**A**

**PROJECT REPORT**

**ON**

**FILE INTEGRITY MONITORING**

**Submitted in partial fulfilment of the requirements for the award of the degree**

**of**

**BACHELOR OF TECHNOLOGY HONS**

**IN**

**INFORMATION SECURITY & ASSURANCE**

**BY**

**TINASHE GONDWA (H170192A)**

**Under the guidance of**

**MS. E HUKUIMWE**

**(Lecturer, Department of Information Security & Assurance)**



**School of Information Sciences and Technology**

**Department of Information Security & Assurance**

**Harare Institute of Technology**

**July 2021**

# CERTIFICATE

**This is to certify that research work embodied in this thesis entitled *“File Integrity Monitoring for Host Machines”* was carried out by Tinashe Gondwa (Registration No: H170192A) studying at Harare Institute of Technology for partial fulﬁllment of Bachelor of Technology degree in Information Security and Assurance to be awarded by the Harare Institute of Technology. This research work has been carried out under my guidance and supervision and it is up to my satisfaction.**

**Date: ……………………………………………………………………………………………**

**Place: …………………………………………………………………………………………..**

**Ms. E. Hukuimwe Signature: ……………………………...**

Supervisor & Lecturer from ISA Department, **Stamp**

Harare Institute of Technology,

Harare, Zimbabwe

# COMPLIANCE CERTIFICATE

**This is to certify that research work embodied in this thesis entitled *“File Integrity Monitoring for Host Machines”* was carried out by Tinashe Gondwa (Registration No: H170192A) studying at Harare Institute of Technology for partial fulﬁllment of Bachelor of Technology degree in Information Security and Assurance to be awarded by Harare Institute of Technology. He has completed to the comments given by the Dissertation phase-I as well as Mid Semester Capstone Reviewer to my satisfaction.**

**Date: …………………………………………………………………………………………..**

**Place: ………………………………………………………………………………………..**

**Ms. E.Hukuimwe Signature: …………………………….**

Supervisor & Lecturer from ISA Department, **Stamp**

Harare Institute of Technology,

Harare, Zimbabwe

# PAPER PUBLICATION CERTIFICATE

**This is to certify that research work embodied in this thesis entitled *“File Integrity Monitoring for Host Machines”* was carried out by Tinashe Gondwa (Registration No: H170192A) studying at Harare Institute of Technology for partial fulﬁllment of Bachelor of Technology degree in Information Security and Assurance to be awarded by Harare Institute of Technology has published article entitled *“File Integrity Monitoring for Host Machines”* for publication by the Harare Institute of Technology School of Information Sciences and Technology Journal. (HiTSiSTJ).**

**Date: ……………………………………………………………………………………………**

**Place: …………………………………………………………………………………………..**

**Ms. E. Hukuimwe Signature: ……………………………...**

**Supervisor & Lecturer from ISA Department, Stamp**

**Harare Institute of Technology,**

**Harare, Zimbabwe**

**Signature and Name of Dean**

………………………………………………………………………………………………….

# THESIS APPROVAL CERTIFICATE

**This is to certify that research work embodied in this thesis entitled *“File Integrity Monitoring for Host Machines”* was carried out by Tinashe Gondwa (Registration No: H170192A) studying at Harare Institute of Technology for partial fulﬁllment of Bachelor of Technology degree in Information Security and Assurance to be awarded by Harare Institute of Technology has published the article entitled *“File Integrity Monitoring for Host Machines”* for publication by the Harare Institute of Technology School of Information Sciences and Technology Journal. (HiTSiSTJ)**

**Date: ……………………………………………………………………………………………**

**Place: …………………………………………………………………………………………**

**Examiners Name and Signature:**

**(………………………………………………….…………………………………………)**

# STATEMENT OF ORIGINALITY

WE HEREBY CERTIFY THAT WE ARE THE SOLE AUTHORS OF THIS THESIS AND THAT NEITHER ANY PART OF THIS THESIS NOR THE WHOLE OF THE THESIS HAS BEEN SUBMITTED FOR A DEGREE TO ANY OTHER UNIVERSITY OR INSTITUTION.

WE CERTIFY THAT, TO THE BEST OF OUR KNOWLEDGE, THE CURRENT THESIS DOES NOT INFRINGE UPON ANYONE'S COPYRIGHT NOR VIOLATE ANY PROPRIETARY RIGHTS AND THAT ANY IDEAS, TECHNIQUES, QUOTATIONS OR ANY OTHER MATERIAL FROM THE WORK OF OTHER PEOPLE INCLUDED IN OUR THESIS, PUBLISHED OR OTHERWISE, ARE FULLY ACKNOWLEDGED IN ACCORDANCE WITH THE STANDARD REFERENCING PRACTICES. FURTHERMORE, TO THE EXTENT THAT WE HAVE INCLUDED COPYRIGHTED MATERIAL THAT SURPASSES THE BOUNDARY OF FAIR DEALING. WE CERTIFY THAT WE HAVE OBTAINED A WRITTEN PERMISSION FROM THE COPYRIGHT OWNER(S) TO INCLUDE SUCH MATERIAL(S) IN THE CURRENT THESIS AND HAVE INCLUDED COPIES OF SUCH COPYRIGHT CLEARANCES TO OUR APPENDIX

WE DECLARE THAT THIS IS A TRUE COPY OF THESIS, INCLUDING ANY FINAL REVISIONS, AS APPROVED BY THESIS REVIEW COMMITTEE.

WE HAVE CHECKED WRITE UP OF THE PRESENT THESIS USING ANTI-PLAGIARISM DATABASE AND IT IS IN ALLOWABLE LIMIT. EVEN THOUGH LATER ON IN CASE OF ANY COMPLAINT PERTAINING OF PLAGIARISM, WE ARE SOLE RESPONSIBLE FOR THE SAME AND WE UNDERSTAND THAT AS PER HIT NORMS, THE INSTITUTE CAN EVEN REVOKE BACHELOR OF TECHNOLOGY DEGREE CONFERRED TO THE STUDENT SUBMITTING THIS THESIS.

**Date: ……………………………………………………………………………………………**

**Place: ………………………………………………………………………………..…………**

**…………………………………. ………………………………….**

Tinashe Gondwa (H170192A), Ms. E. Hukuimwe

Harare Institute of Technology, Harare Institute of Technology

Harare, Zimbabwe Harare, Zimbabwe

# Acknowledgements

**Table of Contents**

[CERTIFICATE ii](#_Toc75699646)

[COMPLIANCE CERTIFICATE iii](#_Toc75699647)

[PAPER PUBLICATION CERTIFICATE iv](#_Toc75699648)

[THESIS APPROVAL CERTIFICATE vi](#_Toc75699649)

[STATEMENT OF ORIGINALITY vii](#_Toc75699650)

[Acknowledgements viii](#_Toc75699651)

[CHAPTER 1: PROPOSAL 1](#_Toc75699652)

[1.1. Introduction 1](#_Toc75699653)

[1.2. Background 1](#_Toc75699654)

[1.2.1. Overview 1](#_Toc75699655)

[1.2.2. FIM Global Market Share 2](#_Toc75699656)

[Global FIM Market: Segmentation Analysis 2](#_Toc75699657)

[1.3. Problem Statement 3](#_Toc75699658)

[1.4. Aims 3](#_Toc75699659)

[1.5. Objectives 4](#_Toc75699660)

[1.6. Signiﬁcance of the Project 4](#_Toc75699661)

[1.7. Methodology 4](#_Toc75699662)

[*i.* *Authentication of Users* 4](#_Toc75699663)

[*ii.* *Identification of files to be monitored* 4](#_Toc75699664)

[*iii.* *Creation of Baseline values for files to be monitored* 5](#_Toc75699665)

[*iv.* *Monitoring changes* 5](#_Toc75699666)

[*v.* *Sending an alert* 5](#_Toc75699667)

[*vi.* *Reporting results* 5](#_Toc75699668)

[1.8. Scope of the Project 5](#_Toc75699669)

[*i.* *Host Machine Environments* 5](#_Toc75699670)

[*ii.* *Files and file types* 5](#_Toc75699671)

[1.9. Deﬁnition of Key Variables 6](#_Toc75699672)

[*i.* *Files* 6](#_Toc75699673)

[*ii.* *Environment* 6](#_Toc75699674)

[*iii.* *FIM* 6](#_Toc75699675)

[*iv.* *Baseline Value* 6](#_Toc75699676)

[*v.* *Compromised File* 6](#_Toc75699677)

[1.10. Conclusion 6](#_Toc75699678)

[CHAPTER 2: LITERATURE REVIEW 8](#_Toc75699679)

[2.1. Introduction 8](#_Toc75699680)

[2.2. Synthesis of Literature 8](#_Toc75699681)

[2.3. Conclusion 8](#_Toc75699682)

[CHAPTER 3: REQUIREMENTS ANALYSIS 9](#_Toc75699683)

[3.1. Introduction 9](#_Toc75699684)

[3.2. Current System 9](#_Toc75699685)

[*3.2.1.* *Context Level Diagram* 9](#_Toc75699686)

[*3.2.2.* *Process Flow Diagram* 9](#_Toc75699687)

[*3.2.3.* *Use-case* 9](#_Toc75699688)

[3.3. Feasibility Study 9](#_Toc75699689)

[*3.3.1.* *Technical Feasibility* 9](#_Toc75699690)

[*3.3.2.* *Economic Feasibility* 10](#_Toc75699691)

[3.4. Requirements Analysis 10](#_Toc75699692)

[3.4.1. Functional Requirements 11](#_Toc75699693)

[3.4.2. Context Level DFD 11](#_Toc75699694)

[3.4.3. DFD Level 1 11](#_Toc75699695)

[3.4.4. DFD Level 2 (Optional) 11](#_Toc75699696)

[3.4.5. Use-case 11](#_Toc75699697)

[3.4.6. Non-functional Requirements 12](#_Toc75699698)

[3.5. Interface Requirements 12](#_Toc75699699)

[3.6. Technical Requirements 12](#_Toc75699700)

[3.7. Assumptions 13](#_Toc75699701)

[3.6 Conclusion 13](#_Toc75699702)

[CHAPTER 4: DESIGN 14](#_Toc75699703)

[4.1. Introduction 14](#_Toc75699704)

[4.2. Proposed Solution 14](#_Toc75699705)

[4.3. Solution Architecture 14](#_Toc75699706)

[4.4. Constraints 14](#_Toc75699707)

[4.5. Security Design 14](#_Toc75699708)

[4.6. Systems Design Models 14](#_Toc75699709)

[4.6.1. UML-Activity Diagram 14](#_Toc75699710)

[4.6.2. UML-Class Diagram 14](#_Toc75699711)

[4.6.3. UML-Sequence Diagram 14](#_Toc75699712)

[4.6.4. UML-Deployment Diagram 14](#_Toc75699713)

[4.7. Database Modelling 14](#_Toc75699714)

[4.7.1. E-R Diagram 14](#_Toc75699715)

[4.7.2. Data Dictionary 14](#_Toc75699716)

[4.7.3. Relational Schema 14](#_Toc75699717)

[4.8. Algorithm Design 14](#_Toc75699718)

[4.9. Interface Design 14](#_Toc75699719)

[4.10. Conclusion 14](#_Toc75699720)

[Chapter 5 Implementation 15](#_Toc75699721)

[5.1 Introduction 15](#_Toc75699722)

[5.2 Coding Conventions 15](#_Toc75699723)

[5.3 Coding Strategy 15](#_Toc75699724)

[5.4 Coding Review 15](#_Toc75699725)

[5.5 Conclusion 15](#_Toc75699726)

[Chapter 6 Systems Testing 16](#_Toc75699727)

[6.1 Introduction 16](#_Toc75699728)

[6.2 Testing Categories and Results 16](#_Toc75699729)

[6.2.1 White Box 16](#_Toc75699730)

[6.2.2 Black Box 16](#_Toc75699731)

[6.3 Types of Testing and Results 16](#_Toc75699732)

[6.3.1 Functional Testing 16](#_Toc75699733)

[6.3.2 Non-Functional Testing 16](#_Toc75699734)

[6.4 Test Cases 16](#_Toc75699735)

[6.5 Levels of Testing and Results 16](#_Toc75699736)

[6.5.1 Unit Testing 16](#_Toc75699737)

[6.5.2 Integration Testing 16](#_Toc75699738)

[6.5.3 Validation Testing 16](#_Toc75699739)

[6.5.4 Systems Testing 16](#_Toc75699740)

[6.5.5 Acceptance Testing 16](#_Toc75699741)

[6.6 Security Testing 16](#_Toc75699742)

[6.6 System Evaluation 16](#_Toc75699743)

[6.8 Conclusion 16](#_Toc75699744)

[Chapter 7 Conclusion 17](#_Toc75699745)

[7.1 Introduction 17](#_Toc75699746)

[7.2 Scope of Future Work 17](#_Toc75699747)

[REFERENCES 18](#_Toc75699748)

List of Figures

List of Tables

Abbreviations

Abstract

# CHAPTER 1: PROPOSAL

## Introduction

File integrity monitoring (FIM) is simply the technique of keeping watch of the data, who accesses and makes changes to it as well as how it has been changed. Integrity is one of the pillars of the CIA triad of information security; hence the scope of this project is cyber security. To better understand this concept, the phrase can be broken down into individual words – file, integrity and monitor. The term *file* refers to data or information that is actually stored on the computer or network. The *integrity* of the file means that it has not been maliciously altered or modified. *Monitoring* refers to constantly checking the file to see whether there have been any modifications done to it. This check will verify whether the integrity of the data is maintained or not [1].

## Background

### Overview

In the business world, the most important assets to any firm are the records of its clients. Hackers are tirelessly working towards illegally gaining access to and modifying these critical assets. Cyber security personnel are therefore devising counter-measures to detect these intrusions and contain the impact of any malicious modification. FIM is an internal control or process that performs the act of validating the integrity of operating system and application software files using a verification method between the current file state and a known, good baseline. This comparison method often involves calculating a known cryptographic checksum of the file's original baseline and comparing with the calculated checksum of the current state of the file. Other file attributes such as timestamps can also be used to monitor integrity. [2] says the general act of performing file integrity monitoring is automated using internal controls such as an application or process and that such monitoring can be performed randomly, at a defined polling interval, or in real-time.

The FIM solutions also involve examination of the files and displaying information about when and how it was changed and who changed it, as well as any further action that can be done to restore those files if modifications are unauthorized.

J. v. Ogden summarises the components of FIM into three distinct features:

1. A Database: This database stores information on the original state of your files and configurations as cryptographic hashes.
2. Agents: These technical components measure your hardware and applications and send data back to your database for comparison.
3. User Interface: This is the visual component of the FIM for administrative users, which serves as the centralized portal for reporting, evaluation, change monitoring, and change control [3].

FIM technology is considered as a major part of cyber security processes and technology, owing to its ability to scan, analyse, and report unexpected changes to important files in an IT environment such as operating system (OS), database, and application software files. [3] elaborates on some of the benefits of FIM such as faster incident response times and real-time, continuous monitoring.

### FIM Global Market Share

According to [4], the File Integrity Monitoring Market was valued at USD 703.98 Million in 2019 and is projected to reach USD 1869.87 Million by 2027, growing at a CAGR of 14.01% from 2020 to 2027. [4] explains how the involvement of bodies such as the PCI-DSS and FISMA has led to FIM becoming one of the major requirements and act as a major factor driving the global file integrity monitoring market growth. They go on to describe how the rapid complexity of cyber-attacks and increasing threats to IT infrastructure are also among some of the major factors driving the global file integrity monitoring market. However, market growth is being hampered by the high costs associated with setting up advanced file integrity monitoring solutions and financial constraints among SMEs [4].

### Global FIM Market: Segmentation Analysis

[4] describes how the global FIM market is segmented based on Organisation Outlook, End-Use Outlook and Geography.

1. By Organisation Outlook
2. SMEs
3. Large Enterprises

Cutting-edge FIM solution is costly in terms of development and this has become a major restraint for the growth of the market. The high cost of Research and Development (R&D) expenses needed to develop advanced FIM solutions has led to high pricing of the security solutions. As the frequency of security breaches has increased over the past 5 years, organizations have increased their IT security investments to protect against advanced threats [4].

1. By End-Use Outlook
2. BFSI
3. Government
4. Healthcare
5. Education
6. IT & Telecom
7. Others

FIM solutions and services are provided to end-users to cater for their specific business requirements, compliance and security needs.

1. By Geography
2. North America
3. Europe
4. Asia Pacific
5. Rest of the world

According to [4], North America is estimated to hold the largest market size. Increasing penetration of the internet and incidents of attacks on enterprise IT infrastructure has driven the need for FIM solutions. Furthermore, rapid economic growth in the developing countries, along with improving regulatory reforms and economic stability is driving the file integrity monitoring market growth in APAC [4].

## Problem Statement

Enterprises firms and institutions store critical information about their customers on their organisational servers. Each server performs some form of response to client or user requests. If a response is given to a malicious user, it could lead to loss of critical data. The need for scalable security monitoring in such dynamic environments has become a critical need.

## Aims

This project aims to provide system and device administrators with a centralised tool where they can be able to monitor the files kept on their computers and network. Each file’s existence or lifetime on the computer can be monitored for any malicious alterations or modifications. Once a file is flagged to have been altered, the system will alert the responsible administrative agent to make changes or take precautionary action against the incident.

## Objectives

1. To generate checksums for files to be monitored
2. To compare the stored checksum with the current checksum
3. To send an alert if the compared checksums are not equal

## Signiﬁcance of the Project

With the rapid increase in workflow in the workspace, there is a huge demand for automation in order to cope with and ease the pressure on responsible personnel. This solution does that by automating the whole process of monitoring a host machine’s files and triggering the necessary action when a compromised file is detected. Continuous monitoring means that any malicious modification is quickly detected, hence data is kept in its original state.

## Methodology

This section describes and clarifies the entire FIM process from the initial setup stages to the point where a file is flagged as compromised and recommended action taken to restore the previous state of the file. As mentioned before, FIM is a process and is broken down into small sequential stages as follows:

### *Authentication of Users*

As a basic part of every secure system, we would need to authorize and keep track of users and their activities. The first step is to have responsible staff members register to use the system. Having users sign up onto the system allows for analytics at a later stage in the project.

### *Identification of files to be monitored*

Once a user has been granted access into the system, the next step is for them to identify the files they would like the system to monitor. As different departments have different personnel handling different workloads and files, this step would need to be divided into departmental level. Responsible personnel locate the directory that they would like for the tool to monitor. However, the overall FIM process would need to be overseen by the IT engineer responsible for managing the whole process.

### *Creation of Baseline values for files to be monitored*

Before they can be actively monitored for changes, the files must have an initial baseline based on their current state. The system therefore generates hashes for the files within the specified location. This is achieved by performing an SHA256 Hash calculation for each individual file. In addition, the creation date and modification date of the file will also be taken into account.

### *Monitoring changes*

With a detailed baseline, we can proceed to monitor all designated files for changes. A regular interval after which a system scan will be carried out will be set, for example a minute.

### *Sending an alert*

This step is dependent on the results from the scan in (*iv)*. If a file’s scanned SHA256 checksum calculated in *(iv)* is different from that generated in *(iii)*, it means that the file has been modified, in which case the system will send an alert to the user that a change has been detected.

### *Reporting results*

For analytics purposes, the system will log all activities and allow for a report of user activity to be generated.

## Scope of the Project

It should be taken into consideration that the project is built after the following factors:

### *Host Machine Environments*

This project is limited to monitoring files based on the host machine on which the tool will be installed. It does not focus on cloud-based files or server files that are hosted elsewhere. Administration and monitoring is limited only to the host machine or environment in which the software tool is running.

### *Files and file types*

Since, when a file is changed or updated, its hash value changes, the project focuses on files that are considered as archive files, whose information need not be changed. If archived file’s hash values are changed, then it means the files have been tampered with thus meaning the files have been compromised. For dynamic files, the system would only generate false positives as each alteration or modification of the original document would result in a different baseline value for the file, hence triggering an alert.

## Deﬁnition of Key Variables

### *Files*

A ‘file’ in this project refers to any document or database record contained and stored in the host machine. These files are organisational documents that are of utmost importance to them and can be of, but not limited to, the following file extensions: ***.pdf, .docx, .xlsx, .sqlite3, .py****.*

### *Environment*

The environment refers to the platform on which the system will run. We will create a virtual environment using Python in which Django, a Python Web Framework, and other dependencies and libraries, will be installed.

### *FIM*

FIM is an abbreviation that stands for File Integrity Monitoring. This refers to the whole process of tracking the integrity of files contained in a specified directory in the host machine by constantly comparing the current hash value against a baseline value already saved in the database.

### *Baseline Value*

The baseline value is the unique SHA256 checksum that is assigned to each file within the monitored directory. The stored baseline will be compared with the current checksum at any given time. If the file’s contents are changed, the hash value changes and will therefore be different from the saved baseline value meaning the file will have been compromised.

### *Compromised File*

A compromised file is one that has been maliciously accessed and had its contents modified. For the purposes of this project and in order to prevent false positives, we will be focusing on archived records whose baseline value is set to be constant because there would be no need to alter their contents.

## Conclusion

This chapter sought to summarise the research problem and describe the proposed solution, its functionality and how it is intended to solve the existing problem. Having analysed the current technologies in the FIM industry and the gap that has been identified, it can be concluded that a more viable, robust and convenient solution is needed, hence this FIM tool.

# CHAPTER 2: LITERATURE REVIEW

## Introduction

This chapter analyses existing systems and critiques them in order to point out gaps or room for improvement that the proposed system will cover. Its purpose is to then find a resolution for the identified flaws in the current systems.

## Synthesis of Literature

## Conclusion

# CHAPTER 3: REQUIREMENTS ANALYSIS

## Introduction

This chapter entails an analysis of the resources – software, hardware and others – that are needed for the proposed system to function or work properly. It also provides a brief summary of the existent system based on DFDs and Process Flow diagrams.

## Current System

### *Context Level Diagram*

### *Process Flow Diagram*

### *Use-case*

## Feasibility Study

A feasibility study is an analysis of the practicality and viability of a proposed solution while emphasising on the potential hurdles that can be met. This section looks to critique the applicability of the proposed system with respect to various aspects such as profitability, technicality and whether it is economic.

### *Technical Feasibility*

#### *Hardware Requirements*

The hardware specifications for this project are a development PC with a minimum of 2GB installed RAM, a Core i3 Processor with processing speeds of 1.5GHz and 60GB of disk space. The table below illustrates a comparison of the minimum, recommended and available hardware requirements for the system to run.

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Minimum** | **Recommended** | **Available** |
| **Installed RAM** | 2GB | 8GB | 4GB |
| **Processor** | Core i3 @ 1.5GHz | Core i7 @ 3.0GHz | Core i5 @ 2.5GHz |
| **Hard Disk** | 60 GB | 150GB | 1TB |

Technically speaking, in terms of hardware, the proposed project is feasible since the available hardware surpasses the minimum hardware required for it to run.

#### *Software Requirements*

The following software requirements are needed for the project to begin running:

Any Python 3 version, which is the programming language used to develop the system. Django, the Python Web Framework (and associated libraries to be installed using the pip command), Hyper-Text Mark-up Language (HTML), Cascading Style Sheets (CSS) and JavaScript (JS), the front-end technology for User Interface design, and Visual Studio Code, a text editor, for writing the code, as well as any modern browser for displaying the rendered results obtained or retrieved from the database.

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Minimum** | **Recommended** | **Available** |
| **Python** | 3.1.0 | 3.8.2 | 3.8.2 |
| **Text Editor** | Notepad++ | Visual Studio Code | Visual Studio Code |
| **Web Browser** | Microsoft Edge | Mozilla Firefox | Mozilla Firefox |

The recommendations above are given with particular concern on the RAM of the PC in use. They require less RAM to run and are therefore ideal for any PC with minimal hardware requirements.

### *Economic Feasibility*

This section analyses the development cost of the project with regards to the purchase of the tools that are required for the entire development process as well as other additional costs. It tries to assess the cost-benefit of the given project.

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Quantity** | **Cost** | **Capital Cost** |
| Samsung Laptop | 1 | Free | Researcher |
| Python Software | 1 | Free (Open Source) | Freeware(Open License) |
| Visual Studio Code | 1 | Free (Open Source) | Freeware(Open License) |

## Requirements Analysis

Requirements analysis entails the process of evaluating whether customer expectations for a proposed system are achievable. It elaborates on the services that the system has to provide while highlighting constrains that need to be accounted for during operation.

### Functional Requirements

Functional requirements describe the services that the proposed system is supposed to provide; how it should respond to user inputs as well as its expected behaviour in particular scenarios.

The following are the proposed system’s expected functional requirements:

#### *Generation of hashes*

* The system should be able to generate hash values or checksums using the SHA256 algorithm, for every file in the specified directory.

#### *Storage of hashes*

* The system should be able to push the generated hash value, along the file’s associated properties, to the database.

#### *Retrieval and display of hashes*

* The system should be able to retrieve the stored information from the database and display it on the User Interface.

#### *Comparison of hashes*

* The system should perform a scan and compared the stored checksum of a file against the checksum of the file’s current state.

#### *Alerting*

* The system is supposed to send an alert to the user if a file’s checksum has changed.

#### *User Registration and Login*

* The system should allow for users to sign up for new accounts and login to already existing accounts.

### Context Level DFD

### DFD Level 1

### DFD Level 2 (Optional)

### Use-case

### Non-functional Requirements

These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process, and constraints imposed by standards. Non-functional requirements often apply to the system as a whole, rather than individual system features or services.

#### *Performance*

The code base of the system allows for robust performance by providing features such as minimal page loading time.

#### *Usability*

The system has a User-Friendly interface that is easy to use and navigate around.

#### Security

With regards to security, the system will be able to provide the following features:

##### *Authentication*

* The system will authenticate users and only authenticated users be granted access to particular features.

##### *Encryption*

* It will store passwords securely using the SHA1 encryption algorithm, bundled together with the Django Framework. The files will be encrypted using the SHA256 cryptographic hash function.

### Interface Requirements

The system interface will offer ease of use for all members who will be using it.

* The interface will use breadcrumbs to guide the user of the current page or path they have navigated to
* Once a user is authenticated, the interface must provide all the navigation options that are available to them.

### Technical Requirements

#### *Hardware Requirements*

* 2GB RAM(minimum)
* Core i3 processor
* At least 60GB Hard disk

#### *Software Requirements*

* Visual Studio Code
* Windows Operating System
* Python
* Django
* Web browser

## Assumptions

The following assumptions were taken into consideration during the design and development of the project:

* The folder from which the cryptographic hash values are to be generated is predefined
* The members who are signing up onto the system are part of one department or organization

## Conclusion

This chapter focused on establishing the feasibility of the proposed system with respect to the available technological and economic limitations. It also discussed the requirements analysis in order to ascertain the capacity of the available resources to support the project in terms of technical and non-technical requirements.

# CHAPTER 4: DESIGN

## Introduction

## Proposed Solution

## Solution Architecture

## Constraints

## Security Design

## Systems Design Models

### UML-Activity Diagram

### UML-Class Diagram

### UML-Sequence Diagram

### UML-Deployment Diagram

## Database Modelling

### E-R Diagram

### Data Dictionary

### Relational Schema

#### 1st Normal Form

#### 2nd Normal Form

#### 3rd Normal Form(optional)

## Algorithm Design

## Interface Design

It centres around envisioning what clients may essential to do and guaranteeing that the UI has mechanisms that are easy to access, understand, and use to inspire those actions.

Python was used in the whole development of this web base application using the flask web framework. Since it’s a microframework it provides a light weight application that is scalable and interoperable over many devices with little or no overhead on resources and data. To design forms, it implements Jinja and WTF forms. Jinja allows one to take advantage of templates already in the framework to design html pages. This means the developer gets more time in developing logic of the application rather than basic html tagging and CSS of the application.

The microweb framework also comes with its own development server and is ideal for rapid prototyping, ensuring that each change made by the developer can be realised instantly. When designing an interface there are various important concepts that must be considered.

These include:

* **User Familiarity**

Exploit user familiar understanding, helping them to get oriented with the unfamiliar material more quickly.

* **Consistency**

The interfaces should have the same look and feel as this helps avoid confusion. This aids learning and memorizing how to use the system.

* **Minimal Surprise**

The user should never be taken aback by the behaviour of a system.

* **Recoverability**

The system should handle error handling and have limitations that help avoid the user from doing any damage to the system.

* **User Guidance**

There should also be some form of help documentation which the user can refer to when they need additional information.

Below are mock designs of the final application. This is to help in developing the final product for a swift prototyping to get quick user feedback regarding the final product.

PROVIDE SCREENSHOTS HERE

## Conclusion

1. This marks the end of the system design, following the designed mention in the whole chapter the proposed system can be implemented and tested to verify and validate if it addresses the specific objectives it is designed for. The designing phase managed to helped in classifying and positioning the program modules to produce the ultimate system interfaces, data storage structures and output formats.

# CHAPTER 5 IMPLEMENTATION

## 5.1 Introduction

This chapter seeks to describe how implementation and evaluation process can be conducted on the proposed system. In a nutshell, it describes the process of putting a decision or plan into effect; executing and putting emphasis on the actual coding of the proposed system. It entails the development platforms that were used in the development of the project. In this case a Django environment was utilized and the SQLite3 database was responsible for data storage. This phase in software development makes sure that the proposed system is being developed within the coding principles and the user objectives are being met. Also, the debugging of the code is conducted in this phase to reduce the amount of errors in the system.

## 5.2 Coding Conventions

These are a set of guiding principle for a specific programming language that endorse programming style, practices, and approaches for each characteristic of a program developed in that language. These conventions typically cover file and folder structure, indentation, comments, declarations, statements, white space, naming conventions, program design practices, programming principles, programming rules of thumb, architectural best practices. These are guiding principle for software operational quality. Software developers are extremely suggested to follow these rules to help improve the readability of their source code and make software maintenance easier.

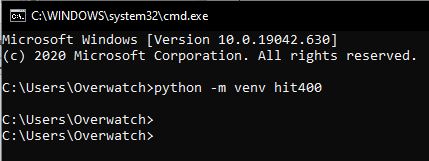
The proposed system follows the shared network architecture.

For the Web Platform the following technologies were implemented

1. JavaScript
2. Bootstrap (HTML5 and CSS3)
3. Python (Django Framework)

venv

This is a tool in python used to create a virtual workspace in which your applications can run without interacting with the actual files of the system environment. The researcher used this tool to create a virtual workspace in which all of the software and libraries required for the project were to be installed.



* **Python uses Python Enhancement Proposals 8**

Python Enhancement Proposals 8 is the bench mark code practice guide intended for Python and it mentions the following:

* **Single declaration per line:**

It is of good practice to have separate code declarations on distinct lines of code. This makes your code more readable and allows other developers who want to modify your code to clearly understand its flow from one declaration to another.

* **Verify value of a constant to that of a variable:**

Instead of expressly comparing value to a constant, just do the equating or evaluation in an if statement. Take advantage of these conditional statements to evaluate your values.

* **Dictionary elements access:**

Try not to utilize the dict.has\_key () technique. Rather, utilize x in d language structure, or pass a default contention to dict.get ().

* **List Manipulation:**

Use the map () and filter functions to manipulate lists. These two functions use a very base standard syntax.

* **File Read:**

Utilize with open functions to open, write and read files. Automatically the files are closed after accessing them which is secure.

* **Continuation of Lines:**

When a rational line of code is extended than acknowledged limit, there is reason towards dividing it over many physical lines.

* **Indentation:**

use tab or four spaces to indent your code. This is meant for readability. Failure can result to run time errors.

* **Value return:**

At the point when a function develops in unpredictability it isn't phenomenal to utilize numerous return declarations inside the function's body. Nevertheless, to retain a vibrant determination and a maintainable readability level, it is desirable over avoid from returning significant qualities from numerous output points in the body

* **Unloading:**

There is flexibility to allocate titles to elements with unpacking assuming you are aware of tuple or list length.

## 5.3 Coding Strategy

Since coding is the core method that is implementation to allow basic communication between humans and machines and strategy being a plan of action designed to achieve a desired objective. Thus, coding strategy is a plan of action taken in the development of the proposed platform. Since the size of the proposed system was a bit extensive, thus plan evolve from a simple command to a mind plan of commands, which later developed into tasks lists. This in turn mutated into functions list, and which later turned into behaviour list which showed how the system is supposed to handle input and process to produce the desired outcome. The whole process is based on perfecting the mistakes that will be made in each stage by evaluating the project scopes. This coding strategy plan size save much time.

This web application was developed using the MVC (Model View Controller) design pattern. This design pattern enables the arrangement of the application source code into loosely coupled objects that help exact initiatives as separate.

MODEL

CONTROLLER

VIEW

Retrieve files

Render Data

View Files

Update Model

**Models**

These contribute as data access layers in which data is accessed from the database and is returned in formats usable within the application. For instance, in the designed application, it does such models models.py files.

* Authentication Model: which securely executes all the authentic and authorization request to the database queries and the required logic.
* User Model: There are necessary for executing user profile creations, updating and all other related user functionality
* Functional Model: It is responsible for handling the

**Controllers**

Are responsible for taking care of the request, process the information from the models and send the information to views, which then send in the reply.

For example, the developed system has such controllers

* Access Controller: for handling logins and logouts
* User Controller: for handling all the user functionality
* Classifier and Model Controller: for handling the model and classifier functionality

**Views**

are presentation templates that are shown in the reply to the web browser, in this case the views were coded in python embed in HTML.

For example, the developed system has view such as,

* Student request for a certificate

**Development Methodology**

This project will implement Agile software development life cycle. It gathers information from different sources and user experiences. This form of software development makes sure the users are involved in the overall process of development. This means the final product to be deployed would have catered for all user requirements

|  |  |
| --- | --- |
| Module | Webpages |
| Platform | Ubuntu Server 18.04.2 LTS |
| Application Server | Flask Development Server |
| Programming | Python,HTML,JAVASCRIPT |
| IDE | Visual Studio Code |
| Methodology | Agile software Development |
| Coding pattern | MVC |
|  |  |
| Module | Model |
| Platform | Ubuntu Server 18.04.2 LTS |
| Framework | Bitcoin Library |
| Database | MongoDB |
| Application Server | Bitcoin core testnet3 Server |
| Programming Language | Python |
|  |  |

## 5.4 Coding Review

This is a systematic inspection of software source code; it is mainly done so as to hunt for and find bugs. This is done to evaluate the source code quality. It is an exercise that is implemented in software development with reason of purifying the quality of the final product software by human review of the source code.

Code Review Checklist

|  |  |
| --- | --- |
| **General** |  |
| * Does the code work? Does it perform its intended function, the logic is correct etc. | YES |
| * Is all the code easily understood? | NO |
| * Does it conform to your agreed coding conventions? These will usually cover location of braces, variable and function names, line length, indentations, formatting, and comments. | YES |
| * Is there any redundant or duplicate code? | YES |
| * Is the code as modular as possible? | PARTIALLY |
| * Can any global variables be replaced? | YES |
| * Is there any commented out code? | YES |
| * Do loops have a set length and correct termination conditions? | YES |
| * Do the names used in the program convey intent? | YES |
| **Performance** |  |
| * Are there any obvious optimizations that will improve performance? | YES |
| * Can any of the code be replaced with library or built-in functions? | NO |
| * Can any logging or debugging code be removed? | YES |
| **Security** |  |
| * Are all data inputs checked (for the correct type, length, format, and range) and encoded? | YES |
| * Are output values checked and encoded? | YES |
| * Are invalid parameter values handled? | YES |
| **Documentation** | YES |
| * Do comments exist and describe the intent of the code? | YES |
| * Are all functions commented? | YES |
| * Is the use and function of third-party libraries documented? | NO |
| * Are data structures and units of measurement explained? | YES |
| * Is there any incomplete code? If so, should it be removed or flagged with a suitable marker like ‘TODO’? | NO |
| Testing |  |
| * Is the code testable? The code should be structured so that it doesn’t add too many or hide dependencies, is unable to initialize objects, test frameworks can use methods etc. | YES |
| * Do tests exist, and are they comprehensive? | NO |
| * Do unit tests actually test that the code is performing the intended functionality? | NO |
| * Could any test code be replaced with the use of an existing API? | NO |

## 5.5 Conclusion

This chapter discusses and looks at all the coding conventions in a chosen programming language. The conventions are communicated to developers and afterwards a review is performed to verify if all developers followed one strategy. If not, code refactoring is done to ensure coding conventions are met; this may be done using a coding review check list by a senior programmer.

# CHAPTER 6 SYSTEM TESTING

## 3.1. Introduction

This is a procedural execution of the proposed system in anticipation of finding the software bugs related to the system. It can also be described as the technique practised to prove and check that the system in hand actually meets all the business and specialized prerequisites that guides its configuration and future improvement. It is carried out to test the quality of the software against a number of factors such as reliability and usability. It is mandated that this process is executed with great efficiency and within the budget that was stated for the project with considerations on the scheduling limits. In addition, the chapter touches on the best way of implementing the system with respect to the analysis and design presented in the previous chapter.

## 6.2 Testing Categories and Results

### 6.2.1 White Box

This is a type of testing technique that mainly examines program structure and derives test data on the basis of program logic or code. It also referred to names like clear box testing, open box testing, logic-driven testing or path driven testing or structural testing [5].

The sequential steps that are carried out in this method are outlined below:

* Identification of all features, components, and programs to be tested,
* Creation of flow graph, plotting all possible paths in the flow graph,
* Identification of all possible paths from the flow graph.
* Writing test cases for every single path of the flow path.
* Execute, rinse and repeat test cases. [5]

### Black Box

BLACK BOX TESTING, also known as Behavioural Testing is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional [6]. This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. It attempts to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structures or external database access
* Behavior or performance errors
* Initialization and termination errors [6]

The illustration below demonstrates Black Box Testing.

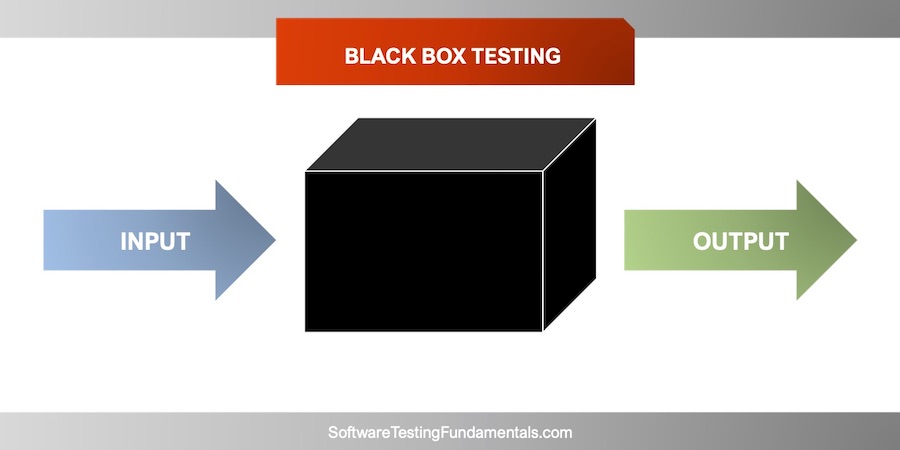


Figure 30:Black Box Testing

### Types of Testing and Results

### 6.3.1. Functional Testing

FUNCTIONAL TESTING is a type of software testing whereby the system is tested against the functional requirements/specifications [7].

Functions (or features) are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application. This type of testing is not concerned with how processing occurs, but rather, with the results of processing. It simulates actual system usage but does not make any system structure assumptions. During functional testing, Black Box Testing technique is used in which the internal logic of the system being tested is not known to the tester Functional testing is normally performed during the levels of System Testing and Acceptance Testing.

Typically, functional testing involves the following steps:

* **Identify** functions that the software is expected to perform.
* **Create** input data based on the function’s specifications.
* **Determine** the output based on the function’s specifications.
* **Execute** the test case.
* **Compare** the actual and expected outputs.

**6.4. Test Cases**

Significant test cases were conducted based on the testing process design. During the project implementation the errors that were discovered were rectified in situ. Below are the comprehensive test cases that were conducted on the developed system:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case** | **Action to execute** | **Expected Outcome** | **Actual Result** | **Comment** |
| **Display User`s Login View** | Open any web browser and type 127.0.0.1:8000 | By default, display User Login View when the current user has no session | By default, display User Login View when the current user has no session | PASSED |
| **User Login into the application** | Prompt user to enter their username and password using the labelled form input fields and press login button | If username/password combination is correct then display the user`s home page. In the case that the credentials were not accepted, display the login view and display the returned message. | If username/password combination is correct then display the user`s home page. In the case that the credentials were not accepted, display the login view and display the returned message. | PASSED |
| **Generate file hashes** | Click Monitor Files, then click on Generate Hashes button | The application should loop through the specified directory and create a hash value for each file, then store it | The application should loop through the specified directory and create a hash value for each file, then store it | PASSED |
| **Verify the hash value of a file** | Click the Verify Hashes button | The application loops through the same folder, generating hash values for the current state of the files | The application loops through the same folder, generating hash values for the current state of the files | PASSED |
| **Sending an email alert** | Click the Verify Hashes button | If there exists a file with a current hash that is different from its stored hash, the application sends an email to the user alerting them of the change | If there exists a file with a current hash that is different from its stored hash, the application sends an email to the user alerting them of the change | PASSED |

### 6.3.2 Non-Functional Testing

This is the type of testing done against the non-functional requirements. Most of the criteria are not consider in functional testing so it is used to check the readiness of a system. Non-functional requirements tend to be those that reflect the quality of the product, particularly in the context of the suitability perspective of its users. It can be started after the completion of Functional Testing. The non-functional tests can be effective by using testing tools. The testing of software attributes which are not related to any specific function or user action like performance, scalability, security or behaviour of application under certain constraints.

Non-functional testing has a great influence on customer and user satisfaction with the product. Non-functional testing should be expressed in a testable way, not like “the system should be fast” or “the system should be easy to operate” which is not testable.

Following testing should consider in non-functional testing types:

* Compatibility testing
* Configuration Testing
* Installation Testing
* Performance testing
* Recovery testing
* Security testing
* Stress testing
* Usability testing

## 6.5 Levels of Testing and Results

Various stages of testing methods were applied to consistently confirm varied viewpoints that would be used to view the system in search of errors. Below are the diverse levels of assessing the intended system applied during the development process of the proposed system.

### 6.5.1 Unit Testing

Unit tests focus on one unit—a program or a program module that performs a specific function that can be tested. The purpose of a unit test is to ensure that the module or program performs its function as defined in the program specification. Unit testing is performed after the programmer has developed and tested the code and believes it to be error free. These tests are based strictly on the program specification and may discover errors resulting from the programmer’s misinterpretation of the specifications. Unit tests are often conducted by the systems analyst or, sometimes, by the programmer who developed the unit. [**21**]

### 6.5.2 Integration Testing

Integration tests assess whether a set of modules or programs that must work together do so without error. They ensure that the interfaces and linkages between different parts of the system work properly. At this point, the modules have passed their individual unit tests, so the focus now is on the flow of control among modules and on the data exchanged among them. Integration testing follows the same general procedures as unit testing: the tester develops a test plan that has a series of tests. Integration testing is often done by a set of programmers and/or systems analysts. There are four approaches to integration testing: user interface testing, use scenario testing, data flow testing, and system interface testing. [**21**]

### 6.5.3 Validation Testing

It puts more emphasis on the “user visible actions and recognizable output from the system. This testing is only considered successful when software functions are in a manner that is expected by the customer”. Two types of validation tests were carried out to the researcher’s satisfaction as follows:

1. Alpha testing, which is operational testing through the use of potential “users or an independent test team at the developers' site”. The second approach which is the use of an independent test team was carried out in the development of this project. The reason for the choice of this testing is based on the software being an off-the-shelf software, thus there is need to undertake this as a form of internal acceptance testing. The system proved to be quite beneficial in identifying the software problems prior to beta testing.
2. Beta testing this is the next testing which is performed “right after alpha testing. Versions of the software (beta versions), were released to a limited audience outside of the programming environment”. The software was released to only two different users and administrator so that further testing could ensure the product had few faults or bugs

### 6.5.4 Systems Testing

System tests are usually conducted by the systems analysts to ensure that all modules and programs work together without error. System testing is similar to integration testing, but is much broader in scope. Whereas integration testing focuses on whether the modules work together without error, system tests examine how well the system meets business requirements and its usability, security, and performance under heavy load. It also tests the system’s documentation. [**21**]

### 6.5.5 Acceptance Testing

Acceptance tests are done primarily by the users with support from the project team. The goal is to confirm that the system is complete, meets the business needs that prompted the system to be developed, and is acceptable to the users. Acceptance testing is done in two stages: alpha testing, in which users test the system using made-up data, and beta testing, in which users begin to use the system with real data and carefully monitor the system for errors. The users’ perceptions of the new system will be significantly influenced by their experiences during acceptance testing. Since first impressions are sometimes difficult to change, analysts should strive to ensure that acceptance testing is conducted only following rigorous (and successful) system testing. In addition, listening to and responding to user feedback will be essential in shaping a positive reaction to and acceptance of the new system by the users. [**21**]

## 6.6 Security Testing

## 6.6 System Evaluation

The developed solution successfully passed the diverse testing paradigms and the sample users accepted system to be, dependable, sustainable, useful solution that provide trusted career insights and advice. All the objectives of this proposed system were met.

## 6.8 Conclusion

This stage helps evaluate the quality of the application. It requires matching results against those anticipated at initial stages. This will help detect mistakes, missing requirements of the client. This stage can be done automatically or using automated tools to make the task easier and faster to complete. Use of automated tools makes it easier for developers to quickly see mistakes and make changes instantly, rather than doing it manually and spending lots of time checking than fixing the actual problem.

# CHAPTER 7: CONCLUSION

## Introduction

This chapter is aimed at drawing conclusions after the completion of the application development process. It also mentions all the constraints and how they can be overcome in the future by anyone who would like to carry this hypothesis further and make improvements.

## Scope of Future Work

The system can be improved to detect file changes not only in their contents (using cryptographic checksums), but also using other properties like Group Properties and file Permissions as well as checking relevant timestamps as to when the file was created, accessed or modified. For a larger setup, it can also be improved give detailed analytics per given time interval.

## Recommendations

I recommend that further research be taken into the integrity of data and information. In reality it is not all organizational files that will be edited while they are in their original folders, hence the need to determine any access into a directory containing the monitored files, not just when they have been modified.

# REFERENCES

|  |  |
| --- | --- |
| [1] | C. Peter, "What Is File Integrity Monitoring and How Does It Work? - DZone Big Data," 25 November 2020. [Online]. Available: https://dzone.com/articles/what-is-file-integrity-monitoring-and-how-it-works. [Accessed 18 June 2021]. |
| [2] | WHOA.com, "File Integrity Monitoring PCI - DSS - FIM - PCI - DSS Compliance | WHOA.com," WHOA.com, 23 October 2013. [Online]. Available: https://www.whoa.com/file-integrity-monitoring. [Accessed 18 June 2021]. |
| [3] | J. v. Ogden, "How Does File Integrity Monitoring Work?," CIMCOR, 14 6 2016. [Online]. Available: https://www.cimcor.com/blog/how-does-file-integrity-monitoring-work. [Accessed 18 6 2021]. |
| [4] | V. M. Research, "Global File Integrity Monitoring Market Size By Organization Outlook, By End-Use Outlook, By Geographic Scope And Forecast," Verified Market Research, Amsterdam, 2021. |