CHAPTER 3
METHODOLOGY

CHAPTER 3
METHODOLOGY

3.1 Introduction

This chapter outlines the research approach and methodology used throughout the project to ensure its successful completion. The methodology served as a guideline for the project's development, enabling it to stay on track with its timeline. Additionally, this chapter covered the systematic process of planning, designing, developing, and testing the proposed project **Secure Cloud Storage with Malware Analyzer and AES Encryption**.

3.2 Project Framework

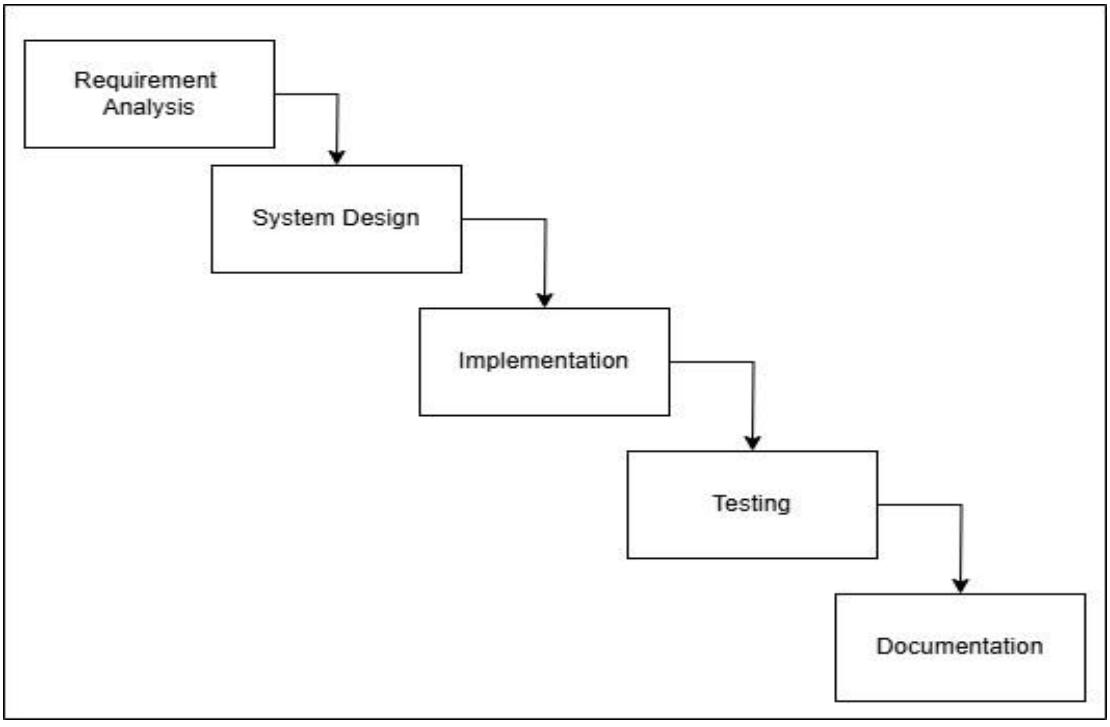

Figure 3. 1 Waterfall model

Figure 3.1 illustrated the Waterfall model used to developed the "**Secure Cloud Storage with Malware Analyzer and AES Encryption**" project, aiming to achieve the

objectives outlined in Chapter 1. This research methodology consists of six major phases, each with its own processes: Requirement Gathering and Analysis, Design, Implementation, Testing and Documentation

Table 3. 1 Waterfall model table

STAGES	TASK	DELIVERABLES
Requirement & Analysis	Preliminary research and literature review on the following topic: 1) Performed initial research and review existing literature on Cloud Computing, Malware Analysis, Data Breaches, Encryption Algorithms. 2) Outlined the background of the study, including the problem statement, objectives, scope, and significance of the research. 3) Compared various mechanisms and techniques, identifying their strengths and weaknesses in relation to the proposed project	1) A comprehensive review of literature on Cloud Computing, Data Breaches, Encryption Algorithms, and Malware 2) Clear definition of the background, problem statement, objectives, scope, and significance of the study 3) A comparative analysis of different mechanisms and techniques, highlighting their strengths and weaknesses in the context of the Encryption Algorithm and Malware analysis.

STAGES	TASK	DELIVERABLES
System Design	Designed the system architecture, including the encryption algorithm, flowchart, use case diagram, entity-relationship diagram, and user interface mock-up.	Detailed designs of the algorithm, flowchart, use case diagram, entity-relationship diagram, and user interface mock-up, providing a blueprint for the implementation phase. 1. Implement the AES encryption and decryption functions. 2. Implement malware analysis API in the project for file analysing 3. Develop the web application using appropriate technologies such as PHP, VirusTotal API, HTML, CSS
Implementation	• Implemented the AES encryption and decryption functions. • Implemented malware analysis API in the project for file analysing • Developed the web application using appropriate technologies	A functional web-based secure cloud storage system with Malware Analyzer and AES encryption to protect data.

STAGES	TASK	DELIVERABLES
Testing	• Performed functional testing to ensure all components work as expected. • Generated test cases to validate the system's functionality, performance, and security.	Test results verifying that the system meets the specified requirements and functions correctly.
Documentation	Gathered and arrange all the project related information.	1) Final report 2) System of a proposed project

3.2.1 Phase 1: Requirement & Analysis

The first phase focused on conducting initial research and reviewing existing literature on key topics, including Cloud Computing, Data Breaches, Encryption Algorithms, and Malware. This foundational step provides the necessary theoretical grounding to inform the project's design and implementation. During this stage, the study's background is outlined, including a problem statement that highlights existing gaps in secure file storage systems, particularly in the context of cloud environments. Objectives are defined to establish what the system aims to achieve, while the scope specifies the project's boundaries. The significance of the research is also emphasized to underline the impact and necessity of the system.

A comprehensive literature review is conducted to analyze various mechanisms and techniques. For example, different encryption algorithms are compared based on factors such as efficiency, security, and performance. Similarly, various implementations of Malware detection and analyzer are examined to determine their applicability to the project. This comparative analysis identifies the strengths and weaknesses of existing methods, ensuring that the proposed system's design integrates proven solutions while addressing identified shortcomings.

3.2.2 Software

Software requirements refer to the specific programs, tools, and platforms necessary for the system to function as intended. This includes operating systems, programming languages, development frameworks, and application software that support the implementation and execution of the system's features. For example, PHP, MySQL, and a web server are needed to develop and deploy the project.

Table 3. 2 Software Requirements

Software Component	Operating System	Processor	Memory	Description
HTML	-	-	-	Hypertext Markup Language is used to design the structure and content of online pages.
CSS	-	-	-	Cascading Style Sheets employed for styling and formatting HTML elements, enhancing the visual presentation of web pages.
JavaScript	-	-	-	Programming language used to create dynamic and interactive effects within web browsers.
Bootstrap	-	-	-	Open-source front-end framework for developing responsive and mobile-first websites using HTML, CSS, and JavaScript.
WSL (Windows Subsystem for Linux)	Windows 10 or later	1.5 GHz dual-core processor or better	2 GB RAM	Provides a Linux environment directly on Windows, enabling the use of Linux tools and utilities.
MySQL	Windows, Linux,	1.6 GHz dual-core	4 GB RAM	Relational database management system (RDBMS) used for storing and

Software Component	Operating System	Processor	Memory	Description
	macOS	processor or equivalent		managing project data.
PHP	Windows, Linux, macOS	2.0 GHz dual-core processor or equivalent	2 GB RAM	PHP serves as the core programming language for the backend, managing file encryption (AES), malware scanning integration (via the VirusTotal API), and server-side logic. It ensures seamless processing of user requests and secure file handling.
VirusTotal` API	-	-	-	An API service used to analyze uploaded files for potential malware. Provides integration for secure file scanning before storing in the cloud.
Google Cloud Platform (GCP)	Windows, Linux, macOS	2.0 GHz dual-core processor or equivalent	2 GB RAM	GCP is cloud-based platform designed to simplify the deployment of web applications, APIs, and static websites. It is particularly popular for hosting applications due to its developer-friendly approach, seamless integration with Git, and support serverless architecture and pay-per-use feature.
Visual Studio Code	Windows, Linux, macOS	1.6 GHz or faster processor	1 GB RAM	Microsoft released a free source-code editor for Windows, Linux, and macOS. Supported features include debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and Git integration.

3.2.3 Hardware

Hardware requirements define the physical components and devices essential for the system's operation. These include servers, storage devices, networking equipment, and any input/output peripherals required for functionality. The hardware must be selected based on the system's performance demands, scalability, and compatibility with the software environment.

Table 3. 3 Hardware Requirements

Hardware Component	Requirement Specifications
Laptop	11th Gen Intel ® Core ™
Chipset	I5-1135g7
Ram	8.0 GB
Operating System	Windows 10

3.2.4 Project Timeline

Table 3. 4 Gantt Chart

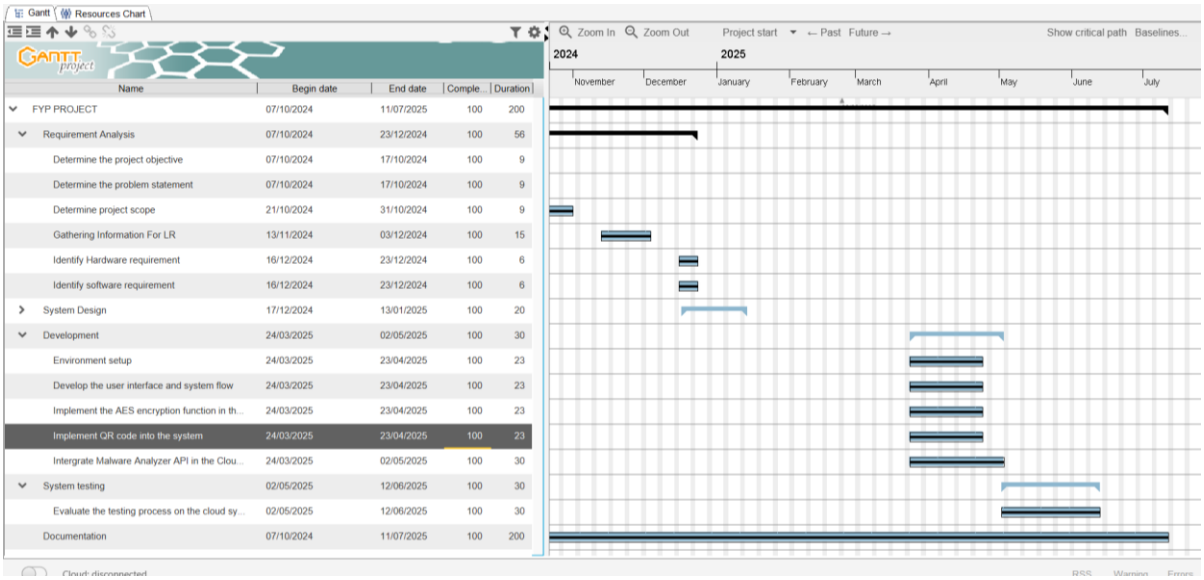

Table 3.4 Provided the project timeline table for the whole timeline development for the completed project.

3.3 Phase 2: System Design

The system design phase of the **"Secure Cloud Storage with Malware Analyzer and AES Encryption"** project focuses on creating a comprehensive architecture that integrates malware analysis and encryption mechanisms for enhanced data security. This phase emphasized the logical and physical components of the system, defined the structure and processes required to deliver a secure and user-friendly cloud storage solution. The design phase is the process to design the system. Within this process, the use case, flow chart, and interface design are created to illustrate how the proposed system work.

3.3.1 Use Case Diagram

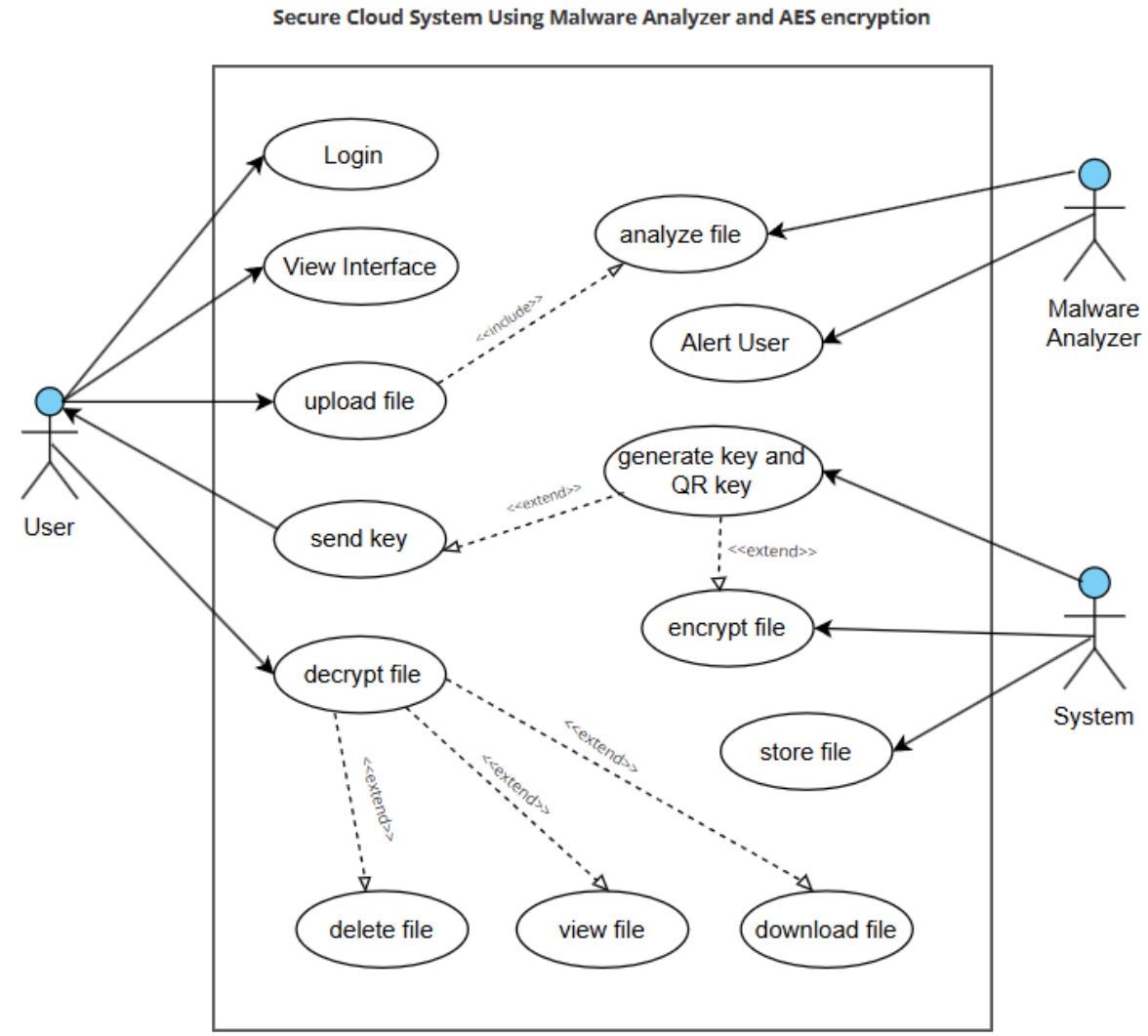

Figure 3. 2 Use case diagram

Figure 3.2 provides illustrations of the interactions between the user, the system, and the malware analyzer. It highlighted key functionalities such as file upload, malware analysis, encryption, secure storage, and file management, ensuring a streamlined and secure process for handling sensitive data in the cloud. The details of proposed use case will be explain at table 3.5 below.

Table 3. 5 Use Case Description Table

Use Case Diagram	Login
Actors	User
Description	The user provides login credentials to gain access to their account in the system. **Basic Path:** 1. User enters valid credentials and is authenticated by the system. **Alternate Path:** Invalid credentials trigger an error and retry prompt.

Use Case Diagram	View Interface
Actors	User
Description	After logging in, the user accesses the main interface, which displays options for upload file and view dashboard **Basic Path:** 1. The interface loads successfully. 2. User choose the action on the interface **Alternate Path:** System fails to load the interface and shows an error message.

Use Case Diagram	Upload File
Actors	User
Description	The user uploads a file to the system, **Basic Path:** 1.The user chooses the file to be upload. 2.The fie is successfully uploaded. **Alternate Path:** Upload fails due to file format issues, size, or network errors.

Use Case Diagram	Analyze File
Actors	Malware Analyzer
Description	Malware analyzer scans the uploaded file for potential malware and alerts the user if threats are detected **Basic Path:** 1. Malware analyzer scans the file 2.Results will be noticed to the user **Alternate Path:** The analyzer encounters errors and notifies the user to retry.

Use Case Diagram	Alert User
Actors	Malware Analyzer, User
Description	The system alerts the user about the malware analysis results, recommending further actions if threats are found. **Basic Path:** 1. The user receives an alert regarding threats. **Alternate Path:** Alerts fail, and results are displayed manually.

Use Case Diagram	Generate Key and QR Key
Actors	System
Description	The system generates a unique AES key to encrypt the uploaded file, and QR from AES Key generated to ease the user to save the key while ensuring its security. **Basic Path:** 1. Key and QR Key is generated successfully. **Alternate Path:** Key generation fails, and the user is prompted to retry.

Use Case Diagram	Encrypt File
Actors	System, User
Description	The uploaded file is encrypted using the generated AES key, ensuring its confidentiality. **Basic Path:** 1. User input the AES key generated 2. File is successfully encrypted. **Alternate Path:** Encryption fails due to resource or compatibility issues, prompting the user to retry.

Use Case Diagram	Store File
Actors	System
Description	The encrypted file is securely stored in the system's cloud storage for future access. **Basic Path:** 1. File is stored securely after encrypted. **Alternate Path:** Storage fails, and the user is notified of insufficient space or system errors.

Use Case Diagram	Send Key
Actors	System
Description	The system securely sends the encryption key to the user or user can save the key during encryption for later use during decryption. **Basic Path:** 1. Key is securely sent or stored by the user **Alternate Path:** Key transfer fails, and the user is notified of the error.

Use Case Diagram	Decrypt File
Actors	User, System
Description	The user requests the decryption of a file, and the system retrieves the key to decrypt it. **Basic Path:** 1. User input the AES key saved from the system 2. File is decrypted successfully. **Alternate Path:** Decryption fails due to mismatched keys or system errors, prompting an error message.

Use Case Diagram	Delete File
Actors	User, System
Description	The user deletes a file from the cloud storage, permanently removing it from the system. **Basic Path:** 1. User decrypt the file using matched AES key 2. User performs action to delete the file 3. The file is deleted successfully. **Alternate Path:** Deletion fails due to system issues, and the user is notified to retry.

Use Case Diagram	View File
Actors	User, System
Description	The user views the contents of a decrypted file through the system interface. **Basic Path:** 1. User decrypt the file using matched AES key 2. User performs action to view the file 3. File content is displayed successfully. **Alternate Path:** File fails to load, and an error message is displayed.

Use Case Diagram	Download File
Actors	User, System
Description	The user downloads a decrypted file from the system to their local device. **Basic Path:** 1. User decrypt the file using matched AES key 2. User performs action to download the file 3. File is downloaded successfully **Alternate Path:** Download fails due to network issues, prompting the user to retry.

Table 3.5 summarized the key use cases for the Secure Cloud System Using Malware Analyzer and AES Encryption, outlined the interactions between the user, system, and malware analyzer. These use cases include logging in, uploading files, analyzing files for malware, alerting the user, encrypting files using AES, securely storing and retrieving files, and managing file access and deletion.

3.3.2 Flowchart

The system flowchart is shown during the design process to illustrate the system's flows and is represented by symbols. It shows the sequence of steps needed to complete the process to secure a file in the **Secure Cloud System using malware analyzer and AES encryption**.

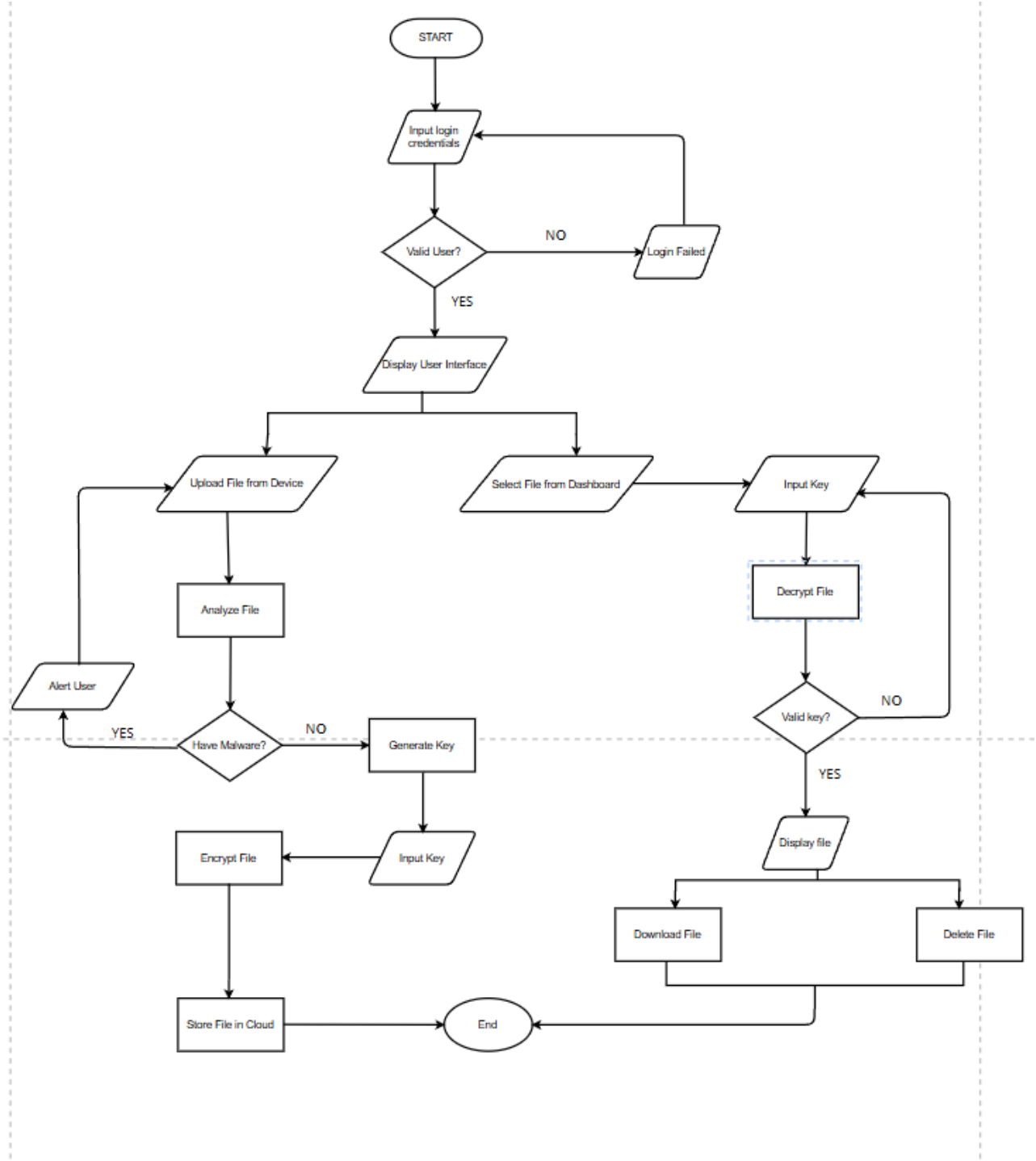

Figure 3. 3 Flowchart of the system

Figure 3.3 provides a detailed process of a **Secure Cloud System Using Malware Analyzer and AES Encryption**, showing the whole flow process of a proposed system. Below is a comprehensive explanation based on the flowchart:

1. Start and Login Process

Input Login Credentials: The process begins with the user entering their credentials (username and password).

Validation: The system checks whether the credentials are valid.

1) If the credentials are invalid, the system displays a "Login Failed" message, and the process terminates.
2) If valid, the system allows the user to proceed by displaying the user interface.

2. User Interface Options

From the user interface, the user has two primary options:

1) Upload File from Device
2) Select File from Dashboard

3. File Upload and Malware Analysis

If the user uploads a file from their device:

1) The system analyzes the file for malware.
2) Decision Point - Malware Detection:
    a) If malware is detected, the system alerts the user, and the process stops for that file.
    b) If no malware is found, the system proceeds to generate the key and QR key for the user for encryption process.

4. File Encryption and Storage

If no malware is detected:

1) The system prompts the user to input a key.
2) The file is encrypted using the provided key.
3) The encrypted file is securely stored in the cloud.

5. File Decryption and Management

If the user selects a file from the dashboard:

1) The system prompts the user to input the decryption key.
2) Decision Point - Key Validation:
    a) If the decryption key is invalid, the decryption fails, and the user is prompted to re-enter the key.
    b) If the key is valid, the system successfully decrypts the file and displays its content.

6. File Actions

After decryption, the user has two options:

1) Download File: The user can save the decrypted file locally.
2) Delete File: The user can delete the file from the cloud storage.

7. End

The process concludes after the file is either downloaded or deleted or the new file stored in cloud storage

This flowchart illustrates a comprehensive, secure workflow for managing files in the cloud. It ensures files are protected with malware analysis before storage and safeguarded using AES encryption. The system incorporates user authentication, malware alerts, encryption, decryption, and file management to provide a robust and secure user experience.

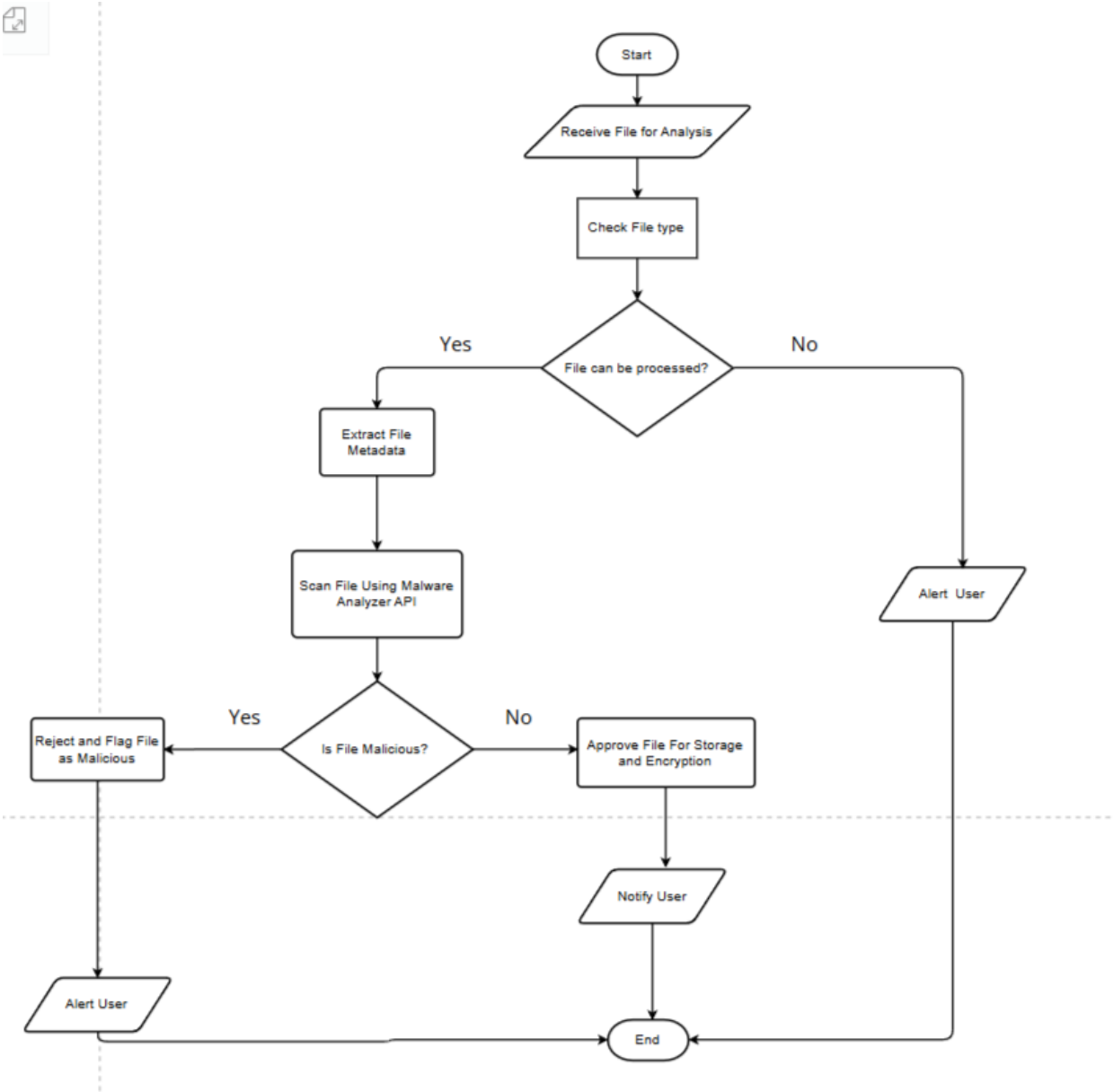

Figure 3. 4.1 Flowchart of the Malware Analyzer API

Figure 3.3.1 provides a detailed process of Malware Analyzer API, showing the whole flow of a file being processed and analyzed before procced to the file storing in the system. Below is a comprehensive explanation based on the flowchart:

1. Start

The process begins when a user uploads a file for analysis.

2. Receive File for Analysis

The system accepts the file and prepares it for further processing.

3. Check File Type

The file type is checked to determine if it is supported by the system.

a) If the file cannot be processed: The system alerts the user and ends the process.

b) If the file is supported: The process moves to the next step.

4. Extract File Metadata

Essential file details such as size, type, and format are extracted to aid in the analysis.

5. Scan File Using Malware Analyzer API

The file is scanned for potential malware threats using integrated malware analyzer API.

6. Is File Malicious?

The results of the scan are analyzed:

a) If the file is malicious: The system rejects the file, flags it as a threat, and alerts the user.

b) If the file is safe: The file is approved for storage and encryption.

7. Approve File for Storage and Encryption

Safe files will be proceed to another process which encryption and storing file in the cloud.

8. Notify User

The user is notified about the status of their file (whether it was safely stored or rejected due to malicious content).

8. End

The process concludes after the necessary actions have been taken

3.3.3 Entity-Relationship Diagram

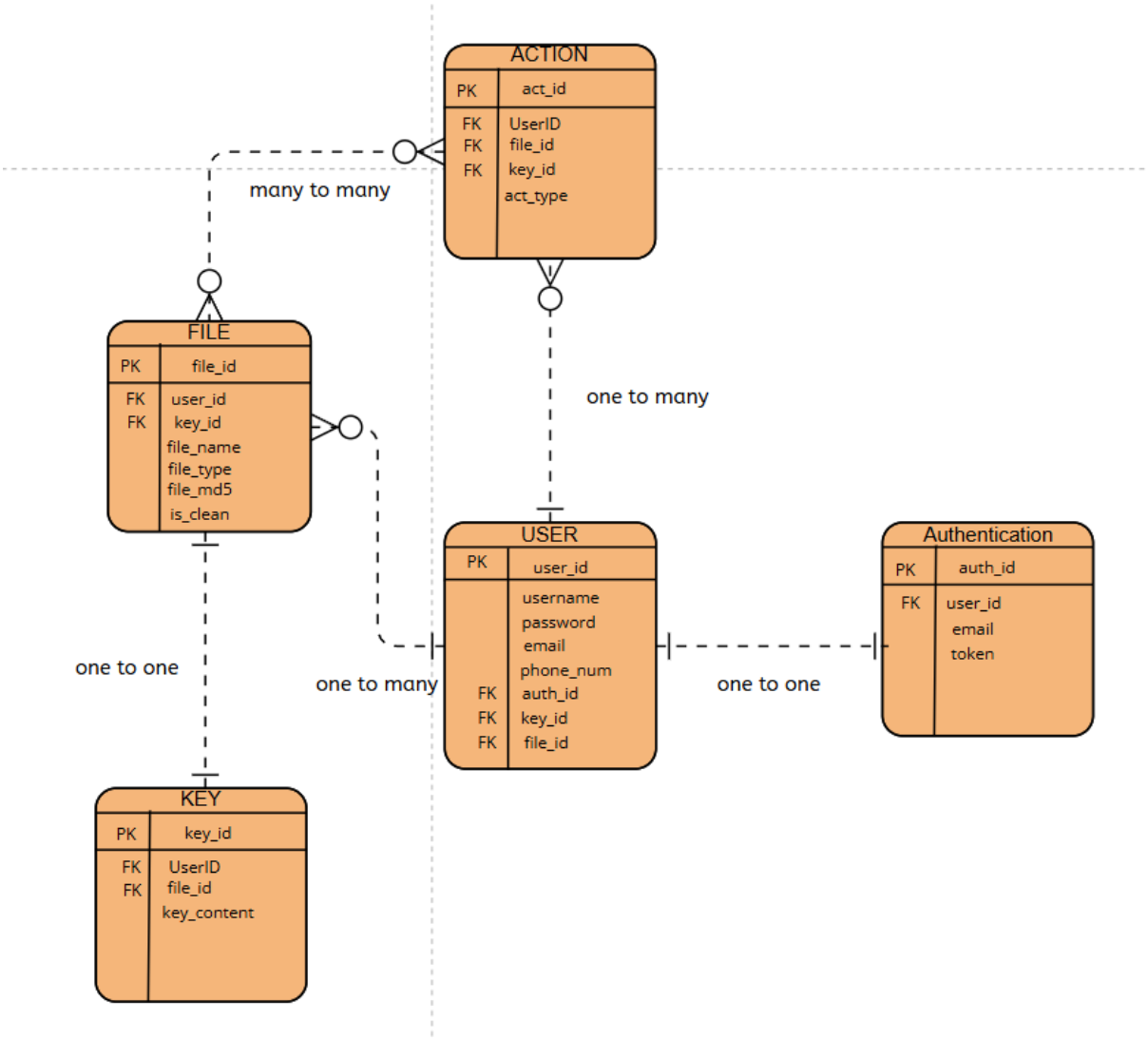

Figure 3. 5 Entity-Relation Diagram

Figure 3.4 shows ERD for the project Secure Cloud System using Malware Analyzer and AES encryption. This diagram provides a clear view for each entity and attributes for the system.

3.3.3.1 USER Entity

Table 3. 6 User Entity

Attribute	Description	Key	Data Type
user_id	Unique identifier for the user	Primary Key	Integer
username	Username of the user		Varchar/String
password	Password of the user		Varchar/String
email	Email address of the user		Varchar/String
phone_num	Phone number of the user		Varchar/String

Table 3.6 shows User Entity to represents the users of the system.

Relationships:

1) One-to-One with AUTHENTICATION (for managing secure user login and tokens).
2) One-to-Many with ACTION (a user can perform multiple actions).
3) One-to-Many with FILE (a user can upload multiple files).

3.3.3.2 AUTHENTICATION Entity

Table 3. 7 Authentication Entity

Attribute	Description	Key	Data Type
auth_id	Unique identifier for authentication	Primary Key	Integer
user_id	References user_id in USER	Foreign Key	Integer
email	Email used for authentication		Varchar
token	Authentication token		String

Table 3.7 shows Authentication Entity to stores user authentication details.

Relationship:

1) One-to-One with USER (linked to user login).

3.3.3.3 FILE Entity

Table 3. 8 File Entity

Attribute	Description	Key	Data Type
file_id	Unique identifier for the file	Primary Key	Integer
user_id	References user_id in USER	Foreign Key	Integer
key_id	References key_id in KEY	Foreign Key	Integer
file_name	Name of the file		Varchar/String
file_type	Type of the file (e.g., .txt)		Varchar/String
file_md5	MD5 hash for file integrity		Varchar/String
is_clean	Indicates malware status		Boolean

Table 3.8 shows File Entity to represents files stored in the cloud

Relationships:

1) One-to-Many with USER (multiple files can belong to one user).
2) One-to-One with KEY (each file has a unique encryption key).
3) Many-to-Many with ACTION (files are associated with various actions).

3.3.3.4 KEY Entity

Table 3. 9 Key Entity

Attribute	Description	Key	Data Type
key_id	Unique identifier for the key	Primary Key	Integer
user_id	References user_id in USER	Foreign Key	Integer
file_id	References file_id in FILE	Foreign Key	Integer
key_content	Content of the encryption key		Varchar/String

Table 3.9 represents Key Entity to manages encryption keys for secure file storage.

Relationship:

1) One-to-One with FILE (each file is assigned one unique key).

3.3.3.5 ACTION Entity

Table 3. 10 Action Entity

Attribute	Description	Key	Data Type
act_id	Unique identifier for the action	Primary Key	Integer
user_id	References user_id in USER	Foreign Key	Integer
file_id	References file_id in FILE	Foreign Key	Integer
key_id	References key_id in KEY	Foreign Key	Integer
act_type	Type of action (e.g., upload)		Varchar/String

Table 3,10 represents Action Entity to tracks actions performed in the system, such as uploading, downloading, or scanning files.

Relationships:

1) Many-to-Many with FILE.
2) One-to-Many with USER.

This ERD ensures secure cloud storage by integrating malware analysis and AES encryption, with clear relationships between users, files, actions, keys, and authentication details

3.3.4 User Interface Mock-up

The user interface mock-up is presented during the design process to make a design prototype before proceeding to develop the front end.

3.3.4.1 Log in Interface

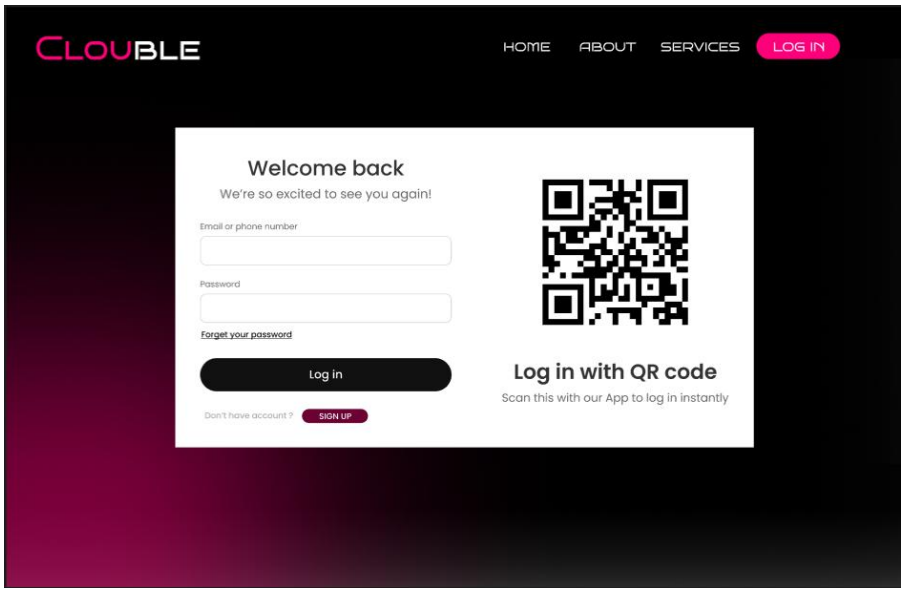

Figure 3. 6 Log in Interface

Figure 3.5 shows the login interface in the cloud system for the user

3.3.4.2 Cloud Interface

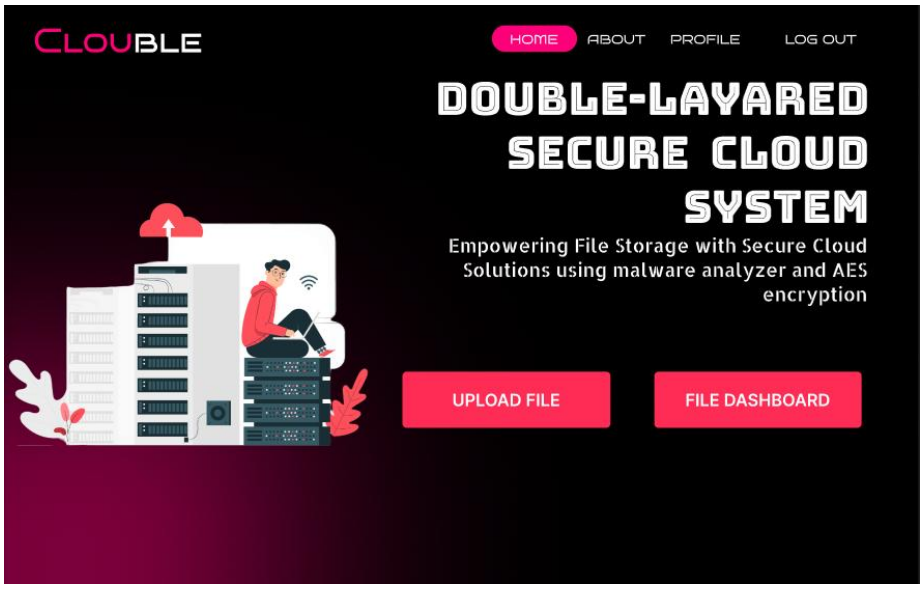

Figure 3. 7 Cloud Homepage

Figure 3.5 shows the homepage in the cloud system for the user after successful login.

3.3.4.3 Upload File Interface

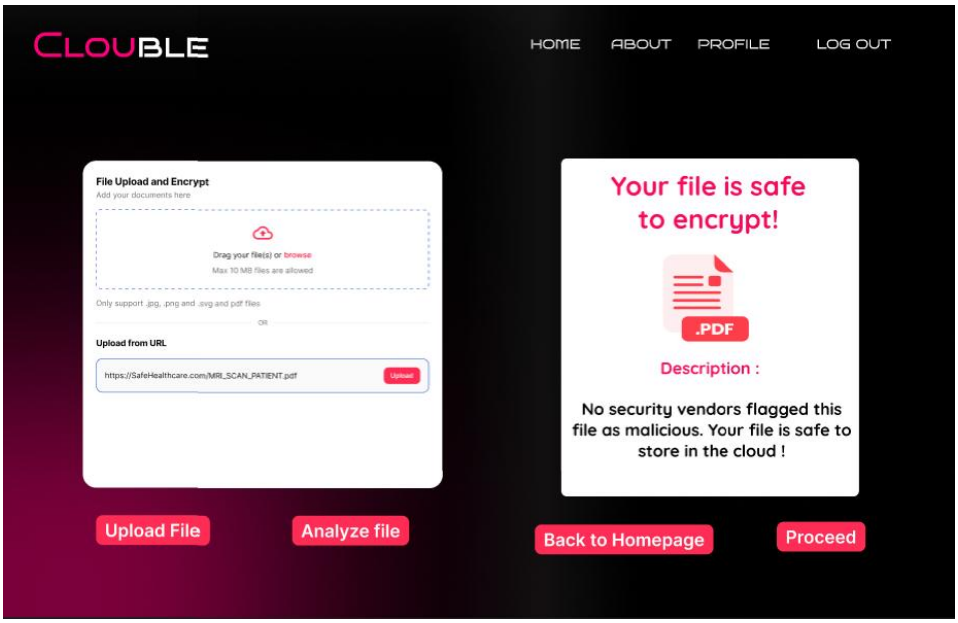

Figure 3. 8 Upload File Interface

Figure 3.5 shows the upload interface in the cloud system for the user when user clicks the upload button. This page also contains malware analyzing before the user allowed to encrypt the file.

3.3.4.4 Encryption Interface

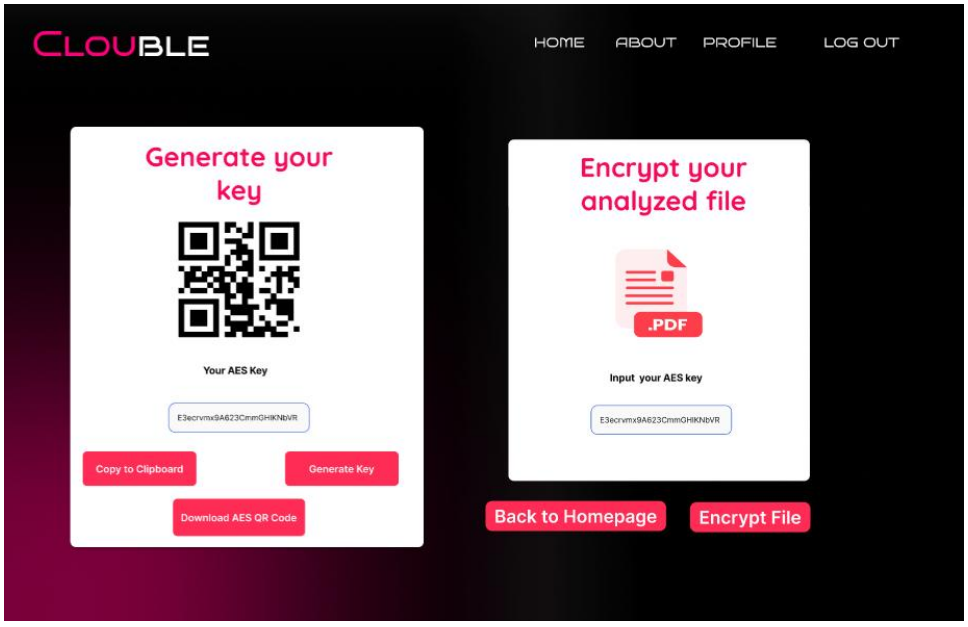

Figure 3. 9 Encryption Interface

Figure 3.8 shows the encryption interface in the cloud system for the user after user has clean results for malware analyzing. Encryption key and QR key will be generated before user can input key and encrypt file.

3.3.4.5 Successful Encrypt Interface

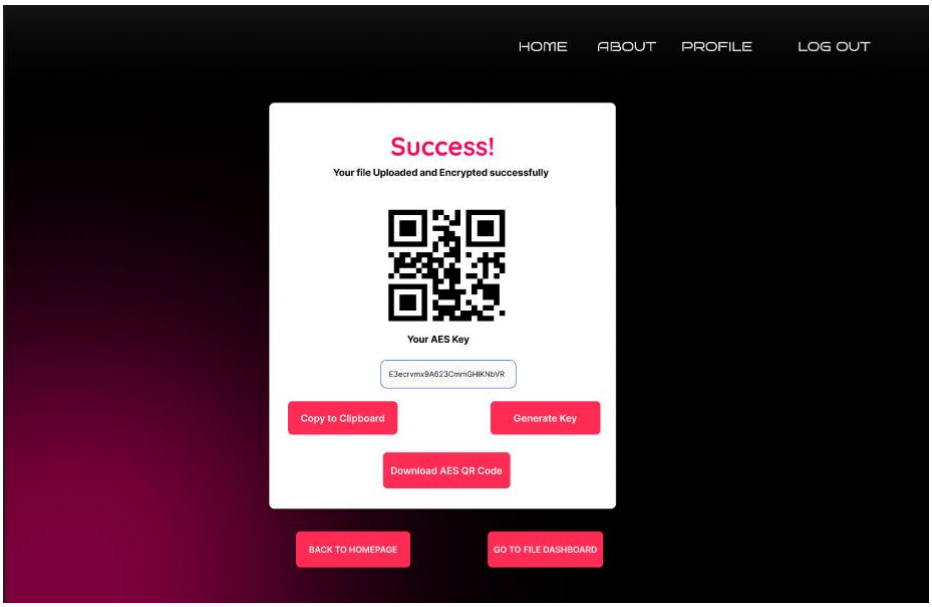

Figure 3. 10 Successful Encrypt Interface

Figure 3.8 Shows the results for the user when the file is successfully encrypted. Key input and QR key also will be shown for the user to save the key.

3.3.4.6 File Dashboard Interface

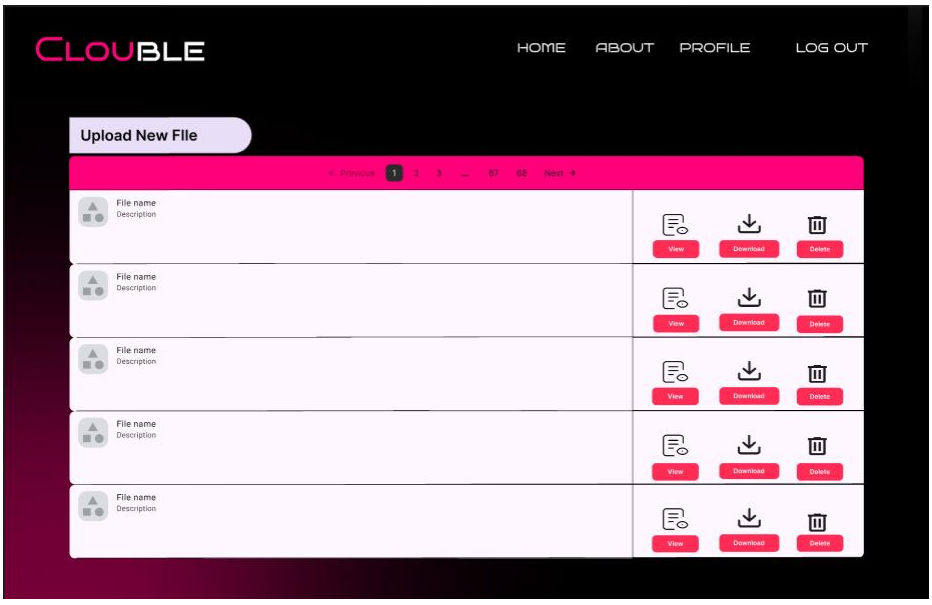

Figure 3. 11 File Dashboard Interface

Figure 3.10 shows the upload interface in the cloud system for the user when user clicks the upload button. This page shows the user the lists of encrypted file and action whether the user wants to view, download or delete the encrypted file.

3.3.4.7 File Decrypt Interface

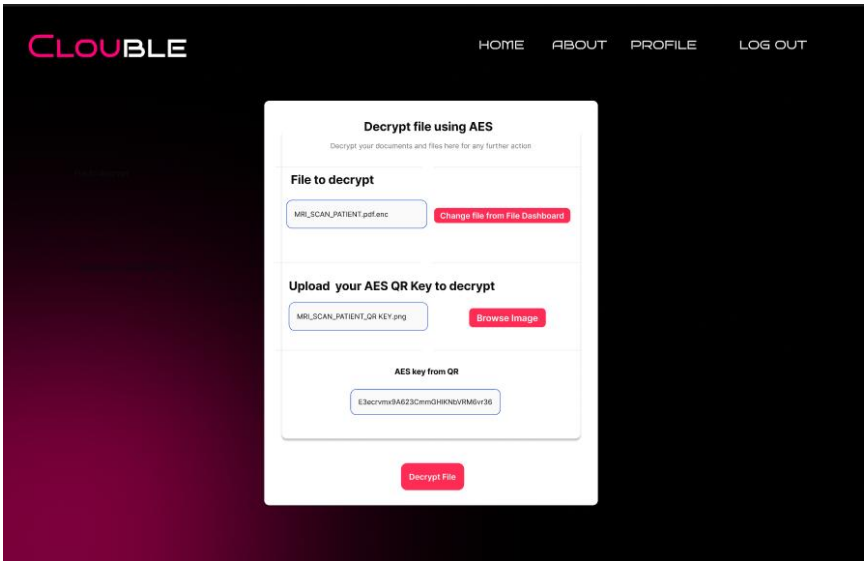

Figure 3. 12 File Decrypt Interface

Figure 3.11 shows the file decryption interface for the user after user done choosing their encrypted files and action. This page requires the user to upload the AES key or QR AES key before user can proceed to do actions on file.

3.3.4.7 Successful Decrypt Interface

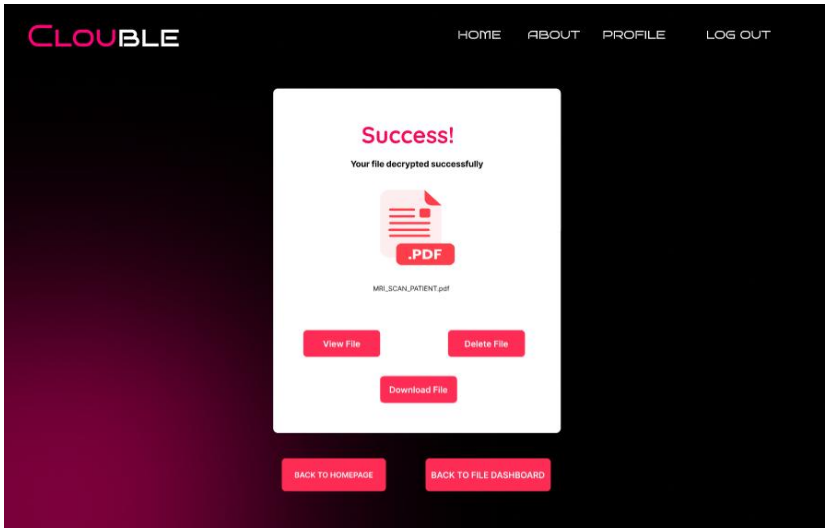

Figure 3. 13 Successful Decrypt Interface

Figure 3.12 Shows successful decrypt interface after user enter the right key to decrypt the file. Only then can the user proceed with the action.

3.4 Phase 3: Implementation

The deployment of such secure cloud storage system exercised an orderly system of integrating malware scanning and application of AES encryption to develop highly secure platform. The initial move were to come up with a cloud-based infrastructure which would enable direct file uploading by the users. In this framework, there were also a malware detection module which scanned the files to check if they were a potential threat before they are uploaded. This module were using the latest malware signature databases and heuristic methods so that threat can be observed. Files then underwent malware check and those that passed were then encrypted with the Advanced Encryption Standard (AES) and then stored. AES also made sure that the information was confidential and it could only be accessed by using the right decryption key. Moreover, the access to the cloud storage were checked and recorded to guarantee the adherence to the standards of data integrity. Testing stages confirm the functionality of the system, among them the correctness of malware identification, encryption stability and the overall performance at different levels of loading. Lastly, the platform had been deployed with scalability where both an individual user and an organization can enjoy the high degree of protection capability the platform offers. The design of the system typified efficiency and none of these protective measures compromised the usability and performance of the system.

3.5 Phase 4: System Testing

The waterfall technique then moves on to system testing. The major goal of system testing was to guarantee that the developed website fits all requirements and functionality. Functional testing are used as a kind of system testing in this project.

3.5.1 Functional Testing

Functionality testing is a type of testing that ensures that the system performs as expected and meets its criteria. It is frequently used to test the system's general functionality, including registration, login, and encryption procedures.

During functional testing, test cases are created to address all the system's functional needs, such as user registration, login, displaying encrypted files, and the server-side client. Testers then run the test cases and compare the actual results to the predicted outcomes. Any differences between the actual and expected results are then identified as flaws or defects.

3.5.2 Test Case Generation

The process of developing a collection of test cases that were used to confirm that a software system satisfies its functional requirements and performs as planned is known as test case generation. Before the system is made available to end users, test case generation seeks to find and capture as many potential flaws or problems as possible. The table for creating test cases for this project is shown below.

Table 3.11 Test Case Generation

Test ID	Test Case	Test Description	Preconditions	Test Input	Expected Output
1.	Login	Verify that a registered user can login	User has registered and confirmed email address	Valid email and password	User is granted access to the main page
2.	Login	Verify that an unregistered user or invalid login information is rejected	None	Invalid email or password	Error message displayed

Test ID	Test Case	Test Description	Preconditions	Test Input	Expected Output
3,	Login	Verify that a registered user receives an OTP and can enter it to login	User has registered and been authorized by admin	Valid email, password, and OTP	User is granted access to the main page
4.	Upload File	Verify that a logged-in user can upload a file	User is logged in	Valid file upload	File is encrypted and uploaded successfully; success message displayed
5.	Download File	Verify that a logged-in user can download an encrypted file and decrypt it	User is logged in and file is uploaded	Request to download a specific file	File is downloaded and decrypted correctly
6.	Delete File	Verify that a logged-in user can delete a file	User is logged in and file is uploaded	Request to delete a specific file	File is deleted successfully, success message displayed
7.	Verify MD5 Hash	Verify that the MD5 hash before encryption and after decryption matches	User is logged in and file is uploaded	Upload and then download a file	MD5 hash of the original file and decrypted file matches
8.	Generate Key	Verify that a user can generate an AES key and download the QR code	User is logged in	Click “Generate Key” button	AES key is generated, QR code is displayed and downloadable

Test ID	Test Case	Test Description	Preconditions	Test Input	Expected Output
9.	Analyze file	Evaluate whether the file processed using malware are accurate and precise before encryption	File must be uploaded before analyzing	Upload file and analyze	Alert user when there are malware in the file
11.	File Encryption	Verify whether file can be encrypted successfully before store to the cloud	User uploaded files and analysed	Encrypt uploaded file using key given	System handles encryption efficiently, performance metrics recorded
12.	File Decryption	Verify whether file can be decrypted successfully before any further action on files	User must have the matched key to decrypt	Decrypt using matched AES key	System handles encryption efficiently, performance metrics recorded

Table 3.11 represents test case, including preconditions, test input and expected output for every test case in the test case generation table.

3.6 Phase 5: Documentation

In the Waterfall model, the final stage of software development is the Documentation phase. During this stage, all relevant materials, including technical guides, system specifications, and development procedures, were prepared to ensure the system's longevity and ease of use. Comprehensive documentation facilitates understanding for both end-users and technical teams, ensuring that the system can be effectively operated, maintained, and updated in the future. This phase also involved compiling records of the development process, including testing results and design decisions, to serve as a reference for future enhancements or troubleshooting.

3.7 Summary

This chapter outlined the research methodology utilized for the project. The research framework is based on the Waterfall model, which comprises five key stages. The first stage involves gathering and analyzing requirements. The second stage, design, focuses on developing the algorithm, flowcharts, and use case diagrams for the web-based system. The third stage, development, specifies the required software and hardware components. The fourth stage, system testing, includes functional testing to ensure system reliability. Finally, the documentation phase marked the conclusion of the process