**UNIVERSITI TEKNOLOGI MARA**

**DAFFMAN: HARDENING NETWORK-ATTACHED STORAGE SYSTEMS USING AES ENCRYPTION AND MD5 HASHING ALGORITHM**

**MUHAMMAD DANIAL BIN ASMAWI**

**BACHELOR OF COMPUTER SCIENCE (Hons.) COMPUTER NETWORKS**

**JULY 2022**

**UNIVERSITI TEKNOLOGI MARA**

**DAFFMAN: HARDENING NETWORK-ATTACHED STORAGE SYSTEMS USING AES ENCRYPTION AND MD5 HASHING ALGORITHM**

**MUHAMMAD DANIAL BIN ASMAWI**

**Thesis submitted in fulfilment of the requirements for Bachelor of Computer Science (Hons.) Computer Networks**

**College of Computing, Informatics & Media**

**JULY 2022**

# SUPERVISOR APPROVAL

**APPLICATION OF AES FILE ENCRYPTION AND MD5 CHECKSUM VERIFICATION FOR NETWORK-ATTACHED STORAGE SYSTEM**

By

**MUHAMMAD DANIAL BIN ASMAWI**

**2021151647**

This thesis was prepared under the supervision of the project supervisor, Supervisor’s Name. It was submitted to the Faculty of Computer and Mathematical Sciences and was accepted in partial fulfilment of the requirements for the degree of Bachelor of Computer Science (Hons.) Computer Networks.

Approved by

.............................................

Shahadan bin Saad

Project Supervisor

8th February, 2023

# STUDENT DECLARATION

I certify that this thesis and the project to which it refers is the product of my ownwork and that any idea or quotation from the work of other people, published or otherwise are fully acknowledged in accordance with the standard referring practicesof the discipline.

....................................................

MUHAMMAD DANIAL BIN ASMAWI

2021151647

8th February, 2023

# ACKNOWLEDGEMENT

I would like to express my gratitude to Allah swt for the completion of this project through the previous two semesters. I would also like to thank the supercisor of this project, Encik Shahadan bin Saad for his consistent and thorough guidance and teachings needed from the beginning of the project. And also credits to those who have contributed to this project directly and indirectly; the Final Year Project lecturer, Pn. Syafnidar Abdul Hamid, my parents, and my friends who are also completing their own project.

# ABSTRACT

Network-attached storage has played many crucial roles in most of today’s network. Despite the wide usage, there are still many issues encountered, especially issues related to file security and integrity, such as man-in-the-middle attack or packet sniffing attack. To solve these issues, a project is made with two objectives; to design a web-based network-attached storage system and to evaluate its reliability in terms of security and performance. From this project, DAFFMAN NAS system is produced. DAFFMAN system is a small-scale network-attached storage solution, constructed using only Python and Python Flask web framework for server-side. The main purpose of DAFFMAN is to fortify small-scale network-attached storage (NAS) systems with AES encryptions to prevent possible cyber attacks on the NAS system. DAFFMAN is also designed so that the system is highly accessible to all as long as the device used is connected to the local area network used for DAFFMAN deployment, either by wired or wireless connections. The usage of DAFFMAN has rendered the files unreadable and unusable when stored in the server, meaning that no modifications can be performed on the files until the files are decrypted by its owner from the client device used. DAFFMAN system can be improved in the future so that the system can be expanded in larger-scale networks and able to handle larger files more efficiently.

# TABLE OF CONTENTS

**CONTENT**

[SUPERVISOR APPROVAL iii](#_Toc126700699)

[STUDENT DECLARATION iv](#_Toc126700700)

[ACKNOWLEDGEMENT v](#_Toc126700701)

[ABSTRACT vi](#_Toc126700702)

[TABLE OF CONTENTS vii](#_Toc126700703)

[LIST OF FIGURES x](#_Toc126700704)

[LIST OF TABLES xi](#_Toc126700705)

[LIST OF ABBREVIATIONS xii](#_Toc126700706)

[CHAPTER 1 1](#_Toc126700707)

[INTRODUCTION 1](#_Toc126700708)

[1.1 Project Background 1](#_Toc126700709)

[1.2 Problem Statement 3](#_Toc126700710)

[1.3 Project Goals 4](#_Toc126700711)

[1.4 Objectives 4](#_Toc126700712)

[1.5 Project Scope 4](#_Toc126700713)

[1.5.1 System Deployment 4](#_Toc126700714)

[1.5.2 End Users 5](#_Toc126700715)

[1.5.3 Encryption and Hashing Algorithm 5](#_Toc126700716)

[1.6 Project Significance 5](#_Toc126700717)

[1.7 Chapter Summary 6](#_Toc126700718)

[CHAPTER 2 7](#_Toc126700719)

[LITERATURE REVIEW 7](#_Toc126700720)

[2.1 Network-Attached Storage Systems 7](#_Toc126700722)

[2.2 Data Security And Integrity 7](#_Toc126700723)

[2.2.1 Hashing and Message-Digest 5 (MD5) Algorithm for File Checksum 7](#_Toc126700724)

[2.2.2 Advanced Encryption Standard (AES) Algorithm 8](#_Toc126700725)

[2.2.3 Avalanche Effects 8](#_Toc126700726)

[2.3 Related Works 9](#_Toc126700727)

[2.3.1 Remotely Accessible, Low Power Network Attached Storage Device (Lanka & Garzevas, 2018) 9](#_Toc126700728)

[2.3.2 Home Server and NAS using Raspberry Pi (Shrivasta & Gadge, 2017) 11](#_Toc126700729)

[2.3.3 Research and Implement of an Encrypted File System Used to NAS (Huang et al., 2004) 11](#_Toc126700730)

[2.4 Chapter Summary 12](#_Toc126700731)

[CHAPTER 3 13](#_Toc126700732)

[METHODOLOGY 13](#_Toc126700733)

[3.1 Project Phases and Tasks 13](#_Toc126700735)

[3.2 Detailed Project Phases 17](#_Toc126700736)

[3.2.1 Project Timeline and Gantt Chart 17](#_Toc126700737)

[3.2.2 Project Planning 18](#_Toc126700738)

[3.2.3 Information Gathering 18](#_Toc126700739)

[3.2.4 System Design 20](#_Toc126700740)

[3.2.5 System Development 23](#_Toc126700741)

[3.2.6 System Testing 24](#_Toc126700742)

[3.2.7 Documentation 24](#_Toc126700743)

[3.3 Chapter Summary 25](#_Toc126700744)

[CHAPTER 4 26](#_Toc126700745)

[ANALYSIS AND DISCUSSIONS 26](#_Toc126700746)

[4.1 Performance Testing: Hashing Speed on Client-side Browsers 26](#_Toc126700748)

[4.2 Security Testing: Reading Transferred Files Through HTTP Packet Sniffing Using Wireshark 28](#_Toc126700749)

[4.3 Discussions 30](#_Toc126700750)

[4.3.1 Application of DAFFMAN in the real world 30](#_Toc126700751)

[4.3.2 System Performance in Client Devices 30](#_Toc126700752)

[CHAPTER 5 32](#_Toc126700753)

[CONCLUSIONS AND RECOMMENDATIONS 32](#_Toc126700754)

[5.1 Project Achievements 32](#_Toc126700756)

[5.2 Project Contribution 32](#_Toc126700757)

[5.3 Issues and Limitations 33](#_Toc126700758)

[5.4 Recommendations 33](#_Toc126700759)

[5.5 Conclusion 34](#_Toc126700760)

[References 35](#_Toc126700761)

[Appendices 39](#_Toc126700762)

[Appendix A: File Hashing Speed Experiments 40](#_Toc126700763)

[Appendix B: File Security Testing using Wireshark File Sniffing Technique 42](#_Toc126700764)

[Appendix C: Web Interfaces of the DAFFMAN System 43](#_Toc126700765)

[Appendix D: Ouriginal Report 46](#_Toc126700766)

# LIST OF FIGURES

[**Figure 1.1** Basic Architecture of NAS system (Andriani et al., 2021) 2](#_Toc126714702)

[**Figure 2.1** Avalanche Effect In AES For Modified Message And Constant Key 9](#_Toc126714703)

[**Figure 3.1** Waterfall Model of the project 13](#_Toc126714704)

[**Figure 3.2** Project Timeline and Gantt Chart 17](#_Toc126714705)

[**Figure 3.4** Entity Relationship Diagramof the Web NAS system 23](#_Toc126714706)

[**Figure 3.4** Network Architecture for The System 23](#_Toc126714707)

[**Figure 4.1** Graph of time taken to hash files corresponding to file size 29](#_Toc126714708)

[**Figure 4.3** The HTTP packet header captured during file upload process containing the unencrypted file. 31](#_Toc126714709)

[**Figure 4.4** The HTTP packet header captured during file upload process containing AES-encrypted file 32](#_Toc126714710)

# LIST OF TABLES

[**Table 2.1** Specifications of Raspberry Pi Model 3B used as file server 10](#_Toc126714712)

[**Table 2.2** Comparison between the proposed NAS\_CFS and NFS protocol 11](#_Toc126714713)

[**Table 3.1** Hardware and Software used for the NAS System 18](#_Toc126714714)

[**Table 4.1** List of files used for upload and hashing speed test. 27](#_Toc126714715)

[**Table 4.2** Time needed to hash the files using browser MD5 algorithm 28](#_Toc126714716)

# LIST OF ABBREVIATIONS

AES Advanced Encryption Standard, previously known as *Rijndael*

DAFFMAN Daniel’s Flask File Manager, the proposed web-based network-attached storage system

DFD Data Flow Diagram

ERD Entity Relationship Diagram

HTTP Hypertext Transfer Protocol

MD5 Message-Digest 5, a hashing algorithm

NAS Network-attached storage system(s)

NFS Network File System

PHP a scripting language used for web development

SHA-2 Secure Hash Algorithm 2, a family of hash functions standardised by US NIST in 2002 (Sarkar & Sanadhya, 2009)

SME small and medium-sized enterprises

SQL Structured Query Language, used to manage relational databases

ZFS Zettabyte File System, a file system designed by Sun Microsystems (Pugh et al., 2009)

# CHAPTER 1

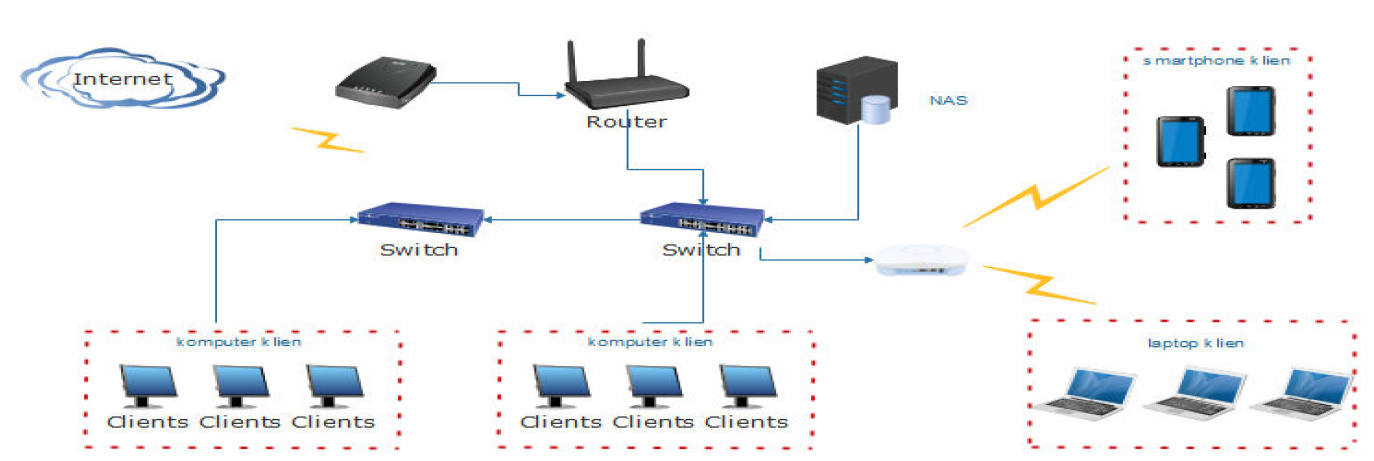
# INTRODUCTION

## Project Background

Due to the rapidly-growing need for digital information storage, most organizations and individuals choose to implement the Network-Attached Storage system in their local network. NAS system has been the main preferences over cloud storage, especially for critical organizational information and files that requires access by multiple users all the time.

**Network-attached Storage (NAS)** system is a centralised data storage server that can be accesses by any end devices in the network via Ethernet. The end devices will access the date stored in the network-attached storage system only by its metadata. Most NAS server host will use a less-featured and simple operating system for its operations, such as the Unix-like FreeBSD and FreeNAS operating system. In order to access the NAS server data, the client devices will use **Network File System (NFS)** protocol. In most of commercial uses of NAS, the NAS storage server will not be directly interacted by the client devices. Instead, the client will retrieve the files stored there by requesting through other servers, which in most cases, is a web server since the NAS system is accessed by using **Hypertext Transfer Protocol Secure (HTTPS)** protocols.

The diagram below shows an example the architectural design of a network with NAS file server. (Jaikar et al., 2016)

****

**Figure 1.1** Basic Architecture of NAS system (Andriani et al., 2021)

One of the main features of network-attached storage system for data storage is that it provides more centralised data distribution to network users who wish to access the needed files. The file owners will no longer need to resend or upload the files to the requester since the needed file is already available in the data storage server. In order to ensure that data integrity is preserved after any form of data exchange, checksum hashing is used. One of the most frequently-used hashing algorithm used is Message-Digest 5 (MD5) algorithm(de Guzman et al., 2019). MD5 algorithm is used because MD5 is one of the most common hashing algorithms and take shorter time to be created for each files before uploading. (Wei et al., 2020) (Mohurle & Panchbhai, 2017)

File encryption has become one of the most crucial and commonly-used method of securing files for public storage purposes. Data encryption is the process of converting unencrypted data or plain text into an almost unreadable data in order to secure them. Encrypted files will require a decryption key or algorithm to be decrypted. The process of encryption includes converting an unencrypted data called plain text into a cipher text. In this project, **AES** or **Advanced Encryption Standard** algorithm will be used for the file encryption. AES, which is also formerly known as ***Rijndael*** and the continuation of **Data Encryption Standard (DES)** (Nuradha et al., 2019), is used because AES algorithm are one of the encryption algorithms that is almost resistant to data cracks due to its complexity and the massive amount of time it usually takes to crack an AES-encrypted files. This is also why AES has become one of the most used encryption methods in many platforms and media.

## **Problem Statement**

The features of network-attached storage allow many network users to store and access files without the need to connect further to the Internet. For this, network-attached storage systems are widely-used to store many files, which may include sensitive or important files that need to be protected from unsecured access or unauthorized editing privileges.

One of the biggest and most critical problems of current network-attached storage systems involves data security. Most NAS users will choose to backup their files in the NAS systems, and this may also include files containing sensitive and confidential data. Due to the limits of current technology of NAS systems, they usually lack of security measures to secure the files from unauthorized access. This vulnerability will expose the stored files to a lot of cyber threats such as data theft and breach of users’ privacy. (Huang et al., 2004)

## Project Goals

The main aim of this project is to develop a web-based network-attached storage system for local area networks that will enforce and apply the file encryptions and checksum verifications algorithm throughout the NAS system in order to ensure better data security and integrity to file storage.

## Objectives

1. To design a web-based network-attached storage system that will implement the AES file encryption method and MD5 checksum hash verification to files stored
2. To evaluate the security of stored files by emulating unauthorized access to the stored files in the NAS system

## Project Scope

### System Deployment

The network-attached storage system is designed for general usage in any type of networks, which also includes small networks such as home or SME network. For testing, the system will be implemented in a small network that consists of at least 1 home router and 1 home end devices such as desktop or laptop computers with Internet capabilities.

### End Users

The users of the NAS system are limited to users of the local area network (LAN) where the NAS system is deployed to. Although the NAS system is browser-based and accessible by LAN, this project will focus to run the system for Windows operating systems, specifically the home desktop version of Windows OS.

### Encryption and Hashing Algorithm

For file integrity checking, the system will use MD5 hashing algorithm. The file hashing process will occur both on client-side and server-side. The SHA-256 hashing algorithm will also be used to store the hash of the file decryption password and user account password used to log into the system.

For the file encryption process, the system will use AES-256 CBC-mode, which is provided in the PyAesCrypt module integrated with the system.

## Project Significance

The project is done to create a secure network-attached storage system that will store encrypted files in the storage server. With the encryption, access to the said files will be greatly filtered and restricted. The encryption processes will also be conducted on the client devices, meaning that it is almost impossible for the stored files to move to and fro the storage server without being encrypted first.

The project will also provide better data integrity preservation along with the secure file encryption by implementing the MD5 checksum verification method to ensure that no file duplicates is created or uploaded to the storage server. Not only that, the MD5 checksum created for each files will also determine if any changes is made to the recently-uploaded files once the new checksum is compared with the checksum of the original file.

## Chapter Summary

This chapter has explained that in this project, in addition of the features of the current network-attached storage systems, the system will also implement the encryption of files using AES encryption algorithm and the hashing of files using MD5 algorithm for file integrity verification. This chapter has also described the current issues that this project has aimed to provide the solution to.

# CHAPTER 2

# LITERATURE REVIEW

This chapter will discuss and review the topics and other works related to this project. All topics discussed are in the area of computer networks and cybersecurity.



## Network-Attached Storage Systems

Network-attached storage (NAS) systems are a networked storage method mostly used among users in the local area network (LAN), although there have been many development researches conducted to make network-attached storage systems accessible to the outside network, as proposed in a research conducted by Lanka and Gargeyas (2018).

## Data Security and Integrity

### Hashing and Message-Digest 5 (MD5) Algorithm for File Checksum

File hashing is the process of generating a length of output corresponding to the content of the hashed file. File hashes will be used for data integrity preservation purposes, such as detecting soft errors found during data transfers or to detect file duplicates for file storage in a networked medium. In the process of generating a hash, a Cryptographic Hash Function is used. Cryptographic Hash Function is a hashing function that takes a random length of input to generate a constant length of output, which can be used to verify data integrity.(Shakya & Karna, 2019). Although MD5 hashing aalgorithm is known as one of the most secure hashing method, there has been few attacks found to be done on the algorithm. One of the attack methods take advantages of the usage of rainbow tables, which is originally used to store hash values and store the location of the passwords. (Libed et al., 2018) (Kumar et al., 2013)

### Advanced Encryption Standard (AES) Algorithm

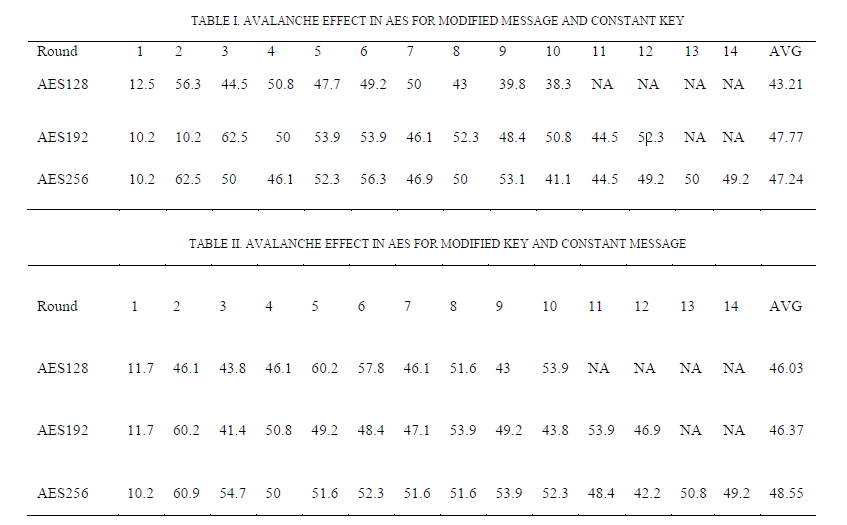
Advanced Encryption Standard, which is previously known as Rijndael, is a symmetric block cipher that process 128 bits data blocks input with 3 cipher keys with the length of 128, 192 and 256 bits. These variations of cipher keys length will be referred as “AES-128”, “AES-192” or “AES-256” (Federal Information Processing Standards, 2001) (Dworkin et al., 2001). Despite these variations, AES-128 are the most frequently-used variation of AES encryption (Opritoiu et al., 2010)

### Avalanche Effects

Avalanche effects are measurements of changes that will occur on the output when the input bits are modified, no matter how little the modification is. Avalanche effect has become a method of analysing the effectiveness of an encryption algorithm. Efficiency of any encryption method is measured using the following formula:

Avalanche Effect = Number of flipped bits / Total Bits

A research conducted by (Sanap & More, 2021) has demonstrated on how avalanche effects is used to review the effectiveness Advanced Encryption Standard (AES) encryption algorithm by measuring the avalanche effect caused on data encrypted using AES algorithm with three different cipher keys, which is AES-128, AES-192, AES-256. The research used Cryptool as a avalanche effect analysis tool.



**Figure 2.1** Avalanche Effect In AES For Modified Message And Constant Key

## Related Works

### Remotely Accessible, Low Power Network Attached Storage Device (Lanka & Garzevas, 2018)

The usage of network-attached storage (NAS) can benefits a lot of users who wishes to increase their data storage space and share it among other local network users. But there are problems occurred regarding the usage of network-attached storage, especially to those who uses it for homes or small businesses. Some of them are the high power consumption by NAS devices due to its requirements to run all the time and the limited accessibility from external networks.

The objective of the research conducted are as below (Lanka & Garzevas, 2018):

1. To create a NAS system with low power consumption
2. To design a NAS system with accesses from network out of the LAN networks.
3. To design a user-friendly NAS system for easier maintenance and configuration.

This research is related to the project because both this project and the referred research are aimed for home use, which needs systems that can run on low-cost and low device specifications. The table below shows the specifications of the Raspberry Pi 3B Model used as the NAS server.

Table 2.1 Specifications of Raspberry Pi Model 3B used as file server

|  |  |
| --- | --- |
| **Attribute** | **Specifications** |
| SoC | Broadcom BCM2837 |
| CPU | 1.2GHz 64-/32- bit quadcore  ARM Cortex- A53 |
| Memory | 1 GB LPDDR2 RAM at  900MHz |
| Storage | MicroSDHC slot |
| Graphics | Graphics Broadcom VideoCore IV |

### Home Server and NAS using Raspberry Pi (Shrivasta & Gadge, 2017)

The purpose of this project conducted by Shrivasta and Gadge (2017) is to create a low-cost user-friendly network-attached storage system, which can be beneficial for home usage as the system is designed for easier maintenance and configuration processes. The network-attached storage system is also designed to run 24/7 with low power consumption for constant availability.

### Research and Implement of an Encrypted File System Used to NAS (Huang et al., 2004)

Huang et. al. (2004) has conducted a research aimed to implement an encryption system for file systems in network-attached storage. The research has designed a file-system level encryption system named NAS\_CFS. NAS\_CFS is a proposed cryptographic file system designed as a mountable kernel modular and applies stackable Vnode interface.

IN NAS\_CFS, all files stored within the system is ensured to be encrypted before read process can be performed on the stored files. The research has also compared their proposed NAS\_CFS to the NFS protocol that is currently used in most networked storage systems.

Table 2.2 Comparison between the proposed NAS\_CFS and NFS protocol

|  |  |  |
| --- | --- | --- |
| **Items** | **NAS\_CFS** | **NFS** |
| Mounting Layer | In Kernel | User-Level |
| Mounting Machine | Server | Client |
| Performance Effect | Low | High |
| Encryption Algorithm | Blowfish | DES / 3DES Blowfish |
| Security | Excellent | Good |
| Convenience | Good | Not Good |
| Portability | Good | Not Good |
| Client Platform | Linux / Windows | Linux |

## Chapter Summary

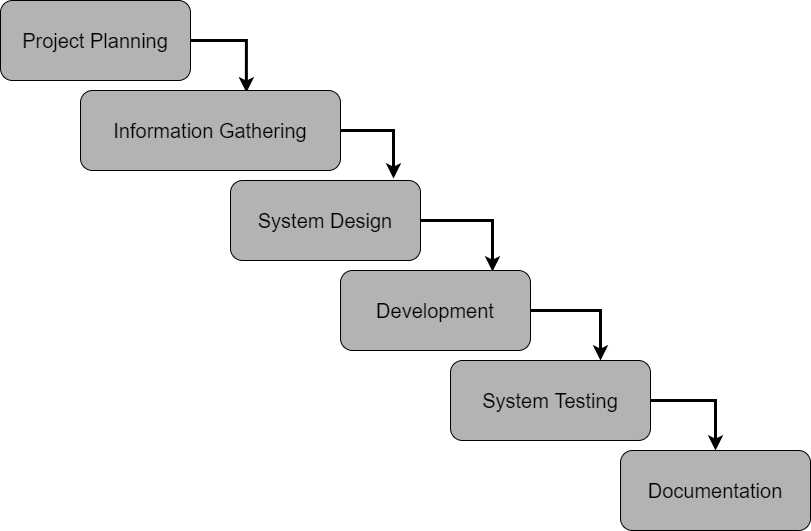
This chapter has discussed all topics related to network-attached storage and cryptography applied to this project. The chapter has also discussed various related works that has applied similar research methods, techniques and attributes that can be made as references for this research

# CHAPTER 3

# METHODOLOGY



## Project Phases and Tasks



**Figure 3.1** Waterfall Model of the project

The Waterfall Model in the diagram above shows the process flow of the project. The project is divided into six phases;

1. Project planning
2. Information Gathering
3. System Design
4. System Development
5. System Testing
6. Documentation

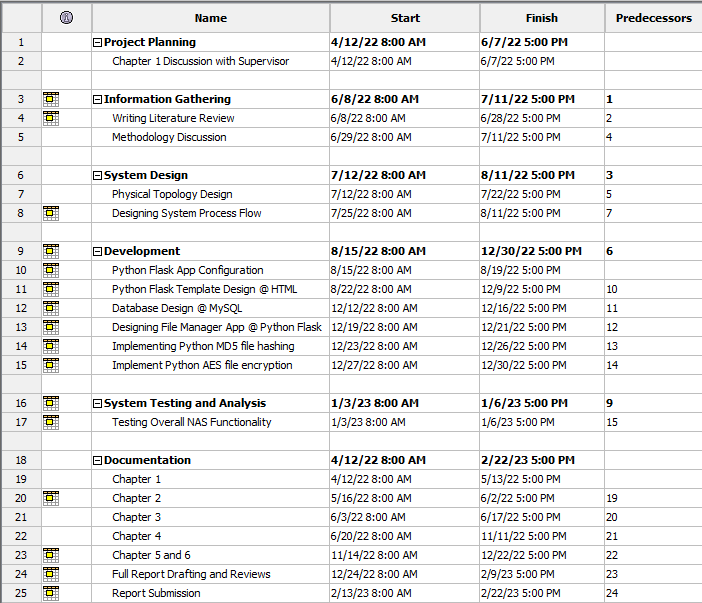
Each phases contains different tasks in many aspects in order to achieve the specific goals set for each phases. The table below shows the activities that will be performed in each phase and what to achieve at the end of the phases.

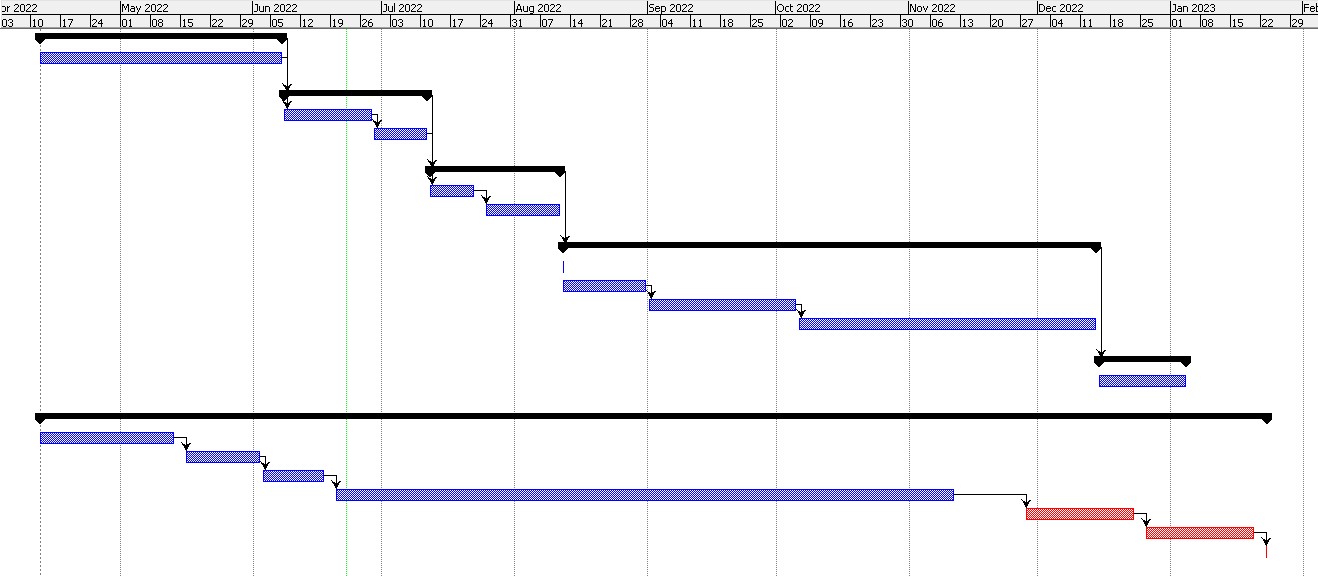
|  |  |  |  |
| --- | --- | --- | --- |
| **Phase** | **Objective** | **Tasks** | **Deliverables** |
| Project Planning | To discuss problems to be solved  To come up with project title and objectives of the project | 1. Discussion with Supervisor on these topics: 2. Area of project 3. Scopes of project | Chapter 1:   * Project Title * Project Background * Objectives * Scopes * Goals and Objectives * Significances |
| Information Gathering | To study the previous works related and collect information on the development of NAS systems | 1. Studying research papers and articles regarding NAS systems and system security 2. Writing Literature Review 3. Methodology Discussion with Supervisor | Chapter 2:   * Related Works * Suitable Hardware and Software Requirements for the system |
| System Design | To design the NAS system in terms of process flow, physical connections needed and user interfaces | 1. Designing the physical architecture of NAS system with the requirements set 2. Designing the process flow 3. Designing user web interfaces using HTML, Bootstrap CSS and Python Flask templates | 1. Chapter 3: Methodology 2. Physical topology of the NAS system 3. System flowchart 4. User web Interfaces of the NAS system |
| System Development | To develop the backend functionality of the NAS system | 1. Configure a web server and NFS file server using Ubuntu OS 2. Creating database for MD5 checksums of the files stored using MySQL DBMS | 1. Chapter 4: Development 2. Fully-working NAS system with AES encryption system |
| System Testing | To test the capabilities of the system to prevent file duplication  To test the effectiveness of file encryptions in the system | 1. Testing by uploading files with similar checksum multiple times 2. Reading files owned by other users stores in the NAS system | 1. Chapter 5: Results and Discussions 2. Chapter 6: Recommendations |

|  |  |  |  |
| --- | --- | --- | --- |
| Documentation | To summarize the project and document a formatted project report | 1. Drafting full reports with correct formats 2. Report drafts review with supervisor and approval 3. Final Submission | 1. Drafts of Report for review by Supervisor before submission 2. Full Project Report covering Chapter 1 to 5 based on results from each phase |

## Detailed Project Phases

### Project Timeline and Gantt Chart





**Figure 3.2** Project Timeline and Gantt Chart

### Project Planning

Project planning phase is the first phase of the project. This phase will heavily involve the supervision and guidance from the supervisor in order to come up with the proper project title for the proposal. Once the title is created, the discussion is done with the supervisor for the problem statement. Overall, the topics discussed in this phase are listed as below:

1. Project Title
2. Project Background
3. Problem Statement
4. Project Objectives
5. Project Scope
6. Project Significance

At the end of this phase, the documentation of Chapter 1 (Project Motivation) is completed with the approval of the supervisor.

### Information Gathering

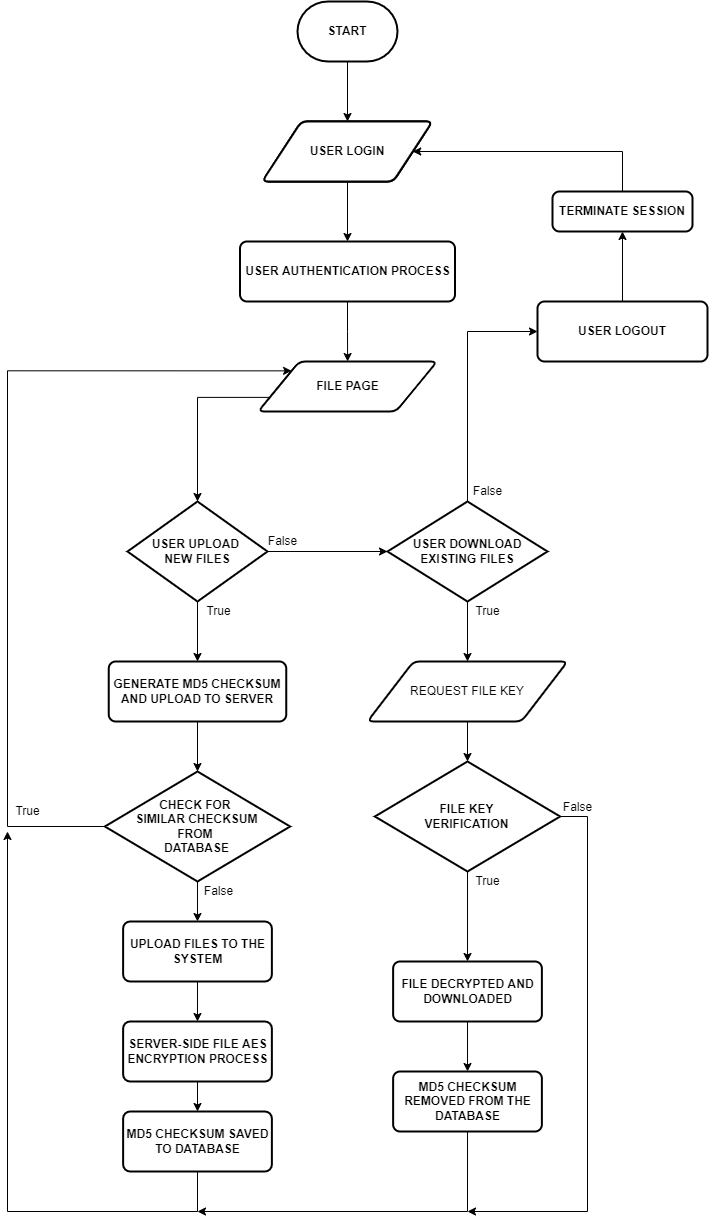
The main objective of this phase is to study research papers and articles related to NAS systems and data security application to NAS systems. In this phase, a number of academic papers collected from various research journals will be reviewed and summarised. Journal articles, research papers, websites are some of the sources collected. A few articles and research papers on previous works related to the development and application of NAS systems will also be reviewed and made as references.

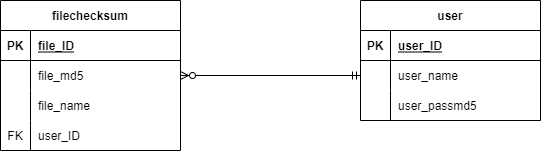
Table 3.1 Hardware and Software used for the NAS System

|  |  |  |  |
| --- | --- | --- | --- |
| Roles | Device | | |
| Hardware | | | |
| File Server | Raspberry Pi Model 4B | | |
| Web Server | Acer Aspire E14 (Model E5-476G-50WA)   |  |  | | --- | --- | | CPU | Intel Core i5-8250U 1.5GHz | | GPU | Nvidia GeForce MX150  (2GB VRAM) | | RAM | 12GB DDR4 | | Storage | 480GB SATA SSD | | Operating System | Ubuntu Desktop 22.10 (Kinetic Kudu) |   Python Flask server is deployed using Gunicorn, a Python WSGI HTTP server for UNIX. (Chesneau, 2021) | | |
| Client Devices | Acer Aspire E14 (Model E5-476G-50WA)   |  |  | | --- | --- | | CPU | Intel Core i5-8250U 1.5GHz | | GPU | Nvidia GeForce MX150  (2GB VRAM) | | RAM | 12GB DDR4 | | Storage | 480GB SATA SSD | | Operating System | Windows 11 Home |   The system will be accessed using Mozilla Firefox web browser. The device used will be connected to the router through wired LAN, using cat6 LAN cable with RJ45 connectors. | | |
| Router | Mercusys 300Mbps MW305R Wireless N Router | | |
| Software | | | |
| Programming Language | | Web Interfaces and Backend | HTML, CSS, Bootstrap CSS framework,  Python Flask Framework  (*Welcome to Flask - Flask Documentation*, 2010) |
| Server Scripting | Python 3 |
| Server OS Shell | | Bash | |
| Database Management System (DBMS) | | MySQL DBMS | |
| Web Server | | Python Flask micro web framework | |
| AES Encryption Tools | | Crypto-JS, a JavaScript encryption library for browsers. | |

### System Design

In this phase, a project flowchart and the system process flowchart will be designed. These diagrams will act as a general guidance on how the project will be executed and how the system will operate once completed. The physical and logical topology of the NAS system consisting of all hardwares listed on the previous phase will also be designed before proceeding to the next phase. The Entity Relationship Diagram (ERD) and Data Flow Diagram (DFD) will also be designed as the main reference during system and database development process.

 **Figure 3.3** System flowchart



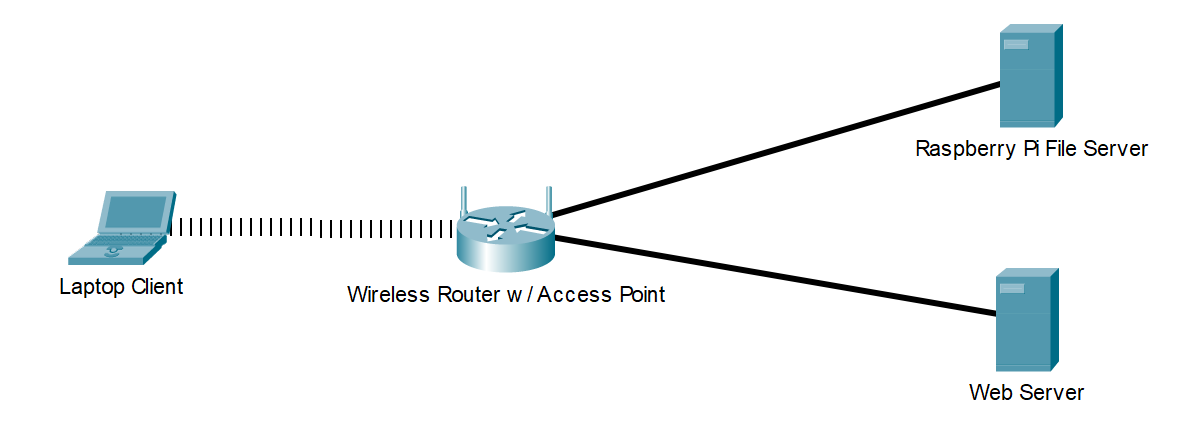
**Figure 3.4** Entity Relationship Diagramof the Web NAS system

### System Development

The system development phase in this report will explain in details on how the development process will be performed. The main objective of this phase is to develop the functionality of the NAS system, both frontend and backend of the system. Before the software development process begin, a testbed consisting of devices listed below will be set up.

* + 1. Router
    2. Raspberry Pi 4B Server
    3. 1TB hard disk drive
    4. Laptops for client device

The diagrams below show the physical and logical topology that will be created for testing purposes.



**Figure 3.4** Network Architecture for The System

Once the testbed has been setup, the Raspberry Pi 4B device will be configured to run a Ubuntu operating system. The Raspberry Pi 4B will then be configured to run needed to run the NAS system services, such as ZFS, MySQL DBMS. The ZFS will act as the file system for the NAS server for data storage redundancy among the hard disks mounted to the file Raspberry Pi file server. (Widianto et al., 2017)

### System Testing

During the testing phase, the overall functionality of the NAS system should be completed. The objective of this phase is to ensure that the NAS system fulfill the proposed objectives. The tasks performed in this phase is based on the system process flowchart during the system design phase.

### Documentation

The documentation phase began once the project title has been approved by the supervisor. The documentation process also takes place throughout the project development starting from project planning phase. The purpose of this phase is to produce a formatted report for final submission. The report will consist of chapters from chapter 1 to 5, covering all tasks and objectives that has been completed throughout the project development. During the documentation phase, drafts of chapters will be submitted for review by the supervisor. Once the chapters are completed, they will be compiled into one full report and a draft report will be reviewed before finalization and approval by the supervisor. Once the documentation phase is completed and the NAS system is fully developed, a presentation will be conducted.

## Chapter Summary

This chapter has highlighted all details and information regarding the methodology and all requirements needed to execute the project accordingly. The chapter has also explained the development phases involved throughout the final year project period, from planning phase to the documentation phase where the project report will be completed and finalized for submission.

# CHAPTER 4

# ANALYSIS AND DISCUSSIONS

This chapter discusses the tests and analyses performed to evaluate the designed system. There are 2 tests conducted to measure and analyze



## Performance Testing: Hashing Speed on Client-side Browsers

One of the significances of this system is that this system implements client-side MD5 file hashing to noticably reduce the file upload time. The process occurs on the browser used by the user to access the system, utilizing the FileReader API implemented by default in today’s latest browsers existed. The system will automatically abort the upload process if the file with similar hash is already uploaded to the server by the same owner, which is determined by running hash checks with all files stored in the user folder.

To test the system hashing speed, a few files of different types and sizes will be selected from the system without uploading it to the file server. For testing purposes, DAFFMAN system has been modified to display the time taken to hash the selected files using MD5 algorithm from browser’s JavaScript CryptoJS library. The hashing will begin immediately after the files has been selected by users. The file used is as described in Table 4.1. and the result of the tests is shown in Table 4.2. The screenshots of the DAFFMAN system displaying the test results is shown in Appendix A.

Table 4.1 List of files used for upload and hashing speed test.

|  |  |  |
| --- | --- | --- |
| **File Name and Type** | **MD5 value** | **File size in KB** |
| readme.txt  (Plain Text file) | 2d01d5d9c24034d54fe4fba0ede5182d | 0.064 |
| resume.pdf  (Document file) | 3748814ea8f1f8ed3c7e458bc9325ceb | 94.325 |
| document.docx  (Microsoft Word file) | 23ad3a62f12892f7b677417b16e792ab | 846.585 |
| image.png  (PNG image file) | c3ffd551ab398d5f77377192d85cf720 | 1445.353  (1.4 MB) |
| video2.mp4  (MP4 video file) | 9670a9ef3b769632840beb5c9372ed94 | 2235.210  (2.2 MB) |
| ReverseEngineering.zip (Zipped file containing documents) | b1d33f07694a3fadc349e99488b6f4a6 | 17863.176  (17.8 MB) |
| video.mp4  (MP4 video file) | 96f1c23a17e901b4ff3c0c6bf6807a5b | 25531.380  (25.5 MB) |
| games.zip  (Zipped file containing media) | 4292400ea275d764de0b5a074dc8c36b | 64903.957  (64.9 MB) |
| anime.mp4  (Video file) | d27687d66055aaf7172d984bcc964512 | 224418.914  (224.4 MB) |

Table 4.2 Time needed to hash the files using browser MD5 algorithm

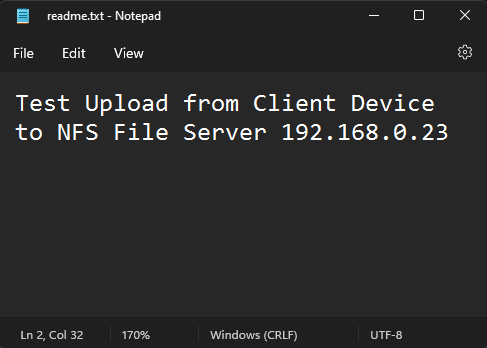
|  |  |  |
| --- | --- | --- |
| **File Name and Type** | **File size in KB** | **Time taken to hash the file (miliseconds)** |
| readme.txt | 0.064 | 0007 |
| resume.pdf | 94.325 | 0086 |
| document.docx | 846.585 | 0168 |
| image.png | 1445.353  (1.4 MB) | 0351 |
| video2.mp4 | 2235.210  (2.2 MB) | 0835 |
| ReverseEngineering.zip | 17863.176  (17.8 MB) | 93028 |
| video.mp4 | 25531.380  (25.5 MB) | 147276 |
| games.zip | 64903.957  (64.9 MB) | 553815 |
| anime.mp4 | 224418.914  (224.4 MB) | 677383 |

From the tests performed, it is found that the time elapsed to hash files experience significance increase when the file size reaches 10 MB. The time elapsed also experience a steeper time increase when the file size has reached or exceeded the 50MB boundary. While any files that are smaller than the 50MB boundary takes only seconds to be hashed. The summary of this tests is shown in graph in Figure 4.1. It is also found that during testing for “anime.mp4” file, which is 224 MB in size, the system tends to crash and render the browser laggy and slow.

**Figure 4.1** Graph of time taken to hash files corresponding to file size

## Data Security Testing: Reading Transferred Files Through HTTP Packet Sniffing Using Wireshark

To test the security of files interacted by the web-based NAS system, a plain text file named “readme.txt” has been created. The content of the file is as shown in Figure 4.1.



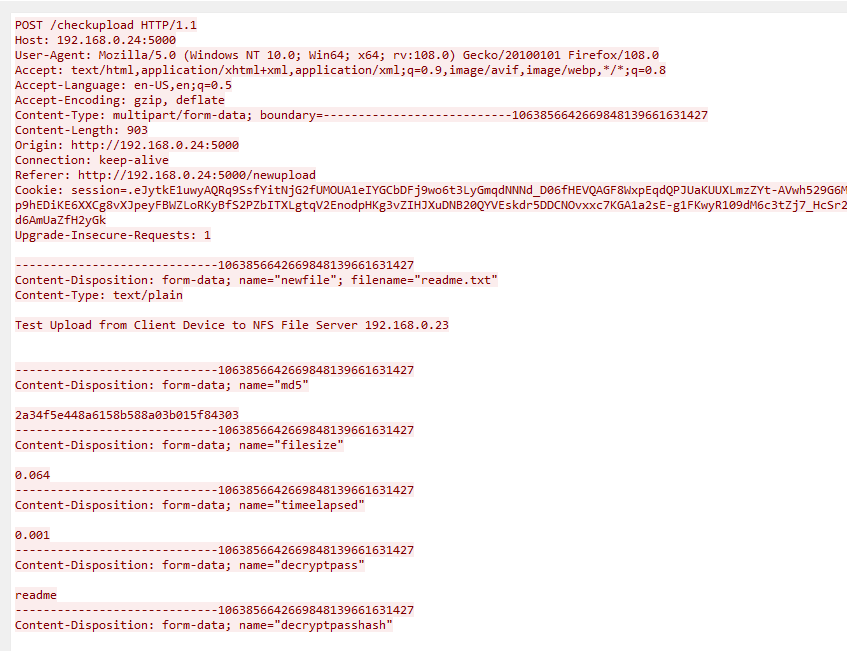
**Figure 4.1** The contents of readme.txt used for security testing

There are multiple steps and setup performed during this test. Those steps can be summarised as following.

1. Make sure that the NAS system is running and all services needed in the web server and file server are running.
2. Run Wireshark on any client devices connected to the local area network.
3. The web server is using port 5000 to serve the web-based NAS system. On Wireshark, filter the packets using the following conditional statements:
   1. ip.addr == [web server IPv4 address] && http
4. Scan through the packets sniffed and extract the packets containing the transferred file.
5. Select the filtered packet and click on “Follow”, followed by “HTTP stream”.
6. The content of the file should be readable in text form for unencrypted files and in unintelligible characters for AES-encrypted file, as shown in Figure 4.3 and Figure 4.4

More detailed steps of running this test are shown in Appendix B, including the screenshots of each step taken to perform the tests.

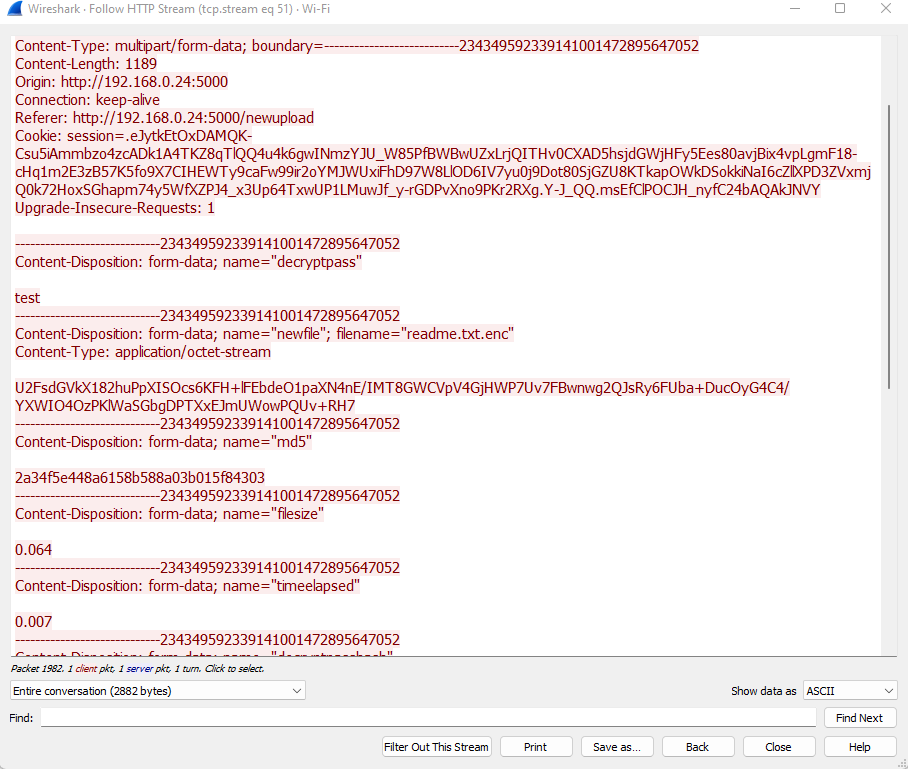
The file will then be uploaded to the file server through the web-based NAS system. A Wireshark is set up to sniff all HTTP incoming and outgoing packets running through the computer port in the client device used to access the web NAS system. The result of this packet sniffing is shown in Figure 4.1. More details, results and screenshots taken during the test is shown in Appendix C.



**Figure 4.3** The HTTP packet header captured during file upload process containing the unencrypted file.

The HTTP packet header shown in Figure 4.3 has shown that the contents of the file “readme.txt” is clear and easily readable by Wireshark packet sniffer. This is because the “readme.txt” file is only encrypted on the server-side once the file has been fully uploaded to the server, which makes the system vulnerable to data breach occuring on the client-side.

But in the proposed system, the AES256 encryption process will occur in the browser in client devices, meaning all file processing and encryption will occur on the client side before being sent to the NFS file server for storage. All files that will leave the network port to be sent to the file server will only leave the device as an AES-encrypted files with unreadable contents, even from the Wireshark sniffer. Only sets of random characters can be read from the sniffed HTTP packet as shown in Figure 4.4.



**Figure 4.4** The HTTP packet header captured during file upload process containing AES-encrypted file

## Discussions

### Application of DAFFMAN in the real world

DAFFMAN is originally proposed to be used in small-scale, such as for home network or SME businesses that require simple network-attached storage solutions at a low cost. This usage scope remains the same due to the performance of the DAFFMAN system that are unable to handle larger storage. The evaluation of the results of hashing speed tests during upload process has determined that at the point of the project completion, DAFFMAN system is only capable of storing small-sized files such as documents, zipped files and short videos with size not more than 50MB. This can be seen as shown in the results of Performance Testing: Hashing Speed on Client-side Browsers.

At the current state of DAFFMAN system, the system is applicable to most small-scale NAS storage system, either for home or SME network. Despite the limited file size processing capability, the proposed system is still capable of storing documents safely with the end-to-end AES encryption algorithm implemented in the system.

### System Performance in Client Devices

One of the most noticable limitations with this web-based NAS system is that the system is unable to handle large files well, notably files exceeding 50MB in size. This can be seen from the graphs shown in Figure 4.1 and Figure 4.2, where it is shown that there is a significant increase in time needed to hash a file when the file size reaches or exceeds 50MB. This is because the encryption method used in this system uses array-based file content reading, which will consume much time to complete. Similar array-based file content reading is also used for the MD5 file hashing algorithm.

The system will also experience a lot of inconsistencies in terms of system performance. This is because system performance is heavily dependant on the connection type, either wired or wireless, and specifications of the device used by the user to access the system. The system tends to perform better with high-end devices with wired LAN connection, compared to low-end devices with WiFi or wireless LAN connection.

## Conclusion

This chapter has discussed the results of the tests and analysis performed on the proposed NAS system, DAFFMAN. From the results achieved from the tests, it has been ensured that all the objectives set for this project has been successfully achieved and completed despite all the issues and limitations encountered, which is to design a web-based NAS system that implements AES encryption and MD5 hashing and to evaluate the security and performance of the proposed NAS system. The DAFFMAN system has fulfilled its purpose of providing end-to-end AES-encryption to all files that will be stored in small-scale NAS storage system.

# CHAPTER 5

# CONCLUSIONS AND RECOMMENDATIONS

This chapter concludes the projects and the tests performed. The chapter also discusses the contribution, issues and recommendations for future development.



## Project Achievements

The completion of this project has successfully achieved both of the objectives set for this project, which is to design a web-based NAS systems that implements AES end-to-end file encryption and to evaluate the system in terms of security and performance.

The DAFFMAN system is constructed from scratch using Python and Python Flask framework for back-end web designing. Multiple JavaScript libraries are also integrated into the system for client-side file processing, which is the major part of DAFFMAN system. The proposed DAFFMAN system is also proven secure based on the results of data security tests performed using packet sniffing attack with Wireshark tool. Files processed in this system are rendered unreadable outside of the client devices used by authorized user due to the AES encryption algorithm applied during upload/download process.

## Project Contribution

This project has produced DAFFMAN, a web-based network-attached storage systems that implements AES-256-CBC client-side encryption and MD5 file hashing. As a web-based system, DAFFMAN has successfully provided a secure way of storing files into the NAS server, all from the web browser with no third-party/external applications installation required.

The fortification of DAFFMAN with AES encryption also prevents possible cybersecurity issues such as man-in-the-middle (MITM) attacks or packet sniffing by unauthorized users, which can compromise the integrity and security of the transferred files if left unencrypted. The encryption process will render the files unreadable until the authorized users decrypt the files themselves with their own decryption key/password.

The low-end specification requirements also make the system deployment cheaper and more efficient budget-wise, making it suitable for a quick solution to a simple NAS storage system in small networks.

## Issues and Limitations

During the development phase, there are many issues encountered involving methods used to implement certain system features and this issue require alternative method that may be more complex. For example, the file decryption process of the DAFFMAN system has to be separated from the file download phase, as opposed in the encryption process, which is integrated with the upload process. This is because most web browsers prevent direct file processing by the browsers after downloading from the servers due to security reasons.

This project also uses low-end server devices, which is the Raspberry Pi 4B for the file server and a Linux laptop computer for web server. This limits the capability of the NAS system to process and store larger files efficiently.

## Recommendations

DAFFMAN system is created to be served solely on Linux/Unix-based server operating systems. In the future, this project can be expanded by integrating Windows-based features into the program so that more server operating systems are capable of running the system, such as Windows Server operating system.

This project can also be improved by implementing different client-side encryption algorithms in the NAS system. The currently-used encryption algorithm is highly complex and file processing will take noticably large amount of time to complete. In the near future, a better browser-based encryption method can help to reduce the file processing time.

## Conclusion

From the results of the tests performed on the proposed DAFFMAN NAS system, it can be concluded that the system can be implemented and deployed in smaller-scale NAS system, such as home or SME network. It can also be concluded that the objectives of this project have been achieved, which is to design a web-based NAS system that implements end-to-end AES encryption and to evaluate the security and performance of the proposed system. As per the completion of the project, the proposed system, DAFFMAN has managed to solve the problem this project has intended to solve, which involves data security and integrity preservation to the files used in network-attached storage systems in most small-scale networks.

# REFERENCES

Andriani, T., Hidayatullah, M., Saputra, D., Esabella, S., & Gunawan, G. (2021). Building data centers using Network Attached Storage (NAS) and Microprocessor Operating Systems. *IOP Conference Series: Materials Science and Engineering*, *1088*(1), 12076. <https://doi.org/10.1088/1757-899x/1088/1/012076>

Chesneau, B. (2021). *Gunicorn - Python WSGI HTTP Server for UNIX*.

de Guzman, L. B., Sison, A. M., & Medina, R. P. (2019). Implementation of enhanced MD5 algorithm using SSL to ensure data integrity. *ACM International Conference Proceeding Series*, 71–75. <https://doi.org/10.1145/3310986.3311027>

Dworkin, M. J., Barker, E. B., Nechvatal, J. R., Foti, J., Lawrence E. Bassham, L. E., Roback, E., & Dray Jr., J. F. (2001). *Announcing the ADVANCED ENCRYPTION STANDARD (AES)*. <http://csrc.nist.gov/csor/>

Federal Information Processing Standards. (2001). *Announcing the ADVANCED ENCRYPTION STANDARD (AES)*. <http://csrc.nist.gov/csor/>

Huang, J., Xie, C., & Cai, B. (2004). Research and Implement of an Encrypted File System Used to NAS. *2nd IEEE International Security in Storage Workshop, SISW 2003*, 73–77. <https://doi.org/10.1109/SISW.2003.10007>

Jaikar, A., Shah, S. A. R., Noh, S. Y., & Bae, S. (2016, April 19). Performance Analysis of NAS and SAN Storage for Scientific Workflow. *2016 International Conference on Platform Technology and Service, PlatCon 2016 - Proceedings*. <https://doi.org/10.1109/PlatCon.2016.7456814>

Kumar, H., Kumar, S., Joseph, R., Kumar, D., Shrinarayan Singh, S. K., Kumar, P., & Kumarr, H. (2013). Rainbow table to crack password using MD5 hashing algorithm. *2013 IEEE Conference on Information and Communication Technologies, ICT 2013*, 433–439. <https://doi.org/10.1109/CICT.2013.6558135>

Lanka, A., & Garzevas, A. (2018). Remotely Accessible, Low Power Network Attached Storage Device. *Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT 2018*, 1083–1088. <https://doi.org/10.1109/ICICCT.2018.8473164>

Libed, J. M., Sison, A. M., & Medina, R. P. (2018). Improved MD5 through the extension of 1024 message input block. *ACM International Conference Proceeding Series*, 18–23. <https://doi.org/10.1145/3278312.3278318>

Mohurle, M., & Panchbhai, V. v. (2017, February 13). Review on realization of AES encryption and decryption with power and area optimization. *1st IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems, ICPEICES 2016*. <https://doi.org/10.1109/ICPEICES.2016.7853276>

Nuradha, F. R., Dwi Putra, S., Kurniawan, Y., & Rizqulloh, M. A. (2019). Attack on AES Encryption Microcontroller Devices With Correlation Power Analysis. *2019 International Symposium on Electronics and Smart Devices*.

Opritoiu, F., Vladutiu, M., Prodan, L., & Udrescu, M. (2010). *A High-Speed AES Architecture Implementation*. Association for Computing Machinery.

Pugh, C., Henderson, P., Silber, K., Carroll, T., & Ying, K. (2009). Utilizing ZFS for the storage of acquired data. *Proceedings - Symposium on Fusion Engineering*. <https://doi.org/10.1109/FUSION.2009.5226462>

Sanap, S. D., & More, V. (2021). Performance Analysis Of Encryption Techniques Based On Avalanche Effect And Strict Avalanche Criterion. *2021 3rd International Conference on Signal Processing and Communication, ICPSC 2021*, 676–679. <https://doi.org/10.1109/ICSPC51351.2021.9451784>

Sarkar, P., & Sanadhya, S. K. (2009). *A New Hash Family Obtained by Modifying the SHA-2 Family*. Association for Computing Machinery.

Shakya, A., & Karna, N. (2019). Enhancing MD5 hash algorithm using symmetric key encryption. *ACM International Conference Proceeding Series*, 18–22. <https://doi.org/10.1145/3309074.3309087>

Shrivasta, A. R., & Gadge, J. (2017). Home Server and NAS using Raspberry Pi. *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI) : 13-16 Sept. 2017.*

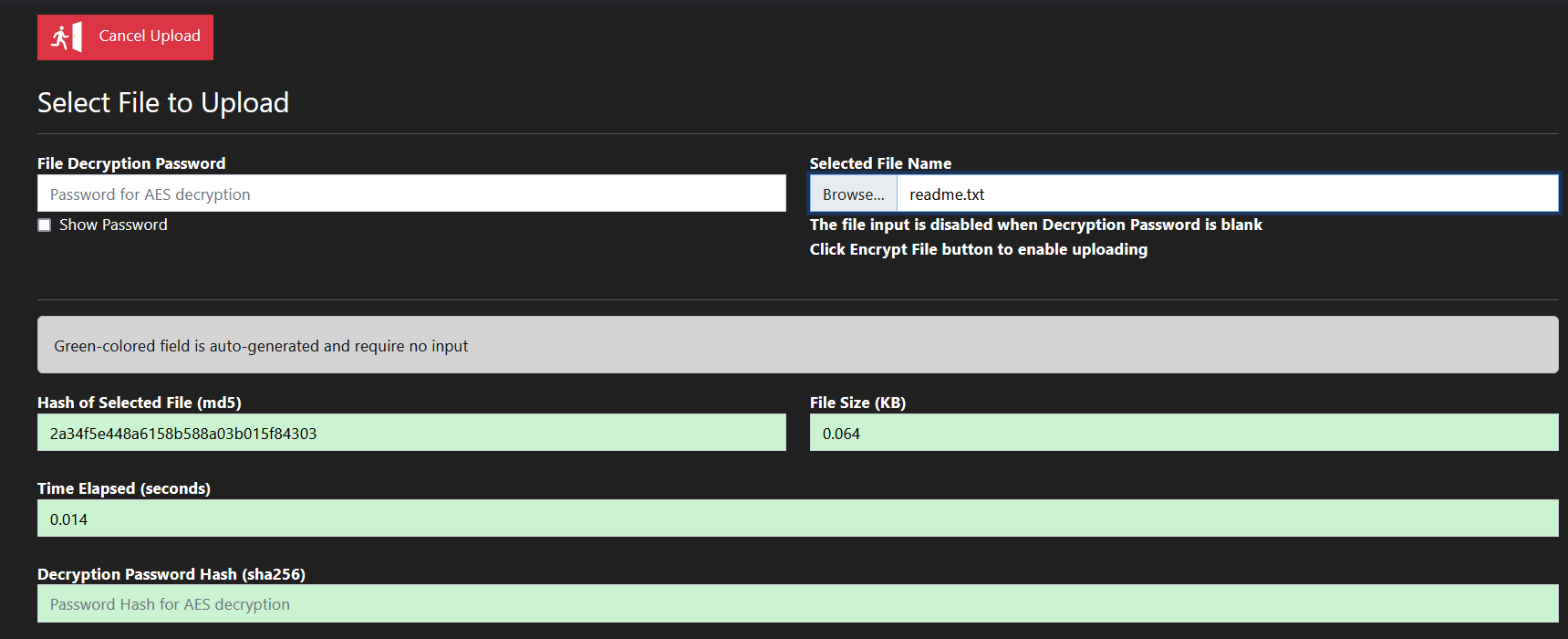
Wei, H., Li, H., & Chen, M. (2020). Architecture design and hardware implementation of AES encryption algorithm. *Proceedings - 2020 5th International Conference on Mechanical, Control and Computer Engineering, ICMCCE 2020*, 1611–1614. <https://doi.org/10.1109/ICMCCE51767.2020.00353>

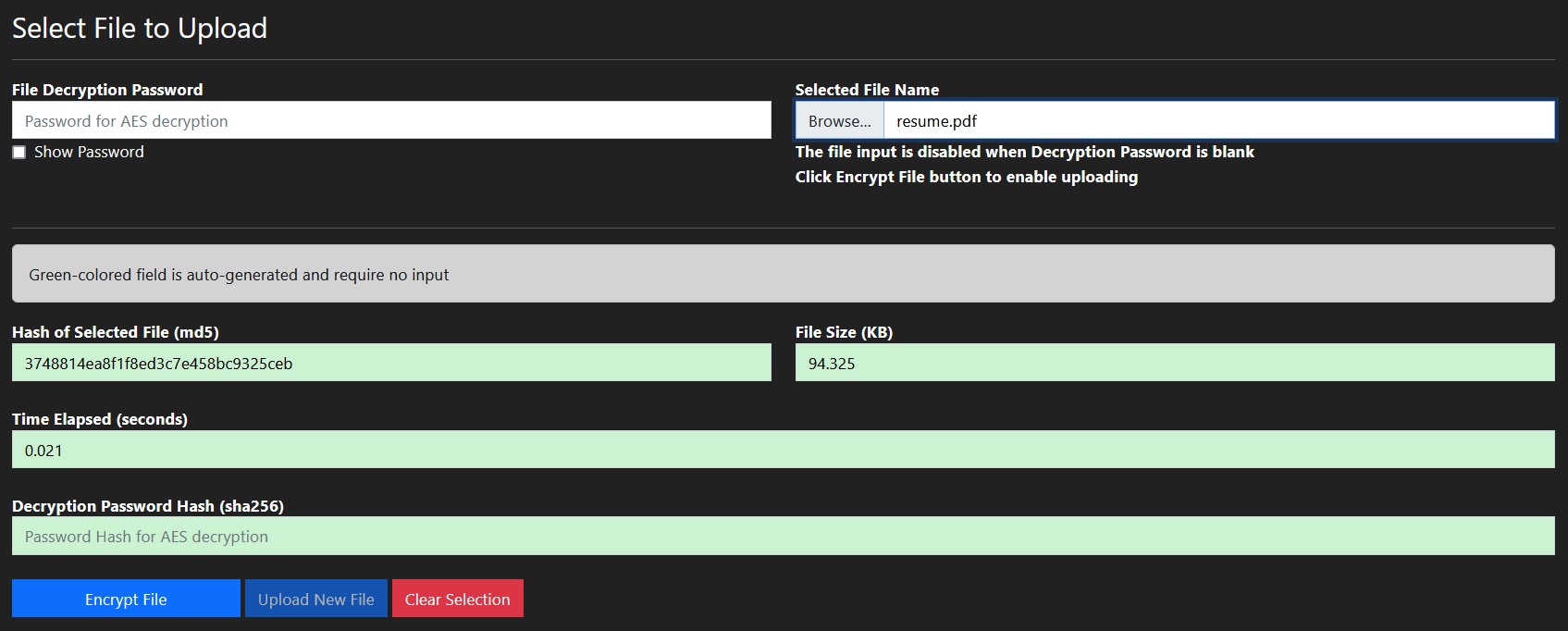
*Welcome to Flask - Flask Documentation*. (2010). <https://flask.palletsprojects.com/en/2.1.x/>

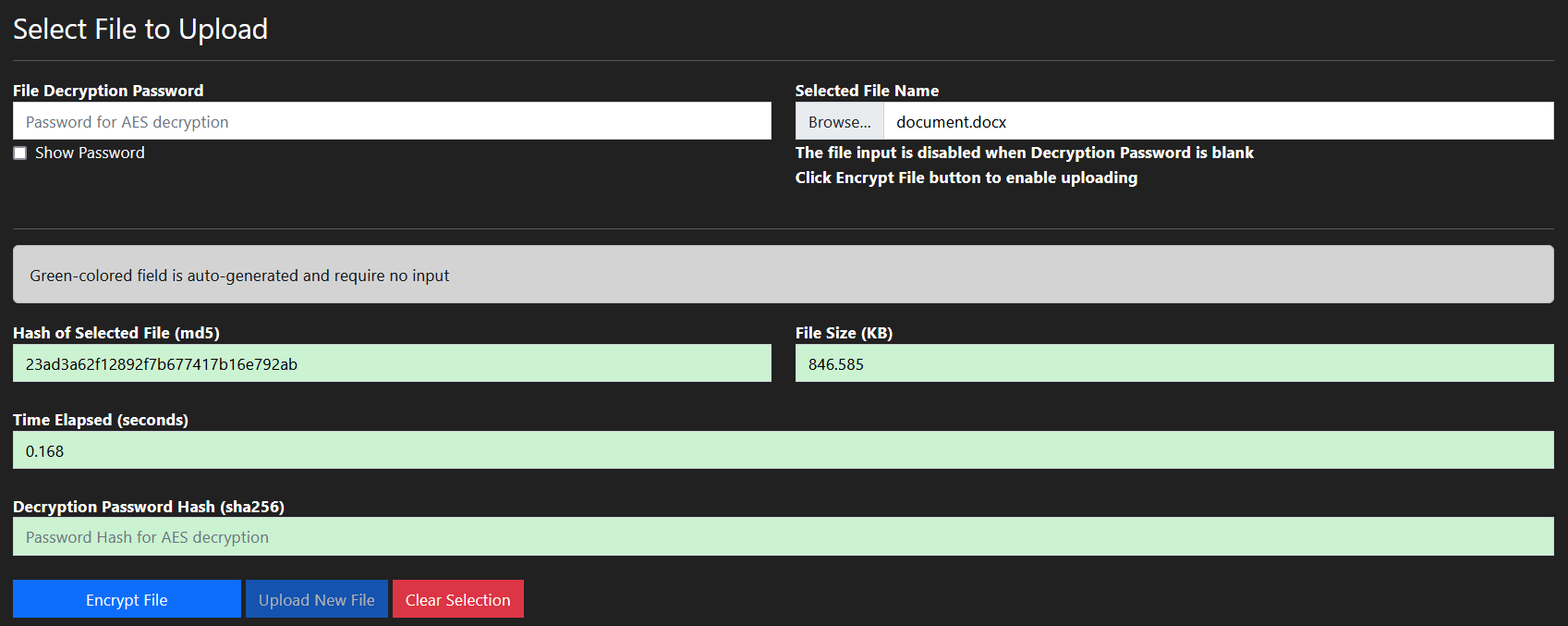
Widianto, E. D., Prasetijo, A. B., & Ghufroni, A. (2017). On the implementation of ZFS (Zettabyte File System) storage system. *Proceedings - 2016 3rd International Conference on Information Technology, Computer, and Electrical Engineering, ICITACEE 2016*, 408–413. https://doi.org/10.1109/ICITACEE.2016.7892481

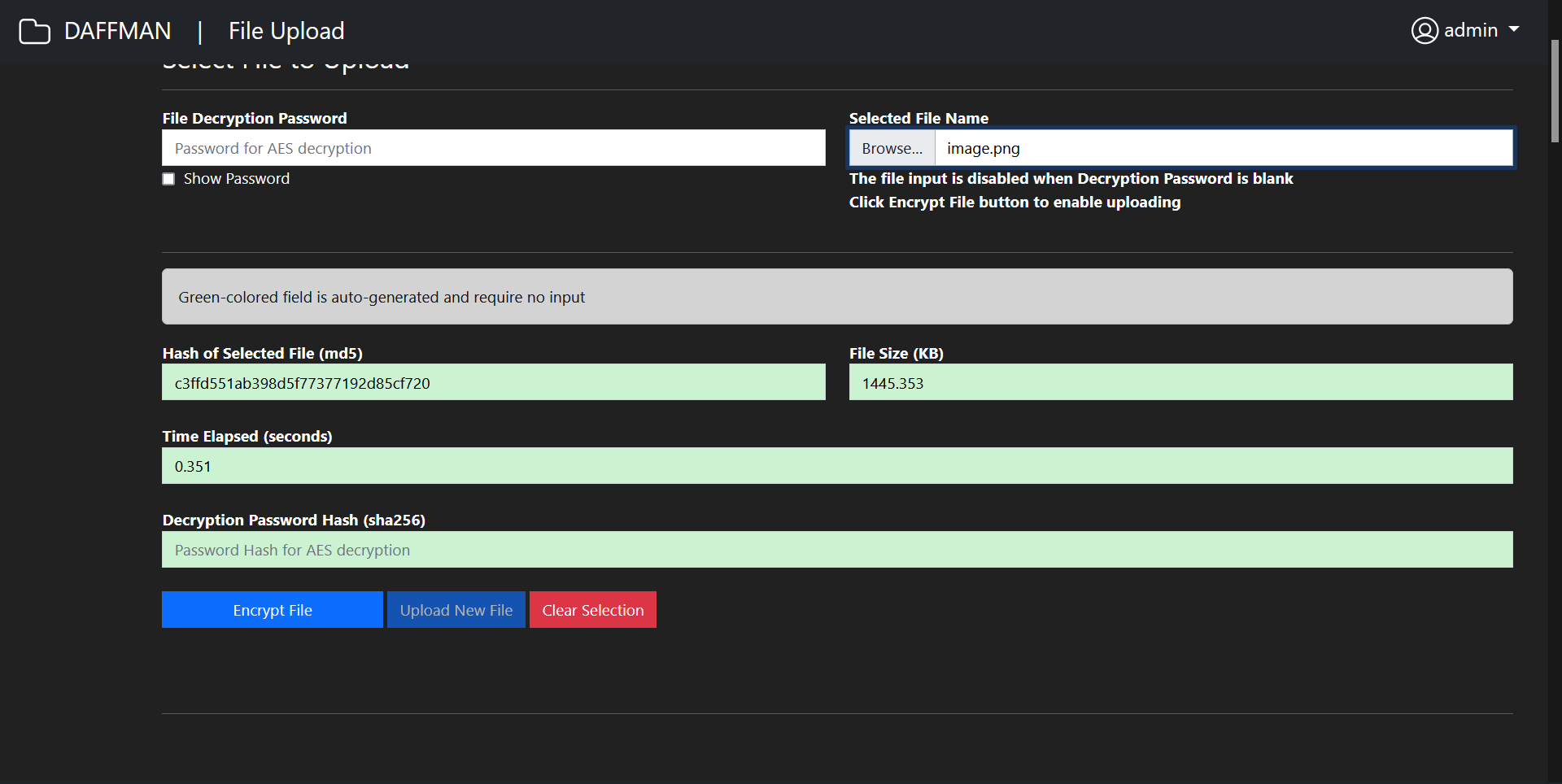
# APPENDICES

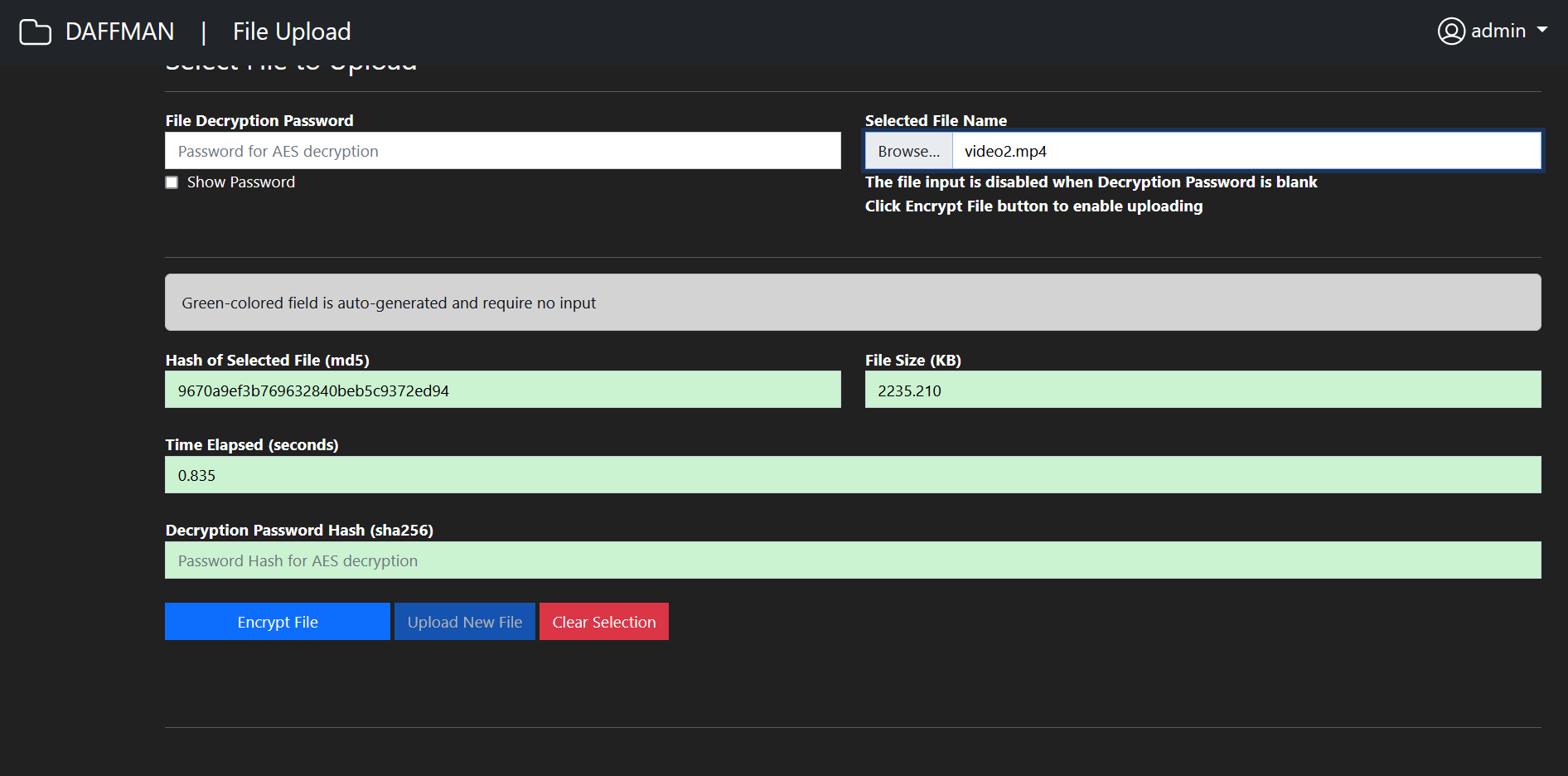
## Appendix A: File Hashing Speed Experiments

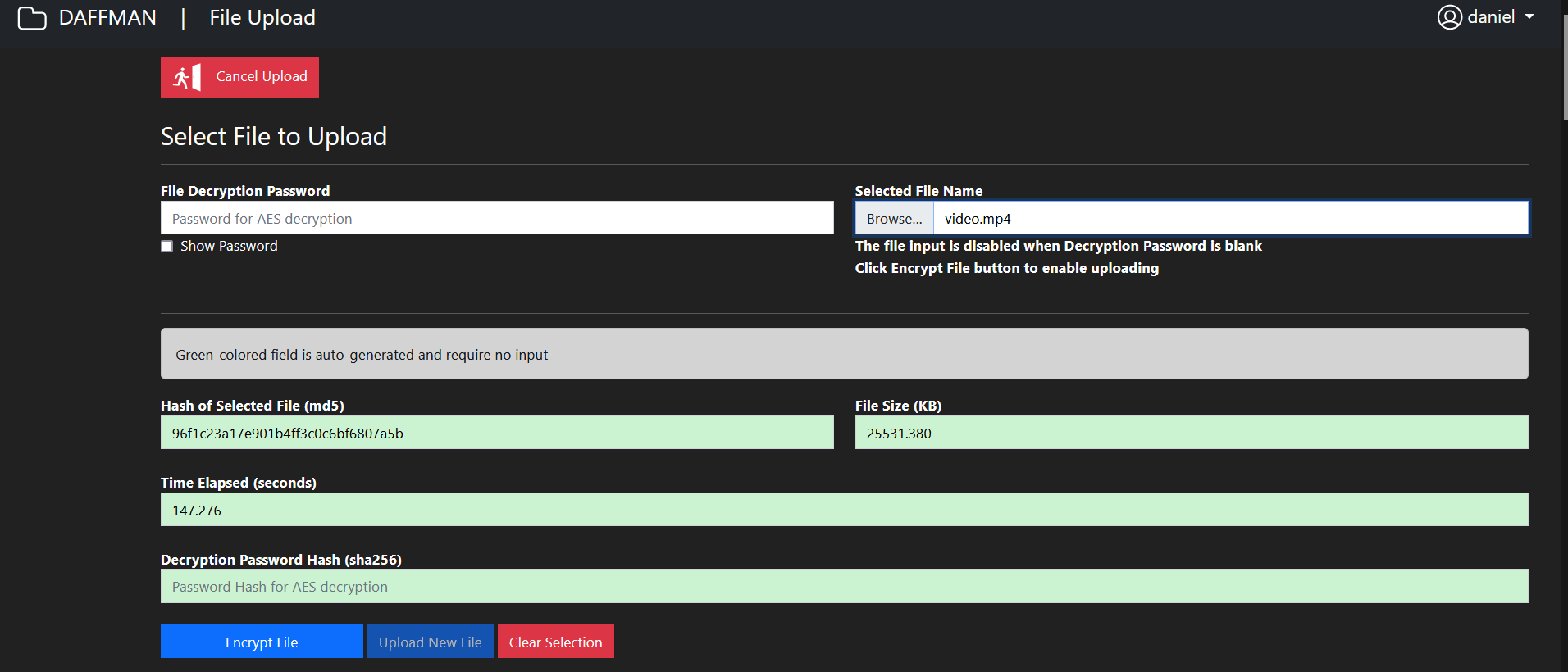


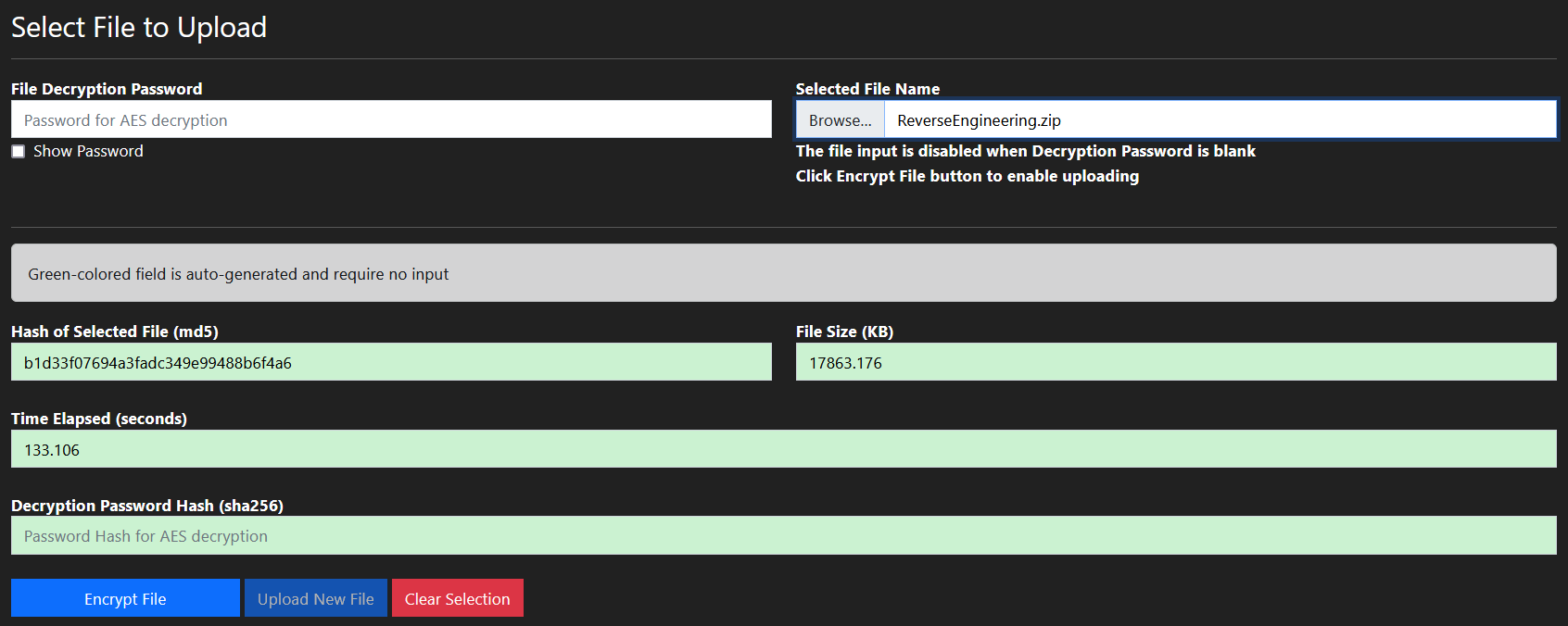


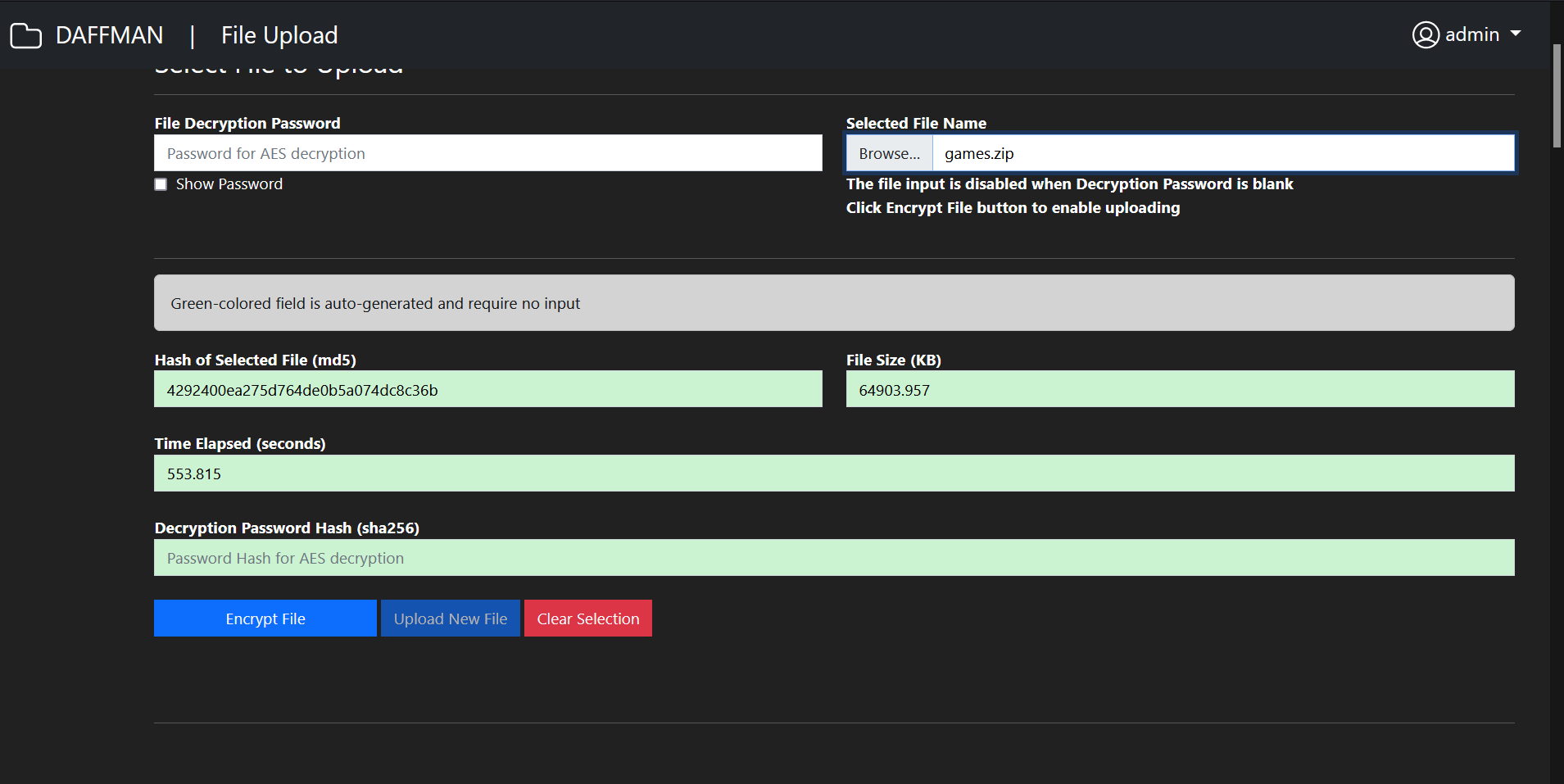




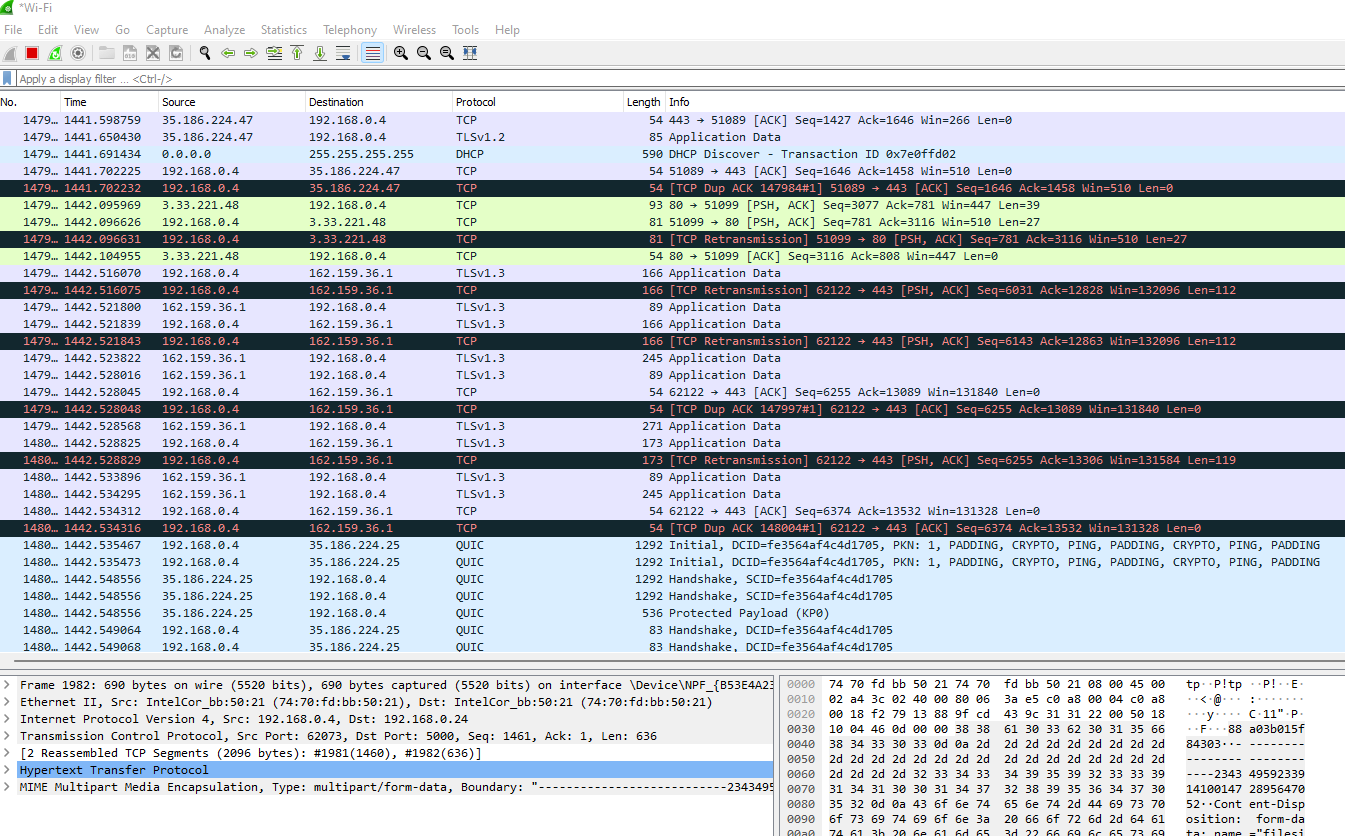


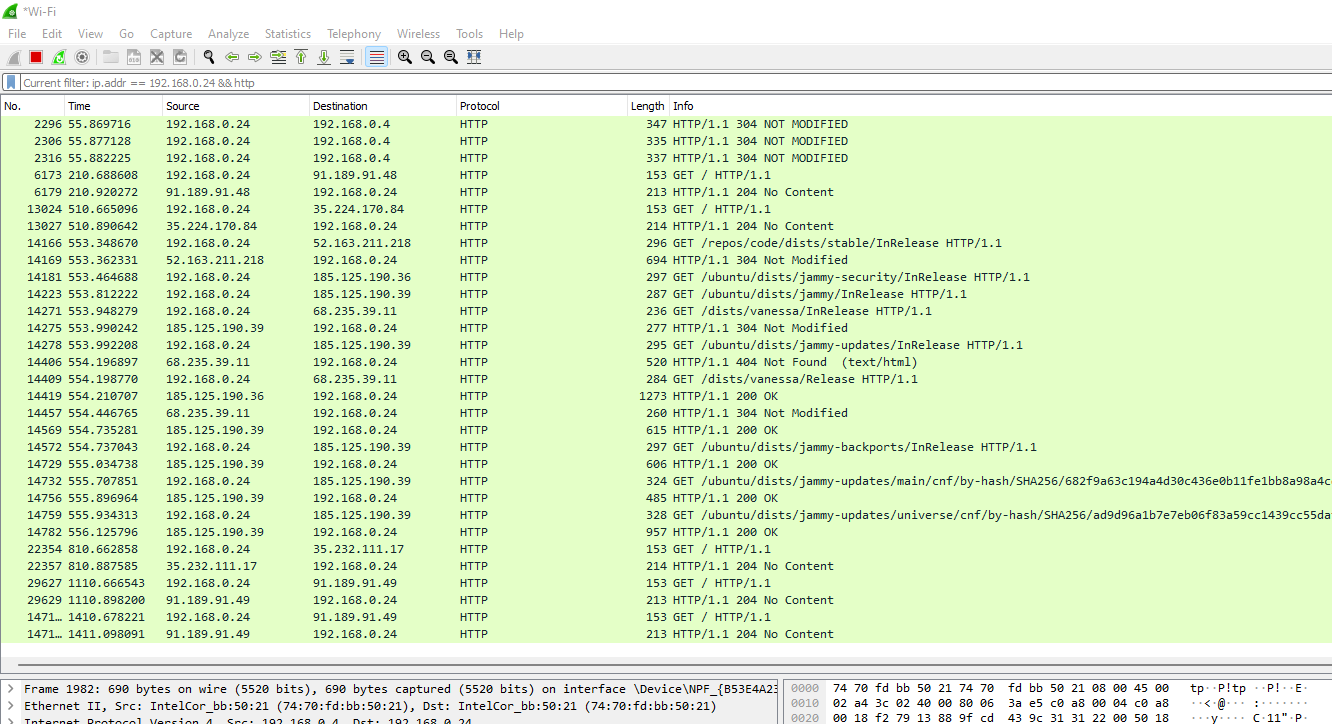


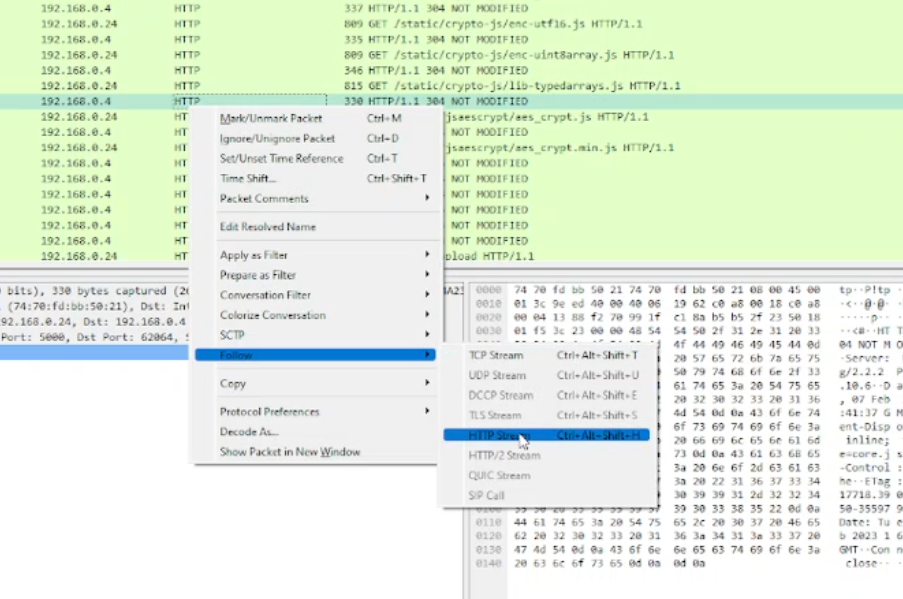




## Appendix B: File Security Testing using Wireshark File Sniffing Technique

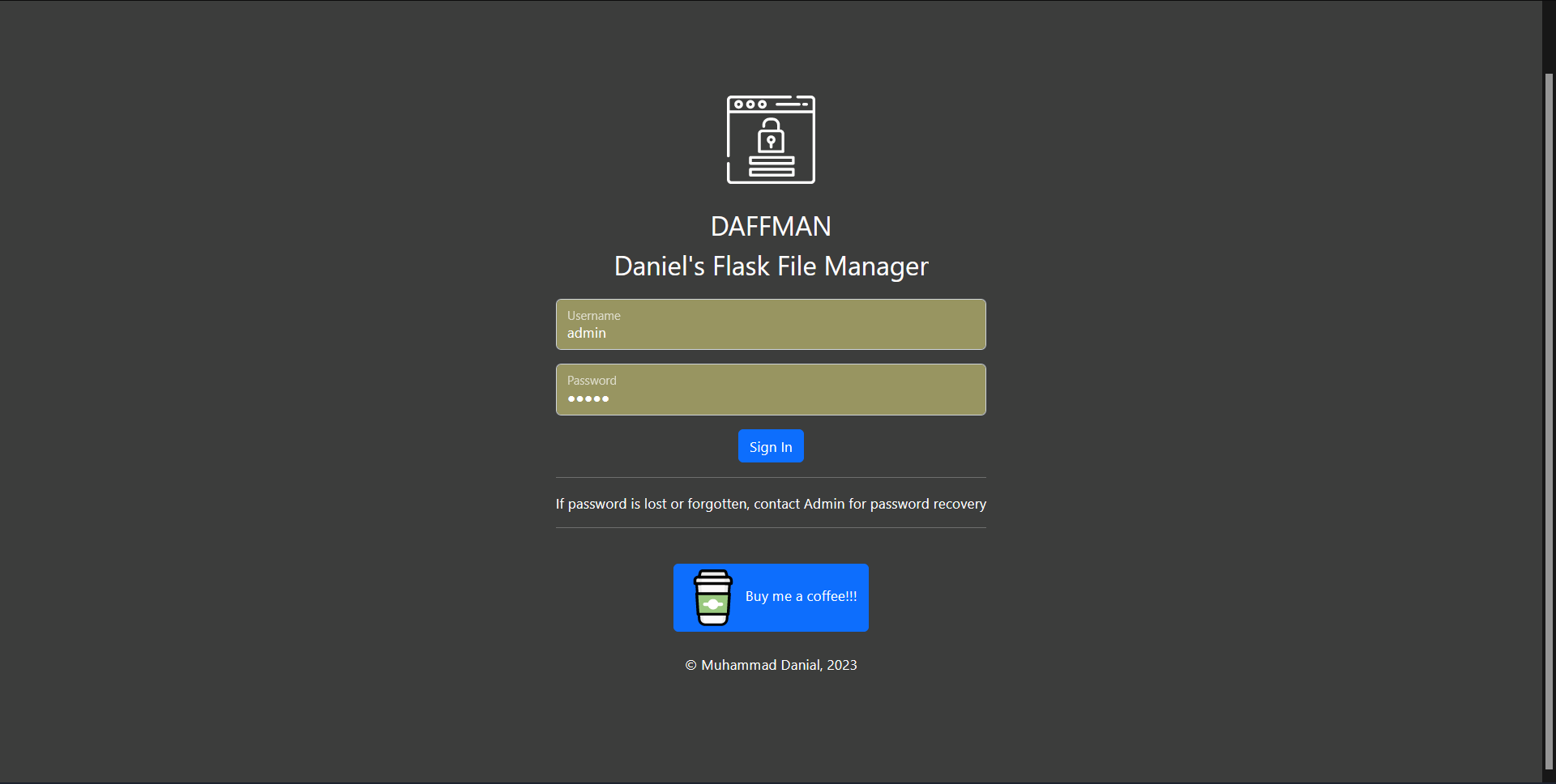




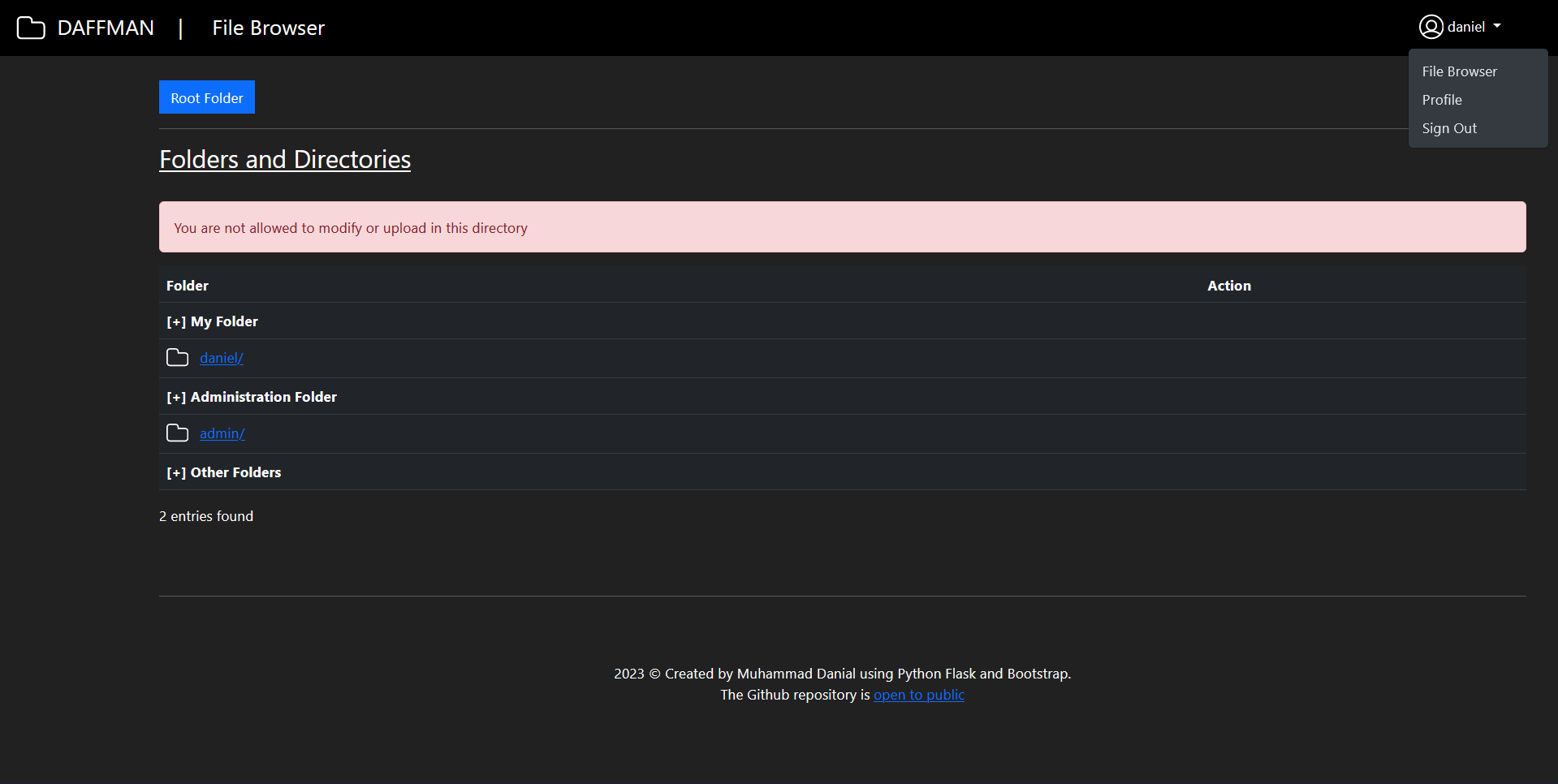
****

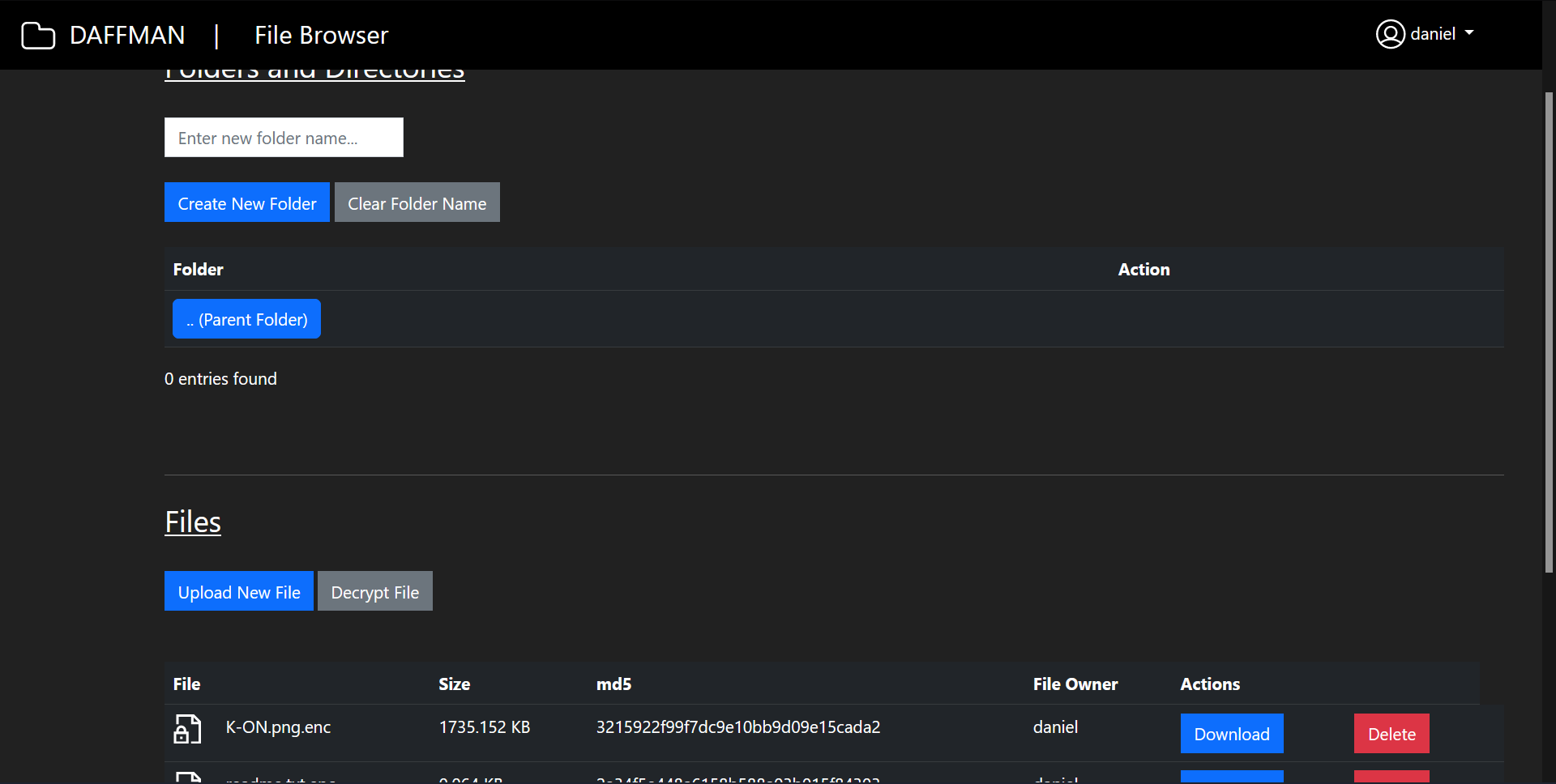
## Appendix C: Web Interfaces of the DAFFMAN System

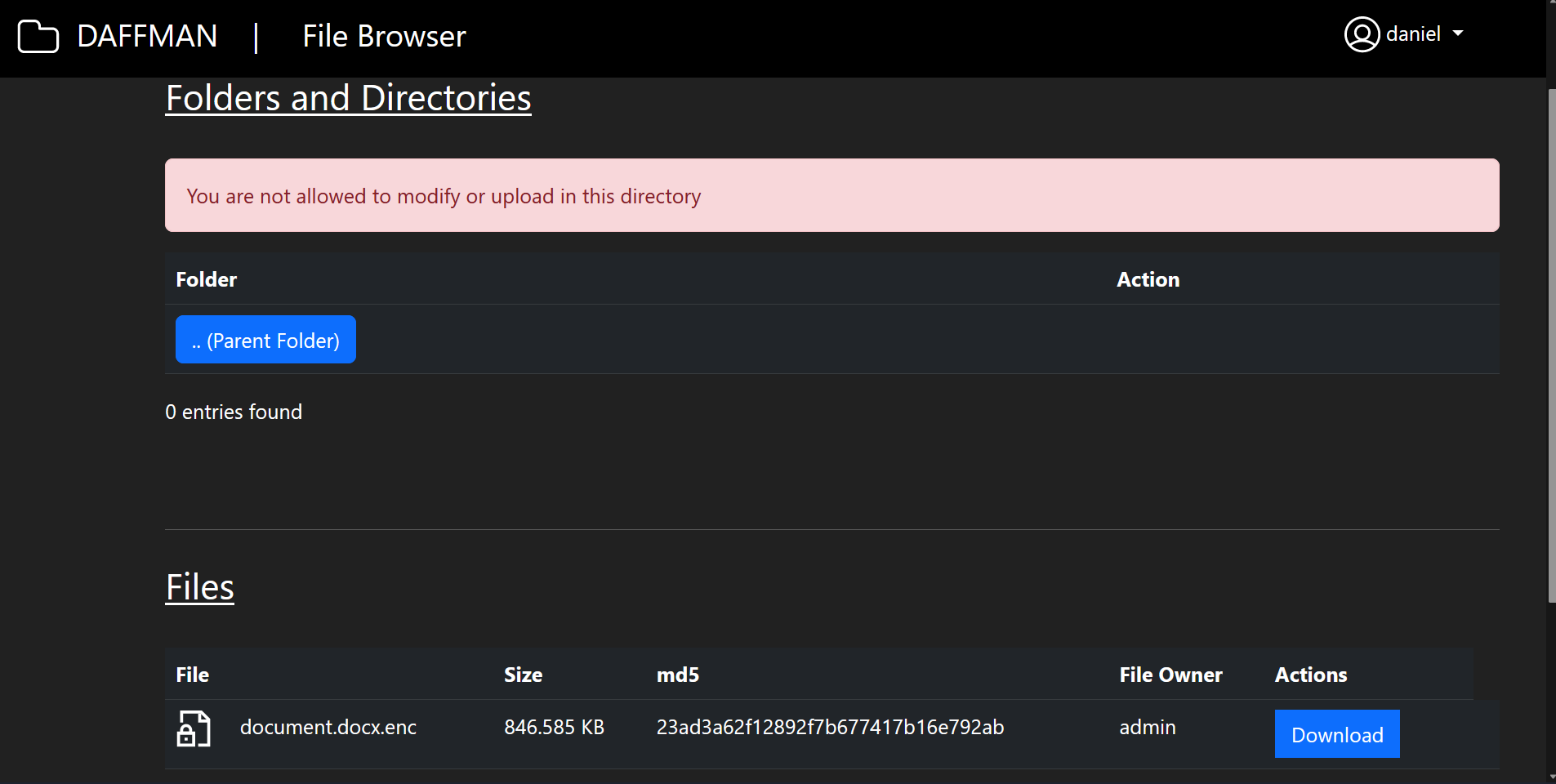
1. User Login page
   * User will input their username and corresponding password to access the file storage system.



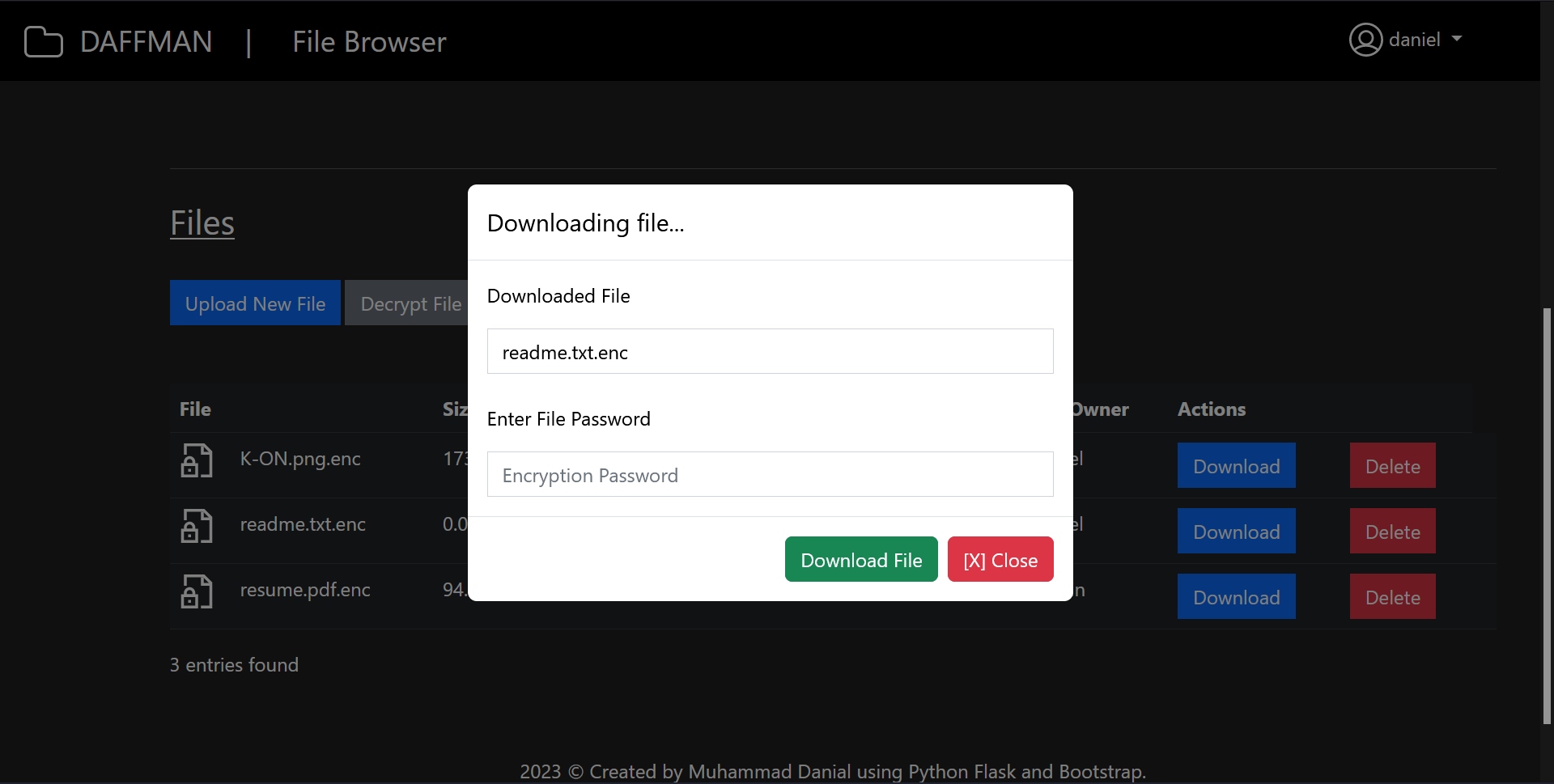
1. File Browser page
   * All files and directory stored in the root directory, “/uploads” will be displayed here and the files will be available for download while file deletion availability is based on user’s authorization on certain folders.
   * Folder creation and deletion will be performed here by the user in their own directory.



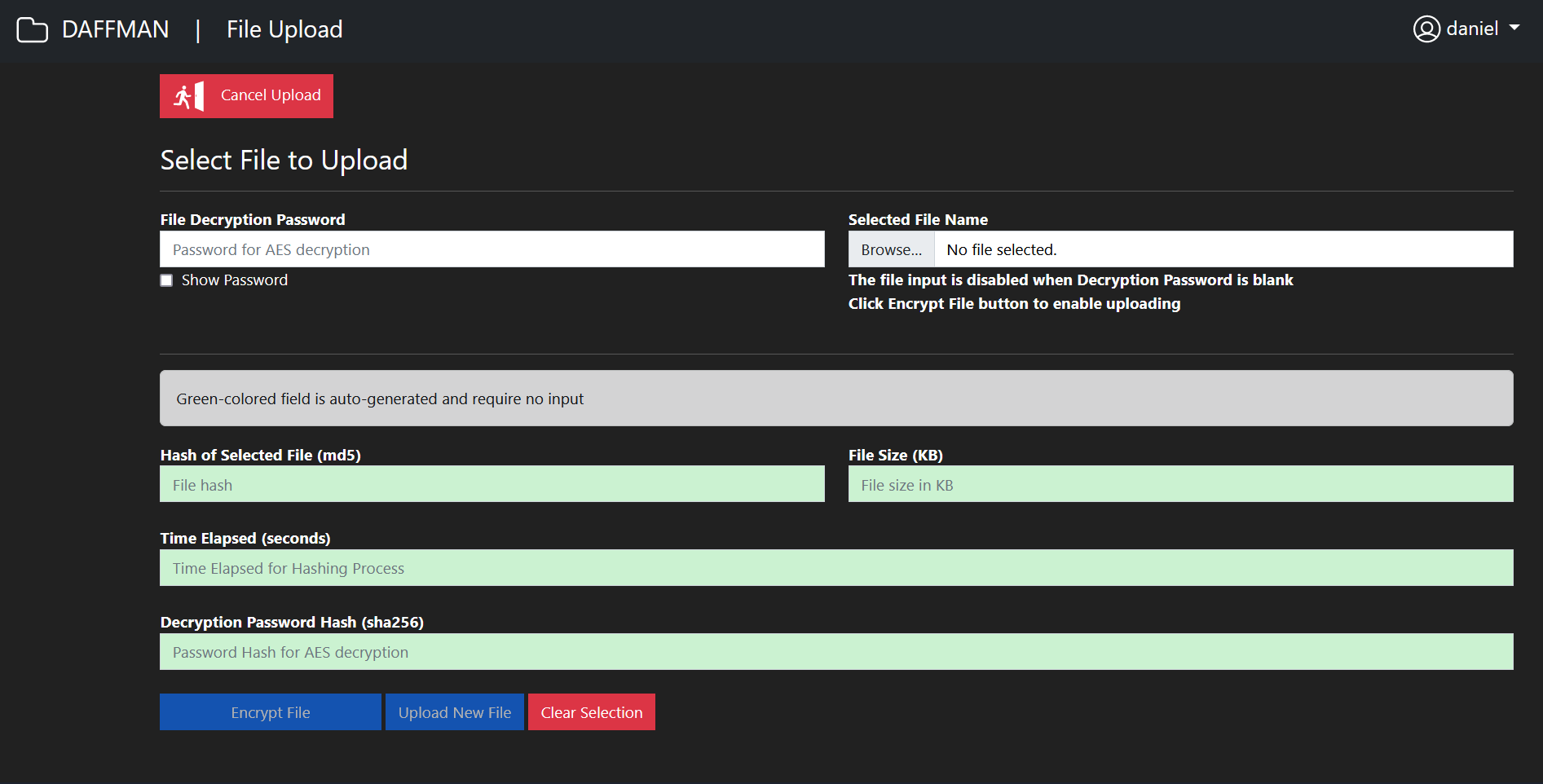




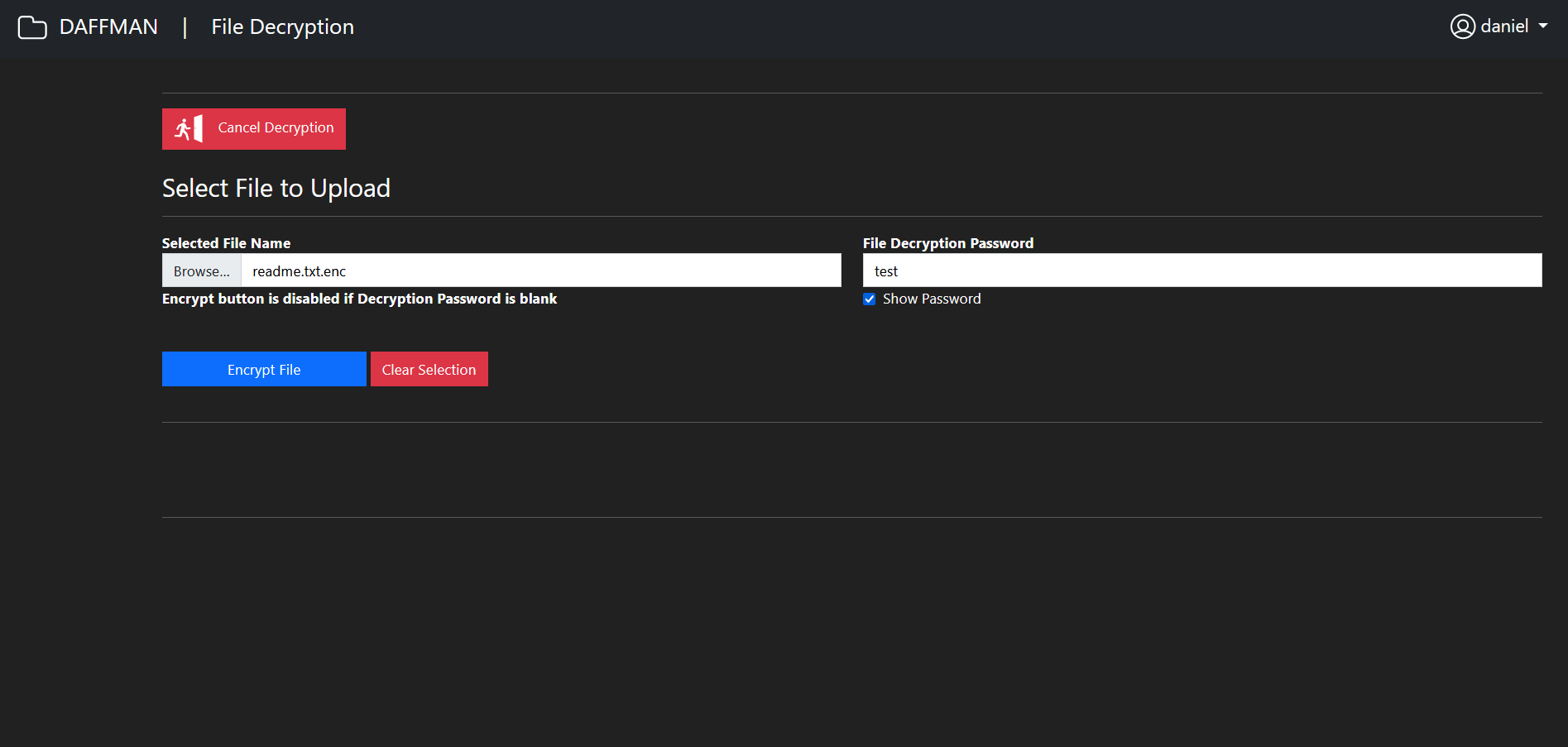
1. File Download popup page
   * The download popup will appear in the File Browser page once the Download button is clicked
   * User will enter the corresponding file password before being permitted to retrieve the file from the server.



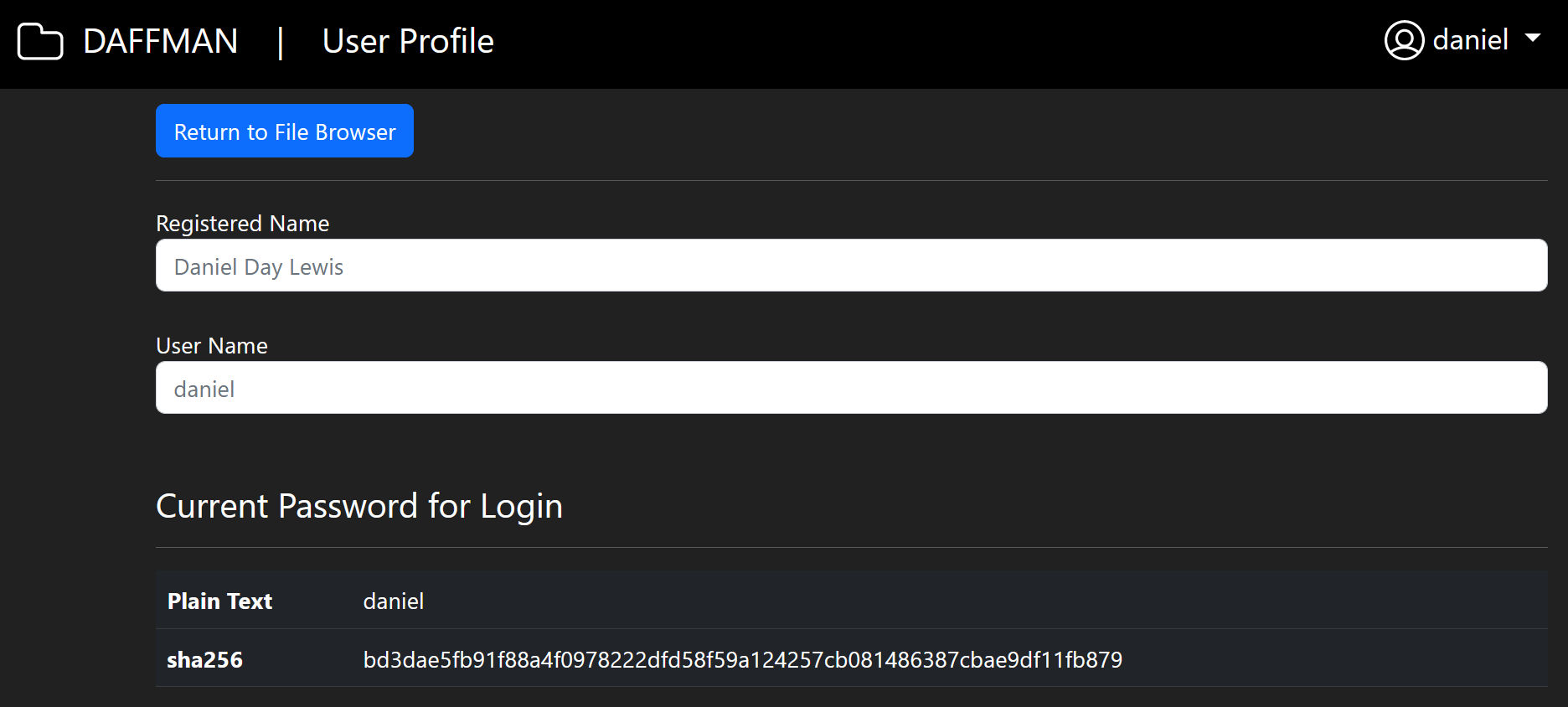
1. File Upload page
   * File to be uploaded will be selected in this page and the MD5 hash will be generated immediately upon file selection.
   * User will also enter the password needed to encrypt and decrypt the files uploaded.

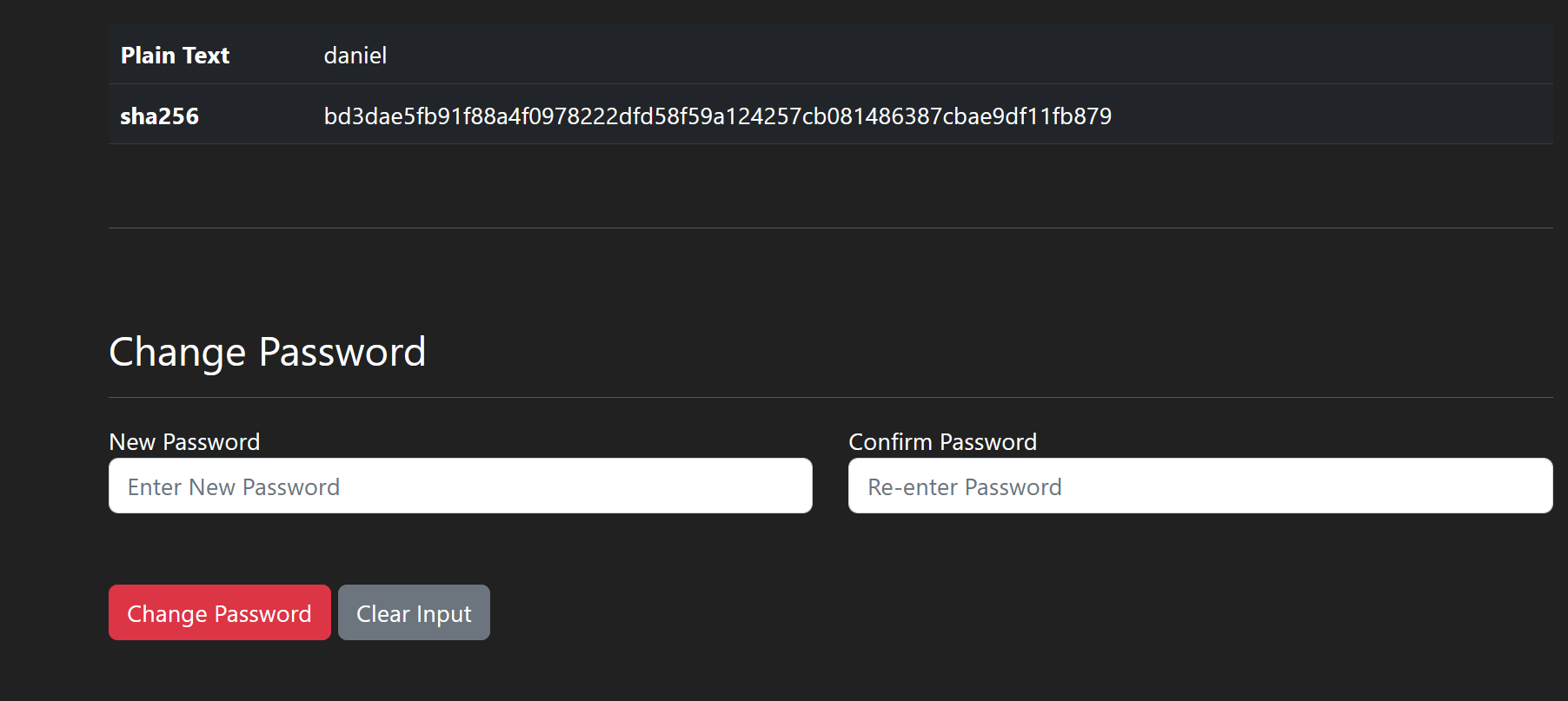


1. File Decryption page
   * Users will decrypt the downloaded file in this page by entering the file password and click Encrypt File
   * If wrong password is entered, the decryption process will still proceed, although corrupted files will be produced. Only correct password will produce a readable and usable decrypted file.

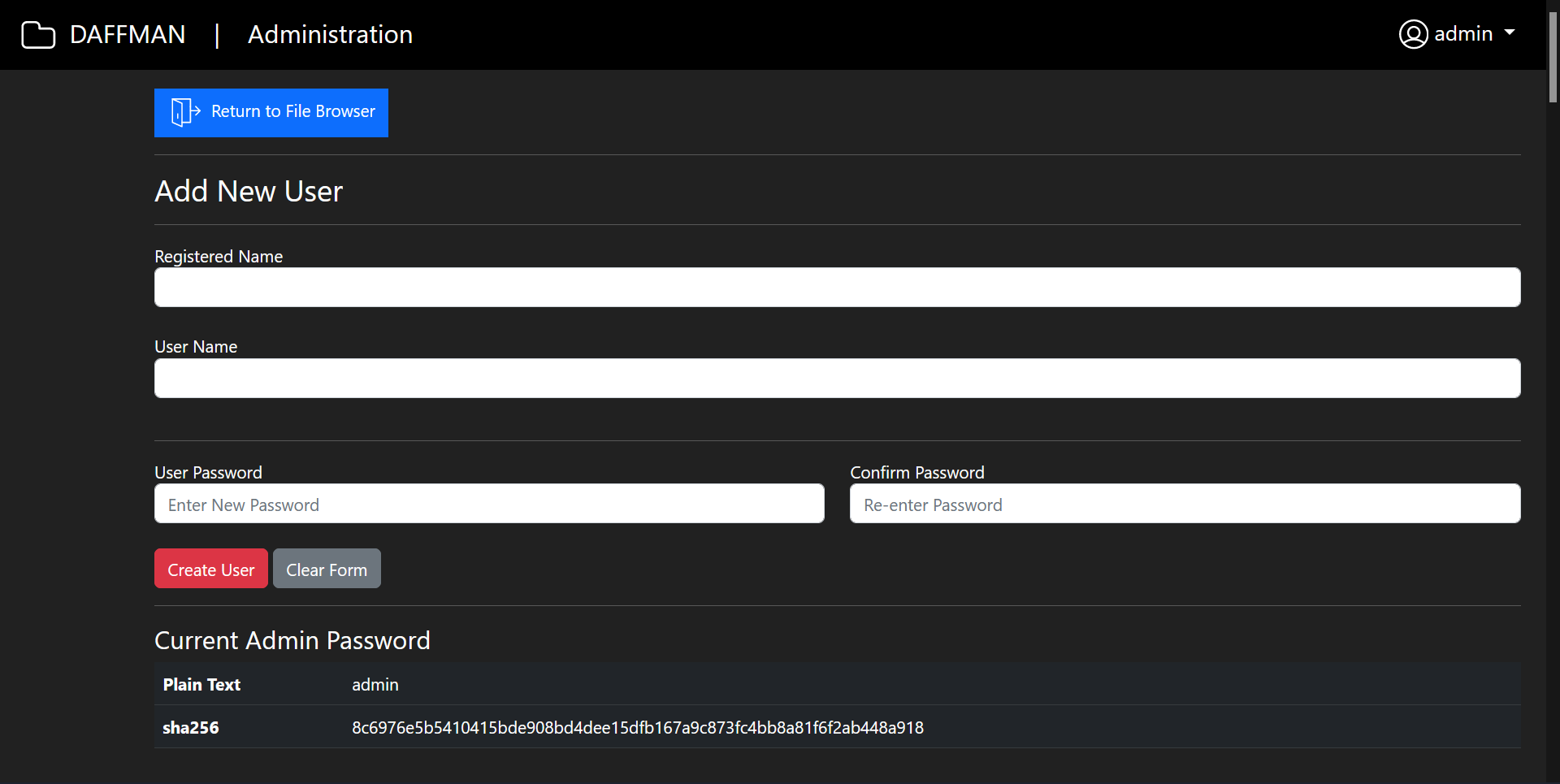
* 

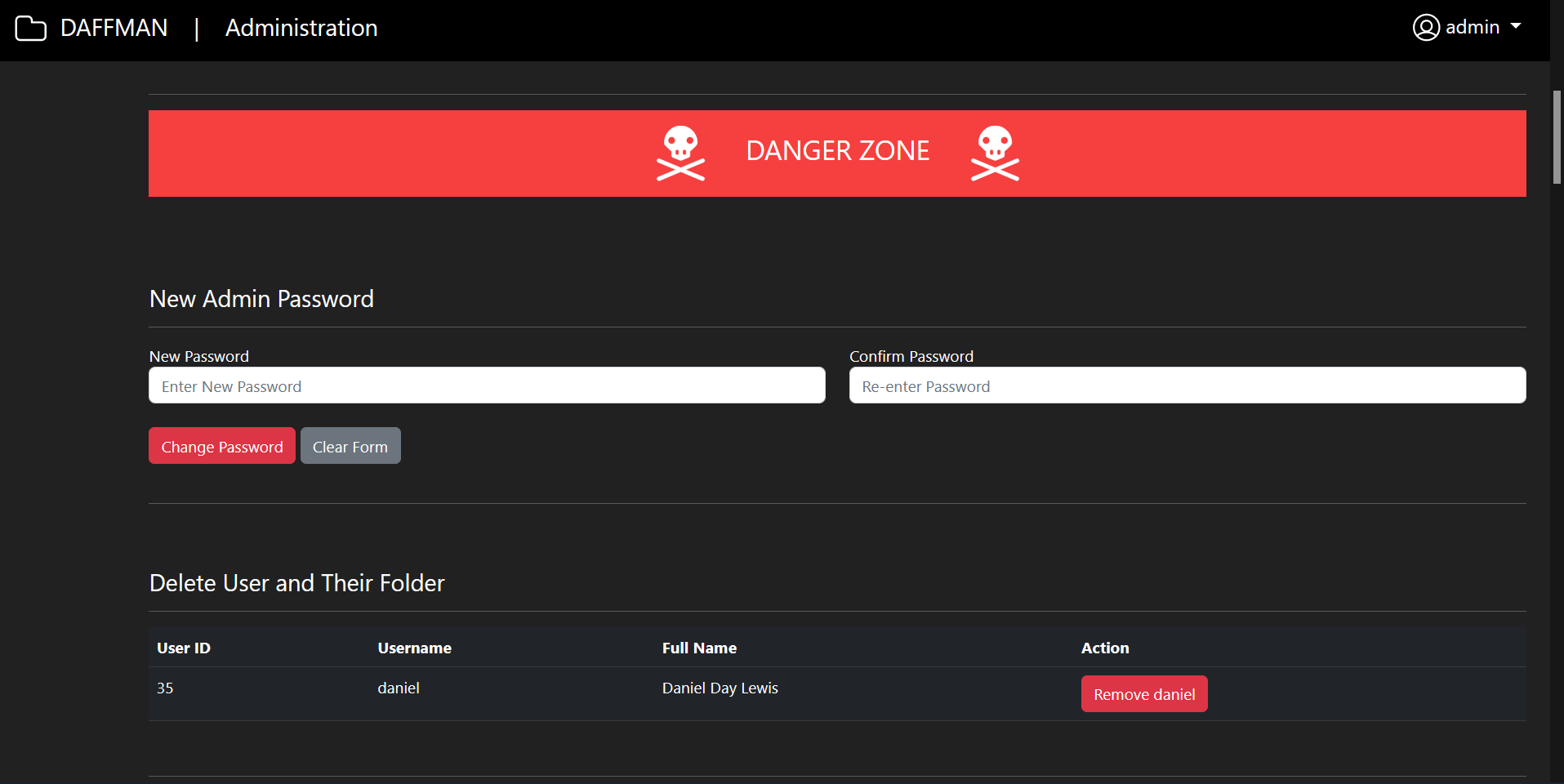
1. Profile page
   * User will use the Profile page to change their login password if needed





1. Administration page
   * Administration page is for admin user only
   * Admin will add new user and password from this page, followed by the creation of user folder in the root folder “/uploads”.
   * Admin will also use this page for user deletion, followed by deletion of the user’s files and folders.





## Appendix D: Ouriginal Report