****

**COMSATS University Islamabad**

**Abbottabad, Pakistan**

**IoT-Based Honeypot Network for Cybersecurity Threat Detection and Attack Analysis**  
Version 1.0

***By***

**Salim Khan CIIT/FA21-BCS-193/ATD**

**Zakirullah Salar CIIT/FA21-BCS-186/ATD**

**Mohammad Zahid Rahmat CIIT/FA21-BCS-222/ATD**

***Supervisor*Sayed Shahab Zarin**

***Bachelor of Science in Computer Science (2021-2024)***

**The candidate confirms that the work submitted is their own and appropriate  
 credit has been given where reference has been made to the work of others**.

**DECLARATION**

We hereby declare that this software, neither whole nor as a part has been copied out from any source. It is further declared that we have developed this software and accompanied report entirely on the basis of our personal efforts. If any part of this project is proved to be copied out from any source or found to be reproduction of some other. We will stand by the consequences. No Portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

Saleem Khan ZakirUllah Sarar Muhammad Zahid Rehmat

--------------------------- ------------------- ---------------------------

**CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (CS) “Project title” was developed by   
**STUDENT 1 NAME (CIIT/FAXX-BCS/SE/TN-000)** and **STUDENT 2 NAME (CIIT/FAXX-BCS/SE/TN-000)** under the supervision of “SUPERVISOR NAME” and co supervisor “CO-SUPERVISOR NAME” and that in (their/his/her) opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Computer Sciences.

---------------------------------------

**Supervisor**

---------------------------------------

**External Examiner**

---------------------------------------

**Head of Department**

**(Department of Computer Science)**

**EXECUTIVE SUMMARY**

The Internet of Things (IoT) revolution has introduced countless opportunities for innovation but also significant vulnerabilities that malicious actors can exploit. The **IoT Honeypot System** is a comprehensive solution designed to proactively address these security challenges by simulating IoT environments to detect, analyze, and mitigate cyberattacks.

This system serves as a strategic tool for cybersecurity professionals and administrators by mimicking the behavior of real IoT devices and protocols, thereby luring attackers into a controlled environment. The captured attack data is analyzed in real-time to uncover patterns, detect anomalies, and generate actionable insights to improve system defenses. Furthermore, the platform incorporates machine learning capabilities, enabling the training of models that predict and adapt to evolving threats.

The **IoT Honeypot System** is equipped with secure authentication, user management, and robust alert mechanisms to ensure seamless operation and secure access. Administrators can monitor real-time activities, receive alerts about malicious activities, and visualize data through intuitive dashboards. The system also logs attack details, providing a valuable dataset for forensic analysis and future cybersecurity enhancements.

By deploying the IoT Honeypot System, organizations can strengthen their security posture, stay ahead of potential attackers, and contribute to the broader effort of improving IoT cybersecurity. It combines innovative technology, advanced analytics, and user-friendly functionalities to deliver an effective, scalable, and secure solution in the fight against IoT-based threats.

This solution not only safeguards critical IoT infrastructure but also empowers organizations to better understand and counteract the techniques employed by malicious actors in an increasingly interconnected world.

.

**ACKNOWLEDGEMENT**

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor “Sir Sayed Shahab Zarin”. Without their personal supervision, advice and valuable guidance, completion of this project would have been doubtful. We are deeply indebted to them for their encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us the values of honesty & hard work.

Saleem Khan ZakirUllah Sarar Muhammad Zahid Rehmat

--------------------------- ------------------- ---------------------------

**ABBREVIATIONS**

|  |  |
| --- | --- |
| **SRS** | Software Require Specification |
| **PC** | Personal Computer |
| **FC** | Functional Requirements |
| **NFC** | Non Functional Requirements |
| **UC** | Use Case |
| DFD | Data Flow Diagram |

**TABLE OF CONTENTS**

[**1.** **Introduction** 12](#_Toc180092202)

[**1.1.** **Brief Introduction** 12](#_Toc180092203)

[**1.2.** **Relevance to Course Modules** 12](#_Toc180092204)

[**1.3.** **Project Background** 13](#_Toc180092205)

[**1.4.** **Literature Review** 14](#_Toc180092206)

[**1.5** **Analysis from Literature Review (in the context of your project.** 14](#_Toc180092207)

[**1.6** **Methodology and Software Lifecycle for This Project** 15](#_Toc180092208)

[**1.6.1.** **Agile Software Development Methodology** 15](#_Toc180092216)

[**2.** **Problem Definition** 18](#_Toc180092217)

[**2.1.** **Problem Statement** 18](#_Toc180092222)

[**2.2.** **Deliverables and Development Requirements** 19](#_Toc180092223)

[**2.3.** **Current System (if applicable to your project)** 20](#_Toc180092224)

[**3.** **Requirement Analysis** 22](#_Toc180092225)

[**3.1.** **Use Case Diagram(s)** 22](#_Toc180092230)

[**3.2.** **Detailed Use Case** 22](#_Toc180092231)

[**3.3.** **Functional Requirements** 37](#_Toc180092232)

[**3.4.** **Non-Functional Requirements** 38](#_Toc180092233)

[**4.** **Design and Architecture** 41](#_Toc180092234)

[**4.1.** **System Architecture** 41](#_Toc180092243)

[**4.2.** **Data Representation [Diagram + Description]** 43](#_Toc180092244)

[**4.3.** **Process Flow/Representation** 51](#_Toc180092245)

[**4.4.** **Design Models [along with descriptions]** 53](#_Toc180092246)

[**5.** **Implementation** 56](#_Toc180092247)

[**5.1.** **Algorithm** 56](#_Toc180092249)

[**5.2.** **External APIs** 56](#_Toc180092250)

[**5.3.** **User Interface** 56](#_Toc180092251)

[**6.** **Testing and Evaluation** 57](#_Toc180092252)

[**6.1.** **Manual Testing** 57](#_Toc180092261)

[This is the sample text 57](#_Toc180092262)

[**6.1.1.** **System testing** 57](#_Toc180092270)

[**6.1.2.** **Unit Testing** 57](#_Toc180092271)

[**6.1.3.** **Functional Testing** 58](#_Toc180092272)

[**6.1.4.** **Integration Testing** 58](#_Toc180092273)

[Table 5.4 shows the integration testing 58](#_Toc180092274)

[**6.2.** **Automated Testing:** 59](#_Toc180092275)

[This is the sample text 59](#_Toc180092276)

[**6.2.1.** **Tools used:** 59](#_Toc180092277)

[Table 5.5 shows the 59](#_Toc180092278)

[**7.** **Conclusion and Future Work** 60](#_Toc180092279)

[**7.1.** **Conclusion** 60](#_Toc180092281)

[**7.2.** **Future Work** 60](#_Toc180092282)

[**8.** **References** 61](#_Toc180092283)

**LIST OF FIGURES**

Fig 3.1 Use Case Diagram 21

Figure 4.2.1 System Sequance Diagram 42

Figure 4.2.2 Sequence diagram for Registration 43

Figure 4.2.3 Sequence diagram for Sign in 44

Figure 4.2.4 Sequence diagram for Manage Profile 45

Figure 4.2.5 Sequence diagram for create Chat Rooms 46

Figure 4.2.6 Sequence diagram for invite members 47

Fig 4.2.7 Sequence diagram for join chat room 48

Fig 4.2.8 Sequence diagram for search Rooms 49

Fig 4.2.9 Sequence diagram for update room 50

Fig 4.2.10 Sequence diagram for delete chat room 51

Fig 4.3.1 Process Flow/Representation 52

Fig 4.4.1 Data Flow Diagram level 0 53

Fig 4.4.2 Data Flow Diagram level 1 54

Fig 4.4.3 Data Flow Diagram level 2 55

**LIST OF TABLES**

Table 3.1 Register ..22

Table 3.2 Sign In ..22

Table 3.3 Manage Profile ..22

Table 3.4 Manage Profile ..22

Table 3.5 Create Chat Room ..22

Table 3.6 Search Room ..22

Table 3.7 Join Room ..22

Table 3.8 Schedule Chat Room ..22

Table 3.9 Create Groups (open/private) ..22

Table 3.10 Leave Room ..22

Table 3.11 Delete chatrooms 9

Table 3.12 log Out ..22

1. **Introduction**

The IoT Honeypot System is a cybersecurity solution designed to simulate the behavior of IoT devices and protocols, attract malicious actors, and analyze their activities in a controlled environment. Unlike conventional security tools, the IoT Honeypot System proactively detects and collects data on real-world attack strategies, helping organizations improve their defenses. This project combines real-time monitoring, data analysis, and machine learning to deliver insights into emerging IoT threats, making it a valuable resource for cybersecurity professionals and researchers.

* 1. **Brief Introduction**

The project focuses on developing the **IoT Honeypot System**, a platform designed to mimic IoT environments, attract attackers, and capture attack data for analysis. Its primary goal is to proactively identify vulnerabilities and threats targeting IoT devices and provide actionable insights to enhance security measures. By simulating IoT protocols and real-time attack monitoring, this system offers a comprehensive approach to understanding and mitigating cyber threats.

**Outcome:**

The project will deliver a functional IoT Honeypot System that features secure authentication, real-time attack monitoring, alert generation, attack data analysis, and machine learning-based threat prediction. These features empower organizations to proactively detect, understand, and counteract IoT-based attacks.

**Tools and Methodology:**

The development of this project follows Agile methodology to ensure continuous feedback and iterative progress. The system is built using tools like Python, Flask for the backend, and MongoDB for data storage, with machine learning models integrated for advanced analytics.

* 1. **Relevance to Course Modules**

Relevance to Course Modules in the Context of the Roomy Proposal:

**1.2.1 Database Management:**  
The IoT Honeypot System involves storing and managing large volumes of attack data, such as logs, timestamps, IP addresses, and attack types. Knowledge of database management ensures efficient data storage, query optimization, and retrieval, enabling effective analysis.

**1.2.2 Web Technologies:**  
The system leverages web technologies for its user interface and backend functionalities. Flask is used to handle backend operations, while the frontend is built using modern web frameworks. MongoDB is chosen for its flexibility in managing unstructured attack data.

**1.2.3 Software Engineering Principles:**  
The project follows core software engineering principles, including requirements gathering, system design, coding, and testing, to ensure the IoT Honeypot System is reliable, scalable, and secure.

**1.2.4 Human-Computer Interaction (HCI):**  
A user-friendly interface is essential for visualizing real-time attacks, alerts, and analytics. The design prioritizes usability, ensuring intuitive navigation for security professionals.

**1.2.5 Project Management:**  
Agile methodology is employed throughout the project lifecycle, featuring sprint planning, stand-up meetings, sprint reviews, and retrospectives. This iterative approach ensures continuous improvements and alignment with project goals.

**1.2.6 Privacy and Security:**  
The system emphasizes secure coding practices, data encryption, and authentication mechanisms to protect sensitive user and attack data. Adherence to data protection regulations is a key focus.

**1.2.7 Quality Assurance and Software Testing:**  
Various testing techniques, including unit testing, integration testing, and user acceptance testing (UAT), ensure the system functions as intended. Emphasis is placed on validating real-time attack monitoring and data capture functionalities.

* 1. **Project Background**

The IoT Honeypot System is designed to address the growing cybersecurity threats faced by IoT devices. With the proliferation of IoT technology, malicious actors often exploit vulnerabilities in these devices. The system creates a simulated IoT environment to attract attackers, analyze their behavior, and generate insights to enhance security measures. Inspired by the need for proactive threat detection, the project aims to provide organizations with a powerful tool to understand and mitigate cyber risks in the IoT domain.

* 1. **Literature Review**

**Honeypots in Cybersecurity:**  
Honeypots have long been used in cybersecurity as a decoy system to attract attackers and study their behavior. Traditional honeypots, however, lack the ability to simulate IoT-specific protocols effectively.

**IoT-Specific Challenges:**  
IoT devices often suffer from weak security measures, such as default credentials and unpatched vulnerabilities. These challenges highlight the importance of tailored solutions like IoT honeypots to address unique IoT threats.

**Existing Solutions:**

* **Conventional Honeypots:** Focus on generic systems but fail to simulate IoT-specific protocols effectively.
* **Specialized IoT Honeypots:** Emerging solutions exist but often lack comprehensive features like machine learning integration and real-time analytics.

**Comparison to IoT Honeypot System:**  
The IoT Honeypot System bridges gaps by offering real-time monitoring, comprehensive attack data analysis, and machine learning-driven insights. Unlike existing solutions, it emphasizes ease of use, scalability, and adaptability to evolving threats.

* 1. **Analysis from Literature Review (in the context of your project.**

The **1.5.1 Identification of Weaknesses in Existing Platforms:**  
Existing solutions often lack real-time monitoring or IoT-specific protocol simulation, limiting their effectiveness in capturing authentic attack data.

**1.5.2 Comparison with Proposed Solutions:**  
The IoT Honeypot System stands out by offering a robust feature set, including real-time attack monitoring, detailed data analysis, and machine learning capabilities.

**1.5.3 Emphasis on Advantages and Benefits:**  
The system provides organizations with actionable insights into IoT threats, allowing them to strengthen their security posture and proactively defend against attacks.

**1.5.4 Scope and Limitations Consideration:**  
While addressing critical IoT security gaps, the system must account for challenges like managing large data volumes and preventing potential system misuse.

* 1. **Methodology and Software Lifecycle for This Project**

For the development of Roomy, a structured methodology and software development lifecycle (SDLC) model are critical to ensuring that the project progresses efficiently, meets user requirements, and delivers a high-quality, functional product. The selected methodology for this project is Agile, as it offers the flexibility and iterative approach needed for dynamic web applications like Roomy.



**Agile Software Development Methodology**  
The Agile methodology ensures iterative development with continuous feedback loops, enabling the IoT Honeypot System to adapt to evolving requirements.

**Key Characteristics:**

* **Iterative Development:** Features like protocol simulation, data analysis, and alert generation will be developed and tested incrementally.
* **Continuous Feedback:** Regular stakeholder feedback ensures alignment with user needs and goals.

**Implementation:**  
The Scrum framework will be applied, featuring sprints focused on delivering specific functionalities. Key practices like daily stand-ups and retrospectives ensure progress and quality.

**1.6.2 Software Development Lifecycle (SDLC)**

**Requirement Analysis:** Identify key functionalities, such as attack simulation, data logging, and machine learning model integration.

**Design:** Develop the system architecture, focusing on scalability, user interface design, and seamless backend integration.

**Development:** Implement features using Python, Flask, and MongoDB, ensuring robust real-time attack monitoring and data analysis capabilities.

**Testing:** Conduct unit, integration, and user acceptance testing to validate system reliability and performance.

**Deployment:** Deploy the system in a real-world environment, testing its ability to handle live attack scenarios.

**Maintenance and Support:** Regular updates, bug fixes, and new feature additions based on user feedback ensure the system remains effective and relevant.

Here’s the rewritten content tailored for an IoT Honeypot System:

**2. Problem Definition**

This section outlines the problem definition for the IoT Honeypot System, including the Problem Statement and Deliverables.

**2.1. Problem Statement**

The IoT Honeypot System addresses the growing security challenges posed by the rapid proliferation of Internet of Things (IoT) devices. With the increased connectivity of IoT devices, malicious actors have more opportunities to exploit vulnerabilities, leading to data breaches, unauthorized access, and botnet attacks. The identified issues include:

**2.1.1. Escalating IoT Security Threats:**

IoT devices often lack robust security measures, making them susceptible to cyberattacks such as Distributed Denial of Service (DDoS) and unauthorized access. Current solutions fail to provide an adaptive and real-time system to detect and analyze these threats effectively.

**2.1.2. Limited Threat Intelligence Collection:**

Existing cybersecurity systems are not specifically designed to capture detailed information about IoT-targeted attacks. This lack of targeted threat intelligence impedes the ability to preemptively secure IoT devices.

**2.1.3. Absence of Scalable and Adaptive Honeypot Solutions:**

Most available honeypot systems are not optimized for IoT environments. They lack scalability, adaptability to diverse IoT protocols, and integration with modern security frameworks.

**2.1.4. Inadequate Real-Time Monitoring and Response:**

Many current systems fail to provide real-time monitoring and actionable insights. This delays response times, allowing attackers to exploit IoT vulnerabilities before mitigation measures can be implemented.

**2.1.5. Complex Deployment and Maintenance:**

Existing solutions often require extensive technical expertise for deployment and maintenance, making them inaccessible for organizations with limited resources or expertise.

**2.2. Deliverables and Development Requirements**

The IoT Honeypot System is designed to address these problems with the following deliverables and features:

**2.2.1. Fully Functional IoT Honeypot Platform:**

Develop a robust platform capable of simulating various IoT devices to attract and monitor malicious actors. The system will log and analyze attack patterns to enhance threat intelligence.

**2.2.2. Real-Time Threat Detection and Analytics:**

Provide real-time monitoring of attacks on IoT devices. The system will include analytics dashboards to visualize and report data on attack vectors, sources, and trends.

**2.2.3. Scalable Architecture:**

Design the system to scale across multiple IoT device simulations, supporting both small-scale and enterprise-level deployments.

**2.2.4. Protocol Emulation:**

Support the emulation of various IoT communication protocols (e.g., MQTT, CoAP, HTTP) to mimic real-world IoT environments.

**2.2.5. Automated Threat Response:**

Integrate an automated threat response mechanism to alert administrators and mitigate attacks in real-time.

**2.2.6. Centralized Log Management:**

Provide centralized logging for all interactions with the honeypot, ensuring efficient storage, retrieval, and analysis of attack data.

**2.2.7. Secure Deployment Framework:**

Ensure secure deployment with minimal exposure to attackers, protecting the honeypot system from exploitation.

**2.2.8. User-Friendly Interface:**

Design a user-friendly interface for system configuration, monitoring, and reporting. The interface will lower the barrier for non-technical users.

**2.2.9. Integration with Existing Security Systems:**

Enable seamless integration with Security Information and Event Management (SIEM) tools and other cybersecurity frameworks for enhanced incident response.

**2.3. Current Systems (if applicable to your project)**

Various systems exist to address cybersecurity threats, including generic honeypot solutions and IoT security frameworks. Below are two notable examples:

**2.3.1. Traditional Honeypot Systems:**

These systems are designed to attract and monitor general cyberattacks. While effective for traditional IT infrastructure, they lack the capability to emulate diverse IoT environments and address specific IoT vulnerabilities.

**2.3.2. IoT-Specific Security Tools:**

Some tools are tailored for IoT security but focus more on endpoint protection and device management rather than threat intelligence collection. These tools often lack real-time attack analysis and adaptive emulation features.

**Limitations of Current Systems:**

Traditional honeypots and IoT-specific tools face the following limitations:

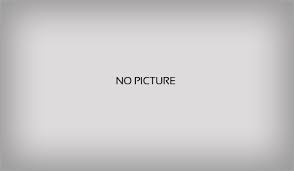
* Lack of scalability to support diverse IoT protocols.
* Inability to simulate complex IoT device interactions realistically.
* Limited real-time monitoring and actionable threat intelligence capabilities.
* Complexity in deployment, making them inaccessible for smaller organizations.

**IoT Honeypot System Improvement:**

The IoT Honeypot System overcomes these limitations by providing:

* A scalable platform with robust protocol emulation for diverse IoT environments.
* Real-time analytics and automated threat responses for enhanced security.
* User-friendly interfaces and secure deployment options to ensure accessibility for all users.

The following figure is a sample figure, Figure 2.1. You are required to follow the same style of numbering and caption for the whole report.



*Figure 2.1: Sample picture*

The following table (Table 2.1) is sample table; you are required to follow the same style of numbering and caption for the whole report.

*Table 2.1: Sample Table*

|  |  |  |
| --- | --- | --- |
| **Header 1** | **Header 2** | **Header 3** |
| Text | Text | Text |
|  |  |  |

The following list style is the sample to consistently follow in the whole report.

* List items 1
* List items 2

1. **Requirement Analysis**

Here is a Use case Diagram of Iot HoneyPot System.

4. 1. **Use Case Diagram(s)**

A diagram of a system

Description automatically generated

Fig 3.1 Use Case Diagram

* 1. **Detailed Use Case**

**Fully dressed Use cases:**

**Simulate IoT Protocols:**

|  |  |
| --- | --- |
| **Use case ID** | UC -01 |
| **Use case Name** | Simulate IoT Protocols |
| **Actors** | Attacker |
| **Description** | The system mimics the behaviour of IoT devices and protocols to attract attackers and capture malicious activity. |
| **Trigger** | An attacker initiates an interaction with the system’s simulated IoT devices. |
| **Preconditions** | The IoT Honeypot system is active and configured to simulate IoT protocols.  Attacker has access to the simulated environment |
| **Postconditions** | Attack data is captured and logged for analysis.  The system remains operational for subsequent interactions. |
| **Normal Flow** | The attacker sends malicious data or exploits vulnerabilities within the simulated IoT environment.  The IoT simulator receives the malicious data or detects the exploit attempt.  The simulator analyzes the received data or the detected activity to identify the attack type and source.  The simulator records the attack data, including timestamp, source (e.g., IP address, device ID), and type of attack.  The simulator continues to operate, potentially implementing mitigation measures based on the captured attack data. |
| **Alternative flow**  **Alternative Flow 1 (Simulation Error):** | If the simulation fails, the system displays an error message and restarts the simulation. |
| **Exceptions** | If the system crashes, attack data is not captured. |
| **Business Rules** | Simulated protocols must mimic real IoT devices to attract attackers effectively.  The system must ensure sufficient security to prevent unauthorized access. |
| **Assumptions** | The attacker is motivated to interact with the simulated environment. |

**Login:**

|  |  |
| --- | --- |
| **Use case ID** | UC -02 |
| **Use case Name** | Login |
| **Actors** | Admin |
| **Description** | The admin uses the system to gain authorized access by providing valid credentials. |
| **Trigger** | The admin initiates the login process by entering their username and password. |
| **Preconditions** | The system is operational.  The admin has valid credentials stored in the database. |
| **Postconditions** | The admin is granted access to the system’s features and functionality.  Unsuccessful login attempts are logged. |
| **Normal Flow** | The admin navigates to the login page.  The admin enters their username and password.  The system validates the credentials against stored data.  If the credentials are valid, the system grants access and displays the admin dashboard.  The system logs the successful login attempt. |
| **Alternative flow**  **Alternative Flow 1 (Invalid Credentials):** | The system displays an error message if the username or password is incorrect.  The admin is prompted to re-enter their credentials. |
| **Exceptions** | If the database is unreachable, the system displays a “Service Unavailable” message.  Multiple failed login attempts lock the admin’s account temporarily. |
| **Business Rules** | Passwords must meet the minimum security requirements (e.g., length, special characters).  The system must prevent brute force attacks by implementing account lockouts. |
| **Assumptions** | Admin knows their username and password. |

**User Management:**

|  |  |
| --- | --- |
| **Use case ID** | UC -03 |
| **Use case Name** | User Management |
| **Actors** | Admin |
| **Description** | The admin manages user accounts, including creating, updating, and deleting accounts. |
| **Trigger** | The admin selects the user management option from the system dashboard. |
| **Preconditions** | The admin is logged into the system.  The database is operational. |
| **Postconditions** | User data is updated successfully in the system.  Logs are generated for all user management activities. |
| **Normal Flow** | The admin accesses the user management interface.  The admin views the list of users.  The admin selects an action (create, update, delete, or view details).  The system performs the requested action and confirms success.  The system logs the action performed. |
| **Alternative flow**  **Alternative Flow 1 (Invalid Data):** | If the admin provides invalid data during user creation or update, the system displays a validation error message. |
| **Exceptions** | If the database connection is lost, the system displays an error message and retries the operation. |
| **Business Rules** | Only admins with sufficient privileges can manage users.  User passwords must be encrypted before being stored. |
| **Assumptions** | The admin has the authority to manage user accounts. |

**Log out:**

|  |  |
| --- | --- |
| **Use case ID** | UC -04 |
| **Use case Name** | Log out |
| **Actors** | Admin |
| **Description** | The admin wants to log out of the system to end their session securely. |
| **Trigger** | The admin is logged into the system. |
| **Preconditions** | The admin is logged into the system. |
| **Postconditions** | The admin’s session is terminated.  The system redirects the admin to the login page. |
| **Normal Flow** | The admin clicks the logout button.  The system terminates the admin’s session.  The system displays a confirmation message and redirects the admin to the login page. |
| **Alternative flow**  **Alternative Flow 1 (Session Timeout):** | If the admin is inactive for a specified period, the system automatically logs them out and displays a timeout message. |
| **Exceptions** | If the system crashes during logout, the admin session might remain active temporarily. |
| **Business Rules** | Admin sessions must be terminated immediately upon logout to prevent unauthorized access.  Session cookies must be cleared after logout. |
| **Assumptions** | he admin logs out to ensure the security of their session. |

**Monitor Real-Time Attacks:**

|  |  |
| --- | --- |
| **Use case ID** | UC -05 |
| **Use case Name** | Monitor Real-Time Attacks |
| **Actors** | Admin |
| **Description** | Admin monitors ongoing attacks on the simulated IoT environment in real time. |
| **Trigger** | Admin accesses the monitoring dashboard. |
| **Preconditions** | Attack data is being actively captured by the system.  Admin has access credentials to the monitoring system. |
| **Postconditions** | Admin gains insights into ongoing attacks.  Real-time attack activity is logged and displayed. |
| **Normal Flow** | The administrator successfully authenticates with the system using valid credentials.  The system presents a real-time dashboard to the administrator. This dashboard displays live attack activity, including:  The source IP address of the attacker.  The specific IoT protocol being targeted (e.g., MQTT, CoAP).  The exact time of each attack attempt.  The administrator reviews the live attack data on the dashboard.  The administrator analyzes the displayed information to gain critical insights, such as:  Identifying the most frequent attack sources.  Detecting emerging attack trends.  Understanding the impact of attacks on the simulated environment. |
| **Alternative flow**  **(No Attack Data)** | If no attack data is available, the system displays “No ongoing attacks” on the dashboard. |
| **Exceptions** | Dashboard fails to load due to system overload. |
| **Business Rules** | Data updates must occur in real time (e.g., every second). |
| **Assumptions** | Admin has basic knowledge of interpreting dashboard information. |

**Generate Alerts:**

|  |  |
| --- | --- |
| **Use case ID** | UC -06 |
| **Use case Name** | Generate Alerts |
| **Actors** | Admin |
| **Description** | The system notifies the admin of critical attack patterns or anomalies in real time. |
| **Trigger** | The system detects unusual activity or a critical threshold breach. |
| **Preconditions** | The system is actively monitoring attacks.  Admin is subscribed to notifications. |
| **Postconditions** | Admin is informed of critical events. |
| **Normal Flow** | The IoT Simulator continuously monitors for suspicious activity.  upon detection of a critical attack pattern by the Security Module, an alert is generated.  The Notification Service then dispatches both an email notification to the administrator's registered email address and an SMS message to their mobile phone number.  Ensuring timely awareness and response to the critical security threat. |
| **Alternative flow**  **(Alert Delivery Failure)** | If notification delivery fails, the system logs the issue and retries. |
| **Exceptions** | Alerts are not generated if attack thresholds are incorrectly set. |
| **Business Rules** | Alert messages must include details like time, type, and severity. |
| **Assumptions** | Admin regularly checks their email or SMS for notifications. |

**Analyze Attack Data:**

|  |  |
| --- | --- |
| **Use case ID** | UC -07 |
| **Use case Name** | Analyze Attack Data |
| **Actors** | Admin |
| **Description** | Allows the admin to study and interpret attack data to identify trends and vulnerabilities. |
| **Trigger** | Admin selects the option to analyze captured attack data. |
| **Preconditions** | The system has captured attack data. Admin has access to the analysis tools. |
| **Postconditions** | Admin obtains insights about attack patterns.  Analytical results are stored for future reference. |
| **Normal Flow** | Admin logs into the system.  Admin navigates to the data analysis section.  Admin selects datasets or attack logs to analyze.  The system processes and displays detailed attack insights, such as trends, attack sources, and vulnerabilities. |
| **Alternative flow**  **(No Data)** | If no data is available for analysis, the system displays an error message and prompts the admin to wait for data collection. |
| **Exceptions** | If the system experiences an error during analysis (e.g., processing overload), it displays an error message and logs the issue for debugging. |
| **Business Rules** | The system must visualize complex data in an intuitive format.  Data should be anonymized when necessary for security compliance. |
| **Assumptions** | Admin is familiar with interpreting analytical charts and statistics. |

**Train Machine Learning Model:**

|  |  |
| --- | --- |
| **Use case ID** | UC -08 |
| **Use case Name** | Train Machine Learning Model |
| **Actors** | Admin |
| **Description** | Allows the admin to train machine learning models using the captured attack data to improve future threat detection. |
| **Trigger** | Admin initiates the machine learning model training process. |
| **Preconditions** | Attack data is pre-processed and ready for training.  Admin has access to the training module. |
| **Postconditions** | A trained machine learning model is saved for deployment.  System accuracy is improved for future detections. |
| **Normal Flow** | * Admin logs in and navigates to the model training section. * Admin selects the datasets to use for training. * The system preprocesses the data, removing outliers and normalizing values. * Admin configures training parameters (e.g., model type, epochs). * The system trains the model and displays progress in real-time. * Upon completion, the system saves the trained model and generates a performance report. |
| **Alternative flow**  **(Training Error)** | If the model training fails (e.g., insufficient data or incompatible parameters), the system displays an error message and logs the issue for troubleshooting. |
| **Exceptions** | System crash during training leads to a loss of progress and requires a restart. |
| **Business Rules** | Ensure high-quality, balanced datasets for accurate training. Prevent model overfitting through appropriate parameter tuning. |
| **Assumptions** | Admin has a basic understanding of machine learning concepts and processes. |

**Visualize Attack Data:**

|  |  |
| --- | --- |
| **Use case ID** | UC -09 |
| **Use case Name** | Visualize Attack Data |
| **Actors** | Admin |
| **Description** | Provides an interactive interface for the admin to view and interpret attack data through charts, graphs, and dashboards. |
| **Trigger** | Admin opens the visualization module. |
| **Preconditions** | Attack data exists in the database. Admin has access to the visualization dashboard. |
| **Postconditions** | Admin gains actionable insights from visualized data. Visualization preferences are saved for future use. |
| **Normal Flow** | 1. Admin logs into the system.  2. Admin navigates to the data visualization dashboard.  3. The system displays attack data in various formats (e.g., bar charts, pie charts, heatmaps).  4. Admin interacts with the visualizations (e.g., filters data by time, type, or source). |
| **Alternative flow**  **(No Data)** | If no data is available, the system displays a message and provides options to refresh or select a different dataset. |
| **Exceptions** | Visualization errors due to missing libraries or rendering issues are logged for debugging. |
| **Business Rules** | Visualization should be intuitive and easy to interpret. Data privacy must be ensured during rendering. |
| **Assumptions** | Admin has basic knowledge of interpreting charts and graphs. |

* 1. **Functional Requirements**

The functional requirements for the IoT Honeypot System outline the key features and capabilities necessary to ensure the system effectively simulates IoT environments, attracts malicious actors, and gathers valuable threat intelligence. These requirements are designed to provide a secure, scalable, and user-friendly platform for monitoring and analyzing IoT-targeted attacks. The following are the functional requirements for the IoT Honeypot System:

**3.3.1. System Deployment and Initialization:**

Provide a mechanism for administrators to deploy and initialize the IoT Honeypot System on different infrastructures (e.g., cloud-based or on-premises).

Ensure the system is easy to configure with options to simulate different IoT devices and protocols.

**3.3.2. Device Emulation:**

Enable the simulation of various IoT devices, including smart sensors, cameras, and home automation devices, to attract attackers.

Support the emulation of diverse IoT communication protocols such as MQTT, CoAP, and HTTP.

**3.3.3. Attack Monitoring and Logging:**

Continuously monitor incoming traffic for suspicious activities targeting the emulated devices.

Log all attack attempts, including source IP addresses, timestamps, attack vectors, and payloads.

**3.3.4. Real-Time Threat Detection and Analytics:**

Provide real-time detection of malicious activities with alerts for specific threats.

Integrate analytics dashboards to visualize attack data, identify trends, and generate reports.

**3.3.5. Temporary Honeypot Configurations:**

Allow administrators to configure temporary honeypot instances for short-term experiments or specific threat analyses.

Schedule these configurations to expire automatically after a defined period.

**3.3.6. Invitation and Access Management:**

Enable secure access for authorized personnel to view and manage the honeypot system.

Provide options for sharing limited access with external collaborators for analysis purposes.

**3.3.7. Search and Analysis of Logs:**

Implement a search feature to quickly find specific log entries or analyze attack data by keyword, source, or protocol.

Support exporting logs for further external analysis.

**3.3.8. Automated Responses and Alerts:**

Include automated response mechanisms such as IP blocking or notifying administrators when predefined thresholds are exceeded.

Ensure the system can send alerts via email or other communication channels for critical threats.

**3.3.9. Honeypot Management and Scalability:**

Allow administrators to schedule, modify, and delete honeypot configurations as needed.

Ensure the system can scale to handle multiple honeypot instances simultaneously across different locations or infrastructures.

**3.3.10. System Security and Authentication:**

Ensure secure access to the system through user authentication mechanisms.

Protect all data logs and configurations with encryption and implement secure user logout options to maintain privacy.

* 1. **Non-Functional Requirements**

**Non-Functional Requirements for IoT Honeypot System**

The non-functional requirements describe how the IoT Honeypot System should operate and the quality standards it must maintain. These focus on aspects such as performance, security, usability, scalability, and maintainability to ensure the system meets its objectives effectively. Below are the non-functional requirements for the IoT Honeypot System:

**3.4.1. Performance**

* **Response Time:** The IoT Honeypot System should process attack data and generate alerts in real-time to ensure immediate detection and response to malicious activities.
* **Scalability:** The system must handle an increasing number of simulated IoT devices and concurrent attacks as the platform grows, without compromising performance.
* **Capacity:** The system should be capable of supporting simultaneous operations, including multiple honeypot simulations, large-scale data logging, and analytics processing, especially during high-traffic periods.

**3.4.2. Security**

* **Data Protection:** All sensitive data, including attack logs, configurations, and user credentials, must be encrypted to protect against unauthorized access.
* **Access Control:** Only authenticated and authorized users should be able to access the system's features, logs, and administrative controls.
* **Activity Logging:** The system should maintain comprehensive logs of all activities, including administrative actions and attacker interactions, to detect suspicious activities and trace potential breaches.

**3.4.3. Usability**

* **Easy to Use:** The IoT Honeypot System’s interface should be intuitive and simple to navigate, allowing administrators to configure simulations, monitor attacks, and access reports with minimal effort.
* **Accessibility:** The platform should adhere to accessibility guidelines to ensure usability for administrators with disabilities.
* **Help and Support:** Provide user manuals, guides, and an integrated help system to assist administrators in understanding and operating the system.

**3.4.4. Reliability**

* **Error Handling:** The system should handle errors gracefully, such as restarting simulations in case of failures, without disrupting overall functionality.
* **High Availability:** The IoT Honeypot System should be operational with minimal downtime, ensuring constant monitoring and data collection, even during updates or maintenance.
* **Data Integrity:** Attack logs and configurations must be protected from data corruption or loss, ensuring that all collected information remains accurate and usable.

**3.4.5. Compatibility**

* **Protocol Support:** The system should support various IoT communication protocols, including MQTT, CoAP, HTTP, and others, to simulate diverse IoT environments effectively.
* **Device Compatibility:** The platform should function seamlessly across different hardware setups, including cloud servers and on-premises infrastructure.
* **Integration Compatibility:** The system should be compatible with existing cybersecurity tools, such as SIEM platforms and threat intelligence databases, to enhance overall functionality.

**3.4.6. Scalability**

* **Add More Resources:** The IoT Honeypot System should allow for the addition of computational resources (e.g., servers or storage) to accommodate increased data and traffic.
* **System Expansion:** The platform should support scaling up by adding more honeypot instances or simulations to monitor larger attack surfaces.

**3.4.7. Interoperability**

* **Integration:** The IoT Honeypot System should integrate seamlessly with third-party tools and APIs, such as log management systems, machine learning frameworks, or external alerting services, to expand its capabilities.

**3.4.8. Maintainability**

* **Easy to Update:** The system’s architecture and codebase should be designed to facilitate easy updates, bug fixes, and feature enhancements.
* **Clear Documentation:** Comprehensive documentation should be provided, detailing system design, configurations, and operational guidelines to support maintenance and onboarding of new developers.

1. **Design and Architecture**

Here is detailed Design and Architecture of Iot HoneyPot System.

8. 1. **System Architecture**

Here is the System Architecture of Iot HoneyPot System:

**4.1.1. Frontend (React.js)**

**Description:**  
This module represents the user interface of the IoT Honeypot System. It is developed using React.js, a dynamic JavaScript library ideal for building responsive and interactive web applications. The frontend provides administrators with a modern, user-friendly interface to monitor and manage the IoT Honeypot System.

**Functionality:**  
The frontend allows administrators to log in securely, configure honeypot simulations, monitor real-time attacks, and analyze captured data through dashboards. It enables users to manage system settings, view alerts, and access data visualizations. Designed to be responsive, the interface adapts to various screen sizes, ensuring seamless operation across desktops, tablets, and mobile devices.

**4.1.2. Backend (Node.js + Express.js)**

**Description:**  
This module constitutes the server-side logic of the IoT Honeypot System. Built using Node.js and Express.js, it manages data flow between the frontend and the database, processes incoming requests, and executes business logic.

**Functionality:**  
The backend handles key operations such as configuring honeypots, capturing attack data, and generating alerts. It processes requests from the frontend, ensuring that actions like simulation management, log retrieval, and analysis requests are completed in real time. It also integrates with the database to securely store and retrieve data, while ensuring reliable and efficient communication between system components.

**4.1.3. Authentication Service**

**Description:**  
This module manages user authentication and access control for the IoT Honeypot System, ensuring that only authorized users can access the platform.

**Functionality:**  
The authentication service provides secure user login and registration, utilizing token-based authentication (e.g., JWT) to verify credentials. It ensures that sensitive user information, such as passwords, is encrypted. This service also enforces role-based access control, granting or restricting permissions based on the user’s role, and provides secure logout functionality to maintain system integrity.

**4.1.4. Simulation Service**

**Description:**  
This module handles the creation and management of IoT simulations, enabling the system to mimic various IoT devices and protocols to attract attackers.

**Functionality:**  
The simulation service allows administrators to configure and deploy honeypot simulations for protocols like MQTT, CoAP, and HTTP. It enables the addition or removal of simulated IoT devices and the customization of protocol settings. The service ensures that attackers can interact with the simulated environment, capturing all malicious activities for further analysis.

**4.1.5. Attack Monitoring Service**

**Description:**  
This module provides real-time monitoring and logging of attack activities targeting the simulated IoT devices.

**Functionality:**  
The attack monitoring service tracks all interactions with the honeypot, capturing details such as timestamps, source IPs, attack vectors, and payloads. It integrates real-time dashboards that display ongoing activities, providing administrators with an overview of attacks as they occur. The service ensures that all activity is logged for detailed analysis and reporting.

**4.1.6. Alert Service**

**Description:**  
This module generates alerts for detected threats and unusual activities within the IoT Honeypot System.

**Functionality:**  
The alert service notifies administrators in real time when specific attack thresholds or conditions are met. Alerts are delivered through email, SMS, or the system interface. The service also integrates with the monitoring module to highlight critical events and ensure immediate action can be taken when necessary.

**4.1.7. Data Analysis Service**

**Description:**  
This module is responsible for analyzing the attack data collected by the honeypot to identify trends, patterns, and vulnerabilities.

**Functionality:**  
The data analysis service processes logs and generates insights into attack behavior, including frequently targeted vulnerabilities and sources. It integrates machine learning algorithms to predict potential future threats. This module provides administrators with visualizations, charts, and detailed reports, enabling proactive security measures.

**4.1.8. User Management Service**

**Description:**  
This module enables administrators to manage system users and their permissions.

**Functionality:**  
The user management service allows for the creation, modification, and deletion of user accounts. Administrators can assign roles (e.g., admin, viewer) and define access levels to system features. The service ensures that only authorized personnel can make changes or access sensitive data, maintaining a secure operational environment.

**4.1.9. Database (ElasticSearch)**

**Description:**  
This module manages the storage and retrieval of data within the IoT Honeypot System. Built using **Elasticsearch**, it is optimized for storing, indexing, and searching large volumes of attack logs and system data. Elasticsearch is a distributed, highly scalable search and analytics engine tailored for real-time data processing.

**Functionality:**

* **Data Storage:** Elasticsearch stores structured and unstructured data, including attack logs, user activity, and honeypot configurations.
* **Real-Time Search:** Provides fast and efficient full-text search capabilities, enabling administrators to quickly retrieve relevant logs and analyze data.
* **Scalability:** Elasticsearch’s distributed architecture allows the system to scale horizontally, ensuring high performance as the volume of logs and system activity grows.
* **Analytics Integration:** Elasticsearch integrates seamlessly with the system’s analytics dashboards, supporting real-time data visualization and reporting.
* **Data Retention Policies:** Allows configuration of data retention periods, ensuring efficient use of storage resources by automatically archiving or deleting older logs.
  1. **Data Representation [Diagram + Description]**

No table of contents entries found.

Here is detailed Design and Architecture of Roomy.

* + 1. **System Sequence Diagram**

Here are Sequence Diagrams for all the use cases

Simulate IoT Protocols:

A screenshot of a computer

Description automatically generated

**Description:**

An attacker sends malicious data to the IoT simulator, which detects, analyzes, and records the attack. A simulation report is generated and sent to the administrator for further analysis and system improvement.

Monitor Real-Time Attacks:

A diagram of a project

Description automatically generated

**Description:**

This SSD focuses on the administrator's ability to monitor real-time attacks. After logging in with valid credentials and successful authentication, the administrator accesses the dashboard, which displays live attack details, including IP addresses, timestamps, and trends. The administrator reviews the live data and analyzes insights to identify frequent attack sources and patterns, aiding in proactive mitigation and strategy formulation.

Generate Alerts:

A diagram of a security system

Description automatically generated

**Description:**

This SSD diagram illustrates the process of generating and notifying the admin about critical attack patterns detected by an IoT system. The IoT Simulator detects a critical attack pattern and forwards the information to the Security Module, which generates an alert. This alert is sent to the Notification Server, which triggers notifications to the admin via email and SMS to ensure immediate awareness of the detected threat

Analyze Attack Data:

A diagram of a system

Description automatically generated

**Description:**

This sequence diagram illustrates how an administrator logs into the system to analyze attack data. The process begins with the administrator providing login credentials, followed by system authentication and granting access. The administrator navigates to the data analysis section, where the system displays the relevant interface. Attack logs are selected for processing, and the system analyzes the selected data, providing insights such as attack sources, trends, and vulnerabilities for further review by the administrator.

Train machine Learning Model:

A screenshot of a computer

Description automatically generated

**Description:**

The admin logs in, selects datasets, configures parameters, and starts the training process. The system processes data, trains the model, notifies the admin upon completion, displays a performance report, and saves the trained model.

Visualize Attack Data:

A screenshot of a computer program

Description automatically generated

**Description:**

The admin logs in, navigates to the visualization section, and applies filters to refine attack data visualizations. The system loads data, displays visualizations, updates them based on filters, and provides interactive, updated views for analysis.

**Process Flow/Representation**

Here is a process flow diagram for Iot HoneyPot System.

A flowchart of a sign-in

Description automatically generated

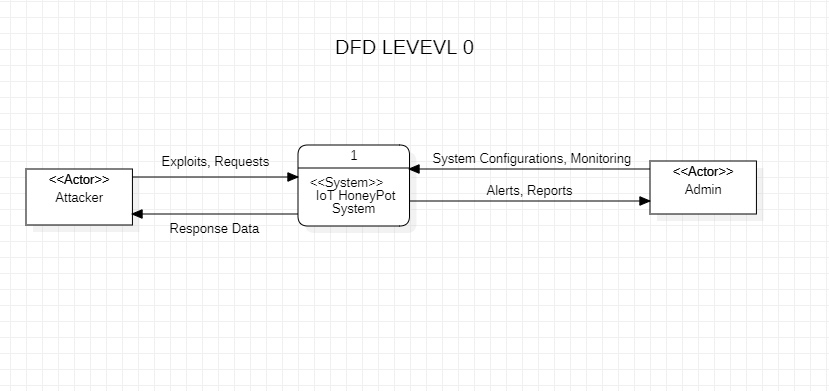
Fig 4.3 process flow diagram

* 1. **Design Models [along with descriptions****]**

Here is Design Models with different levels of diagram

* + 1. **Data Flow Diagram Level 0:**

**DFD LEVEL 0:**



**Description:**

The Level 0 DFD provides an overview of the IoT Honeypot System as a single process with its external entities. It highlights the interactions between attackers and administrators with the system, showing the main input and output data flows. Attackers interact with the system by attempting to exploit vulnerabilities, while administrators manage the system by configuring simulations, monitoring activity, and reviewing reports. The IoT Honeypot System processes these interactions by simulating IoT devices, capturing attack attempts, and providing real-time alerts and detailed reports to the administrators.

**DFD Level 1:**

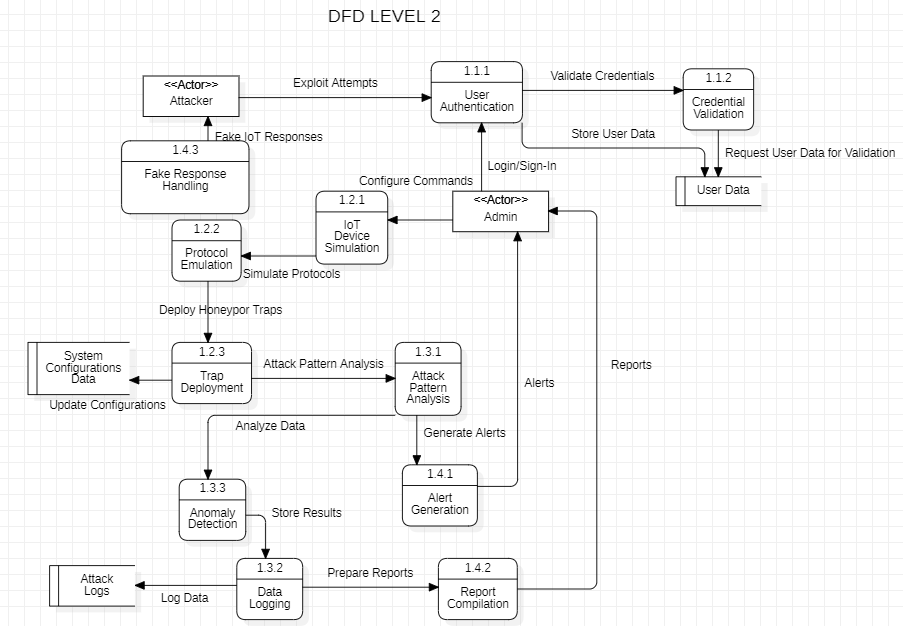
A diagram of a system

Description automatically generated

**Description:**

The Level 1 DFD for an IoT honeypot system breaks down the main process into its key subprocesses, providing more detail on how the system functions. Key subprocesses include user authentication, IoT device simulation, protocol emulation, trap deployment, attack pattern analysis, and alert/report generation. Each subprocess highlights its inputs, outputs, and the data stores involved, such as user data, attack logs, and system configurations. This breakdown offers a clear understanding of how the system manages interactions with attackers and administrators.

**DFD Level 2:**



**Description:**

The Level 2 DFD for an IoT honeypot system details the subprocesses from Level 1, including device simulation, protocol emulation, trap deployment, attack analysis, and alert/report generation. It illustrates how data flows between processes like anomaly detection and data logging, interacting with data stores such as attack logs and configurations. This level provides a comprehensive view of the system’s operations, focusing on detecting, analyzing, and responding to attacks.

1. **Implementation**

This chapter will discuss implementation details supported by UML diagrams (if applicable). You will not put your source code here. Any of the following sections may be included based on your project.

1. 1. **Algorithm**

Mention the algorithm(s) used in your project to get the work done with regards to major modules. Provide a pseudocode **OR** a natural language explanation regarding the functioning of main features. Be sure to use the correct syntax and semantics for algorithm representations.

* 1. **External APIs**

Describe the APIs used in the table 5.1.

Table 5.1 shows that

*Table 5.1: Details of APIs used in the project*

|  |  |  |  |
| --- | --- | --- | --- |
| **Name of API** | **Description of API** | **Purpose of usage** | **List down the function/class name in which it is used** |
|  |  |  |  |
|  |  |  |  |

* 1. **User Interface**

Details about user interface with descriptions.

1. **Testing and Evaluation**

This chapter may include the following sections. (Students are required to perform the testing both manually and automatedly).

8. 1. **Manual Testing**

This is the sample text

6. 1. 1. **System testing**

Once the system has been successfully developed, testing has to be performed to ensure that the system working as intended. This is also to check that the system meets the requirements stated earlier. Besides that, system testing will help in finding the errors that may be hidden from the user. There are few types of testing which includes the unit testing, functional testing and integration testing. The testing must be completed before it is being deploy for user to use.

* + 1. **Unit Testing**

Once the system has been successfully developed

* **Unit Testing 1:** Login as FYP Committee as shown in Table 5.1

**Testing Objective:** To ensure the login form is working correctly

*Table 5.1: Login Unit Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Verify user login after click on the ‘Login’ button on login form with correct input data | Username:  L001  Password:  1234 | Successfully log into the main page of the system as FYP Committee member. | Pass |
| 2. |  |  |  |  |

* **Unit Testing 2:** Edit Profile

**Testing Objective:** To ensure the edit profile form is working properly.

*Table 5.2: Edit Profile Unit Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. |  |  |  |  |

* + 1. **Functional Testing**

The functional testing will take place after the unit testing. In this functional testing, the functionality of each of the module is tested. This is to ensure that the system produced meets the specifications and requirements.

* **Functional Testing 1:** Login with different roles as shown in Table 5.3

**Objective**: To ensure that the correct page with the correct navigation bar is loaded.

*Table 5.3: Login Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Login as a ‘FYP Committee’ member. | Username: L001  Password: 1234 | Main page for the FYP Committee member is loaded with the FYP Committee navigation bar | Pass |
| 2. |  |  |  |  |

* + 1. **Integration Testing**

Table 5.4 shows the integration testing

*Table 5.4: Integration Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Login as “FYP Committee” member | Username: L001  Password: 1234 | Login successful and the FYP Committee page with its navigation bar is loaded and in the view profile page | Pass |
| 2. | Upload student record for Project 1 | - | File successfully uploaded and return to the upload page. Student records are updated. | Pass |
| 3. | View supervising student | - | The list of supervisees shown on the screen. | Pass |

* 1. **Automated Testing:**

This is the sample text

* + 1. **Tools used:**

Table 5.5 shows the

*Table 5.5: Tools used*

|  |  |  |  |
| --- | --- | --- | --- |
| **Tool Name** | **Tool Description** | **Applied on [list of related test cases / FR / NFR]** | **Results** |
|  |  |  |  |
|  |  |  |  |

1. **Conclusion and Future Work**

This chapter concludes the project and highlights future work.

1. 1. **Conclusion**
   2. **Future Work**
2. **References**

References to any book, journal paper or website should properly be acknowledged. Please consistently follow the style. The following are few examples of different resources i.e. journal article, book, and website.

* 1. Lyda M.S. Lau, Jayne Curson, Richard Drew, Peter Dew and Christine Leigh, (1999), Use Of VSP Resource Rooms to Support Group Work in a Learning Environment, ACM 99, pp-2. **(Journal paper example)**
  2. Hideyuki Nakanishi, Chikara Yoshida, Toshikazu Nishmora and TuruIshada, (1996), FreeWalk: Supporting Casual Meetings in a Network, pp 308-314 **(paper on web)** http://www.acm.org/pubs/articles/proceedings/cscw/240080/p308-nakanishi.pdf
  3. Ali Behforooz& Frederick J.Hudson, (1996), Software Engineering Fundamentals, Oxford University Press. Chapter 8, pp255-235. **(book reference example)**
  4. Page Author, Page Title, http://www.bt.com/bttj/archive.htm, Last date accessed**. (web site)**