**Chapter 3**

**RESEARCH METHODOLOGY**

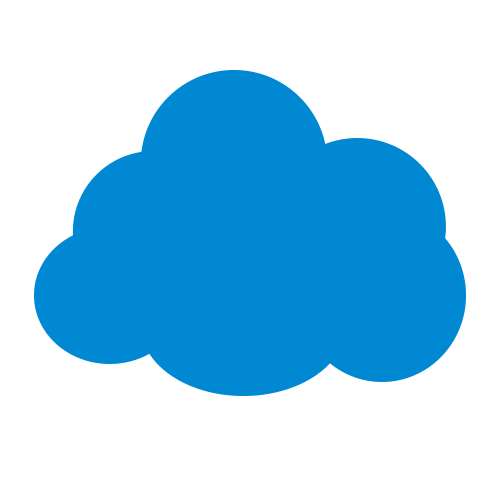
This chapter presents the project design and project development to provide blueprint of the entire system architecture. Operational procedure and method of testing and evaluation are explained to measure its effectiveness based on predefined parameters.

**Project Design**

The Cryptographic Instant Messaging (CIM) system is Instant Messaging (IM) application software that provides encrypted chat communication and file transfer for secure and reliable exchange of information.



**IM Relay Server**



**INTERNET**

**Encrypt**

**Decrypt**





**Receive**

**Send**

**Client X**

**Client Y**

*Figure 2.*Interface Diagram of CIM

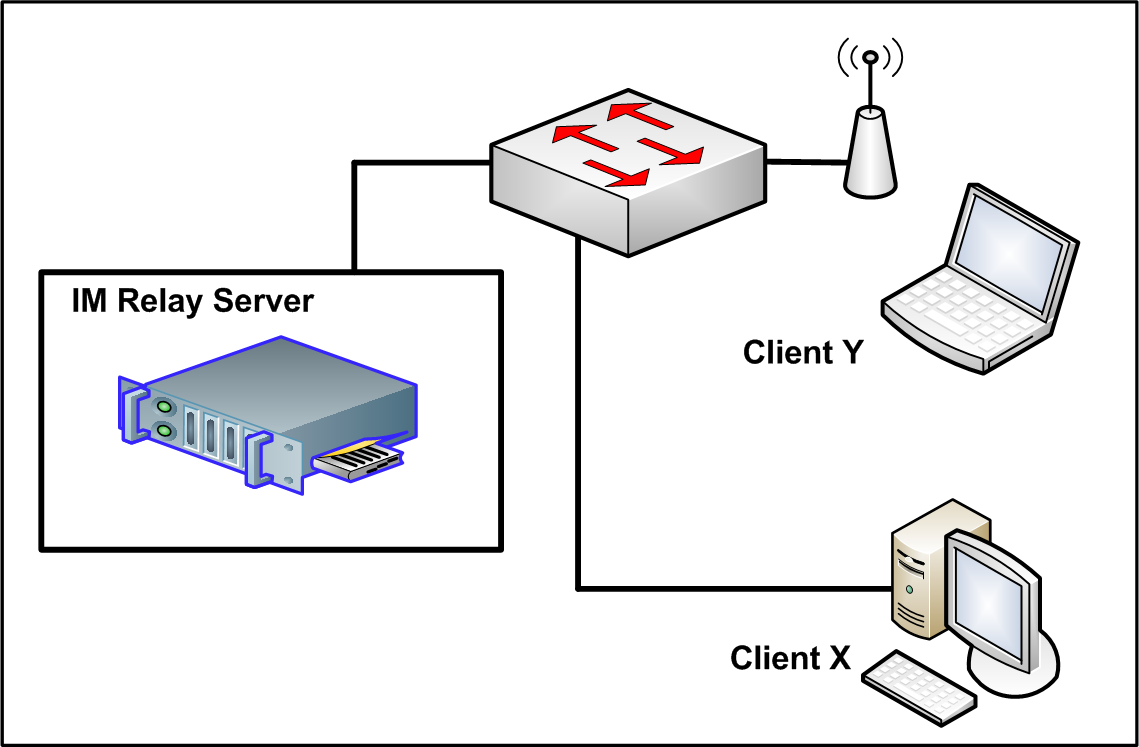
**Operational Concept and Dataflow**

Figure 2 from the previous page illustrates the fundamental structure of the system and flow of data from an end-user’s perspective. Client X initiates a chat conversation with Client Y using the CIM client software. During the chat conversation, the system automatically encrypts the text message prior sending it to the IM Relay Server. The server now searches for the intended recipient from its list of client sockets. When the recipient’s socket is available, the server transmits the encrypted text message to Client Y (recipient) through its socket. The server manages all clients’ sockets. It handles authentication of all incoming clients prior acceptance of the socket. IM Relay Server maintains a minimal database using MS SQL Server, SQL Express, or MS Access for accounts and activity logs. As soon as the text message arrived at Client Y (recipient), the software then decrypts the text using the system-generated key as the default. Unreadable message or randomized characters indicates unmatched passkey. Both Client X (sender) and Client Y (recipient) must encode manually the correct or agreed passkey to decode their conversation. The manual encoding of a passkey is done to deny the system administrator or any authority from decoding the message during the conduct of any special investigation. System generated keys are stored in the server specifically for this purpose. It is strongly suggested that this manual key should be transmitted through another mode of transmission like SMS, Email, or voice call.

A similar procedure can be done during sending and receiving files. Client X starts up the File Transfer window and browse for the file. By default, CIM uses Advanced Encryption Standard (AES) 256-bit to encrypt and decrypt text messages and files. The sender can choose to double encrypt the file by hiding it in an image file using steganography or Hidden in Plain Sight (HIPS) algorithm. A checkbox or option button is provided for this feature. By checking both AES and HIPS, the system first encrypts the file using AES and hides the resulting file into an image. The CIM then transmits the image file to the IM relay server. The server searches for the intended recipient from its list of client sockets. When the recipient’s socket is available, the server relays the file or message to Client Y (recipient). As soon as the encrypted file or image file arrived, the user can now browse for the file and choose to open it with the same application. The encrypted file is extracted from the image file using the HIPS function for file extraction. After the successful extraction, the CIM now tries to decrypt the extracted file using the system-generated key. If unsuccessful, it asks for the passkey which must be encoded by the receiving user to convert the encrypted file back to its original form. Manual encoding of passkey happens when the sender chooses to encode a different key other than the default system-generated key.

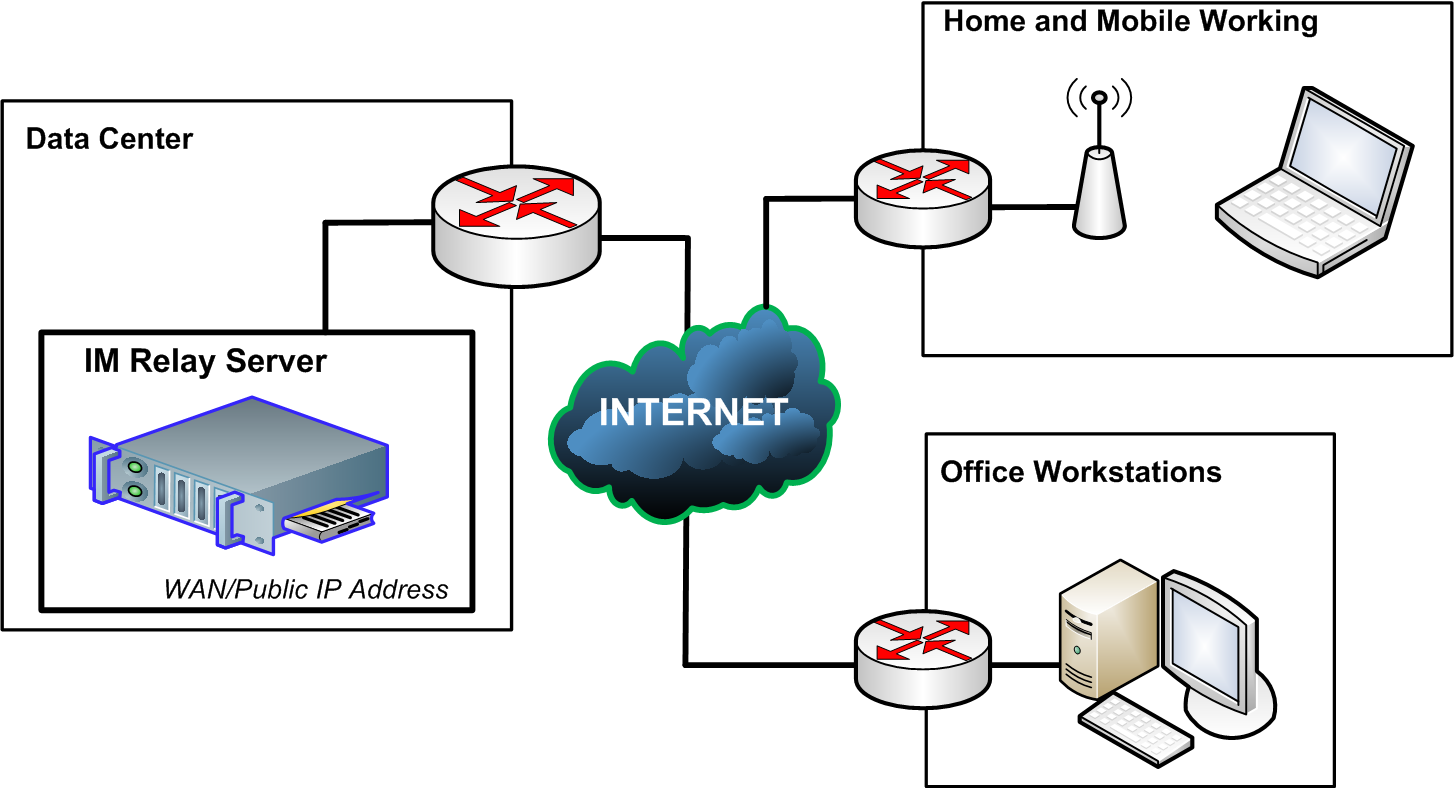
**Network Architecture**

To visualize its deployment from a technical perspective, the network diagram is drawn for this purpose. In figure 3 of the next page, the CIM primarily utilizes the Local Area Network (LAN) for inter-office instant messaging and collaboration. Client X (desktop computer) transmits messages to Client Y (laptop computer) through the adjacent relay server within the same LAN.



*Figure 3.*Local Area Network (LAN) Setup of CIM

The CIM is also applicable to clients outside the office and anywhere in the world. For a Wide Area Network (WAN) setup, the Internet is utilized. The IM Relay Server is assigned with a global or public IP address to allow connection from clients outside of its network. This allows interconnectivity among regional offices, company branches, or widely dispersed units through this system.



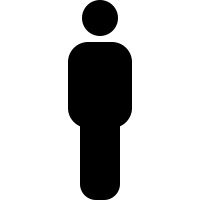
*Figure 4.* Wide Area Network (WAN) Setup of CIM

Users who bring their work at home can now securely communicate and send files to the main office through the CIM. Firewalls and Intrusion Prevention System (IPS) provide optional security reinforcement at the network layer between the CIM relay server and CIM client. Servers are ideally housed in a Data Center for a large-scale deployment.

**System Analysis and Design**

The interaction between the system and users is explained below through the Use Case Diagram where the administrator and the end users are the actors.

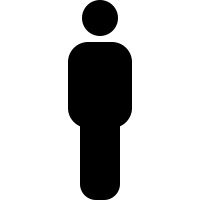
CIM Server Software

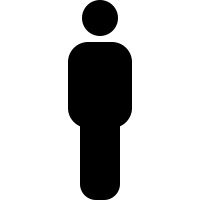


CIM Client Software

**Admin**

CIM Client Software





Send & Receive Chat Message

Send & Receive Chat Message

**User X**

**User Y**

Send & Receive File

Send & Receive File

*Figure 5.*Use Case Diagram of the CIM

Above use case diagram depicts the procedure in utilizing the CIM. The system administrator creates the accounts needed for end users to log in or authenticate. After a successful login, users can now send and receive chat messages as well as transmit files with each other via the relay server.

Encryption and decryption of chat messages and files are done automatically by the system. The CIM adapts AES 256-bit by default as its first layer encryption. Users may choose not to use the system-generated key and encode a different manual key known to both the sender and recipient. This is done to deny the system administrator or any authority from decoding the files during any special investigation.

Start

**FILE DECRYPTION**

**FILE ENCRYPTION**

Start

Encrypted File

Send Encrypted File

Plain File

Encrypt with AES

N

Decrypt the File?

Extract embedded file

Y

Apply HIPS?

Y

Hide Plain File into an image

Decrypt with AES

N

Error?

Enter Key

Y

N

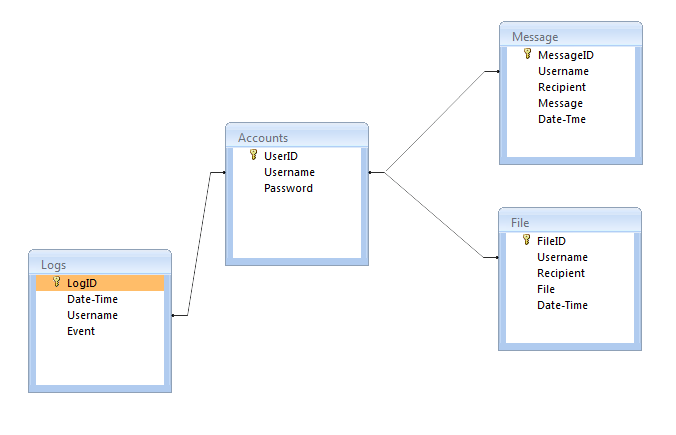
Save Plain File to directory

END

END

*Figure 6.*Flowchart for FileEncryption and Decryption

Figure 6 of the previous page explains the process of encrypting and decrypting files. Users may choose to double encrypt the file using steganography hiding technique. To do this, end users clicks the HIPS radio button to hide the data file in an image prior to the transmission. When the recipient received the encrypted file, the user can now browse for the file and open it with same CIM application. The system extracts the embedded file using HIPS algorithm and then decode it using the AES decrypt function.



*Figure 7.* Table Relationship Diagram of the CIM Database

Above figure shows the different tables to be used for handling client authentication, chat, and file transmission. Tables are connected or joined by Username so that a query and updating can be made through this field. Account table is used for authentication, Message table is for storing chat conversation, and File table is used to record files being transmitted. Apart from these primary tables is the Logs table which is used to monitor and review all activities by the client IM. This is also important for the security of the CIM server and its connecting clients.

**Project Development**

The development of CIM adopted agile methodology so that various modules or components including other third-party modules (e.g. DLL, API, and SDK) can be integrated into the system; vis-a-vis one component can be released or launched even without the other unfinished components making it ideal for evolutionary or continuously improved software.



Base

Encryption

Data Transfer

*Figure 8.*Software Development Life Cycle using Agile Methodology

https://medium.com/

Above figure elucidates the process of SDLC as earlier presented in the conceptual model of this study. The CIM was divided into three (3) components - the base, data or file transfer, and encryption component. These components served as sprints or iterative work activities needed to complete the entire system.

*Figure 9.*Phases of Software Development

**Planning Phase**

Planning involved identifying requirements, determining system specifications, and establishing work breakdown schedules. Every component of the system was born out from the requirements and specifications provided by the stakeholders or end users. Hardware, software, and skill requirements were identified to ensure completion of the project. The base component of the CIM adopted open architecture technique to allow various components to be integrated.

**Designing Phase**

CIM system architecture was mapped out to identify technical specifications of every component. These specifications include forms, screens, reports, databases, modules, and routines. Use case diagram, network diagram, and other related figures previously drawn in this chapter were examined to serve as blueprints for the coding of the system.

**Building Phase**

This is the coding or development phase of the project. The software was written using Visual C# computer programming language. The practice of secure coding was also observed during this phase in order to address any application-specific vulnerability thereby reducing the possibility of getting exploited by a determined hacker.

**Testing Phase**

Each component underwent functional and portability testing prior to its release. All forms, functions, classes, and procedures, were tested to ensure an error-free application. It uses standard testing methodology, test case forms, and test logs. Results of these test cases were documented and formed part of this research.

**Reviewing Phase**

The system or components of this system was reviewed to ensure conformity with the project specification. In this phase, the CIM was demonstrated to fifteen (15) respondents for evaluation. The respondents were composed of five (5) IT Experts, five (5) cybersecurity experts, and five (5) end users. They assessed the CIM based on ISO 25010 criteria for software quality and security.

**Launching Phase**

In this phase, the component is now ready for deployment. The completion of the base component marked the start of the next development cycle for the other components. The CIM, with its base component, was a working IM and can already be deployed in a production environment even without the other components. The completed data transfer and encryption components were then integrated into the base component for a complete CIM system as required in this research.

**Operation and Testing Procedure**

**Operation Procedure**

For CIM System Administrator, the following procedure was conducted:

1. Configured the IP Address of the server.
2. Started the CIM Server software.
3. Added users through the “Add User” button.
4. Informed the end user for their username and password as well as the IP Address of the server they will use to connect.

For the CIM user X (sender), the following procedure was conducted:

1. Checked for internet or network connectivity.
2. Opened CIM Client software.
3. Entered the username and password provided by the administrator as well as the IP address of the server.
4. After a successful login, the main window was displayed showing the hierarchical list of users.
5. Started conversation by selecting the user and clicked the “Chat” button.
6. Sent files by browsing and selecting the desired encryption algorithm or a combination of both the AES 256-bit and HIPS hiding technique.

For CIM user Y (recipient), the following procedure was conducted:

1. Checked for internet or network connectivity.
2. Opened CIM Client software.
3. Entered the username and password provided by the administrator as well as the IP address of the server.
4. After a successful login, the main window was displayed showing the hierarchical list of users; offline messages were also displayed.
5. A file was received; the user browsed for the file and extracted the embedded file from the image.
6. The system decrypted the file using the system-generated key.
7. Started conversation by selecting the user and clicked the “Chat” button.

**Testing Procedure:**

The CIM was subjected to functional and portability testing to ensure an error-free application across all versions of MS Windows operating system. The testing was conducted among computer programmers and system administrators to immediately resolve or debug any fault of the system. Test cases were documented and form part as supplementary material or evidence of this research. The table on the next page enumerates the procedure or steps conducted during the testing of CIM through various scenarios.

Table 1

*Testing Procedure of the CIM*

|  |  |
| --- | --- |
| **Scenario** | **Steps to be undertaken** |
| Client Login | 1. Attempted to login with erroneous entries 2. Login with correct entries 3. Verified the password was encrypted using the Wireshark tool of KALI Linux |
|  |  |
| Send and Receive Chat Message | 1. Sent a long and random chat message to determine a possibility of a crash. 2. Sent readable chat message and verified the accuracy of the message as it reached the recipient. 3. Measured the time it took for the message to reached the recipient 4. Verified the text was encrypted during transmission using Wireshark tool of KALI Linux |
|  |  |
| Send and Receive File | 1. Sent a file and measured the time it took for the file to reach the recipient. 2. Sent large files and checked the possibility of crashes. 3. Verified HIPS hiding technique by locating the image file containing the embedded data file. 4. Verified AES encryption is applied by opening the encrypted file if it is unreadable. |
|  |  |

Test case forms and test incident logs were accomplished or filled up to document the alpha testing of the CIM. These forms included description which explains the purpose of the test scenarios; while the pre-requisite establishes the required state of the system prior to the execution. The previously enumerated steps were then reflected in the test execution steps of this form. The table on the next page is the test case form that was used during the conduct of alpha testing.

Table 2

*Test Case Form*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | |  | | | | **Test Case ID** | | |  | |
| **Test Case Description** | |  | | | | **Test Priority** | | |  | |
| **Pre-Requisite** | |  | | | | **Issue Severity** | | |  | |
| Test Execution Steps: | | | | | | | | | | |
| **Step Nr** | **Action** | | **Inputs** | **Expected Output** | **Actual Output** | | **Test OS** | **Test Result** | | **Test Comments** |
| 1 |  | |  |  |  | |  |  | |  |
| 2 |  | |  |  |  | |  |  | |  |

Ten (10) test cases or scenarios were executed for this research. Results of the test cases were recorded, including whether the condition was “pass” or “fail”. A pass condition means that the actual results meet the expected outcome. On the other hand, a fail means that the system is executed with errors or the actual results do not meet the expected outcome. Test incident log was provided to document un-forecasted errors that occurred during the exploration of the system including the previously documented test scenarios. The table below is the test incident log format that was used during the alpha testing.

Table 3

*Test Incident Log*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Test Scenarios** | **Error Description** | **Case Ref** | **Severity** | **Priority** | **Screenshot** |
| 1 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |

The classification table below was used to determine the severity of error resulted during the test execution. Its corresponding description served as bases of classifying such severity. The severity and priority level was then reflected into the test case form.

Table 4

*Severity Classification and Priority Level of Errors*

|  |  |  |
| --- | --- | --- |
| **Severity** | **Description** | **Priority** |
| Critical  (Severity 1) | These issues are showstoppers and productivity is severely hindered. No workarounds exist and require immediate resolution. | High |
| Major  (Severity 2) | These are issues that have an impact on productivity but can have a workaround. | Medium |
| Minor  (Severity 3) | These are issues that do not prevent productivity to be carried over. All issues have a workaround. | Low |

Test results were summarized to determine if the system meets the required percentage score for a system worthy to be deployed. The following criteria were used to determine if the system or components of the system can be released:

1. All of the test cases are executed.
2. “Pass” result should be no less than eighty percent (80%) of the executed test cases.
3. There must be no critical or Severity 1 issue unresolved.

The table on the next page was used to document the summary of all test cases.

Table 5

*Test Execution Summary*

|  |  |
| --- | --- |
| **Test Execution** | **Expected Results** |
| Total no. of test cases |  |
| No. of test cases executed |  |
| % executed |  |
| No. of test cases passed |  |
| % passed |  |
| No. of test cases failed |  |
| % failed |  |
| No. of test cases not executed |  |
| % not executed |  |

**Portability Testing**

The system ideally runs on any Microsoft Windows platform except for the mobile version of Windows OS since this is a desktop application. Portability testing was conducted to check the CIM for varying errors across all versions of Windows operating systems. This test was also necessary to determine other required libraries or software in order to successfully execute the CIM, The system was developed using .NET framework in which previous versions of Windows OS are not installed by default. The table on the next page was used to document the conduct of portability testing of CIM. Errors that occurred during this testing were also recorded in the test incident log.

Table 6

*Portability Testing Summary*

|  |  |  |
| --- | --- | --- |
| Operating System (Version) | Prerequisites | Expected Results |
|  |  |  |
| MS Windows OS |  |  |
|  |  |  |
|  |  |  |

Vital to the conduct of alpha testing is the security test. It determines the validity of encryption implemented. The table below was used to test the encryption of the CIM against the three (3) commonly used Sniffing or Penetration Testing software – Wireshark, Cain&Abel, and Ettercap as part of Kali Linux.

Table 7

*Security Testing Summary*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Instances | Original Message (Source) | Result Message (Destination) | Penetration Testing Software | | |
| Wireshark | Cain&Abel | Ettercap (Kali) |
| Send Chat Message |  |  |  |  |  |

A parallel test was also conducted to determine the time it takes for the data to arrive at the destination when applied with dual-layer encryption. The Latency Test measures the timeframe of data transmission. The table below was used for the Latency Test of the CIM.

Table 8

*Latency Testing Summary*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Delay  (Time it takes for the data to arrive from sender to recipient) | | | | | |
| Instances | Plain  (No Encryption) | Encrypted with  AES | Encrypted with  HIPS | | | |
| Send Chat Message |  |  |  |  |  |

**Evaluation Procedure**

Software evaluation was conducted to ensure conformity to the required specification and that the system exhibits characteristics that meet the criteria of a quality and secure application software. This study adopted the ISO 25010 standard for evaluating the software quality and validates the security of CIM. The following are the essential criteria of this standard which were used as the evaluation metrics for this study: Functional Suitability, Performance Efficiency, Usability, Reliability, Security, Maintainability, Portability, and Compatibility.

The following activities were conducted to deduce an accurate and justified evaluation of CIM.

1. Five (5) IT experts, Five (5) cybersecurity professionals, and Five (5) end users were invited as respondents/evaluators for this research.
2. The evaluation metrics and the Likert Scale as shown in the table below were presented and discussed to the evaluators; the Likert scale was necessary to measure the respondent’s opinion for the system during the conduct of evaluation.

Table 9

*Likert Scale*

|  |  |
| --- | --- |
| **Numerical Rating** | **Descriptive Rating** |
| 5 | Excellent |
| 4 | Very Good |
| 3 | Good |
| 2 | Fair |
| 1 | Poor |
|  |  |

1. Walkthrough of the system were conducted and evaluators were encouraged to explore the CIM through hands-on experience.
2. The interception of CIM communication between the client and the server were demonstrated through the use of KALI Linux in order to validate the security of the system.
3. The evaluators were given sufficient time to rate and evaluate the system.
4. The evaluation instrument accomplished were collected and tabulated.
5. The score mean was computed to deduce a numerical result of the evaluation.
6. The result was then interpreted using the equivalent descriptive rating as shown in the table below.

Table 10

*Rating Scale for Interpreting the Evaluation Result*

|  |  |
| --- | --- |
| **Mean Rating Scale** | **Descriptive Evaluation** |
| 4.51- 5.00 | Excellent |
| 3.51-4.50 | Very Good |
| 2.51-3.50 | Good |
| 1.51-2.50 | Fair |
| 1.00-1.50 | Poor |
|  |  |

Every criterion or characteristic specified in the ISO 25010 were computed with a mean score and its equivalent descriptive rating. A poor rating means unacceptability of the system to be released in the production environment and should be subjected for re-development. A fair rating only equates to ordinary software which contains only the basic feature and does not offer any advantage over other IM. A good rating indicates a satisfying performance of the software and exhibit features less found in popular IMs. Very good rating demonstrates outstanding performance of the software and showcased features advantageous over other IM. Excellent rating bespeaks superb performance and harbors unparalleled features deemed to be the IM of choice if released in the market.