**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](#_Toc75890681)

[COMPLIANCE CERTIFICATE iii](#_Toc75890682)

[PAPER PUBLICATION CERTIFICATE iv](#_Toc75890683)

[THESIS APPROVAL CERTIFICATE v](#_Toc75890684)

[STATEMENT OF ORIGINALITY vi](#_Toc75890685)

[Acknowledgements vii](#_Toc75890686)

[CHAPTER 1: PROPOSAL 1](#_Toc75890687)

[1.1. Introduction 1](#_Toc75890688)

[1.2. Background 1](#_Toc75890689)

[1.2.1. Overview 1](#_Toc75890690)

[1.2.2. FIM Global Market Share 2](#_Toc75890691)

[Global FIM Market: Segmentation Analysis 2](#_Toc75890692)

[1.3. Problem Statement 3](#_Toc75890693)

[1.4. Aims 3](#_Toc75890694)

[1.5. Objectives 3](#_Toc75890695)

[1.6. Signiﬁcance of the Project 3](#_Toc75890696)

[1.7. Methodology 3](#_Toc75890697)

[*i.* *Authentication of Users* 3](#_Toc75890698)

[*ii.* *Identification of files to be monitored* 4](#_Toc75890699)

[*iii.* *Creation of Baseline values for files to be monitored* 4](#_Toc75890700)

[*iv.* *Monitoring changes* 4](#_Toc75890701)

[*v.* *Sending an alert* 4](#_Toc75890702)

[*vi.* *Reporting results* 4](#_Toc75890703)

[1.8. Scope of the Project 4](#_Toc75890704)

[*i.* *Host Machine Environments* 4](#_Toc75890705)

[*ii.* *Files and file types* 4](#_Toc75890706)

[1.9. Deﬁnition of Key Variables 5](#_Toc75890707)

[*i.* *Files* 5](#_Toc75890708)

[*ii.* *Environment* 5](#_Toc75890709)

[*iii.* *FIM* 5](#_Toc75890710)

[*iv.* *Baseline Value* 5](#_Toc75890711)

[*v.* *Compromised File* 5](#_Toc75890712)

[1.10. Conclusion 5](#_Toc75890713)

[CHAPTER 2: LITERATURE REVIEW 6](#_Toc75890714)

[2.1. Introduction 6](#_Toc75890715)

[2.2. Synthesis of Literature 6](#_Toc75890716)

[2.3. Conclusion 6](#_Toc75890717)

[CHAPTER 3: REQUIREMENTS ANALYSIS 7](#_Toc75890718)

[3.1. Introduction 7](#_Toc75890719)

[3.2. Current System 7](#_Toc75890720)

[*3.2.1.* *Context Level Diagram* 7](#_Toc75890721)

[*3.2.2.* *Process Flow Diagram* 7](#_Toc75890722)

[*3.2.3.* *Use-case* 7](#_Toc75890723)

[3.3. Feasibility Study 7](#_Toc75890724)

[*3.3.1.* *Technical Feasibility* 7](#_Toc75890725)

[*3.3.2.* *Economic Feasibility* 8](#_Toc75890726)

[3.4. Requirements Analysis 8](#_Toc75890727)

[3.4.1. Functional Requirements 8](#_Toc75890728)

[3.4.2. Context Level DFD 9](#_Toc75890729)

[3.4.3. DFD Level 1 9](#_Toc75890730)

[3.4.4. DFD Level 2 (Optional) 9](#_Toc75890731)

[3.4.5. Use-case 9](#_Toc75890732)

[3.4.6. Non-functional Requirements 10](#_Toc75890733)

[3.5. Interface Requirements 10](#_Toc75890734)

[3.6. Technical Requirements 10](#_Toc75890735)

[3.7. Assumptions 11](#_Toc75890736)

[3.8. Conclusion 11](#_Toc75890737)

[CHAPTER 4: DESIGN 12](#_Toc75890738)

[4.1. Introduction 12](#_Toc75890739)

[4.2. Proposed Solution 12](#_Toc75890740)

[4.3. Solution Architecture 12](#_Toc75890741)

[4.4. Constraints 12](#_Toc75890742)

[4.5. Security Design 12](#_Toc75890743)

[4.6. Systems Design Models 12](#_Toc75890744)

[4.6.1. UML-Activity Diagram 12](#_Toc75890745)

[4.6.1.1. UML-Class Diagram 12](#_Toc75890746)

[4.6.2. UML-Sequence Diagram 12](#_Toc75890747)

[4.6.3. UML-Deployment Diagram 12](#_Toc75890748)

[4.7. Database Modelling 13](#_Toc75890749)

[4.7.1. E-R Diagram 13](#_Toc75890750)

[4.7.2. Data Dictionary 13](#_Toc75890751)

[4.7.3. Relational Schema 13](#_Toc75890752)

[4.8. Algorithm Design 13](#_Toc75890753)

[4.9. Interface Design 13](#_Toc75890754)

[4.10. Conclusion 14](#_Toc75890755)

[CHAPTER 5 IMPLEMENTATION 15](#_Toc75890756)

[5.1 Introduction 15](#_Toc75890757)

[5.2 Coding Conventions 15](#_Toc75890758)

[5.3 Coding Strategy 16](#_Toc75890759)

[5.4 Coding Review 18](#_Toc75890760)

[5.5 Conclusion 20](#_Toc75890761)

[CHAPTER 6 SYSTEM TESTING 21](#_Toc75890762)

[6.1. Introduction 21](#_Toc75890763)

[6.2 Testing Categories and Results 21](#_Toc75890764)

[6.2.1 White Box 21](#_Toc75890765)

[6.2.2 Black Box 21](#_Toc75890766)

[6.3 Types of Testing and Results 22](#_Toc75890767)

[6.3.1. Functional Testing 22](#_Toc75890768)

[6.3.2 Non-Functional Testing 24](#_Toc75890769)

[6.5 Levels of Testing and Results 24](#_Toc75890770)

[6.5.1 Unit Testing 24](#_Toc75890771)

[6.5.2 Integration Testing 24](#_Toc75890772)

[6.5.3 Validation Testing 25](#_Toc75890773)

[6.5.4 Systems Testing 25](#_Toc75890774)

[6.5.5 Acceptance Testing 25](#_Toc75890775)

[6.6 Security Testing 25](#_Toc75890776)

[6.6 System Evaluation 25](#_Toc75890777)

[6.8 Conclusion 25](#_Toc75890778)

[CHAPTER 7: CONCLUSION 26](#_Toc75890779)

[7.1. Introduction 26](#_Toc75890780)

[7.2. Scope of Future Work 26](#_Toc75890781)

[7.3. Recommendations 26](#_Toc75890782)

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

* SMEs
* 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

System design is conducted in order to clearly develop the architectural design of the proposed system. The system is broken down into single standalone units, with a clear definition of the interfaces between them. The handling and parsing of data for each unit with the system is taken into consideration so as to meet the stated requirements. The user requirements are defined based on which the system will be developed with the intention of satisfying the business needs.

## Proposed Solution

## Solution Architecture

## Constraints

The major constraint with the development of the system is to do with

## Security Design

This is a Secure by design technique that was implemented, in software engineering of the system. This means the system was designed from the roots up to be secure. Much care was taken to lessen the impact when a security vulnerability that can be exploited is discovered.

Least Privilege Design Principle

This is an approach that is guided by the principle relate to minimalism in the provision of access rights to users of the system. On implementing this principle effort were put in place for access rights to be time based. This was done to assure that the resources are access at the time they are necessary only in the completion of the job at hand. The reasons for not giving access outside this space will not permit for pointless access and thereof mitigate against the risk associated with integrity of the data since user can edit or update the data. This principle attempts to mitigate against damage happening by coincidence or error. This is possible by limiting the interaction time the users have with the data or resources.

**Access Control Requirements**

Some of the data stored in the system need to be kept confidential; some data need special controls on who is allowed to change or delete them. Personnel records, for example, should be readable only by the personnel department and the employee’s supervisor; changes should be permitted to be made only by the personnel department. Access control requirements state who can access what data and what type of access is permitted—whether the individual can create, read, update, and/or delete the data. The requirements reduce the chance that an authorized user of the system can perform unauthorized actions.

**Complete Mediation Design Principle**

The principle is based on validating every access to every resource for authorization. This was achieved by using authorization functions on all pages. The authorization functions first check if the user is logged on, after it checks user level and sees if the user is authorized then the resource is presented to the user.

**User Authentication**

The system will allow only authorized users to access their permitted data through the use of user account credentials. Students accounts can only be created by their registered institution administrator and their passwords are to be securely hashed using SHA (256) and salted before being stored into the database.

I implemented validation using FLASK-FORM and FLASK-SQLALCHEMY and FLASK-LOGIN

* **Session based authentication**

Session based authentication is fulfilled entirely by the Flask-Login extension

* **Role management**

Flask principle makes this possible by isolating users’ role when they login to the application

* **Password hashing**

This is handled by bcrypt making it possible to store hashed passwords.

* **Basic HTTP authentication**

Achieved using view methods in actual application decorating in the views.py.

* **Token based authentication**

Usually is permitted by retrieving the user auth token by execution an HTTP POST with the authentication details as JSON data against the authentication endpoint

## Systems Design Models

### UML-Activity Diagram

### UML-Class Diagram

### UML-Sequence Diagram

### UML-Deployment Diagram



## Database Modelling

### E-R Diagram

This type of diagram shows the connection between the entities involved in the system.



### Data Dictionary

### Relational Schema

#### 1st Normal Form

#### 2nd Normal Form

#### 3rd Normal Form (optional)

## Algorithm Design

## Interface Design

Python was used in the whole development of this web based application using the Django web framework. To design forms, Django 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.

When designing an interface there are various important concepts that must be considered.

* *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*

No unusual system behaviours must occur.

* *Recoverability*

The system should be able to handle all forms of errors and have limitations that help avoid the user from doing any damage to the system.

* *User Guidance*

The user should be guided on how the system should be used if ever they get lost while navigating around it.

Below are mock designs of the final application.

PROVIDE SCREENSHOTS HERE

## Conclusion

This chapter dwelt on the design of the proposed system, with particular emphasis on the UML diagrams.

# CHAPTER 5 IMPLEMENTATION

## 5.1 Introduction

This chapter seeks to describe how the implementation and evaluation processes 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.

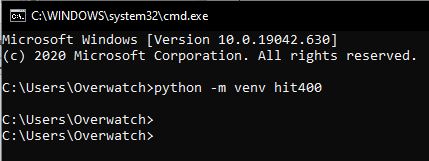
The proposed system follows the shared network architecture.

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 Enhancement Proposals 8

This is a bench mark code practice guide intended for Python and it mentions the following:

* *Single declaration per line:*

It is good practice to have one variable declaration per line. This enables code readability and allows other developers to understand the flow in terms of variables.

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

Evaluation of values and constants should be done in conditional if statements.

* *Dictionary elements access:*

Passing a default contention to dict.get() or utilizing the “x in d” language structure instead of using the dict.has\_key() technique.

* *List Manipulation:*

Using the map () and filter() functions to manipulate lists.

* *File Manipulation:*

Using the context manager (with statement) to open, write to and read from files. The files are automatically closed after accessing them which makes this method 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:*

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

## 5.3 Coding Strategy

Coding is the core method that is implemented to allow basic communication between humans and machines. A strategy is a plan of action designed to achieve a desired objective. Therefore, a coding strategy is a plan of action taken in the development of the proposed platform. The proposed system is relatively small but modular so the plan designed saved time.

This web application was developed using the MVC (Model View Controller) design pattern.

**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 that are usable within the application. For instance, in the designed application, it does such in *models.py* files.

* Authentication Model: responsible for authenticating users on the system
* User Model: There are necessary for executing user account creations, updating and all other related user functionality
* File Model: It is responsible for handling the file table creation

Controllers

They are responsible for taking care of the request, processing the information from the models and sending it to the information to views, which then render the response.

For example, the developed system has such controllers

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

Views

These are presentation templates that are shown in the response to the web browser; in this case the views were coded in python embedded in HTML.

For example, the developed system has view such as a display of all stored file hashes.

Development Methodology

This project will implement Agile software development life cycle which 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, meaning that the final product to be deployed would have catered for all user requirements

|  |  |
| --- | --- |
| **Web Server** | Django Development Server |
| **Programming** | HTML, JavaScript |
| **IDE** | Visual Studio Code |
| **Methodology** | Agile software Development |
| **Coding pattern** | MVC |
| **Module** | Model |
| **Platform** | Windows 10 |
| **Framework** | Django |
| **Database** | SQLite3 |
| **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** | |
| 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 discussed on the coding conventions for 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

## 6.1. 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:

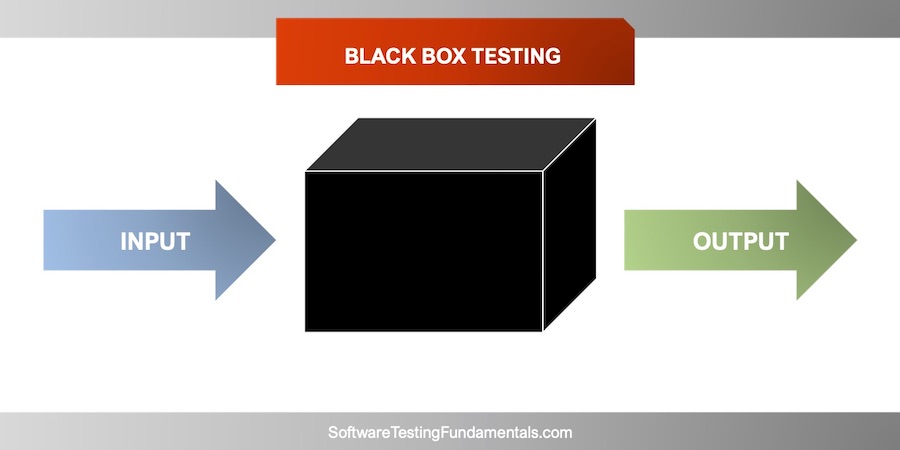
* Identification of all features, components, and programs to be tested,
* Creation of flow graph, plotting all possible paths in the flow graph,
* Identification of all possible paths from the flow graph.
* Writing test cases for every single path of the flow path.
* Execute, rinse and repeat test cases. [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
* Behavioural or performance errors
* Initialization and termination errors [6]

The illustration below demonstrates Black Box Testing.



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 [8]. Functional testing ensures that the application properly satisfies the requirements. It is concerned with the result of processing rather than the processing itself. It mimics actual system usage but does not make any assumptions about the system structure. During this testing, the Black Box Testing technique in which the tester is unaware of the system is used. It is normally performed during the levels of System Testing and Acceptance Testing [8].

The various steps involved in this testing are mentioned below:

* Determining the functionality of the product that needs to be tested - this includes testing the main functionalities, error condition, and messages, usability testing, that is, whether the product is user-friendly or not, etc.
* Creation of input data - for the functionality to be tested as per the requirement specification.
* Determine output for the functionality under test.
* Execution of prepared test cases.
* Comparison of actual output with expected output [9].

**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 this testing 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 [10]. It is usually preceded by Functional Testing. The effectiveness of non-functional tests can be attained by using testing tools. It is used to test software attributes which are not related to any specific function or user action like performance, scalability, security or behaviour of application under certain constraints [10]. It has a great influence on customer and user satisfaction with regards to the product. It should be expressed in a testable way, not like “the system should be fast” or “the system should be easy to operate” ways, which are not testable.

The following methods are considered 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. Shown 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 a single unit — a program or 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. This type of testing is usually performed by the systems analyst or, sometimes, by the programmer who developed the unit.

### 6.5.2 Integration Testing

Integration tests verify that different modules or services used by your application work well together [11]. 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. It follows the same general procedures as unit testing wherein the tester develops a test plan that has a series of tests. It is often done by a set of programmers and/or systems analysts.

### 6.5.3 Validation Testing

Validation testing is the process of ensuring if the tested and developed software satisfies the client /user needs [12]. Two types of validation tests were carried out to the researcher’s satisfaction as follows:

1. Alpha testing – carried out by the developers on the application before release at the software development site. It involves black and white box testing.
2. Beta testing – is carried out at the customer side after the product is developed and deployed [12].

### 6.5.4 Systems Testing

System Testing means testing the system as a whole. All the modules/components are integrated in order to verify if the system works as expected or not. It is done after Integration Testing and plays an important role in delivering a high-quality product. [13].

### 6.5.5 Acceptance Testing

Once the System Testing process is completed by the testing team and signed-off, the entire Product/application is handed over to the customer/few users of customers/both, to test for its acceptability i.e., Product/application should be flawless in meeting both the critical and major Business requirements [14]. The goal is to confirm completeness of the system, whether it meets the business needs, and is acceptable to the users.

## 6.6 Security Testing

Security testing is an integral part of software testing that deals with discovering the weaknesses, risks, or threats in the software application. It also helps in stopping nasty attacks from outsiders, ensuring the security of the software applications.

## 6.6 System Evaluation

The developed system successfully passed the diverse testing methods and the potential users approved the system as dependable and requisite tool for monitoring file changes. All the objectives of this proposed system were met.

## 6.8 Conclusion

This stage dealt with evaluating the quality of the application. It involves matching actual results against those predicted at initial stages and can be done manually or automatically using automated tools to make the task easier and faster to complete.

# 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. |
| [5] | XenonStack, "White Box Testing Techniques and Advantages - XenonStack," XenonStack, 4 December 2018. [Online]. Available: https://www.xenonstack.com/insights/what-is-white-box-testing/. [Accessed 3 June 2021]. |
| [6] | S. T. Fundamentals, "Black Box Testing - SOFTWARE TESTING Fundamentals," SOFTWARE TESTING Fundamentals, 17 September 2020. [Online]. Available: https://softwaretestingfundamentals.com/black-box-testing/. [Accessed 3 May 2021]. |
| [7] | S. T. Fundamentals, "Software Testing Types - Software Testing Fundamentals," Software Testing Fundamentals, 16 September 2020. [Online]. Available: https://softwaretestingfundamentals.com/software-testing-types/. [Accessed 4 May 2021]. |
| [8] | S. T. Fundamentals, "Functional Testing - SOFTWARE TESTING Fundamentals," SOFTWARE TESTING Fundamentals, 7 September 2020. [Online]. Available: https://softwaretestingfundamentals.com/functional-testing/. [Accessed 8 May 2021]. |
| [9] | S. Zalavadia, "Functional Testing: A Complete Guide with Types and Examples," Software Testing Help, 28 June 2021. [Online]. Available: https://www.softwaretestinghelp.com/guide-to-functional-testing/. [Accessed 13 June 2021]. |
| [10] | Kanoah, "Non-Functional Testing," SlideShare, 20 July 2016. [Online]. Available: https://www.slideshare.net/kanoahinc/nonfunctional-testing-64219889. [Accessed 14 June 2021]. |
| [11] | S. Pittet, "The different types of testing in software | Atlassian," Atlassian CI/CD, 15 June 2017. [Online]. Available: https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing. [Accessed 27 June 2021]. |
| [12] | S. T. Help, "Validation Testing Ultimate Guide," Software Testing Help, 28 June 2021. [Online]. Available: https://www.softwaretestinghelp.com/validation-testing/. [Accessed 29 June 2021]. |
| [13] | S. T. Help, "What Is System Testing – A Ultimate Beginner’s Guide," Software Testing Help, 28 June 2021. [Online]. Available: https://www.softwaretestinghelp.com/system-testing/. [Accessed 29 June 2021]. |
| [14] | S. T. Help, "What Is Acceptance Testing (A Complete Guide)," Software Testing Help, 30 May 2021. [Online]. Available: https://www.softwaretestinghelp.com/what-is-acceptance-testing/. [Accessed 14 June 2021]. |
| [15] | J. Dhobale, "System Implemantation," 13 December 2017. [Online]. Available: https://www.slideshare.net/JaipalDhobale/system-implemantation. [Accessed 19 May 2021]. |