it represents that the upcoming data serves as a buffer when the Header size increases. Calculating the actual size of the Header is done as follows:

$$BeginPad - BeginHeader = TotalHeaderSize$$

- End Header: Put after the extra padding⁴ necessary for the Header, the padding is calculated as follows:

$$EndHeader - BeginPad = TotalPadding$$

²JSON is an open standard file format and data interchange format that uses human-readable text to store or transmit data objects consisting of attribute–value pairs and arrays.

³Its the initial dictionary bytes at the beginning of the file representing the Vault.

⁴Padding is the addition of extra bytes to a data structure to ensure that the structure has extra allocated space to increase in size without affecting other structures.

Files: All files added into the vault, concatenated with each other without any extra buffer or markings. They can be of any type, and must have a size greater than zero. The Vault is able to handle large size files, but performance when it comes to encrypting or decrypting them individually decreases. There's no effect on file extraction however, the default chunk size handle is *50MB*.

Footer⁵: Simple dictionary which consists of the history log needed for the Vault. It contains two magic identifier values:

- Begin Footer: Start point after the last file, where the dictionary representing the full Footer begins.
- End Footer: Put after the ending of the actual Footer. Size of the Footer can be calculated as follows:

$$EndFooter - BeginFooter = TotalSize$$

3.1.1 Header Decomposition

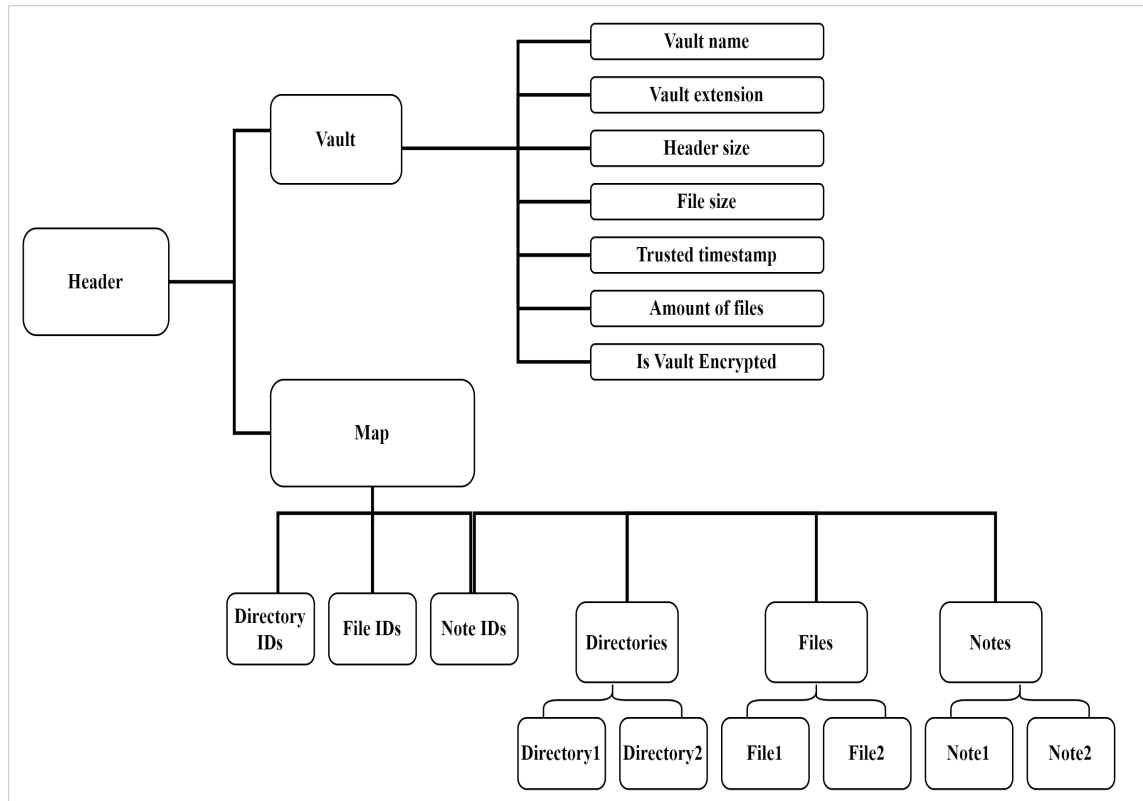


Figure 3.2: Header Structure

⁵Its the last dictionary bytes at the ending of the file representing the Vault.

Header is composed into two sections, where the main one is the Map, and the second one is the Vault details.

- **Vault:** A dictionary key consisting of 7 values which are used to represent general Vault data that can be viewed in the Settings:
 - **Vault Name:** a string representing the name used to create the Vault.
 - **Vault Extension:** a string representing the extension the Vault had during creation.
 - **Header Size:** the size of the decrypted Header.
 - **File Size:** the total size of all files in the Vault, the size is calculated after encryption.
 - **Trusted Timestamp:** the timestamp when the Vault was created.
 - **Amount of files:** total amount of files in the Vault, it also includes Notes.
 - **Is Vault Encrypted:** a boolean value to represent whether the Vault is fully decrypted.
- **Map:** A dictionary key consisting of 6 values which are used to organize how Files are accessible, Directories are organized, and where the Notes exist:
 - **Directory IDs:** a list of folder IDs that exist in the map.
 - **File IDs:** a list of file IDs that exist in the Vault.
 - **Note IDs:** a list of Note IDs that are attached into files.
 - **Directories:** a dictionary contains key-value pairs where each key is the ID of a Directory, and each value is the corresponding sub-dictionary.
 - **Files:** a dictionary contains key-value pairs where each key is the ID of a File, and each value is the corresponding sub-dictionary.
 - **Notes:** a dictionary contains key-value pairs where each key is the ID of a Note, and each value is the corresponding sub-dictionary.

3.1.2 File Decomposition

Every file in the map consists of critical data representing where it exists in the concatenation of files, and what sort of information with it is included, the structure is visualized in Figure 3.3.

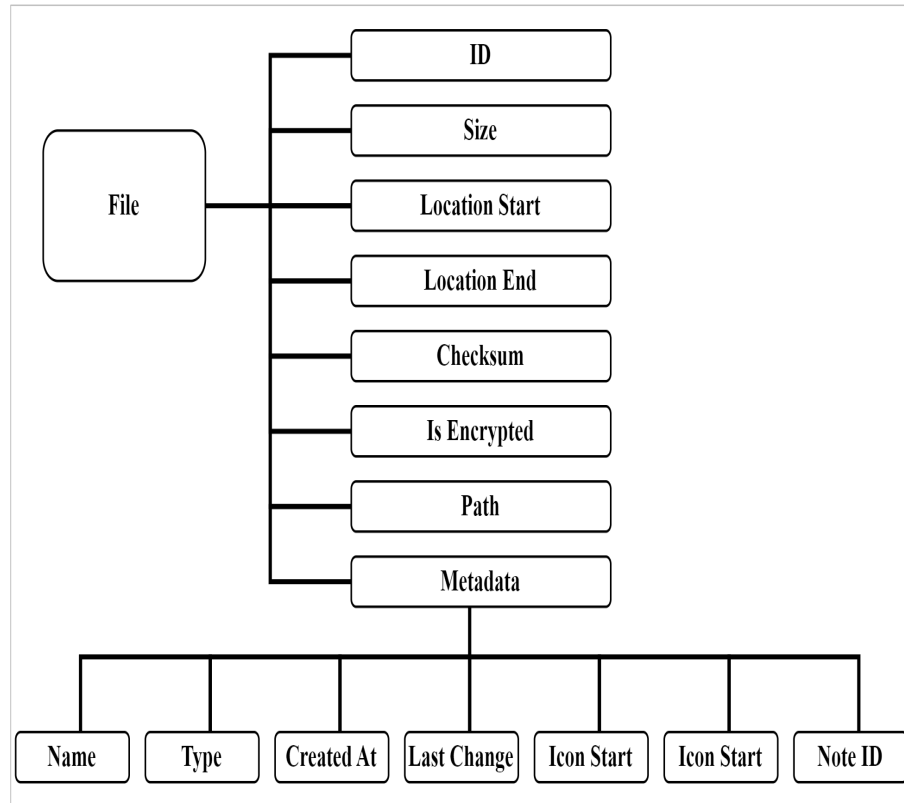


Figure 3.3: File Structure

Most Notable ones to go over are:

- **Path:** The ID of the directory this file belongs to.
- **Location Start:** The exact byte index the requested file bytes begin in the Vault.
- **Location End:** The exact byte index where the requested file bytes end in the Vault.
- **Icon Start:** Index representing where the icon of the file starts, useful for custom EXEs, value is -1 if it does not exist.
- **Icon End:** Index representing where the icon of the file ends.
- **Note ID:** The Note ID attached to this file, -1 if does not exist.

3.1.3 Directory Decomposition

Folder or simply a Directory, the name in this context are the same. This serves as a way to point towards paths, and chain other folders which belong under it via a combination of the ID or path.

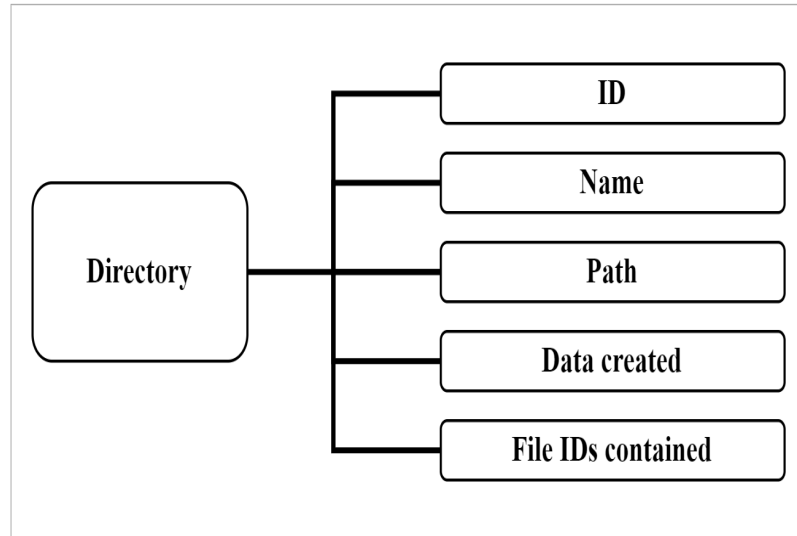


Figure 3.4: Directory Structure

- **ID:** The actual ID of the directory, where other directory or files can belong under. This is used as the *Path* value which other Directories or Files use to be saved under.
- **Path:** The ID of the parent directory this directory belongs to.
- **File IDs:** The file IDs which have their *Path* set to this directory.

Displaying the full file or folder location requires recursively traversing all available paths until reaching the path *'0'*, which is considered the *'Root'* path. This path will always exist and is the only path without an *'UpOneLevel'* navigation button because its the parent. An example path looks like:

`/MyInformation/Documents/TopSecret/`

Where the initial *'/'* path is the **Root Directory**⁶. This structure closely resembles what Linux or UNIX-like systems offer. Only its contents can be removed.

⁶The root directory contains all folders and files on the system

3.1.4 Note Decomposition

Originally, the design decision for the notes was to be only *Voice Notes* which are used to be represented as additional Metadata for any file that exists in the Vault. This has evolved to the attachment of any file type which serves as an attachable Note into the Vault.

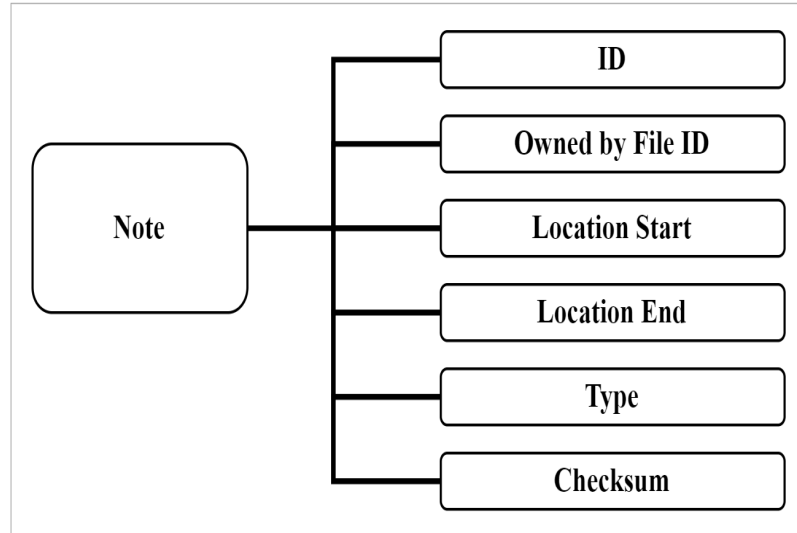


Figure 3.5: Note Structure

- **Owned by File ID:** The File ID this Note belongs under.
- **Location Start:** The exact index the requested note bytes begin in the Vault.
- **Location End:** The exact index where the requested note bytes end in the Vault.

The notes and icons representing the files in the Vault are the only items that are not encrypted, as their sole purpose is to serve as simple representations of items. Therefore, they do not require encryption. In theory, any type of file can be inserted as a note, but there is a **10MB** limit to prevent exploitation of this function.

3.1.5 Footer Decomposition

The underlying data structure is also a dictionary. The purpose of the footer is to save any error data or warning session logs that occur in the Vault. One major benefit of the footer is that it is **not mandatory** to have it; it is self-healing and can always be regenerated, although any previously saved data will be wiped.

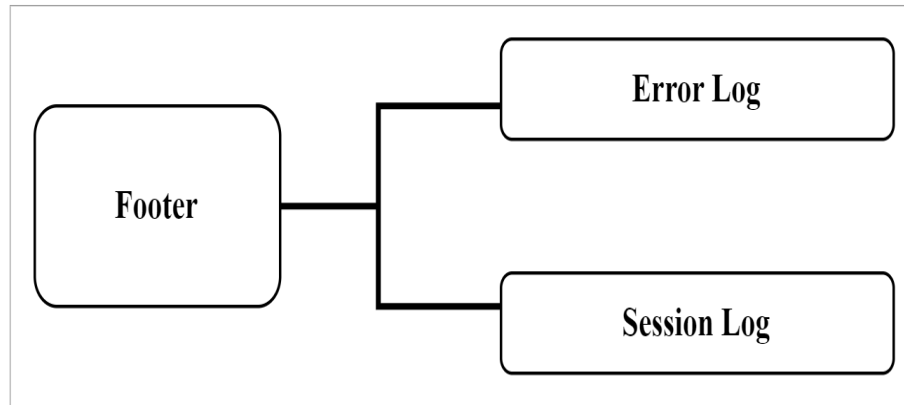


Figure 3.6: Footer Structure

- **Error Log:** Any ERROR logs generated throughout the lifetime of the Vault.
- **Session Log:** Any WARNING, or ATTENTION logs generated throughout the lifetime of the Vault.

3.1.6 Logging

Logs generated by the Vault are not following any Logging conventions such as the obsolete *RFC 3164* [3] or the newer *RFC 5424* [4] which are used in different applications. its a mix of various recommended log types, the purpose is to have a general log which outputs the time, message, and severity. For example:

```
2024-05-22 07:27:14 [ERROR]: Message: The Vault did not contain a footer!
```

That's why a log type such as *ATTENTION* can be found. The *INFO* type is the only type that is not saved into the footer after the Vault session ends, it is automatically deleted after exiting the Vault as it is not necessary to keep track of.

Examples:

```
2024-05-22 14:26:35 [ATTENTION]: Message: Vault Fresh1.gg created!
```

```
2024-05-22 14:27:13 [INFO]: Message: Inserted D:/sample.txt into the vault
```

```
2024-05-22 14:29:22 [WARNING]: Message: Couldn't decrypt x.x! Incorrect Password
```

Thus, only 4 types exist: *ERROR*, *ATTENTION*, *WARNING*, *INFO*

3.1.7 Vault View Design

The interface is designed to resemble that of a file archiver, featuring key functionalities such as Add, Extract, View, Delete, and Find.

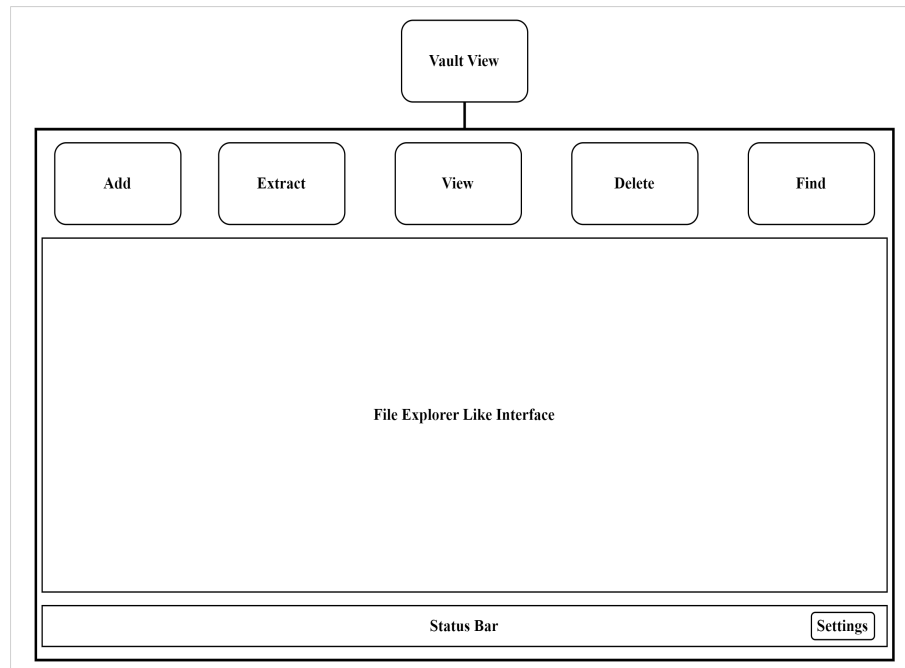


Figure 3.7: Vault View

Heavy tasks such as adding, extracting, or deleting triggers a progress bar to indicate the approximate remaining time. This holds true for finding a Vault on the system as well.

The status bar of the main view indicates what Vault is currently being shown. One of the key features is to keep the Vault as hidden as possible on the computer, any valid character extension name, any valid file name for the Vault can be used.

Right-clicking on any button will display information about its purpose. It's worth mentioning that navigation actions can also be performed using the keyboard: the **Up** and **Down** arrow keys navigate through folders, **Enter** opens a folder, and **Backspace** returns to the previous folder. Currently, no special navigation features are implemented for visually impaired users. Only the **TAB** button can scroll through every intractable item.

3.1.8 File Explorer

The view window for the Vault is uniquely tailored, to be as closely representative of an actual Windows File Explorer ⁷ without using any available APIs provided by Microsoft itself, and this was inspected with SpyPlusPlus[5]. Rendering is done uniquely by reading the File attributes, and displaying them adequately with the required columns.

The search is rendered based on the data provided from the Map. Most of the inspiration was taken from WinRAR [6] and 7-zip [7].

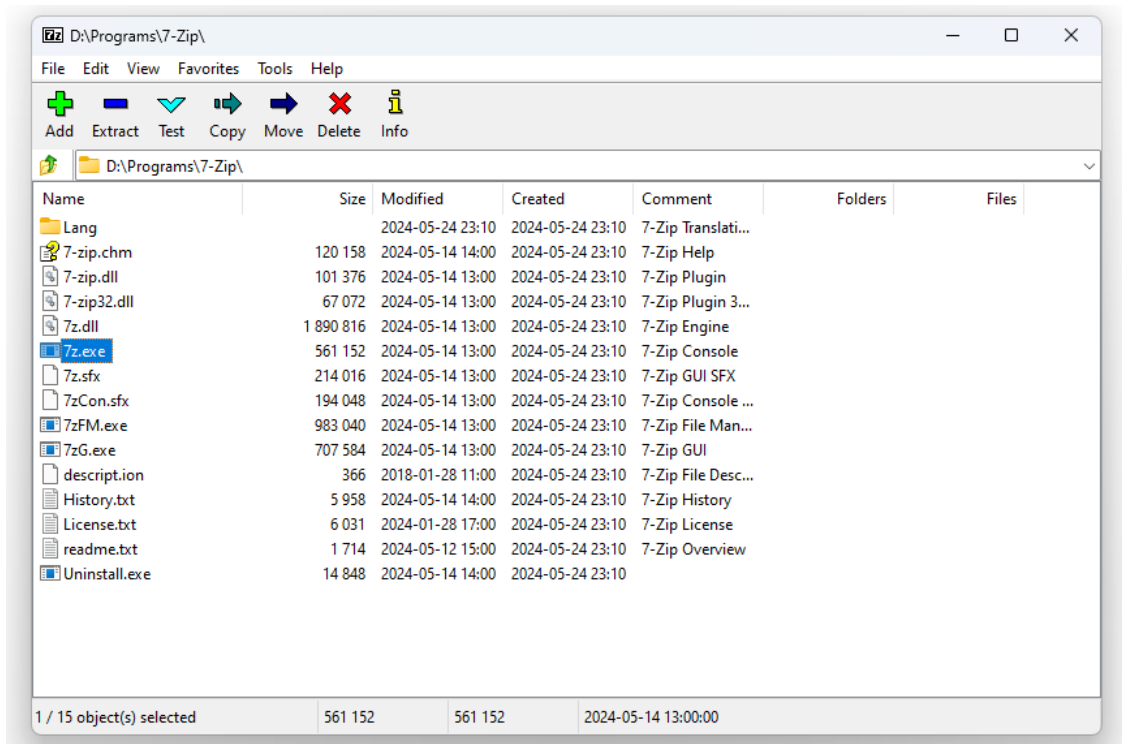


Figure 3.8: 7zip interface

3.1.9 View File Information

Checking the Metadata of the File allows the user to continue with various interactions, such as individual encryption, deletion, and attaching a Note.

⁷File Explorer is a file manager application and default desktop environment that is included with releases of the Microsoft Windows operating system.

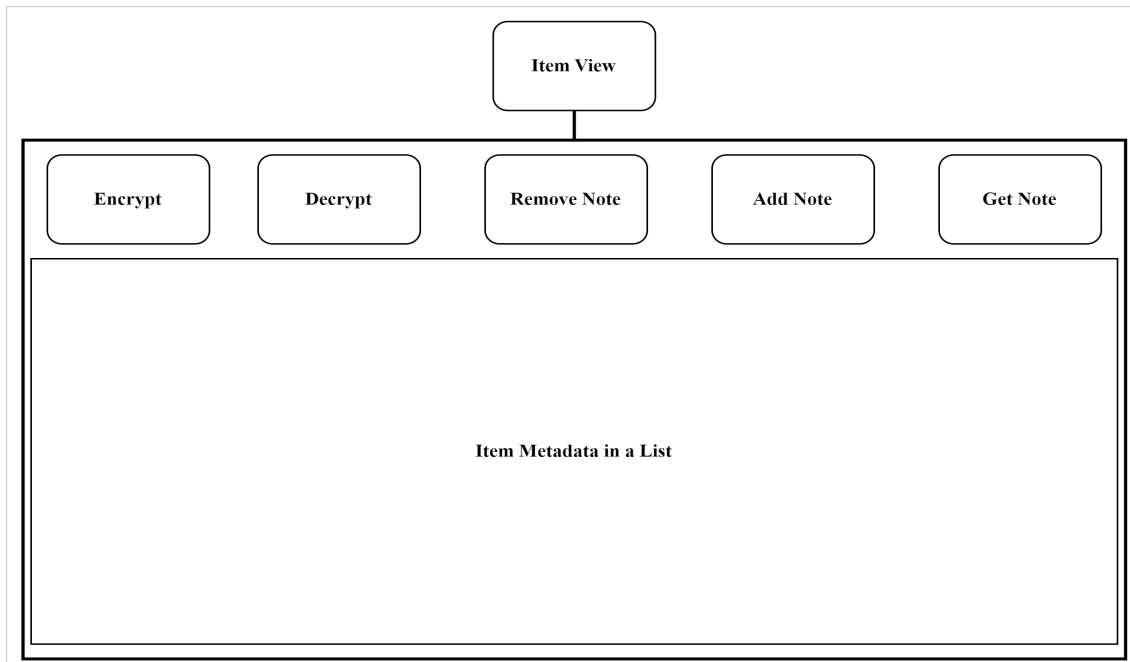


Figure 3.9: Item View Window

To emphasize, the purpose of Adding a Note is to serve as additional Metadata related to the File. This is useful if the file is encrypted to understand why, or have a hint for it.

The idea of individual encryption is to prevent:

- **Extraction:** Its not possible to extract a File which is individually encrypted in the Vault. The special password used for it must be given, otherwise, the File shall exist indefinitely until deleted or decrypted.

Its not peculiar to have this sort of encryption, first, the file is encrypted by its own special password, and then encrypted again with the Vault password. The Vault itself is only capable of decrypting the layer which was added by the Vault, this is visualized in Figure 3.10.

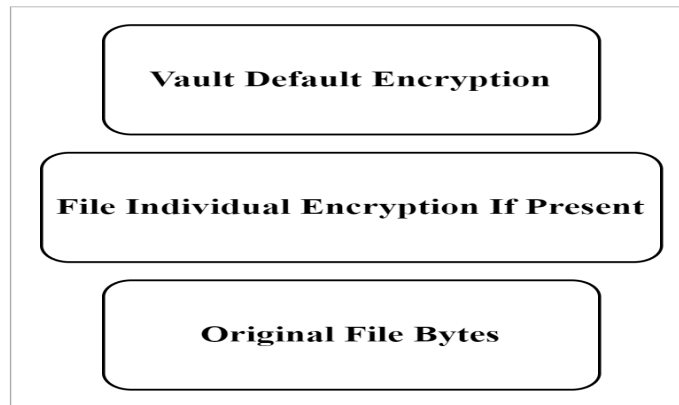


Figure 3.10: Individual Layer Encryption

It is important to note the distinction between extraction and deletion. Extraction keeps the given files or folders in the Vault; as it only creates an encrypted copy for the Vault to have. While deletion completely removes the existence of the files or folder from the Vault.

3.1.10 File Search

Searching detailed information about the files in Vault is also possible, it is not as plain as the Windows search bar provides, but rather more options are included.

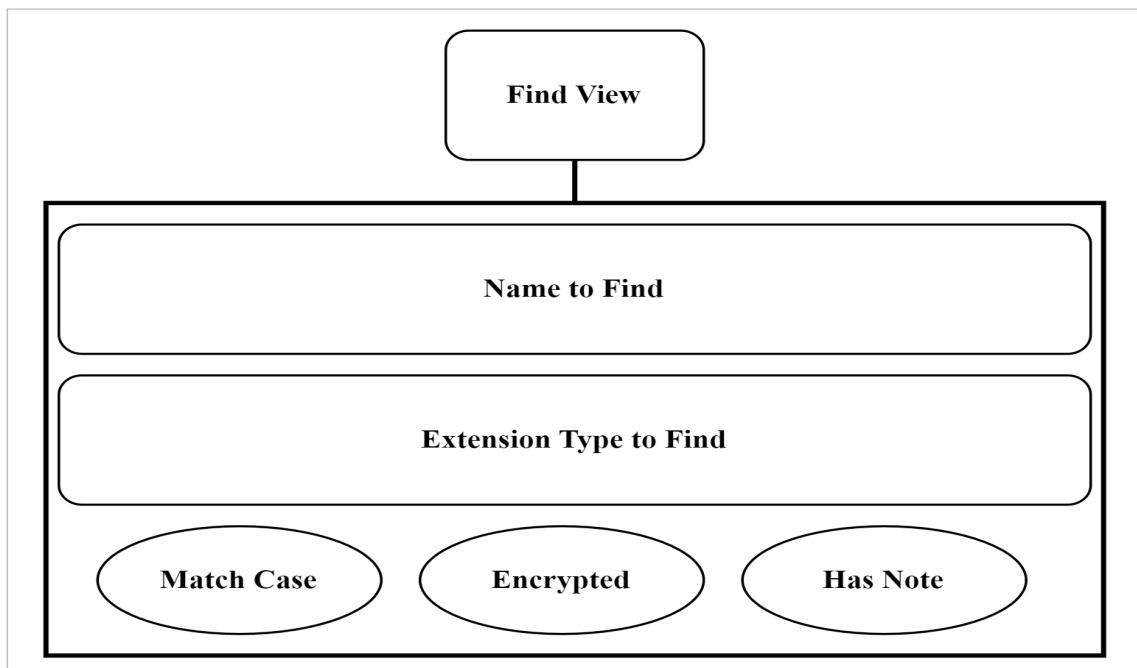


Figure 3.11: Find File View

All possible interactions are limited to what does the Vault have to offer at initial release which can be summed up to: Custom Notes and Individual Encryption

3.1.11 Vault Settings

The settings of Vault can be changed to reflect reality saved in the Vault.

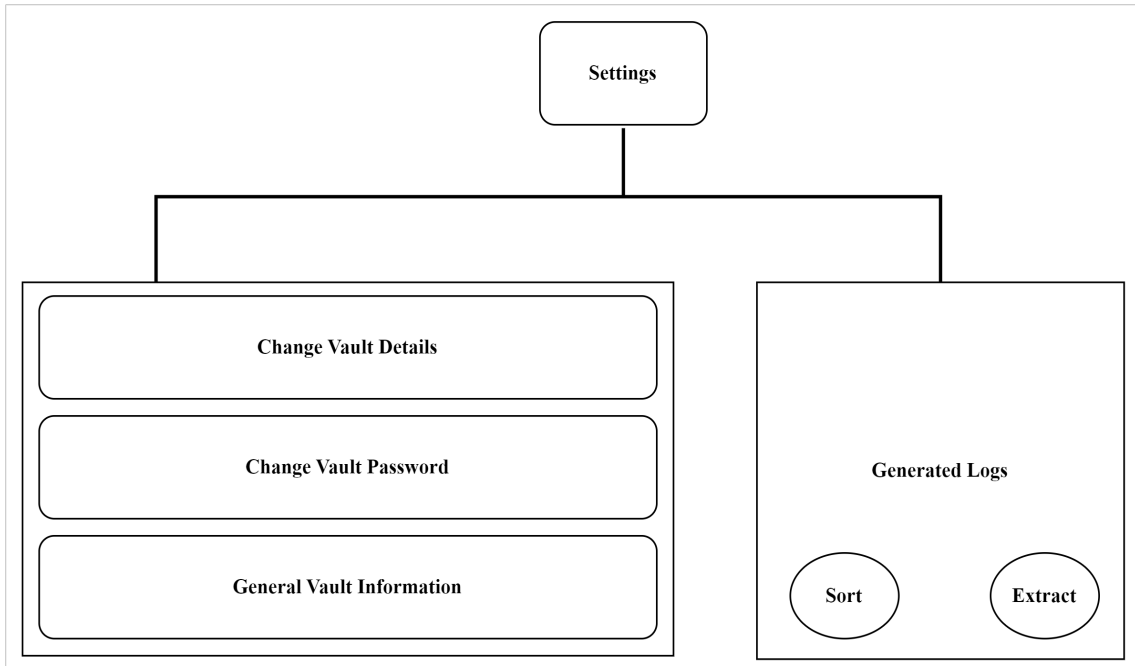


Figure 3.12: Settings Window

Main idea to provide a way to change the current password of the Vault, display Vault details such as the timestamp of when the Vault was created, and an extraction of the already saved logs in the Vault. Vault details are represented in:

- **Vault Name:** The real name of the Vault
- **Vault Extension:** The real extension of the Vault

It is referred to as 'Real' name or extension, since its possible to simply modify the extension and the name of the Vault manually via FileExplorer or even a command line terminal. This is why reflecting reality of what is internally built must be given.

It must be noted that, with the current Design, a special way to get the 'Password Hint⁸' of the Vault is implemented, and it only exists throughout the life time of the initial footer created by the Vault. Therefore; if the initial footer is destroyed, the hint is deleted out of existence.

3.1.12 Interaction Diagram

The full possible interaction with the Vault is illustrated in Figure 3.13.

⁸A text of a maximum size of 32 bytes representing the hint of the Vault added by the user

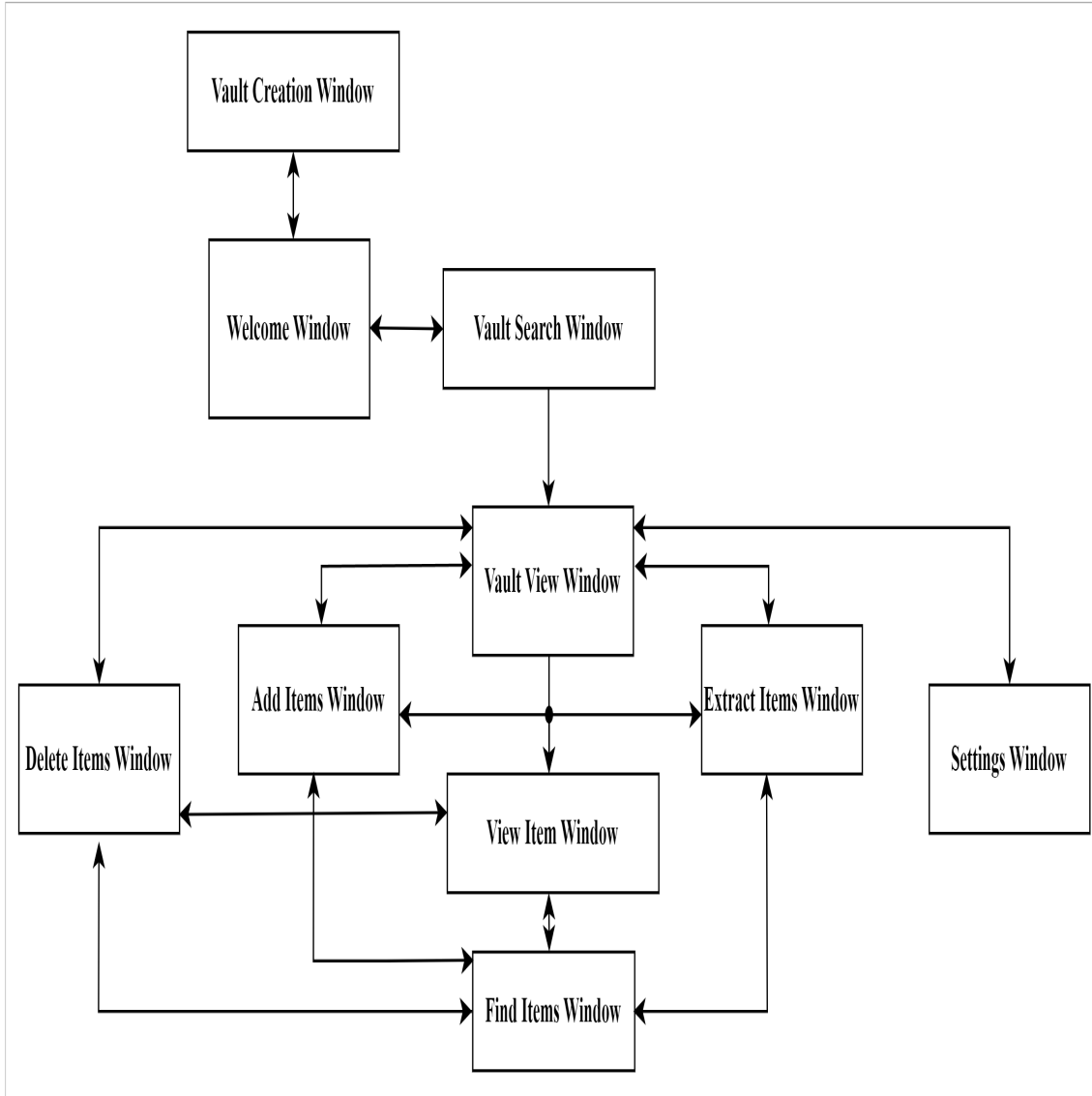


Figure 3.13: Possible User Actions

This is a close representation of how many intractable windows can be open simultaneously, its possible to use all of them together because they are Non Modal except for the *Settings Window*. However, certain Window open buttons are disabled if special Window is open. For example, *Delete Items Window* will disable the Add or Extract item Windows.

3.2 Limitations

The design architecture has a few limitations which need to be put under consideration:

- **No debuffer⁹ for the Header:** The Header by default, has a buffer of **4KB**, it expands whenever the estimated size increases. Consequently, after long-term Vault usage, if much of the data is deleted, the Header size decreases but without the buffer. The largest allocated buffer remains in the Vault. Its beneficial in a scenario where a large amount of Folders or Files will be added in the future, as there is less possibility for shifting bytes in order to increase the padding for the Header, however, the multiple of **4KB** can still be mitigated but the implementation is expensive since various aspects like constant deletion or constant addition must be considered.
- **Unrecoverable Header:** As the header is placed at the very beginning of the Vault file, any slight external modification to it can cause corruption. A possible way to mitigate this is to cluster headers and distribute them in small quantities throughout the Vault file. This approach reduces the risk of complete corruption, limiting potential damage to only partial corruption. By splitting the header into smaller segments that are aware of each other's locations, we can enhance the Vault's resilience to corruption.
- **Byte Precision:** Since the bytes of the files in the Vault are shifted (overwritten) from one place to another, any byte-shifting or manipulating functions are at high risk of corrupting the file in question. A chain of file corruption would be imminent if even a single byte is misplaced.
- **Slow file deletion in a case of a large Vault:** If the Vault is extremely large, such as **10 GB**, removing a file located at the beginning would be slow. This can be mitigated by using temporary files to store data, though this approach requires additional temporary storage that will be consumed and removed during the operation.

⁹Debuffing means removing the extra buffer bytes from the end of the header.

Chapter 4

Implementation

This chapter differs from the user documentation. It introduces the development process from a developer's perspective, covering both the backend and the frontend. The software requirements will be analyzed, and any underlying technical limitations will be mentioned.

4.1 Software Requirements

As part of an application development process, backend, frontend, and a main programming language shall be chosen, thus, the conclusive aspect for this project is the usage of:

`PyQt 6.7` , `Python 3.12`

As part of the Qt[8] family, PyQt[9] is used for backend and frontend development, and its based on Python[10]. It served as a good candidate to start GUI development, and along the way, the full capability of the software to provide functional backend logic behind the scenes was realized. PyQt allows for simple connections of functions with their respective GUI elements, consequently facilitating adequate organization of the code.

4.1.1 Chosen Software limitations

Originally throughout the development process, the version 6.4 of PyQt and Python 3.9 were used. Issues encountered were:

- **PyQt 6.4:** Inconsistent handling of pixmap¹ bytes extraction for the Icon of the File. The occurrence of this was rather random, as it worked with bulk File import, but sometimes not. The same applies for individual File import. The source code to prove it is as follows:

```
1         file_info = QFileInfo(file_location)
2         file_icon_provider = QFileIconProvider()
3         icon = file_icon_provider.icon(file_info)
4         if not icon or icon.isNull():
5             return None
6         # start culprit line of code
7         pixmap = QPixmap(icon.pixmap(QSize(16, 16)))
8         # end culprit line of code
9         byte_array = QByteArray()
10        buffer = QBuffer(byte_array)
11        buffer.open(QBuffer.OpenModeFlag.WriteOnly)
12        pixmap.save(buffer, "PNG", quality=1)
13        byte_array = buffer.data()
14        buffer.close()
15        return byte_array.data()
```

Code 4.1: Pixmap extraction in Python

Upon investigation with SysInternals[11] ProcessExplorer², PyQt 6.4 was calling WINMM.dll³ with a separate thread that caused unmanageable freezing and hanging in the program. To solve the issue, two solutions could be used:

- **Use Default Icons:** Completely disregard saving Icons from Files inside the Vault.
- **Upgrade Library:** Update the Library to the latest version.

And the latter proved to be a better solution than the former.

- **Python 3.9:** EOL⁴ date is 2025, and it's the last compatible version with the latest release of PyQt 6.7. Thus, a simple Python upgrade ought to suffice. Also, Python 3.9 is the version which works with *pyqt6-tools*.

¹A pixmap stores and displays a graphical image as a rectangular array of pixel color values.

²Process Explorer shows information about which handles and DLLs processes have opened or loaded.

³Windows Multimedia API, it also serves as a sound library.

⁴End of life refers to the point in time when hardware and software systems reach the end of their useful life.

- **PyQt internal implementation:** Its based on C/C++ Objects, careful management of deleting them when their existence is not necessary shall be followed, otherwise there would be a potential possibility for memory leakage⁵.
- **PyQt sub-optimal documentation:** Multiple functions and class implementations had to be referred to the original C++ Qt implementation for more details. Some code examples were written in C++ , and had to be translated manually to Python to proceed with the development.

4.2 Class Diagrams

The Main class that shall be elaborated upon is the "***VaultView***" class. As it is the one responsible for all of the interactions which can be done with the Vault. One important decision was to keep the QWindow References, they go out of scope and are garbage collected, which will destroy the underlying C++ obj thus resulting in a Runtime error if improper deletion happens. Same applies to any class requiring additional QWidgets, the references are also kept.

- **ViewManager:** This class is only used to manage the navigation between the Search, Create, and View Vault Windows. It essentially manages the destruction of the underlying C++ objects to avoid memory leakage or runtime errors. However, this Object is destroyed once the VaultView Window is open.
- **VaultSearchWindow:** A representation of the files and folders available on the system itself. It is a data field reference for the class ***ViewManager***.
- **VaultCreateWindow:** Used to get the necessary information from the user to create a Vault. It is a data field reference for the class ***ViewManager***.

Any other interaction dialogs, custom windows, and custom widgets are created on demand and destroyed once not needed anymore.

As for package management, all essential graphical user interfaces classes are under the GUI folder, all other utility functions are in their respective folders and are imported where necessary or where required by a component.

⁵a memory leak is a type of resource leak that occurs when a computer program incorrectly manages memory allocations in a way that memory which is no longer needed is not released.

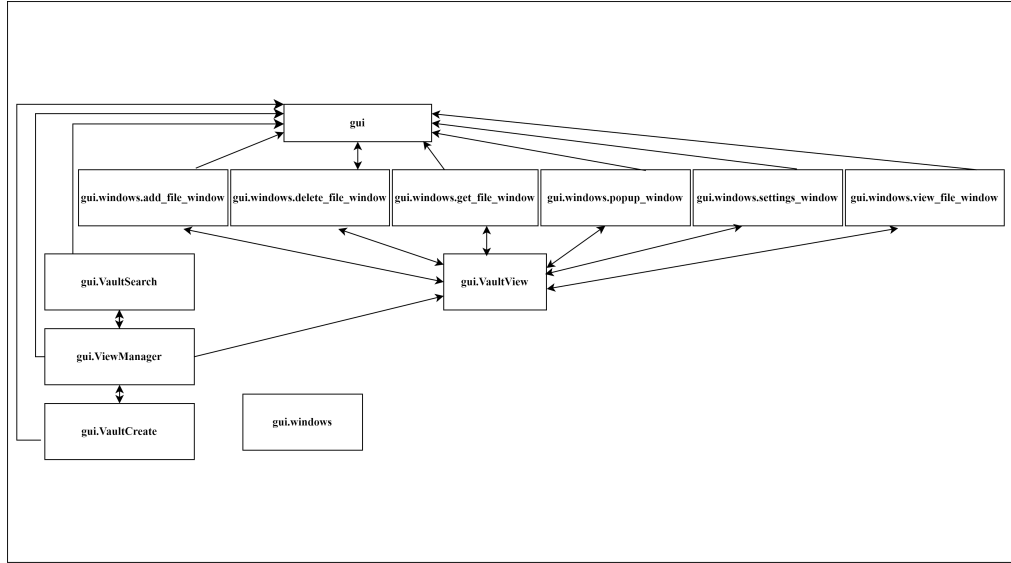


Figure 4.2: Main Packages

A *requirements.txt* file is included with the code base to manage the packages and to ensure that all necessary cryptographic and GUI functions are available.

4.3 Backend implementation

Since some PyQt's QWidgets⁷ such as '*QButton*' or '*QLine*' offer connecting them into Python functions, thus, the organization of the code can be visualized in Figure 4.3. CustomWdigets, Interactions, and Windows are extensions of QWidgets, these are effectively used by the VaultView.

⁷Widgets are the basic building blocks for graphical user interface (GUI) applications built with Qt. Each GUI component (e.g. buttons, labels, text editors) is a widget that is placed somewhere within a user interface window, or is displayed as an independent window.

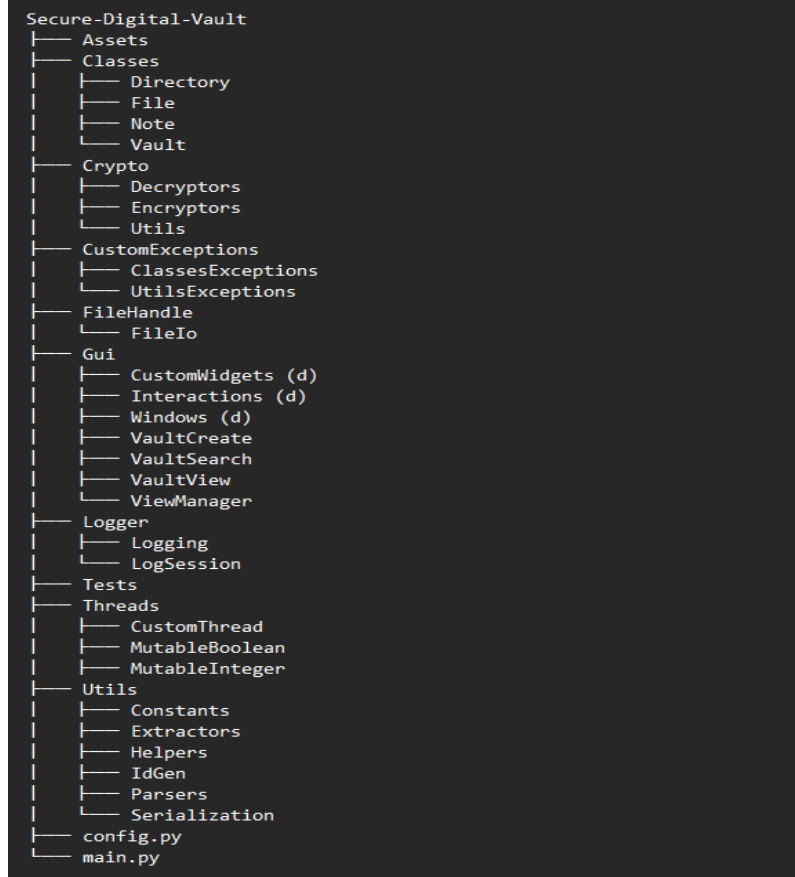


Figure 4.3: Code Structure

This allows for the seamless import of any functions into the *Gui* directory, effectively separating the backend from the frontend. This also facilitates easier testing, as the backend and frontend can be tested separately.

4.3.1 Creating a Vault

```

1      class Vault:
2          def __init__(self, password : str, vault_path : str):
3              self.__header = {}
4              self.__footer = {}
5              self.__map = {}
6              self.__vault_path = vault_path
7              self.__password = password
8              self.__hint = "No Hint"
    
```

Code 4.2: Simple Vault

One of the most notable functions within this file is: '*update vault file*' where it updates the vault with the newly encrypted header, and updates the disk. All relevant developer functions in the code are documented using Python Docstring[12].

The calling of the functions sequentially is done from the GUI, where first the Vault file is fetched, decrypted, and then attached to the Vault. This provides a better Object-Oriented approach when it comes to data encapsulation ⁸ as the Vault's priority is to Store the Header, and only update the Disk when necessary.

4.3.2 Utilities

All important functions are split in folders, and the main ones are organized as follows:

- **filehandle:** Core functions for I/O operations.
- **utils:** Most of the helper functions for better organization.
- **crypto:** Any cryptographic related functions exist in this location.

4.3.3 Main Override Function

As the idea of the Vault is to:

- **Not Create a .tmp file:** Any operation done in the Vault, shall not create a temporary file to aid in speed, organization or efficiency.
- **Not Insert Large Data Blocks Into The RAM:** Any operation done in the Vault, shall limit the usage of the RAM, and handle the Blocks in chunks that reach the maximum size of **50MB**.
- **Must be Storage and RAM Efficient:** If previous two points are adhered, then the matter of the Storage size or RAM size is not coming under consideration, as there is no need for a temporary file to handle the cutting and reattaching of data, or no large data blocks will be taking the capacity of the RAM during Vault activities.

⁸refers to the bundling of data, along with the methods that operate on that data, into a single unit.

Therefore, during any operation that requires either: Insertion, or Deletion, effectively shifts the bytes after a chunk from the chosen index. Illustration Code is as Follows:

```
1     file = open(file_path, "rb+")
2     move = len(given_bytes) - byte_loss
3     move = max(0, min(move, file_size - at_location))
4     # Read the chunk that will be displaced by the new data
5     file.seek(at_location + move)
6     lost_chunk = file.read(chunk_size)
7     file.seek(at_location)
8     # Write the given bytes at the specified location
9     file.write(given_bytes)
10    save_location = file.tell()
11    # Calculate the next byte loss
12    written = len(given_bytes)
13    file.seek(0, 2)
14    remaining = file.tell() - save_location
15    file.seek(save_location)
16    next_byte_loss = len(lost_chunk) - written
17    # If next byte loss is negative, handle edge cases
18    if next_byte_loss < 0:
19        if remaining == 0:
20            to_write = abs(next_byte_loss)
21            file.write(lost_chunk[:to_write])
22            return file
23        if next_byte_loss + save_location >= file_size:
24            file.write(lost_chunk)
25            return file
26        return file
27    # If lost chunk is empty, return file
28    if len(lost_chunk) == 0:
29        return file
30    # Recursively handle the next chunk
31    return override_bytes_in_file(file_path, lost_chunk,
        next_byte_loss, at_location=save_location, chunk_size=
        chunk_size, fd=file)
```

Code 4.3: Override Bytes in File

As this is a basic logical illustration of how shifting happens, it does not exactly reflect what is in the original code base, as that is constantly improved for efficiency.

4.3.4 Encryption

The Vault is utilizing AES[13] encryption, which is being used in CBC[14] mode. As the key is set to 32 bytes (256 bits), thus the AES variant is of AES-256 standard. Padding is of PKCS7[15] type, for integrity checks, SHA-256[16] is used as the checksum algorithm.

To name a few benefits and drawbacks which were considered during implementation can be summed in:

- **Benefits:**

- **CBC:** is secure against certain types of attacks, such as replay attacks, which can be an issue with simpler modes like ECB (Electronic Codebook) mode.
- **Chaining Mechanism:** each plaintext block is XORed with the previous ciphertext block before being encrypted, making it reliant on the previous ciphertext.
- **Increased Security:** By incorporating an initialization vector (IV), CBC mode ensures that identical plaintext blocks will encrypt to different ciphertext blocks.

- **Drawbacks:**

- **Padding:** CBC mode requires that the plaintext to be a multiple of the block size (usually 16 bytes for AES). This necessitates padding the plaintext, which can add a slight overhead.
- **Vulnerable to Oracle Padding Attack.**[17]
- **Sequential Processing:** Encryption and decryption in CBC mode cannot be parallelized, which can impact performance.

4.3.5 Token Generation

In order to recover the Vault, it is possible to recreate more than one key to unlock the Vault. These are generated when the Vault is being created. There are multiple different ways of recovering the Vault which could've been implemented, but the necessity of no internet access was mandatory. Thus, derivation of the the Password

key with 3 tokens is done by default everytime a Vault comes into existence. These can be used if the **Token** option is ticked when searching for a Vault.

```
1 def generate_password_token(password : str) -> bytes:
2     tokenizer = [83, 101, 99, 117, 114, 101, 45, 68, 105, 103,
3                 105, 116, 97, 108, 45, 86, 97, 117, 108, 116]
4     cipher = xor_magic('').join([xor_magic(chr(n)).decode() for
5                                  n in tokenizer])).decode()
6     token = encrypt_bytes(password.encode(), cipher)
7     return token
```

Code 4.4: Token Generation

The logic uses a substitution cipher with a special value, these bytes will be converted to Base64⁹ encoded string.

4.3.6 Magic Values

The Vault contains 5 Magic Values, which are XORed with a special Vault Constant Value as part of a substitution cipher:

- **HeaderStart:** Mandatory Value existing in the Beginning of the File.
- **HeaderPad:** Mandatory Value existing after the Ending of the Header JSON.
- **HeaderEnd:** Mandatory Value existing after the last amount of buffer for the header.
- **FooterStart:** Non-Mandatory Value existing after the last File in the Vault.
- **FooterEnd:** Non-Mandatory Value existing after the Ending of the Footer JSON.

These Values shall not exist more than once in the Vault, for at least the first Three. As they will directly promotes lack of practicality for Vault if found in incorrect place, as simply the Vault will indicate that it couldn't decrypt the given Header. However, these are not likely to corrupt the Vault as they are located in the very beginning of the file. Furthermore, the other Values can exist more than once, and the Vault is able to handle an incorrect Footer.

⁹Base64 is a group of binary-to-text encoding schemes that transforms binary data into a sequence of printable characters, limited to a set of 64 unique characters.

4.3.7 Decryption

The Decryption follows the reversal of encryption using the original provided password. We must note that when changing the Password of the Vault, it follows the '*Emergency Shutdown*' protocol, where data must be re-encrypted incase of abrupt closure. However, the Vault remains fully encrypted during operation and password change. This is illustrated in Figure 4.4.

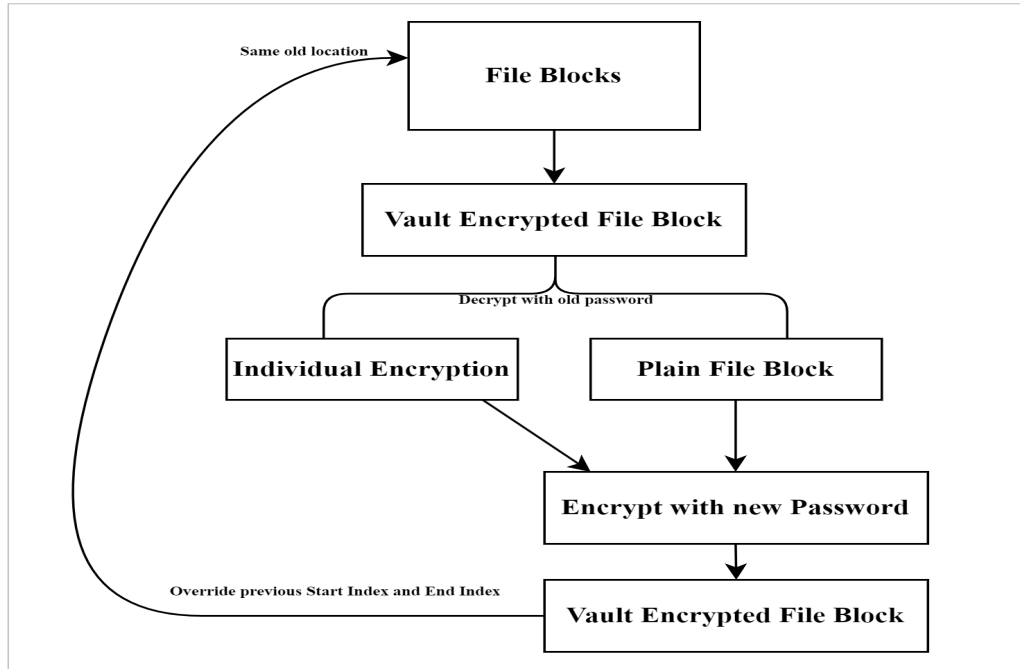


Figure 4.4: Password Change Logic

Therefore, the Vault remains fully encrypted during the duration of password change. For tokens, these bytes are converted from a Base64 encoded string. It follows the reversal of generating tokens and results in the same original password:

```

1  def resolve_token(token : bytes) -> str:
2      tokenizer = [83, 101, 99, 117, 114, 101, 45, 68, 105, 103,
3                  105, 116, 97, 108, 45, 86, 97, 117, 108, 116]
4      cipher = xor_magic(''.join([xor_magic(chr(n)).decode() for
5                                  n in tokenizer])).decode()
6      token = decrypt_bytes(token, cipher)
7      return token.decode()

```

Code 4.5: Token Resolution

4.3.8 Logging

A custom Logger, designed as a Singleton¹⁰, is implemented for the Vault. This approach ensures efficiency within the application's context. Given that certain logs must persist throughout the session, a single Logger instance suffices to generate and store logs. Notably, these logs do not directly alter the Footer, a task reserved for the very end of the Vault's activity will handle that.

```
1     def __log(self, level:str, log_message:str, is_error:bool=False
      , no_session=False):
2         timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%
          M:%S")
3         log_message = f"{timestamp} [{level}]: Message: {
          log_message}"
4         try:
5             with self.__lock:
6                 if no_session:
7                     return log_message
8                 if is_error:
9                     self.__log_session.add_error_log(log_message)
10                else:
11                    self.__log_session.add_normal_log(log_message)
12        except Exception as e:
13            msg = f"{log_message} -
              FAILED_LOCK_ACQUISITION_EXCEPTION: {type(e).__name__
              } with message: {str(e)}"
14            self.__log_session.add_error_log(msg)
```

Code 4.6: Logging

The session class is a simple concatenation of logs separated by a new line character. It makes it easier to display them in the Vault settings.

This Logger is capable of sending a PyQt signal¹¹ to certain GUI elements, but its current usage is only to send a signal to the GUI in order to display a Popup Window for 3 seconds that a log event has occurred.

¹⁰The singleton pattern is a software design pattern that restricts the instantiation of a class to a singular instance.

¹¹Each PyQt widget, which is derived from QObject class, is designed to emit 'signal' in response to one or more events.

4.3.9 Signaling

Most of the signals handled in the Vault are related to Progress Bar updating, this way, it is possible to maintain an accurate progress bar when a resource or time consuming function is running. Usually, since these functions are handling in a separate thread, the function that thread takes always has the progressbar signal which uses it to emit a value to use for updating.

```
1 # Example Function
2 def __change_password(self, old_password : str, new_password :str,
    vault : Vault, progress_signal : pyqtSignal) -> None:
3     # Code
4     for i in range (0,10):
5         # Code
6         progress_signal.emit(10)
7     # End Loop
8     progress_signal.emit(100)
9
10 # Main Code
11 self.worker = Worker(self.__change_password, password, new_password
    , self.__vault)
12 self.worker.args += (self.worker.progress, )    # Force add signal
13 self.worker.progress.connect(self.update_progress_bar)
14
15 # Updater
16 def update_progress_bar(self, num_to_update_with : int) -> None:
17     if num_to_update_with == 100 or self.progress_bar.value() >=
        100:
18         self.progress_bar.stop_progress(False)
19         return
20     current_value = self.progress_bar.value()
21     if num_to_update_with + current_value >= 100:
22         self.progress_bar.setValue(99)
23     else:
24         self.progress_bar.setValue(num_to_update_with +
            current_value)
```

Code 4.7: Signaling Example

4.3.10 Threading and Mutable Values

One of the limitations with Python is proper multi-threading, the GIL¹² lock in Python can be challenging to deal with, thus, the multi-threading is less effective compared to languages without such limitations.

One solution could be is to use Python processes, however, threads are lighter than processes, and share the same memory space. The critical multi-threading functions have a need to share resources. QThread with special timers are used for operation. The Timers purpose is to give an allocated amount of time for the thread to run, if finished, then the interruption of the thread begins.

As for Mutable Values, Python does not offer Mutable Integers or Booleans, so instead of using lists to mitigate that, Objects to wrap the Integers and Booleans were created instead in the following way for better organization:

```
1 class MutableInteger:
2     def __init__(self, start_val: int = 0):
3         self.__int = start_val
4     def __str__(self) -> str:
5         return str(self.__int)
6     def get_value(self) -> int:
7         return self.__int
8     def set_value(self, value: int) -> None:
9         self.__int = value
10
11 class MutableBoolean:
12     def __init__(self, start_val: bool = True):
13         self.__bool = start_val
14     def __str__(self) -> str:
15         return str(self.__bool)
16     def get_value(self) -> bool:
17         return self.__bool
18     def set_value(self, value: bool) -> None:
19         self.__bool = value
```

Code 4.8: Mutable Classes

¹²Global Interpreter Lock (GIL) simply is a mutex (or a lock) that allows only one thread to hold the control of the Python interpreter.

4.4 Frontend implementation

As the user cannot directly add extra class attributes to the default QWidgets, multiple custom classes were derived, they hold additional fields, functions, and even may include other widgets.

4.4.1 Frontend challenges

These challenges occurred during the development of the Windows:

- **Decision not to use any Windows APIs:** For Vault search, it would be possible to use Windows APIs to display the Files or Folders on the system in a custom window, but this would introduce extra unnecessary imports just for one Window.
- **Documentation based on C++:** Most of the examples for any QWidget usage is done in C++, extra effort to correctly translate code to Python was necessary.
- **Deleting the underlying C++ Object:** This was carefully implemented in order to avoid the RuntimeError:

```
wrapped C/C++ object of type QWidget has been deleted
```

which in turn crashes the Application. After ending the usage of a Window, the custom `exit()` method was implemented to sweep up any QWidget objects related to the Window itself.

- **QTree:** The QTree model was used instead of a QList model to hold all the items (Folders,Files) in display. It may add extra RAM overhead, but the decision was to properly implement the visibility of data columns, and easier access for extra fields rather than using a QList.

The Frontend is split into two parts, Managers, and SubWindows.

4.4.2 Managers

Vault Create Window, Vault Search Window, and the Main Window 'VaultView' Window. The first two can only be shown before the VaultView.

The Vault Create Window, introduces the user with the options to have a Vault formed, and one of the key features to present is that the Password must be of a good quality, mostly can be summed to:

- Password length should be at least 6 characters
- Password should contain at least one uppercase letter
- Password should contain at least one lowercase letter
- Password should contain at least one digit
- Password should contain at least one special character

This way, its more difficult to guess the password. Certainly, based on that, the token creation can be done.

The Vault Search Window, can use a secondary Thread to be able to detect a certain file with a specific extension, it cannot be spammed as the button gets locked until the time limit for the thread ends. This time limit is 10 seconds.

4.4.3 VaultView

This Window is the one which contains a the 'Vault' class itself as a field, these fields are name mangled¹³ to at least provide some sort of encapsulation.

This View, and the Settings View are responsible for directly modifying the Vault object with modify functions. No secondary threads exist for extra help with these as they are called in different Windows or Views. Its worth noting, that this View is not modal ¹⁴ because its the Parent Window, but the Settings View is.

¹³Mangling is used for class attributes that one does not want subclasses to use which are designated as such by giving them a name with two or more leading underscores and no more than one trailing underscore.

¹⁴A modal window creates a mode that disables user interaction with the main window but keeps it visible.

4.4.4 Dialogs and Windows

Dialogs are a form of a default modal interaction type of Window, the application uses the following custom ones:

- **FindFileDialog:** It forces the user to start the search, or click on the close button to have it fully closed.
- **InteractDialog:** This dialog is used to get the password of an individually encrypted file.
- **LogExtractDialog:** The purpose to keep the extraction modal is not to allow any interaction with the Settings Window.
- **NoteDialog:** A modal dialog to use the available options regarding Notes.

These are not precisely data modifying sort of interactions, except for NoteDialog, but as per the design decision its recommended to have at least some Modal Window types to prevent every single possible Window of being open. However, the user is capable of Adding Files, Extracting Files, and Viewing a certain File at the same time. Every action will trigger its respective secondary Thread, and no freezing shall occur in this heavy scenario.

Popup Window only shows up when a log event is registered, its lifetime is destroyed once the Vault is closed. It is unnecessary to have this QWidget to be constantly recreated in the background whenever a log is created.

Add, Delete, Get Windows these windows prevent certain interaction when under use, the reason is not to allow any concurrent data modification. For example, deleting a File that exists within the list of the 'Get' files to be extracted. As this would not result in an undefined behavior, but rather, it will simply make the Vault fail to extract that current element.

Abortion of certain operations can be done, such as Add, Delete. These operations are resource heavy, therefore, if too much time is ought to be taken, its possible to abort without any Vault modification as the mechanism is integrated

within the function. A mutable boolean in such scenario is passed to the given function, and before any critical section, its checked first. The following example can show both ways of using "Don't Abort" and "Abort":

```
1     def __process_import(self, selected_items : list[tuple[str, str
2         ],
3
4         continue_running : MutableBoolean,
5         recursions : int, signal : pyqtSignal,
6         call_num : MutableInteger = MutableInteger
7             (0), id_to_insert_into : int = 0):
8
9     # Handle ID REMOVAL INCASE OF ABORT
10    if not continue_running.get_value():
11        self.parent().remove_folder_without_files(
12            id_to_insert_into) # This id valid, since in the
13                               next recursion abort may be turned on.
14        return None
15
16    # Progress bar details
17    progress_increase = 0
18    if recursions < 1:
19        progress_increase = 100
20    else:
21        progress_increase = floor(100 / recursions)
22
23    amount_of_files = 0
24    # Go through dirs:
25    for folder in selected_items:
26        if continue_running.get_value() and folder[0] == "
27            Folder":
28                # Code
29
30            # Code
31
32            # Code
33        # Go through files
34        for file in selected_items:
35            if not continue_running.get_value():
36                return None
```

Code 4.9: Abort During Import Example

The structure of all GUI elements, and utility functions allow for future modifications. Furthermore, there are no custom interfaces, or regular Python Base Class

representatives to inherit from. Even though, it is definitely possible to implement and update the source code, the required effort to do that will not pose any significant improvement over the functionality of the Vault and readability of the code.

4.5 Technical limitations

Its not to deny that there are certain limitations in the Vault that are considered as bugs and actual limitations, In this section, every known limitation will be discussed and what sort of possible improvements can be considered for development. Most of the limitations showed up during testing of the Vault, and some others due to the time constraint.

4.5.1 Interaction Based

- **Concurrent Vault File Modification:** If any modifying function such as Add, Encrypt, or Decrypt for example are used, then these functions do not lock the Vault File itself during byte shift or byte override. In the case of only byte override where no shifting is involved, its completely valid not to lock the Vault File. However, any other function which shifts the bytes can have a major effect on the Vault, thus, possibly corrupting the Vault.
- **Huge Log Size:** During the lifetime of the Vault, **WARNING**, **ATTENTION**, and **ERROR** logs can be generated. As of the release, there is no user-friendly way to pause the savings of the logs into the footer. Therefore, the longer the Vault exists, the greater the size of the footer gets.
- **Unrecoverable Item:** An item can become unrecoverable if and only if its location bytes are corrupted, or, its individually encrypted and the password is forgotten. In this scenario, the only way to get rid of the item is by deleting it. However, its definitely possible to create extra tokens for this item, but it would take more development time.
- **Delete:** Testing has detected that the delete function at the time of release poses a significant risk of Vault corruption, where incorrect byte shifting occurs and an invalid Header update happens. The function is able to delete the bytes,

but not reliably. This may be due not to locking the Vault File during deletion so no File Descriptors exist other than the one overriding and shifting the bytes to delete. It is immediately detectable if a file is corrupted when its Icon is not showing the in the Vault. Luckily enough, this does not result in a chain reaction effect for every single file existing in the Vault. Therefore, at the time of release, at least the removal button of a note from a file is disabled.

- **Forcing a Hint:** By default, a hint for the password is forced to be existent. Since it lies after the footer in a special XOR-ed value, it cannot be rid of unless the Footer is corrupted or destroyed. An attacker can extract this value and try to narrow down correct password search criteria.

4.5.2 Security

- **Possible Memory Leakage in Python due to PyQt:** It is worth to make heavy inspections with professional tools like GammaRay[18] to ensure there is no memory leakage, but in the current implementation, this does not pose a high risk.
- **Dictionary Attack¹⁵:** The Vault is not resilient to a regular dictionary attack, as there is no wait time after inputting the incorrect password, but at least a modal popup window shows. An attacker can still create a script to automate the process, but regardless, it makes it difficult to guess that the password is always greater than or equal to six characters.
- **Actual Password:** is saved in memory as a plain text, it is worth to save it encrypted with a special cached value, but in this scenario it does not pose any security threat as it will not be in the memory unless the user actually knows the password.

4.6 Improvements

The current notable improvements to be added for the Vault will be summed in this section. These improvements are the ones which are known at the initial release.

¹⁵A dictionary attack is based on trying all the strings in a pre-arranged listing.

4.6.1 GUI

Additional items of interaction can be introduced for the users, this will promote Vault customization and usability. Items to consider:

- **Logs:** More log level options to consider, as well as, a way to sort them by level or delete them.
- **Colorful Gui:** Representation of viewing the file could be more user friendly.
- **Additional Functions:** Test for any extra bytes which can be deleted, rename files or folders, and more in depth analyzing of folder contents.
- **Accessibility:** A set of custom interactions for any visually impaired user to be able to use the Vault.
- **File or Folder View Columns:** By request, add more data about the File to be saved.
- **Note Display:** Display the Note to the user instead of forcing them to extract it.
- **Find:** More options to search by, such as searching only for Folder names, or search by data creation or modification.

4.6.2 Storage Optimization

Some optimization can be done, and won't pose significant overhaul of logic:

- **Header Size Optimization:** Currently by default, the initial header size is approximately 300 bytes. Adding items into the header, will increase its size. Each addition of a File dictionary is estimated to be 130 bytes, and a Folder dictionary is estimated to be 100 bytes. Therefore, this increases the size of the Header significantly overtime. Comparing it to WinRAR:
 - File: approximate of 40 bytes extra bytes when addition happens.
 - Folder: approximate of 35 bytes when adding a plain directory.

An example of a heavily optimized entry for the Header can be as follows:


```
1      {  
2          "1" : "17101055*17100810*0x6cxf9dds*0*4*(Mahalo x  
              DLMT (feat. Lily Denning) - So Cold)*mp3  
              *1710081055*1710081069*-1"  
3      }
```

Code 4.10: Optimized Header Entry

This string can be split using the delimiter ‘*’. The components are as follows:

- **First two values (17101055*17100810):** Indicate where the file exists.
- **Third Value (0x6cxf9dds):** Checksum.
- **Fourth Value (0):** 1 or 0 representing whether it is individually encrypted.
- **Fifth Value (4):** Folder Path it belongs under.
- **Sixth Value (Mahalo x DLMT ft. Lily - So Cold):** File name.
- **Seventh Value (mp3):** File extension.
- **Next two values (1710081055*171008106):** Indicate where the icon of the file exists, this can be -1 for both values as well.
- **Last value:** Attached Note ID.

The main advantage of this is that it will approximately decrease the to 50 bytes per File, thus, providing a chance to compete against WinRAR’s optimization.

- **Footer Size Optimization:** The Footer can be filtered further, there’s no need to store every WARNING Log permanently, further customization options can be implemented for what to save and what to ignore.
- **Icon Saving:** All File icons are saved in the Vault, its redundant data, and can be mitigated by using default icons for extensions.

Chapter 5

Testing

Testing was conducted in two ways, Unit Tests (Automatic Testing) for Vault functionality, and End-To-End Tests for User Interface (Manual Testing).

5.1 Unit Tests

The total amount of these tests are: 109, and they can be all executed at once by running the included Powershell script:

```
./run_tests.ps1
```

The report will cover all available unit test cases, which are primarily focused on the utility functions. Given the complexity of extracting specific integrated functions from the GUI, a combination of automatic and manual testing is required. Consequently, the utility functions were tested automatically.

Figure 5.1 shows a sample report generated by the test tool.

```
===== test session starts =====
platform win32 -- Python 3.12.0, pytest-8.2.1, pluggy-1.5.0
rootdir: D:\Thesis
plugins: html-4.1.1, metadata-3.1.1
collected 109 items

Secure-Digital-Vault\tests\classes\directory_test.py ..... [ 48%]
Secure-Digital-Vault\tests\classes\file_test.py ..... [ 53%]
Secure-Digital-Vault\tests\classes\note_test.py ..... [ 58%]
Secure-Digital-Vault\tests\classes\vault_test.py ..... [ 63%]
Secure-Digital-Vault\tests\crypto\decryptors_test.py ..... [ 68%]
Secure-Digital-Vault\tests\crypto\encryptors_test.py ..... [ 73%]
Secure-Digital-Vault\tests\crypto\utils_test.py ..... [ 78%]
Secure-Digital-Vault\tests\custom_exceptions\classes_exceptions_test.py ..... [ 83%]
Secure-Digital-Vault\tests\custom_exceptions\utils_exceptions_test.py ..... [ 88%]
Secure-Digital-Vault\tests\file\handle_file_test.py ..... [ 93%]
Secure-Digital-Vault\tests\logger\log_session_test.py ..... [ 98%]
Secure-Digital-Vault\tests\logger\logging_test.py ..... [100%]
Secure-Digital-Vault\tests\utils\helpers_test.py ..... [ 82%]
Secure-Digital-Vault\tests\utils\id_gen_test.py ..... [ 88%]
Secure-Digital-Vault\tests\utils\parsers_test.py ..... [ 96%]
Secure-Digital-Vault\tests\utils\serialization_test.py ..... [100%]

===== Generated html report: file:///D:/Thesis/report.html =====
===== 109 passed in 0.05s =====
```

Figure 5.1: Unit Tests Execution

report.html

Report generated on 23-May-2024 at 19:39:19 by [pytest-html](#) v4.1.1

Environment

Python	3.12.0
Platform	Windows-11-10.0.22631-SP0
Packages	<ul style="list-style-type: none">pytest: 8.2.1pluggy: 1.5.0
Plugins	<ul style="list-style-type: none">html: 4.1.1metadata: 3.1.1
JAVA_HOME	C:\Program Files\Java\jdk-17.0.2

Summary

109 tests took 284 ms.

(Un)check the boxes to filter the results.

☒ 0 Failed,

☒ 109 Passed,

☒ 0 Skipped,

☒ 0 Expected failures,

☒ 0 Unexpected passes,


☒ 0 Errors,

☒ 0 Reruns

Show all details

/

Hide all details

Result 	Test	Duration	Links
Passed	Secure-Digital-Vault/tests/utis/serialization_test.py::test_deserialize_dict	0 ms	
Passed	Secure-Digital-Vault/tests/utis/serialization_test.py::test_serialize_dict	0 ms	
Passed	Secure-Digital-Vault/tests/utis/serialization_test.py::test_formulate_footer	0 ms	
Passed	Secure-Digital-Vault/tests/utis/serialization_test.py::test_formulate_header	1 ms	
Passed	Secure-Digital-Vault/tests/utis/parsers_test.py::test_get_last_folder	0 ms	
Passed	Secure-Digital-Vault/tests/utis/parsers_test.py::test_remove_trailing_slash	0 ms	
Passed	Secure-Digital-Vault/tests/utis/parsers_test.py::test_show_as_windows_directory	0 ms	
Passed	Secure-Digital-Vault/tests/utis/parsers_test.py::test_parse_file_name	1 ms	
Passed	Secure-Digital-Vault/tests/utis/parsers_test.py::test_parse_directory_string	0 ms	
Passed	Secure-Digital-Vault/tests/utis/parsers_test.py::test_parse_timestamp_to_string	0 ms	
Passed	Secure-Digital-Vault/tests/utis/parsers_test.py::test_parse_from_string_to_size	0 ms	
Passed	Secure-Digital-Vault/tests/utis/parsers_test.py::test_parse_size_to_string	0 ms	
Passed	Secure-Digital-Vault/tests/utis/parsers_test.py::test_parse_json_safely	0 ms	
Passed	Secure-Digital-Vault/tests/utis/id_gen_test.py::test_gen_id_near_limit	138 ms	
Passed	Secure-Digital-Vault/tests/utis/id_gen_test.py::test_gen_id_large_values	2 ms	
Passed	Secure-Digital-Vault/tests/utis/id_gen_test.py::test_gen_id_with_clashes	1 ms	
Passed	Secure-Digital-Vault/tests/utis/id_gen_test.py::test_gen_id_with_gaps	0 ms	
Passed	Secure-Digital-Vault/tests/utis/id_gen_test.py::test_gen_id_no_clashes	0 ms	
Passed	Secure-Digital-Vault/tests/utis/id_gen_test.py::test_gen_id_empty_list	0 ms	
Passed	Secure-Digital-Vault/tests/utis/helpers_test.py::test_force_garbage_collect	21 ms	
Passed	Secure-Digital-Vault/tests/utis/helpers_test.py::test_count_digits	1 ms	
Passed	Secure-Digital-Vault/tests/utis/helpers_test.py::test_get_file_size	1 ms	
Passed	Secure-Digital-Vault/tests/utis/helpers_test.py::test_is_proper_extension	1 ms	
Passed	Secure-Digital-Vault/tests/logger/logging_test.py::test_singleton_reset	0 ms	
Passed	Secure-Digital-Vault/tests/logger/logging_test.py::test_get_current_time	0 ms	

5.2 Manual Tests

It definitely is possible to automate the Manual tests via a comprehensive framework such as Squish[19]. However, due to the time constraint, manual user testing was conducted in order to ensure delivery of the basic functionality.

5.2.1 Vault Creation Window Test

Given	When	Then
The VaultCreateWindow is displayed	The User does the following: 1. Clicks on the 'Save' button	The data in the input fields becomes read-only
The VaultCreateWindow is displayed	The User does the following: 1. Clicks on the 'Edit' button	The data in the input fields becomes editable
The VaultCreateWindow is displayed	The User does the following: 1. Clicks on the 'Create' button	The vault creation process begins and the progress bar updates accordingly
The VaultCreateWindow is displayed	The User does the following: 1. Enters valid data into the input fields	The data is accepted and saved when the 'Save' button is clicked
The VaultCreateWindow is displayed	The User does the following: 1. Enters invalid data into the input fields	An error message appears indicating the validation errors
The VaultCreateWindow is displayed	The User does the following: 1. Enters a valid password in the password field, 2. Clicks on the 'Create' button	The vault is successfully created, and tokens are generated and saved
The VaultCreateWindow is displayed	The User does the following: 1. Enters an invalid password in the password field, 2. Clicks on the 'Create' button	An error message appears indicating that the password is not strong enough
The VaultCreateWindow is displayed	The User does the following: 1. Clicks on the 'Return' button	The application navigates back to the main menu

Table 5.1: E2E Testing Scenario: Vault Creation Window

5.2.2 Welcome Window Test

Given	When	Then
The ViewManager is displayed	The User clicks on the 'Find Vault' button	The VaultSearchWindow is shown
The ViewManager is displayed	The User clicks on the 'Create Vault' button	The VaultCreateWindow is shown

Table 5.2: E2E Testing Scenario: ViewManager Buttons

5.2.3 Vault Search Window Test

Given	When	Then
The VaultSearchWindow is displayed	The User clicks on the 'Detect' button	The application searches for a vault with the specified extension in the current directory
The VaultSearchWindow is displayed	The User clicks on the 'Insert' button	The application updates the tree view with the specified path
The VaultSearchWindow is displayed	The User clicks on the 'Reset' button	All fields (address bar, extension, tree view, password, location, token checkbox) are reset to their initial states
The VaultSearchWindow is displayed	The User clicks on the 'Import' button with valid inputs	The application attempts to decrypt the vault with the specified password and location
The VaultSearchWindow is displayed	The User clicks on the 'Return' button	The application returns to the main menu (ViewManager)

Table 5.3: E2E Testing Scenario: VaultSearchWindow Buttons

Given	When	Then
The VaultSearchWindow is displayed	The User clicks on the 'Detect' button	The application searches for a vault with the specified extension in the current directory
The VaultSearchWindow is displayed	The User clicks on the 'Insert' button	The application updates the tree view with the specified path
The VaultSearchWindow is displayed	The User clicks on the 'Reset' button	All fields (address bar, extension, tree view, password, location, token checkbox) are reset to their initial states
The VaultSearchWindow is displayed	The User clicks on the 'Import' button with valid inputs	The application attempts to decrypt the vault with the specified password and location
The VaultSearchWindow is displayed	The User clicks on the 'Return' button	The application returns to the main menu (ViewManager)
The VaultSearchWindow is displayed	The User enters a folder	The tree view navigates inside the selected folder
The VaultSearchWindow is displayed	The User clicks on the 'Token' checkbox	The password input field is enabled for entering a token password
The VaultSearchWindow is displayed	The User double clicks or presses enter on 'UpOneLevel'	The tree view navigates to the parent folder
The VaultSearchWindow is displayed	The User opens the drop down menu	The application displays the available drives

Table 5.4: E2E Testing Scenario: VaultSearchWindow Functionalities

5.2.4 Vault View Window Test

Given	When	Then
VaultViewWindow is initialized with a Vault object	Vault contains items	Items in the vault are displayed in a list within the window
VaultViewWindow is displayed	User clicks the "Add Item" button and inputs new item details	New item is added to the vault and the list of items in the window updates to include the new item
VaultViewWindow is displayed	User selects an existing item and clicks the "Edit" button, then changes item details	Selected item's details are updated in the vault and the changes are reflected in the list of items in the window
VaultViewWindow is displayed	User inputs search criteria in the search bar and presses enter	List of items in the window filters to show only the items matching the search criteria
VaultViewWindow is displayed	User double-clicks an item in the list	A detailed view of the selected item is displayed in a new window or pane
VaultViewWindow is displayed	User clicks on a column header to sort items	List of items in the window is sorted based on the selected column (e.g., by name, date added)
VaultViewWindow is initialized with an empty vault	Vault is empty	A message or indication is shown that there are no items in the vault

Table 5.5: E2E Testing Scenario: VaultViewWindow

5.2.5 Add Items Test

Given	When	Then
AddFileWindow is initialized	User inputs a path in the address bar and presses enter or clicks the insert button	Address bar is updated with the input path, and the tree view is updated with the contents of the selected path
AddFileWindow is initialized	User clicks the reset button	All fields in the window are reset to their initial state
AddFileWindow is initialized	User selects items in the tree view	Amount of content text field is updated with the number of selected folders and files
AddFileWindow is initialized	User clicks the import button	Progress bar shows the progress of the import operation
AddFileWindow is initialized	User clicks the abort button during import	Import operation is aborted
AddFileWindow is initialized	User clicks the return button	Window closes, and the VaultView window is displayed

Table 5.6: E2E Testing Scenario: AddFileWindow Interactable Items

5.2.6 Extract Items Test

Given	When	Then
GetFileWindow is initialized	User inputs a path in the address bar and presses enter or clicks the extract button	Address bar is updated with the input path, and the list widget is updated with the contents of the selected path
GetFileWindow is initialized	User clicks the extract button	Progress bar shows the progress of the extraction operation
GetFileWindow is initialized	User clicks the abort button during extraction	Extraction operation is aborted
GetFileWindow is initialized	Extraction operation completes	Files are extracted to the specified location on disk

Table 5.7: E2E Testing Scenario: GetFileWindow Interactable Items

5.2.7 View Items Test

Given	When	Then
Initialization		
ViewFileWindow is initialized with a parent VaultView and a CustomQTreeWidgetItem	ViewFileWindow is instantiated with the specified parameters	Buttons are updated based on the file meta-data and size
Encryption		
A file exceeding CHUNK_LIMIT is displayed	File size exceeds CHUNK_LIMIT	Encryption and decryption buttons are disabled with context informing the user about the file size limit
User clicks the Encrypt button after entering a strong password	User clicks the Encrypt button and enters a strong password	Encryption process begins, progress bar shows progress
User clicks the Decrypt button after entering the correct decryption password	User clicks the Decrypt button and enters the correct decryption password	Decryption process begins, progress bar shows progress
User clicks the Encrypt button after entering a weak password	User clicks the Encrypt button and enters a weak password	User is notified about the weak password, encryption process does not start
User clicks the Decrypt button after entering the incorrect decryption password	User clicks the Decrypt button and enters an incorrect decryption password	User is notified about the incorrect password, decryption process does not start
Note Management		
User clicks the Add Note button	User clicks the Add Note button	NoteDialog is opened to add a note to the file
User clicks the Get Note button	User clicks the Get Note button	NoteDialog is opened to display the existing note for the file

Table 5.8: E2E Testing Scenario: ViewFileWindow Functionality

5.2.8 Find Items Test

Given	When	Then
Search Parameters		
User inputs a string to find in file names	User enters a string in the "String to find" field	String is captured for searching
User inputs an extension type to find	User enters an extension type in the "Extension type to find" field	Extension is captured for searching
User selects the "Match string case" checkbox	User checks the "Match string case" checkbox	Case-sensitive search is enabled
User selects the "Encrypted" checkbox	User checks the "Encrypted" checkbox	Encrypted file search is enabled
User selects the "Note Attached" checkbox	User checks the "Note Attached" checkbox	Files with attached notes search is enabled
Search Action		
User clicks the Search button	User clicks the "Search" button	Search for files based on the provided parameters is initiated
Error Handling		
Invalid extension provided	User provides an invalid extension and the "Encrypted" checkbox is not checked	User is notified about the invalid extension

Table 5.9: E2E Testing Scenario: FindFileDialog Functionality

5.2.9 Note Test

Given	When	Then
Add Note		
User inputs the location and extension for the note file	User enters the file location and extension	Location and extension are captured for adding a note
User clicks the Import button after providing a valid extension, matching extension in the file location, and a file size within the limit	User clicks the "Import" button	Note is added to the vault with the provided location and extension
User inputs an invalid extension	User enters an invalid extension	User is notified about the incorrect extension
User inputs a file location with an extension that does not match the provided extension	User enters a file location with a mismatched extension	User is notified about the mismatch between the extension and file location
User inputs a file location with an invalid size exceeding the note limit	User enters a file location with a size exceeding the note limit	User is notified about the oversized file
Get Note		
User inputs the location for retrieving the note	User enters the file location	Location is captured for retrieving the note
User clicks the Get button after providing a valid file location	User clicks the "Get" button	Note is retrieved from the vault for the provided location
User inputs an invalid file location	User enters an invalid file location	User is notified about the invalid file location

Table 5.10: E2E Testing Scenario: NoteDialog Functionality

5.2.10 Settings Test

Given	When	Then
Change Vault Details		
User provides a new name and extension for the vault	User enters a new name and extension	New vault name and extension are captured
User provides an invalid extension	User enters an invalid extension	User is notified about the invalid extension
User provides a valid new name and extension, and optionally, a new password and hint	User enters valid new details	Details are saved for future execution
User saves the vault details	User clicks the "Save" button	Details are saved for potential execution
User clicks the "Edit" button after saving the details	User clicks the "Edit" button	Details editing mode is activated
Change Password		
User provides the old and new passwords along with an optional hint	User enters old and new passwords	Password change process begins
User provides incorrect old password	User enters incorrect old password	User is notified about the incorrect password
User provides a weak new password	User enters a weak new password	User is notified about the weak password
User provides the same old and new passwords	User enters the same old and new passwords	User is notified to use a different password
User saves the password changes	User clicks the "Execute" button	Password change process is executed
Vault Information Logs		
User sorts the logs in ascending order	User clicks the "Sort" button	Logs are sorted in ascending order
User sorts the logs in descending order	User clicks the "Sort" button again	Logs are sorted in descending order
User extracts the logs	User clicks the "Extract" button	Logs are extracted from the vault

Table 5.11: E2E Testing Scenario: SettingsWindow Functionality

5.3 Additional Testing

For extended testing, given a larger time frame, several different types of testing is possible.

5.3.1 Security Testing

It's possible to check whether the Vault executable generated by the source can be detected by virus scanners as a virus.

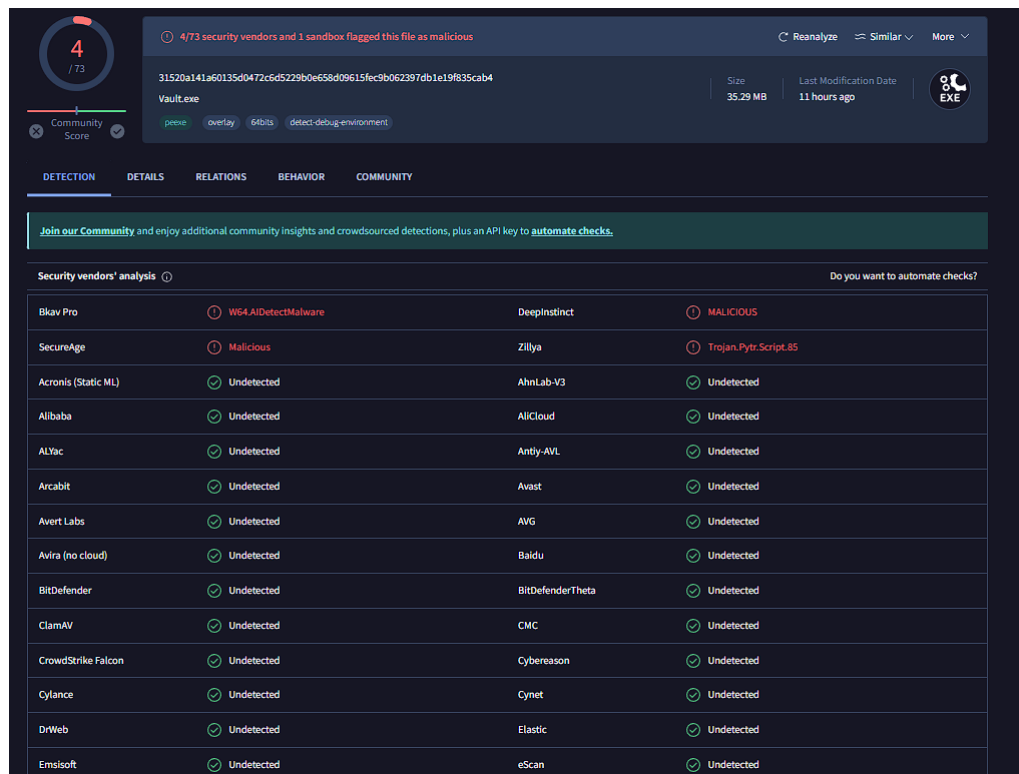


Figure 5.2: Vault Marking

The Vault itself is not a virus its detection by virus scanners reason could be because of how the executable was generated. More security testing frameworks can be used, such as **Metasploit Framework**[20].

5.3.2 Performance Testing

It's possible to measure the time taken for each operation by using timers. Additionally, monitoring CPU and disk utilization during Vault file processing is feasible. One example of an application framework that can accomplish this is **Prometheus**[21].

Chapter 6

Conclusion

The main purpose of the project behind the scenes was to create an exercise that integrates as many university courses as possible. These courses include, but are not limited to, Concurrency, Graphical User Interface Design, Software Technology, Python Programming, Algorithms and Data Structures, Object-Oriented Programming, and more.

As for the motivation behind the project stems from an unfortunate ransomware attack that occurred approximately a decade ago. An attempt to update Windows drivers using DriverEasy[22] resulted in the full encryption of a personal disk drive, leading to the loss of personal and memorable data. While this application is nowhere near being actual ransomware¹, the idea of allowing users the freedom to encrypt and decrypt their files remained in mind. Additionally, there was a desire for a comprehensive learning experience in GUI programming, and architectural design, which contributed to the development of this idea.

The source code will remain public, and anyone interested in learning from it or suggesting improvements is always welcome. While the project may not be perfect, it effectively conveys its educational purpose and demonstrates the challenges that can arise from even a seemingly simple idea.

¹Ransomware is a type of cryptovirological malware that permanently blocks access to the victim's personal data unless a ransom is paid.

Bibliography

- [1] Seagate Technology LLC. “Everything You Want to Know About Hard Drives”. In: *What Does a Hard Drive Do?* (2024). URL: <https://www.seagate.com/blog/everything-you-wanted-to-know-about-hard-drives-master-dm>.
- [2] Intel Corporation. “What is RAM?” In: *What Is RAM?* (2024). URL: https://www.intel.com/content/www/us/en/tech-tips-and-tricks/computer-ram.html#articleparagraph_cop.
- [3] The Internet Society (2001). “The BSD syslog Protocol”. In: *The BSD syslog Protocol Memo* (2001). URL: <https://www.rfc-editor.org/rfc/rfc3164>.
- [4] IETF Trust. “The Syslog Protocol”. In: *The Syslog Protocol* (2009). URL: <https://datatracker.ietf.org/doc/html/rfc5424>.
- [5] Spy++ Developers. “Download Spy++”. In: *graphical view of the system’s processes* (2022). URL: <https://learn.microsoft.com/en-us/visualstudio/debugger/introducing-spy-increment?view=vs-2022>.
- [6] win.rar GmbH. “File archiver utility”. In: *RAR Products* (1995). URL: <https://www.win-rar.com>.
- [7] Igor Pavlov. “File archiver with a high compression ratio”. In: *File archiving* (1997). URL: <https://www.7-zip.org>.
- [8] The Qt Company. “Qt Framework”. In: *Qt development framework* (2024). URL: <https://doc.qt.io/qt-6>.
- [9] The Qt Company. “Python Bindings for Qt”. In: *Design GUI with Python* (2024). URL: <https://www.qt.io/qt-for-python>.
- [10] Python Developers. “Metasploit”. In: *Python Documentation* (2024). URL: <https://docs.python.org/3.12>.

- [11] Winternals Software Microsoft. “Process Explorer”. In: *Process Explorer Introduction* (2023). URL: <https://learn.microsoft.com/en-us/sysinternals/downloads/process-explorer>.
- [12] David Goodger. “Docstring Conventions”. In: *PEP 257 – Docstring Conventions* (2001). URL: <https://peps.python.org/pep-0257>.
- [13] NIST. “ADVANCED ENCRYPTION STANDARD”. In: *ENCRYPTION STANDARD* (2023). URL: <https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf>.
- [14] Wiki. “Cipher Block Chaining”. In: *CBC Mode* (2024). URL: [https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Cipher_block_chaining_\(CBC\)](https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Cipher_block_chaining_(CBC)).
- [15] B. Kaliski. “PKCS7: Cryptographic Message Syntax”. In: *Cryptographic Message Syntax* (1998). URL: <https://datatracker.ietf.org/doc/html/rfc2315>.
- [16] NIST. “SHA-256”. In: *SHA-256 Hashin* (2015). URL: <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>.
- [17] OWASP. “Testing for Weak Cryptography”. In: *Testing for Padding Oracle* (2024). URL: https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/09-Testing_for_Weak_Cryptography.
- [18] KDAB. “Qt Inspection”. In: *Examine and manipulate the internals of Qt applications at runtime* (2024). URL: <https://www.kdab.com/development-resources/qt-tools/gammaray>.
- [19] Qt. “Qt Gui Testing”. In: *Qt Gui Test Automation* (2024). URL: <https://www.qt.io/product/quality-assurance/squish/platform-qt-gui-test-automation>.
- [20] rapid7 metasploit. “Metasploit”. In: *Penetration testing framework* (2024). URL: <https://www.metasploit.com>.
- [21] Prometheus Authors. “Prometheus”. In: *open-source monitoring solution* (2024). URL: <https://prometheus.io>.
- [22] Driver Easy. “How to UPDATE YOUR DRIVERS”. In: *What is Driver Easy* (2024). URL: <https://www.drivereasy.com>.

List of Figures

2.1	Vault Creation Window	8
2.2	Ready Vault Creation Window	9
2.3	Vault Search Hint	10
2.4	Vault Search Hint	10
2.5	Vault Search Window With Token	11
2.6	Vault Search Window Without Token	11
2.7	Right Click on Any Button	12
2.8	Plain Empty Vault	13
2.9	Vault With Items	13
2.10	Add Content Window	14
2.11	Add File Window	15
2.12	Regular Vault View After Addition	16
2.13	Viewing a File	16
2.14	Encrypting a File	17
2.15	Decrypting an Encrypted File	17
2.16	Add a Note	18
2.17	Get a Note	18
2.18	Extracting Files or Folder	19
2.19	Extracting Files or Folder	19
2.20	Find File View	20
2.21	Vault Settings	21
2.22	Vault General Information	21
2.23	Vault Change Password	22
2.24	Vault Log Sorting	23
2.25	Vault Logs Extraction	23
2.26	Vault Byte Checks	24
2.27	Vault Popup Notification	25

2.28	Moving or Copying a Vault	25
2.29	AntiVirus marking the Vault	26
3.1	Main Structure	29
3.2	Header Structure	30
3.3	File Structure	32
3.4	Directory Structure	33
3.5	Note Structure	34
3.6	Footer Structure	35
3.7	Vault View	36
3.8	7zip interface	37
3.9	Item View Window	38
3.10	Individual Layer Encryption	39
3.11	Find File View	39
3.12	Settings Window	40
3.13	Possible User Actions	41
4.1	Main Structure of the Qt Windows	46
4.2	Main Packages	47
4.3	Code Structure	48
4.4	Password Change Logic	53
5.1	Unit Tests Execution	65
5.2	Vault Marking	76

List of Tables

5.1	E2E Testing Scenario: Vault Creation Window	67
5.2	E2E Testing Scenario: ViewManager Buttons	68
5.3	E2E Testing Scenario: VaultSearchWindow Buttons	68
5.4	E2E Testing Scenario: VaultSearchWindow Functionalities	69
5.5	E2E Testing Scenario: VaultViewWindow	70
5.6	E2E Testing Scenario: AddFileWindow Interactable Items	71
5.7	E2E Testing Scenario: GetFileWindow Interactable Items	71
5.8	E2E Testing Scenario: ViewFileWindow Functionality	72
5.9	E2E Testing Scenario: FindFileDialog Functionality	73
5.10	E2E Testing Scenario: NoteDialog Functionality	74
5.11	E2E Testing Scenario: SettingsWindow Functionality	75

List of Codes

4.1	Pixmap extraction in Python	44
4.2	Simple Vault	48
4.3	Override Bytes in File	50
4.4	Token Generation	52
4.5	Token Resolution	53
4.6	Logging	54
4.7	Signaling Example	55
4.8	Mutable Classes	56
4.9	Abort During Import Example	60
4.10	Optimized Header Entry	64