

# **SECURITY & SURVEILLANCE FOR TEACHERS AND STUDENTS**

**A PROJECT REPORT**

*Submitted by,*

<b>Mr. CHANDRASHEKHAR S</b>	<b>- 20211CCS0065</b>
<b>Ms. SHUBHA K A</b>	<b>- 20211CCS0067</b>
<b>Mr. AUGUSTIAN P B</b>	<b>- 20211CCS0104</b>
<b>Ms. KAVYA JAISHREE J</b>	<b>- 20211CCS0131</b>

*Under the guidance of,*

**Ms. SOUMYA**

*in partial fulfillment for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING (CYBERSECURITY)**

**At**



**PRESIDENCY UNIVERSITY**

**BENGALURU**

**JANUARY 2025**

# PRESIDENCY UNIVERSITY

## SCHOOL OF COMPUTER SCIENCE ENGINEERING

### CERTIFICATE

This is to certify that the Project report “SECURITY & SURVEILLANCE FOR TEACHERS AND STUDENTS” being submitted by “CHANDRASHEKHAR S, SHUBHA K A, AUGUSTIAN P B, KAVYA JAISHREE J” bearing roll numbers “20211CCS0065, 20211CCS0067, 20211CCS0104, 20211CCS0131” in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering (Cybersecurity) is a bonafide work carried out under my supervision.



**Ms. SOUMYA**  
Assistant Professor  
School of CSE & IS  
Presidency University



**Dr. ANANDARAJ S P**  
HoD  
School of CSE & IS  
Presidency University



**Dr. L. SHAKKEERA**  
Associate Dean  
School of CSE  
Presidency University



**Dr. MYDHILI NAIR**  
Associate Dean  
School of CSE  
Presidency University



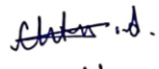

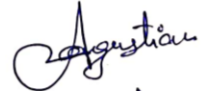

**Dr. SAMEERUDDIN KHAN**  
Pro-Vc School of Engineering  
Dean -School of CSE & IS  
Presidency University

**PRESIDENCY UNIVERSITY**  
**SCHOOL OF COMPUTER SCIENCE ENGINEERING**

**DECLARATION**

We hereby declare that the work, which is being presented in the project report entitled **SECURITY & SURVEILLANCE FOR TEACHERS AND STUDENTS** in partial fulfillment for the award of Degree of **Bachelor of Technology in Computer Science and Engineering (Cybersecurity)** is a record of our own investigations carried under the guidance of **Ms. Soumya**, Assistant Professor, School of Computer Science Engineering & Information Science, Presidency University, Bengaluru.

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

NAME	ROLL NO	SIGNATURE
CHANDRASHEKHAR S	20211CCS0065	
SHUBHA K A	20211CCS0067	
AUGUSTIAN P B	20211CCS0104	
KAVYA JAISHREE J	20211CCS0131	

## **ABSTRACT**

The rapid digitization of education necessitates the creation of secure online environments for students and educators. This report introduces a user-friendly web-based application designed to streamline the management of URL filtering policies, ensuring enhanced cybersecurity in educational institutions. The system leverages pfSense as a firewall, LDAP for authentication and directory services, and MongoDB for efficient database management. The administrator interface offers an intuitive platform where educators can add, remove, or modify URL filtering policies without requiring advanced technical expertise.

By abstracting the complexities of firewall configuration, the application updates policies in real-time through REST APIs with pfSense, enabling seamless integration and improved efficiency. LDAP-based role-based authentication ensures that only authorized users can access and modify policies, while MongoDB serves as a centralized repository for user and policy information.

This approach eliminates the need for direct interaction with the pfSense GUI or command-line interface, making URL policy management accessible to non-technical users. The project demonstrates the integration of these technologies into a scalable, secure, and adaptable solution tailored for educational institutions. It addresses the growing need for simplified yet effective cybersecurity measures, promoting a safer digital environment for students and educators alike.

## ACKNOWLEDGEMENT

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC, School of Engineering and Dean, School of Computer Science Engineering & Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Deans **Dr. Shakkeera L and Dr. Mydhili Nair**, School of Computer Science Engineering & Information Science, Presidency University, and **Dr. Anandaraj S P**, Head of the Department, School of Computer Science Engineering & Information Science, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Ms. Soumya**, Assistant Professor and Reviewer **Ms. Bhavya B**, Assistant Professor, School of Computer Science Engineering & Information Science, Presidency University for her inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the PIP2001 Capstone Project Coordinators **Dr. Sampath A K, Dr. Abdul Khadar A and Mr. Md Zia Ur Rahman**, department Project Coordinators **Dr. Sharmasth Vali Y** and Git hub coordinator **Mr. Muthuraj**.

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

**Chandrashekhar S**

**Shubha K A**

**Augustian P B**

**Kavya Jaishree J**

# **TABLE OF CONTENTS**

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
	<b>ABSTRACT</b>	<b>iv</b>
	<b>ACKNOWLEDGMENT</b>	<b>v</b>
	<b>LIST OF TABLES</b>	<b>ix</b>
	<b>LIST OF FIGURES</b>	<b>x</b>
<b>1.</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Background and Motivation	1
	1.2 Problem Statement	3
	1.3 Significance of the Study	5
	1.3.1. Empowering Teachers and Administrators	5
	1.3.2 Enhancing Network Security	5
	1.3.3 Cost-Effectiveness and Scalability	5
	1.3.4 Practical Contribution to Educational Institutions	6
	1.3.5 Potential for Future Research and Development	6
<b>2.</b>	<b>LITERATURE SURVEY</b>	<b>7</b>
	2.1 Firewall GUI-Based URL Filtering	7
	2.2 Command Line Interface (CLI)	8
	2.3 Third-Party Network Management Tools	9
	2.4 Cloud-Based Filtering	9
	2.5 Proxy-Based Filtering	10
	2.6 DNS-Based Filtering	11
	2.7 Endpoint Security Software	12
	2.8 Manual Whitelisting/Blacklisting	12
	2.9 Browser Plugins	13
	2.10 Machine Learning-Based Filtering	14
<b>3.</b>	<b>RESEARCH GAPS OF EXISTING METHODS</b>	<b>15</b>
	3.1 Complexity for Non-Technical Users	15
	3.2 Lack of Role-Based Access Control (RBAC)	15

3.3	Cost Barriers	15
3.4	Inflexibility in Policy Management	16
3.5	Resource Constraints on Cloud-Based Filtering	16
3.6	Limited Https Filtering in Proxy-Based Methods	16
3.7	Limited Automation and Real-Time Response	16
3.8	Scalability Challenges	17
3.9	Minimal User Empowerment	17
<b>4.</b>	<b>PROPOSED METHODOLOGY</b>	<b>18</b>
4.1	System Overview	18
4.2	Key Components and Methods	18
4.2.1	pfSense Firewall	18
4.2.2	Administrative Webpage	18
4.2.3	MongoDB Database	19
4.2.4	LDAP For Authentication and Role-Based Access Control (RBAC)	19
4.2.5	REST APIs	20
4.3	Integration of Components	20
4.4	Expected Benefits	20
<b>5</b>	<b>OBJECTIVES</b>	<b>21</b>
5.1	Simplify Network Policy Management	21
5.2	Enhance Network Security	21
5.3	Implement Secure User Authentication	22
5.4	Ensure Scalability and Efficient Data Management	22
5.5	Enable Real-Time Policy Updates	23
5.6	Minimize Dependency on IT Staff	23
5.7	Provide A Cost-Effective Solution	24
5.8	Foster A Secure and Productive Digital Learning Environment	24
5.9	Support Future Scalability and Enhancements	24
<b>6</b>	<b>SYSTEM DESIGN AND IMPLEMENTATION</b>	<b>26</b>
6.1	System Design	26
6.1.1	High-Level System Design	27

	6.1.2 Integration of Components	27
	6.1.2.1 pfSense Firewall	27
	6.1.2.2 MongoDB as the Database Layer	28
	6.1.2.3 LDAP for Authentication and Active Directory	28
	6.1.3 Communication Via Rest APIs	28
	6.2 Implementation	29
	6.2.1 Development of the Administrator Webpage	29
	6.2.1.1 Tools and Technologies (Flask, Bootstrap, Python)	29
	6.2.1.2 Webpage Design and User Interface	30
	6.2.2 URL Filtering Policy Management Workflow	30
	6.2.2.1 GET/POST Requests to pfSense via REST API	31
	6.2.2.2 URL Addition and Removal Logic	31
	6.2.2.3 MongoDB Integration for Policy Data Storage and Retrieval	32
	6.2.3 LDAP Configuration	33
	6.2.4 LDAP Integration for Secure Authentication	33
<b>7</b>	<b>TIMELINE FOR EXECUTION OF THE PROJECT</b>	<b>34</b>
<b>8</b>	<b>OUTCOMES</b>	<b>36</b>
<b>9</b>	<b>RESULTS AND DISCUSSIONS</b>	<b>38</b>
	9.1 Policy Update Efficiency and Real-Time Management	38
	9.2 Effectiveness of Role-Based Access Control (RBAC)	38
	9.3 Accuracy and Reliability of URL Filtering	39
	9.4 Web Interface Usability and Design	40
<b>10</b>	<b>CONCLUSION</b>	<b>41</b>
	<b>FUTURE OUTLOOK</b>	<b>42</b>
	<b>REFERENCES</b>	<b>43</b>
	<b>APPENDICES</b>	<b>45</b>



## LIST OF TABLES

Sl. No.	Table Name	Table Caption	Page No.
1	Table 1	pfSense vs Commercial Solutions	3
2	Table 2	Execution plan for web-based url filtering project	34

## LIST OF FIGURES

<b>Sl. No.</b>	<b>Figure Name</b>	<b>Caption</b>	<b>Page No.</b>
1	Figure 1	System architecture of web-based URL filtering	26
2	Figure 2	Query for inserting a URL filtering policy	32
3	Figure 3	Query for retrieving all allowed URLs for a Role	32
4	Figure 4	Gantt Chart for Project Timeline	35

# CHAPTER-1

## INTRODUCTION

As educational institutions continue to embrace digital technologies, ensuring secure and efficient internet usage has become critical. Unrestricted access to the internet poses risks such as exposure to harmful content, reduced productivity, and potential network vulnerabilities. Traditional network management tools like Firepower Management Center (FMC) offer comprehensive solutions but are often too complex and technical for non-expert users, such as teachers, who need simpler ways to manage URL filtering and access policies. These limitations create a need for a more intuitive and efficient solution to manage internet safety and security in educational environments.

This project proposes a comprehensive system to address these challenges by creating a **user-friendly administrator webpage** that enables teachers and administrators to manage Access Control Policies securely and effectively. By integrating open-source tools like **pfSense** for firewall management, **LDAP** for role-based authentication, and **MongoDB** for storing policies and logs, the system ensures both usability and security. The webpage, built with **Flask** and **Bootstrap**, provides a simple and intuitive interface for users to add or remove URLs in real-time while adhering to strict **role-based access controls (RBAC)** to prevent unauthorized modifications.

The proposed system not only simplifies policy management but also empowers educators and administrators to take an active role in maintaining network security. By bridging the gap between technical complexity and user accessibility, this solution provides a scalable, adaptable, and cost-effective alternative to commercial solutions like FMC. Through this project, educational institutions can achieve a balance between usability, security, and efficiency, ensuring a safe and productive digital environment for teachers and students alike.

### 1.1 Background and Motivation

The integration of digital tools and technologies has transformed the way educational institutions operate, enabling improved teaching, learning, and administrative efficiencies. However, this digital transformation also brings significant challenges, particularly in ensuring network security, regulating internet usage, and providing safe online environments

for students and staff. Unrestricted access to the internet can expose users to harmful or non-educational content, pose security risks such as malware attacks, and lead to reduced productivity. Therefore, maintaining robust control over internet policies is a crucial aspect of digital governance in educational institutions.

Traditional solutions for managing network security, such as Firepower Management Center (FMC), offer advanced capabilities for administrators to create and modify access control policies. These systems, while powerful, are often designed with network administrators in mind and are not user-friendly for non-technical users such as teachers. Teachers, who are pivotal in shaping student interactions with online content, often require the ability to add or remove URLs or manage filtering policies themselves. However, the technical complexity of traditional firewalls makes this difficult, leading to a dependency on IT staff for even basic changes. This gap between technical complexity and user accessibility necessitates the development of a more intuitive and user-centric solution for managing network policies.

The challenges are further compounded in educational environments, where a variety of roles, including teachers, administrators, and IT staff, need controlled access to network resources based on their specific responsibilities. A solution is needed that not only simplifies the process of managing access control policies but also ensures strict security measures to prevent unauthorized access or modifications. This is where role-based access control (RBAC) plays a critical role, enabling permissions to be tailored to specific roles, thereby balancing accessibility and security effectively.

This project aims to address these challenges by proposing a comprehensive system that simplifies policy management for non-technical users while maintaining robust security. The solution is built around a **user-friendly administrator webpage**, allowing teachers to manage URL filtering policies with ease. By leveraging open-source tools such as **pfSense** for firewall management, **LDAP** for authentication and role management, and **MongoDB** for storing policies and logs, the system ensures scalability, efficiency, and cost-effectiveness. The webpage, developed using **Flask** and **Bootstrap**, provides an intuitive and interactive interface, enabling real-time modifications to access control policies via **REST APIs**.

Through this approach, the project bridges the gap between technical complexity and usability, empowering teachers to take an active role in managing network security. At the same time, it reduces the burden on IT staff and provides administrators with a cost-effective alternative

to commercial solutions like FMC. By addressing the specific needs of educational institutions, the proposed system contributes to creating a secure, productive, and efficient digital environment for both teachers and students.

Table 1. PFSense VS COMMERCIAL SOLUTIONS

Feature	pfSense	Commercial Solutions
Cost	Free (open-source); only hardware costs.	High subscription or license fees.
Scalability	Highly scalable; suitable for small to large networks.	Scalable, but higher costs for larger networks.
Customization	Fully customizable; supports add-ons and third-party integrations.	Limited customization in pre-configured setups.
Ease of Use	Requires some technical expertise but offers extensive documentation.	User-friendly GUIs but less flexibility.
Features	Supports URL filtering, VPN, traffic shaping, and API integration.	Advanced features available in premium tiers.
Community Support	Strong open-source community support.	Paid support with response time guarantees.

### Why pfSense for this project?

By all means, in terms of cost, flexibility, and meeting specific needs for an educational institution, pfSense definitely leads the pack. Being an open-source software, it is the most customizable, supports advanced features like VPN, URL filtering, and real-time traffic monitoring. The REST APIs integration gives an administrator the flexibility to manage the policies and make updates as necessary because it is all inclusive, even for non-technical users

## 1.2 Problem Statement

Educational institutions are increasingly relying on digital tools for teaching, learning, and administration. However, this digital transformation presents several challenges, particularly around managing internet access and maintaining a secure network environment. With teachers, students, and administrators all requiring varying levels of access to online resources, institutions need to ensure that access control policies are not only effective but also easy to manage.

**Traditional solutions** like Firepower Management Center (FMC) provide robust features for controlling access to online resources and securing networks. However, these solutions are

often complex and designed for use by network administrators, making them unsuitable for non-technical users like teachers. Teachers, who frequently need to modify policies such as URL filtering (e.g., allowing access to educational sites or blocking non-educational content), often lack the tools to make changes independently. This leads to a reliance on IT staff for even minor adjustments, which can slow down the response to emerging needs and create bottlenecks in managing network security. The inability for teachers to control URL filtering and access policies directly adds complexity and inefficiency to the management of digital resources in schools.

Furthermore, educational institutions need to ensure that only authorized individuals can modify access control policies, maintaining a secure and controlled environment. Traditional systems do not always offer the flexibility needed to assign specific roles (such as teachers, admins, or IT staff) different levels of access to network management features. Without **role-based access control (RBAC)**, teachers might inadvertently access sensitive configurations, or non-technical users may be excluded from necessary actions.

**The gap in existing solutions** lies in the need for a more intuitive and accessible way for non-technical users, such as teachers, to manage internet access policies while ensuring network security. There is a pressing need for a system that can simplify the process of managing policies, allow for real-time updates, and provide role-based access control without compromising on security. Traditional systems require technical expertise to update and manage policies, and this complexity often leads to delays in policy enforcement or, worse, poor network management.

This project seeks to solve these problems by designing a **user-friendly interface** for managing access control policies, specifically URL filtering, while incorporating **role-based access control** to ensure that users only have access to the features they are authorized to use. The proposed solution should be accessible to teachers and administrators, allowing them to modify policies in real-time without needing extensive technical knowledge, while ensuring that security is not compromised.

### 1.3 Significance of the Study

This study holds significant value for educational institutions that are increasingly adopting digital tools for administrative and educational purposes. The growing reliance on the internet in schools necessitates the establishment of effective security and access control measures. However, the complexity of existing systems often prevents non-technical users, such as teachers, from actively participating in managing and modifying internet usage policies. The proposed system, which simplifies the process of managing access control policies through a user-friendly interface, directly addresses this gap.

#### 1.3.1 Empowering Teachers and Administrators:

One of the primary contributions of this study is the empowerment of teachers and administrators to take an active role in managing network security. By providing them with an easy-to-use web interface, the system allows them to modify **URL filtering policies** and manage internet access with minimal technical knowledge. This reduces the dependency on IT staff for simple tasks, enabling quicker response times to emerging needs, such as blocking or unblocking specific websites. It also improves the overall efficiency of network management, allowing administrators to focus on more complex issues.

#### 1.3.2 Enhancing Network Security:

The system's integration of **role-based access control (RBAC)** and **LDAP authentication** ensures that only authorized users can make changes to critical network policies. Teachers are allowed to modify policies that are relevant to their role, such as managing URL filters, while administrators can access and adjust more advanced configurations. This separation of duties improves the overall security of the network, ensuring that unauthorized changes cannot be made by users without the appropriate permissions.

#### 1.3.3 Cost-Effectiveness and Scalability:

The proposed solution is built on open-source technologies like **pfSense**, **Flask**, **Bootstrap**, and **MongoDB**, making it a cost-effective alternative to commercial solutions like Firepower Management Center (FMC). Educational institutions, which often operate under tight budgets, can benefit from a powerful, scalable solution without the need for expensive licensing fees.

The system's scalability also allows it to be deployed across multiple institutions or departments, making it suitable for use in a variety of educational environments, from small schools to larger university networks.

#### **1.3.4 Practical Contribution to Educational Institutions:**

In addition to the technical benefits, the system has practical significance in improving the management of educational resources. By providing real-time policy updates, the system ensures that educational institutions can adapt quickly to changing needs. Whether it is blocking distracting websites during study hours or ensuring that new educational resources are accessible, the system provides administrators and teachers with the tools they need to manage internet access more effectively. This contributes to a safer and more focused learning environment.

#### **1.3.5 Potential for Future Research and Development:**

Finally, this study opens the door for further research into user-friendly solutions for managing complex network security tasks. It demonstrates that open-source technologies can be effectively used to create systems that balance user accessibility with security. Future work could explore expanding the system to include additional network management features, such as bandwidth control, application filtering, or advanced traffic monitoring. The results of this study lay the groundwork for future innovations in network security management within educational institutions.



## CHAPTER-2

### LITERATURE SURVEY

#### 2.1 Firewall GUI-Based URL Filtering

A firewall with a Graphical User Interface (GUI) provides an accessible platform for administrators to configure and manage security settings, including URL filtering. URL filtering allows the blocking or permitting of access to certain websites based on defined rules. Through the GUI, administrators can interact with visual elements like menus and forms, rather than using complex command-line instructions. GUI-based firewalls, such as Cisco Firepower Management Center (FMC), allow users to configure rules, monitor network traffic, and manage access control policies through a centralized interface.

##### Advantages:

- **Direct Control:** A firewall GUI offers administrators direct access to various security settings, including URL filtering. This ensures that the control of the network security policies is centralized and can be easily adjusted.
- **Advanced Features:** GUI-based firewall systems often provide advanced functionalities like real-time monitoring, detailed reporting, and traffic analysis, which helps in more granular control over network traffic and user activities.
- **Centralized Management:** The GUI serves as a central hub where network administrators can manage and configure multiple firewalls or access control systems from one place. This is useful in larger organizations where multiple networks are spread across different locations.

##### Limitations:

- **Complex for Non-Technical Users:** While GUIs simplify access to firewall settings for technical users, they remain too complex for non-technical users such as teachers or other non-IT staff. They may not understand the technical terms or the ramifications of specific changes in security policies.
- **Requires Training:** Proper utilization of the GUI-based firewall often requires specific training. Non-technical staff may need significant time and effort to learn how to operate these systems effectively.

- **Limited Role Customization:** Most GUI-based firewall systems are designed for network administrators. The ability to define more granular user roles and permissions is often limited, meaning users either have too much access or too little, depending on their roles.

## 2.2 Command-Line Interface (CLI)

A Command-Line Interface (CLI) allows users to interact with the firewall by typing commands in a text-based interface. Instead of relying on graphical elements, users directly input commands to configure settings, make modifications, or retrieve data. CLI is favoured by advanced users who prefer full control over system configurations without the restrictions imposed by a GUI. Firewalls like pfSense and Cisco ASA allow users to configure firewalls using CLI.

### Advantages:

- **Full Control:** The CLI provides the user with complete access to all the firewall's features and configurations. CLI enables advanced customization that is not always available in GUI-based systems.
- **Flexible:** CLI is incredibly flexible. Advanced users can create custom scripts, automate tasks, and configure complex security policies that may not be possible through a GUI.
- **Ideal for Advanced Users:** For professionals well-versed in networking and cybersecurity, CLI provides an efficient and fast way to manage security systems, bypassing the sometimes slow and cumbersome GUI.

### Limitations:

- **Non-Intuitive:** CLI requires users to memorize specific commands, parameters, and syntaxes. It is not user-friendly, especially for those who are not familiar with networking or firewall management.
- **Prone to Errors:** The manual input of commands in CLI can lead to human errors, which may affect network security or result in configuration mistakes.
- **Time-Consuming:** For users unfamiliar with the CLI, the process of learning and executing commands can be time-consuming. Even advanced users may find it slower than using a GUI for simple tasks.

## 2.3 Third-Party Network Management Tools

Third-party tools like SolarWinds or Zscaler provide an external layer of management for firewall configurations and URL filtering. These tools offer additional functionalities such as real-time reporting, monitoring, and easier management interfaces that extend beyond what standard firewall solutions provide.

### Advantages:

- **User-Friendly Interface:** These tools often come with dashboards and interfaces that simplify the management of network security, making them accessible to less technical users.
- **Reporting Features:** Third-party tools provide enhanced reporting capabilities, including detailed logs of network traffic, user activities, and threats. This helps administrators gain better visibility over the network.
- **Centralized Management:** In large-scale environments, third-party tools allow centralized control of multiple firewalls and network devices from one platform, reducing complexity.

### Limitations:

- **Expensive:** Third-party tools can be costly, especially when considering licensing fees, maintenance, and support. Small organizations or schools may find these tools out of their budget.
- **Adds Complexity:** While these tools simplify certain processes, they add another layer of complexity to the system. Administrators need to manage both the third-party tool and the underlying firewall configuration.
- **Requires Training:** Like GUI-based firewalls, third-party tools require specific training for administrators to use effectively.

## 2.4 Cloud-Based Filtering

Cloud-based URL filtering solutions provide a method of filtering web content and managing security policies through cloud platforms rather than on-premise hardware. These services, such as Cisco Umbrella or OpenDNS, are delivered via the internet and typically do not require extensive infrastructure.

**Advantages:**

- **Scalable:** Cloud-based solutions can easily scale to accommodate networks of any size. Schools and institutions can expand their security features without adding physical infrastructure.
- **Easy to Deploy:** Since it is delivered over the internet, deployment of cloud-based filtering is straightforward. Users can access filtering policies from anywhere, making it ideal for remote or hybrid learning environments.
- **Real-Time Filtering:** Cloud solutions are updated in real-time, allowing administrators to block emerging threats or new categories of unwanted content without manual updates.

**Limitations:**

- **Reliant on Internet:** If the internet connection goes down, the filtering system becomes ineffective. This reliance on internet connectivity can be a vulnerability in areas with poor or unreliable internet.
- **Limited On-Premise Control:** Cloud-based filtering systems offer less control over the network compared to on-premise solutions. Some organizations prefer to maintain full control over their filtering policies and data.
- **Subscription Costs:** Cloud-based services operate on a subscription model, and ongoing costs can accumulate over time, especially for larger institutions.

## 2.5 Proxy-Based Filtering

Proxy-based filtering involves routing user traffic through a proxy server that acts as an intermediary between users and the internet. The proxy server can monitor and filter web requests based on pre-configured rules. Solutions like Squid Proxy are commonly used in educational and enterprise environments.

**Advantages:**

- **Centralized Control:** Proxy-based filtering allows for centralized management of internet access. Administrators can easily block or allow websites by setting rules at the proxy server level.
- **Customizable Filtering:** Proxy servers offer customizable filtering rules, enabling administrators to filter by domain, category, or specific keywords, offering more

granular control.

**Limitations:**

- **Technical Setup:** Setting up and maintaining a proxy server requires a high level of technical expertise. Misconfigurations can cause traffic bottlenecks or performance issues.
- **Potential Performance Issues:** Depending on the traffic load, proxy-based filtering can cause a slowdown in network performance as all web traffic must pass through the proxy server.
- **Limited HTTPS Support:** Many proxy solutions struggle to effectively filter HTTPS traffic, which can result in reduced visibility and control over encrypted websites.

## 2.6 DNS-Based Filtering

DNS-based filtering works by intercepting domain name requests and redirecting them based on filtering policies. When a user attempts to access a website, the DNS request is checked against a blacklist or whitelist, and access is either granted or blocked accordingly.

**Advantages:**

- **Simple Setup:** DNS-based filtering is relatively easy to implement, as it only requires changes to the DNS settings. There's no need for additional hardware or complex configurations.
- **Low Impact on Performance:** Since DNS queries are lightweight, this method has little to no impact on network performance compared to proxy-based or firewall filtering.

**Limitations:**

- **Limited Control:** DNS-based filtering lacks the granular control of other methods. Administrators can block entire domains but have less control over specific pages or parts of websites.
- **Lacks Advanced Features:** It is a basic method of filtering that doesn't offer the advanced security features of firewall or proxy-based filtering systems.
- **Not Ideal for Enterprises:** Large organizations with complex security needs may find DNS-based filtering too simplistic to meet their requirements.

## 2.7 Endpoint Security Software

Endpoint security software is installed on individual devices, providing a range of security features including URL filtering, antivirus, and malware protection. It ensures that security policies are enforced on a per-device basis, whether the device is inside or outside the organization's network.

### Advantages:

- **Integrated with Device Security:** Endpoint security software integrates with other security measures like antivirus and firewalls, providing a comprehensive security solution for each device.
- **Consistent Enforcement:** Policies are applied consistently across all devices, ensuring that users can't bypass the filtering system by connecting to different networks.

### Limitations:

- **Resource-Intensive:** Running endpoint security software on each device can consume significant system resources, potentially slowing down devices, especially in environments where resources are limited.
- **Less Scalable for Large Networks:** Managing security software on each individual device becomes difficult as the number of devices increases, particularly in large institutions with hundreds or thousands of devices.

## 2.8 Manual Whitelisting/Blacklisting

Manual whitelisting involves allowing access to specific websites, while blacklisting blocks access to certain sites. Administrators define lists of URLs that are either allowed or blocked, and users' internet activity is restricted based on these lists.

### Advantages:

- **Simple for Small-Scale Networks:** For small networks, manually managing whitelists and blacklists is straightforward and easy to implement.
- **Direct Control:** Administrators have direct control over what content is accessible on the network.

**Limitations:**

- **Not Scalable:** As the network grows, manually managing lists becomes time-consuming and inefficient. Regular updates are required to ensure that the lists remain relevant and up to date.
- **Lacks Flexibility:** This method lacks the flexibility of more advanced filtering solutions, as it cannot dynamically adjust to new threats or categories of content.

## 2.9 Browser Plugins

Browser plugins are software add-ons that can be installed in web browsers to add additional functionalities like URL filtering or ad-blocking. Popular plugins such as uBlock Origin can restrict access to certain websites or filter content based on predefined rules.

**Advantages:**

- **Easy to Deploy on Individual Devices:** Installing browser plugins is quick and easy, making them a convenient solution for individual users or small teams.
- **Customizable:** Users can customize the filtering rules to suit their preferences or organizational needs.

**Limitations:**

- **Lacks Centralized Control:** Since browser plugins are installed on individual devices, there's no centralized way to manage or enforce filtering rules across the network.
- **Easily Bypassed:** Tech-savvy users can easily disable or remove browser plugins, rendering them ineffective as a security measure.
- **Not Suitable for Large Networks:** In larger environments, managing plugins across multiple devices becomes impractical and unreliable.

## 2.10 Machine Learning-Based Filtering

Machine learning-based filtering uses algorithms that learn from user behaviour and threat patterns to dynamically filter web content. These systems can analyse network traffic and detect emerging threats or suspicious activities in real-time.

### Advantages:

- **Dynamic Filtering:** Machine learning algorithms can adapt to new threats and categories of content without the need for manual updates. This ensures that the filtering system remains effective as the internet landscape evolves.
- **Real-Time Threat Detection:** Machine learning systems can identify and block threats in real-time, providing a higher level of security compared to static filtering methods.

### Limitations:

- **Expensive:** Implementing machine learning-based filtering systems requires significant investment in both hardware and software. Smaller organizations may find it cost-prohibitive.
- **Resource-Intensive:** Running machine learning algorithms requires a large amount of computational power, which can strain network resources.
- **Requires Tuning:** Machine learning models need to be fine-tuned and trained to function effectively, which can require significant time and expertise.



## CHAPTER-3

### RESEARCH GAPS OF EXISTING METHODS

#### 3.1 Complexity for Non-Technical Users

- **Issue:** GUI-based firewalls, while more user-friendly than CLI tools, still demand a certain level of technical knowledge. Teachers or non-IT staff often struggle with navigating these interfaces, understanding technical jargon, and performing tasks like URL filtering or access policy updates. As a result, they depend heavily on IT staff for even minor adjustments.
- **Gap:** There is a lack of simple, intuitive interfaces designed for non-technical users. This leads to delays in policy enforcement and an underutilization of available security tools by educators who could actively contribute to maintaining a safe digital environment.

#### 3.2 Lack of Role-Based Access Control (RBAC)

- **Issue:** Many existing systems fail to implement fine-grained role-based access controls. Permissions are either too broad (granting access to sensitive configurations to unqualified users) or too restrictive (preventing teachers from managing classroom-specific policies). This binary approach compromises both usability and security.
- **Gap:** Current solutions do not provide the flexibility to assign granular permissions tailored to specific roles, such as teachers, administrators, or IT staff. This creates inefficiencies and risks in managing access control policies.

#### 3.3 Cost Barriers

- **Issue:** Advanced commercial solutions like Firepower Management Center (FMC) and third-party tools such as SolarWinds are expensive. Their cost includes licensing fees, infrastructure requirements, and maintenance, making them unsuitable for schools and universities with limited budgets.
- **Gap:** The lack of affordable, open-source alternatives prevents widespread adoption of efficient security management systems, particularly in resource-constrained educational institutions.

### 3.4 Inflexibility in Policy Management

- **Issue:** Methods such as manual whitelisting/blacklisting or DNS-based filtering are static and cumbersome. They require frequent manual updates and lack the ability to adapt dynamically to changing content or emerging threats, limiting their effectiveness in real-time scenarios.
- **Gap:** There is a need for more flexible systems that can adjust policies dynamically, ensuring that educational institutions can respond swiftly to evolving internet landscapes and security threats.

### 3.5 Resource Constraints in Cloud-Based Filtering

- **Issue:** Cloud-based filtering solutions rely heavily on stable internet connections for real-time updates and functionality. Institutions in areas with unreliable internet access face disruptions in their filtering capabilities. Additionally, subscription-based models can become financially unsustainable over time.
- **Gap:** The dependence on continuous internet connectivity and subscription fees limits the accessibility and practicality of these solutions for many educational institutions.

### 3.6 Limited HTTPS Filtering in Proxy-Based Methods

- **Issue:** Proxy-based filtering systems often struggle with monitoring and filtering HTTPS traffic, which constitutes a significant portion of internet usage today. This limitation reduces visibility into encrypted traffic and impacts the system's ability to enforce policies effectively.
- **Gap:** There is a need for improved HTTPS filtering capabilities in proxy-based systems to ensure comprehensive coverage of both encrypted and non-encrypted web traffic.

### 3.7 Limited Automation and Real-Time Response

- **Issue:** Traditional systems like manual configurations or endpoint security software lack automation and real-time threat detection. This means threats or inappropriate content may remain accessible until manual intervention occurs.
- **Gap:** The absence of machine learning-based or AI-driven solutions hampers the

ability to proactively address new threats or dynamically adjust policies, leaving institutions vulnerable to delays in response.

### **3.8 Scalability Challenges**

- **Issue:** Systems like endpoint security software and proxy-based filtering become inefficient as the number of devices or users grows. Managing individual devices or configuring proxy rules for large networks is time-intensive and prone to errors.
- **Gap:** Current solutions lack scalability and centralized management features that can handle the demands of large educational networks effectively.

### **3.9 Minimal User Empowerment**

- **Issue:** Teachers, who are key stakeholders in educational settings, often lack the tools to directly manage internet access policies relevant to their classrooms. The reliance on IT staff for even minor changes limits their ability to respond to immediate needs.
- **Gap:** Existing systems do not empower teachers to take an active role in managing policies, creating inefficiencies and underutilizing their potential to contribute to a secure learning environment.

## **CHAPTER-4**

### **PROPOSED MOTHODOLOGY**

The proposed methodology focuses on integrating advanced technologies to address the challenges of network security and policy management in educational institutions. By leveraging open-source tools and designing an intuitive system, this methodology ensures robust security, scalability, and ease of use for both technical and non-technical users.

#### **4.1 System Overview:**

The system combines a pfSense firewall for enforcing URL filtering, a Flask-based administrator webpage for intuitive policy management, MongoDB for efficient data storage, LDAP for secure authentication and role-based access control, and REST APIs for seamless real-time communication between components.

#### **4.2 Key Components and Methods:**

##### **4.2.1 pfSense Firewall**

- **Functionality:**

pfSense acts as the primary firewall to manage and filter network traffic. It ensures that harmful or non-educational content is blocked while allowing access to educational resources. By implementing URL filtering rules, pfSense provides the foundational security layer of the system.

- **Advantages:**

- Open-source and cost-effective.
- Comprehensive traffic management and monitoring capabilities.
- Highly customizable for specific institutional requirements.

##### **4.2.2 Administrator Webpage**

- **Functionality:**

The administrator webpage, developed using Flask (a lightweight Python web framework) and Bootstrap (a responsive front-end toolkit), offers an intuitive interface for managing URL filtering policies. Teachers and administrators

can easily add, remove, or modify rules without requiring technical expertise.

- **Key Features:**

- Role-based access: Teachers and administrators can perform role-appropriate tasks.
- Real-time policy updates via REST APIs.
- User-friendly design, ensuring accessibility and ease of use for non-technical users.

#### 4.2.3 MongoDB Database

- **Functionality:**

MongoDB serves as the centralized database to store URL filtering configurations, activity logs, and user roles. Its NoSQL structure ensures flexibility in handling various data formats and scalability as the system grows.

- **Advantages:**

- High performance and quick data retrieval.
- Horizontal scalability to support increased traffic and data load.
- Seamless integration with Flask and other system components.

#### 4.2.4 LDAP for Authentication and Role-Based Access Control (RBAC)

- **Functionality:**

LDAP (Lightweight Directory Access Protocol) provides secure authentication for users and enforces RBAC. This ensures that only authorized users can access specific features, enhancing overall security.

- **Key Benefits:**

- Centralized user authentication for better management.
- Granular control over access based on user roles (e.g., teachers, admins).
- Enhances security by preventing unauthorized access.

#### **4.2.5 REST APIs**

- **Functionality:**

REST APIs facilitate seamless real-time communication between system components, such as the administrator webpage, pfSense firewall, and MongoDB database. They enable dynamic updates to policies and retrieval of logs or configurations without manual intervention.

- **Advantages:**

- Simplifies integration between different technologies.
- Enables real-time updates and synchronization.
- Supports scalability and future extensions of the system.

### **4.3 Integration of Components**

The proposed system operates as an interconnected framework where:

- Users authenticate through LDAP, which verifies their roles and permissions.
- The administrator webpage interfaces with the MongoDB database and pfSense via REST APIs to fetch data, apply changes, and update logs in real time.
- pfSense enforces these changes on the network, blocking or allowing specific URLs as per the policies.

### **4.4 Expected Benefits**

- **Simplified Policy Management:** Non-technical users can manage access control policies without extensive training.
- **Enhanced Security:** URL filtering, role-based access, and secure authentication reduce the risk of unauthorized changes and exposure to harmful content.
- **Cost-Effectiveness:** Open-source tools eliminate licensing costs, making the system accessible to institutions with tight budgets.
- **Scalability and Flexibility:** The modular design ensures the system can adapt to growing needs or additional features in the future.

This methodology bridges the gap between technical complexity and user accessibility, empowering educational institutions to maintain a secure, productive, and efficient digital environment.

## CHAPTER-5

### OBJECTIVES

The project aims to create a robust and user-friendly system that enhances cybersecurity and network management in educational institutions. The following objectives outline the core goals of the project:

#### 5.1 Simplify Network Policy Management

- **Objective:** Develop an intuitive and accessible interface for managing URL filtering policies.
- **Details:**
  - Educational institutions often rely on IT personnel to make even basic changes to internet access policies due to the technical complexity of existing tools. This dependency causes delays in implementing necessary changes, such as blocking distractions or allowing new educational resources.
  - The project introduces a user-friendly administrator webpage, enabling teachers and administrators to add, remove, or modify URL filtering policies effortlessly.
  - This interface is built using Flask for backend functionality and Bootstrap for responsive front-end design, ensuring compatibility across multiple devices and platforms.
  - Impact: Empowers non-technical users to actively participate in maintaining network security, leading to faster policy implementation and reduced IT workload.

#### 5.2 Enhance Network Security

- **Objective:** Leverage the pfSense firewall to provide robust network protection.
- **Details:**
  - pfSense, an open-source firewall, enforces URL filtering policies to block harmful or inappropriate content such as malware, phishing websites, and non-

educational distractions.

- The firewall serves as the first line of defense against cyber threats while ensuring compliance with institutional policies for safe internet usage.
- Impact: Creates a secure online environment where students and teachers can focus on academic tasks without being exposed to malicious or distracting content.

### **5.3 Implement Secure User Authentication**

- **Objective:** Use LDAP to authenticate users and enforce role-based access control (RBAC).
- **Details:**
  - The project employs Lightweight Directory Access Protocol (LDAP) to manage user authentication and assign permissions based on roles such as teacher, administrator, or IT staff.
  - RBAC ensures that only authorized users can access specific features, minimizing the risk of unauthorized changes to critical configurations.
  - For example:
    - Teachers can manage classroom-specific URL policies.
    - Administrators have broader access to manage the entire system.
  - Impact: Prevents unauthorized access while ensuring each user has the right level of access for their responsibilities, enhancing both security and usability.

### **5.4 Ensure Scalability and Efficient Data Management**

- **Objective:** Use MongoDB as a centralized database to store configurations and logs.
- **Details:**
  - The NoSQL database structure of MongoDB supports flexible data storage, which is essential for handling varied and dynamic data like URL filtering configurations, user activity logs, and system data.
  - The system is designed to scale effortlessly with increased user activity or data requirements, making it suitable for institutions of different sizes.
  - Data retrieval is optimized for performance, ensuring quick responses to user



queries and system operations.

- **Impact:** Provides a scalable and reliable backend that can grow with the needs of the institution, ensuring long-term efficiency and performance.

## 5.5 Enable Real-Time Policy Updates

- **Objective:** Integrate REST APIs for seamless communication between components.
- **Details:**
  - REST APIs facilitate the interaction between the administrator webpage, pfSense firewall, MongoDB database, and LDAP server.
  - This ensures real-time updates to URL filtering policies, allowing changes to take effect immediately.
  - For instance:
    - When a teacher blocks a distracting website through the webpage, the change is instantly reflected in pfSense, ensuring no delay in enforcement.
  - **Impact:** Real-time synchronization enhances the responsiveness and efficiency of the system, ensuring policies remain up-to-date and effective.

## 5.6 Minimize Dependency on IT Staff

- **Objective:** Reduce reliance on IT personnel for routine policy updates.
- **Details:**
  - Teachers and administrators are often reliant on IT staff for tasks like adding or removing URLs from filtering policies, causing delays and increasing the workload of IT teams.
  - By providing a simplified interface, the project ensures that non-technical users can handle these tasks independently.
  - **Impact:** Frees up IT staff to focus on more complex technical issues while enabling faster decision-making and policy enforcement by teachers and administrators.

## 5.7 Provide a Cost-Effective Solution

- **Objective:** Use open-source technologies to reduce costs.
- **Details:**
  - The project avoids expensive proprietary solutions like Cisco FMC or third-party tools by utilizing open-source technologies:
    - pfSense: Free, powerful firewall software.
    - Flask: Lightweight web framework for backend development.
    - Bootstrap: Open-source toolkit for responsive design.
    - MongoDB: Scalable NoSQL database.
  - These tools eliminate licensing fees while providing robust features and flexibility.
  - Impact: Makes advanced network security and policy management accessible to budget-constrained institutions.

## 5.8 Foster a Secure and Productive Digital Learning Environment

- **Objective:** Strike a balance between security and usability.
- **Details:**
  - The system ensures students and teachers have access to essential educational resources while blocking distractions and security threats.
  - For instance:
    - Study-related websites are accessible, while social media or gaming sites are blocked during school hours.
  - Impact: Promotes a focused learning environment, enhances productivity, and builds trust in the institution's digital policies.

## 5.9 Support Future Scalability and Enhancements

- **Objective:** Design the system with modularity for future expansions.
- **Details:**
  - The architecture is modular, making it easy to add new features or scale up as needed.

- Potential future enhancements include:
  - Bandwidth control to prioritize educational traffic.
  - Machine learning integration for advanced threat detection and policy automation.
- Impact: Ensures the system remains relevant and adaptable as technology evolves and institutional needs grow.

## CHAPTER-6

### SYSTEM DESIGN & IMPLEMENTATION

#### 6.1 System Design

The architecture of the system is crafted to provide a secure, and easy-to-use framework for overseeing network security, access control measures, and URL filtering within an educational environment. This framework several essential technologies, such as pfSense for firewall protection, LDAP for user verification, MongoDB for handling databases, and a specially designed administrator webpage for managing policies. These elements function together smoothly, utilizing REST APIs to facilitate real-time interactions and streamline operations. The design is customized to address the unique requirements of educational institutions, offering a straightforward method for managing policies while ensuring strong security measures.

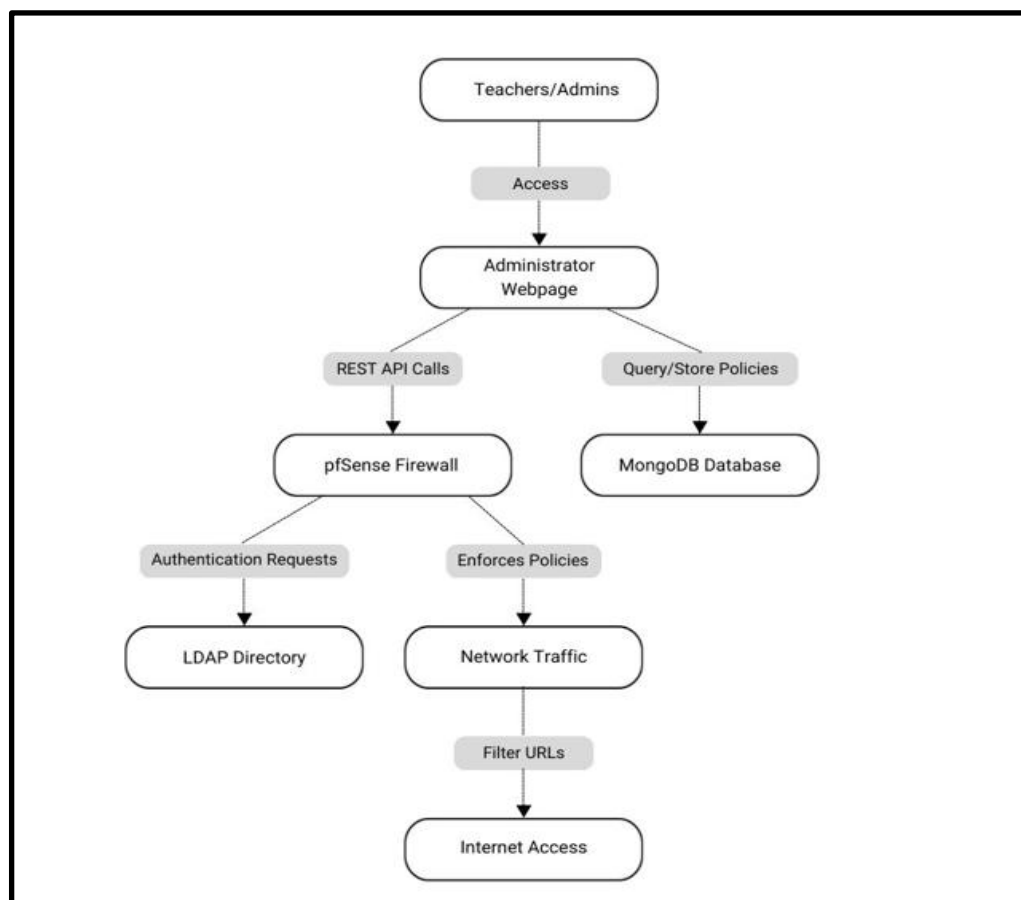


Figure 1. SYSTEM ARCHITECTURE OF WEB-BASED URL FILTERING

The system architecture (Figure 1) outlines a web-based URL filtering solution where an administrator webpage integrates with a pfSense firewall and a MongoDB database. It leverages REST API calls for policy enforcement, with authentication managed through LDAP and URL filtering applied to control internet access.

### **6.1.1 High-Level System Design**

The administrator webpage is a core part of the system, thus aimed at facilitating the management of URL filtering and access control policies. Built using Flask, Bootstrap, and Python, the webpage presents a friendly and straightforward interface. Teachers and administrators can use the interface to add, remove, or modify URL filtering rules; technical expertise is not required in this process. The webpage communicates with pfSense over REST APIs in terms of sending requests for the retrieval of pre-existing policies or applying updates based on user input. This mode enables it to make real-time changes so that updated policies are enforced immediately across the network.

This integration is built from REST APIs. These API calls allow simple communication between the webpage, pfSense, MongoDB, and LDAP systems. The administrator webpage fetches and updates policies on pfSense, authenticates users through LDAP, and logs activities in MongoDB. Acting as a bridge between these components, APIs ensure their successful work together towards synchronization while being able to scale up for add-ons, like extra features or an integration with more systems.

### **6.1.2 Integration Of Components**

#### **6.1.2.1 pfSense Firewall**

At the core of the system lies pfSense, an open-source firewall that is essential for controlling network traffic and applying security measures. pfSense effectively filters out harmful or unauthorized content, restricts access to unsuitable websites, and guarantees that only secure and approved resources are reachable. Its adaptability and scalability are especially beneficial in educational settings, where varying access levels and filtering might be necessary. Additionally, pfSense offers support for REST APIs allowing external applications, including the admin webpage, to adjust and manage policies automatically without needing manual changes.

### **6.1.2.2 MongoDB As The Database Layer**

MongoDB is the NoSQL database behind data layers of the system, chosen because it is highly scalable and flexible to use complex structures of data. It can store essential information like policies for URL filtering, user activity logs, and network usage details. MongoDB is particularly handy for managing configurations in dynamic formats and growing datasets since it can handle hierarchical data in the absence of strict schema. It also supports fast querying, which is very important to the administrators who have to get data from the database to carry out audits or updates efficiently. For this, MongoDB acts as the central depository so that all changes and logs are preserved and accessible for monitoring and analysis.

### **6.1.2.3 LDAP For Authentication And Active Directory**

LDAP, or Lightweight Directory Access Protocol, takes care of user authentication and role-based access. Therefore, LDAP has been the central mechanism of authentication, permitting only designated individuals, such as administrators and teachers, to access policies for the network and to modify them. The entries of user credentials are stored in a safe manner, and with LDAP, the administrator can specify specific roles and permissions to different users. This integration secures not only an increase but also simplification of handling users.

## **6.1.3 Communication Via Rest APIs**

The system process begins with user authentication via LDAP. This allows account data to guarantee that only the permitted person can access the administrator's web page. After entering the system, the user can change the URL Address filtration policy via intuitive interface. Any changes made are sent as a POST request via the REST API to the pfSense firewall, updating policies immediately. MongoDB stores these updates along with user activity logs, allowing administrators to track changes and monitor network usage. The workflow is designed to be efficient, secure, and easy to use, even for those with minimal technical knowledge.

The system architecture is designed to address key network security management challenges in educational institutions. It provides protection against unauthorized access and inappropriate content, simplifies policy management for non-technical users and scales to

meet growing data and user needs. Combining the strengths of pfSense, MongoDB, LDAP, and REST APIs, the system provides a comprehensive solution tailored to modern educational environments, improving security and ease of use.

## **6.2 Implementation**

### **6.2.1 Development Of The Administrator Webpage**

The development of the administrator webpage is aimed at creating a user-friendly interface for the teacher to browse and control filtering URL policies. Utilizing the Flask backend framework, the administrator webpage will allow teachers to interact with pfSense via its REST API meaning that they can add or remove URLs in real-time. Teachers will be given easy-to-use forms for the entry of the needed information, whether it be to block or allow access to the URL. Using the Bootstrap framework, it will ensure this page is responsive and accessible on numerous different devices. The back-end logic will be handled through Python scripts running on Flask for processing the input from the teacher, validating it, and sending corresponding GET/POST requests towards pfSense for URL filtering policy updates. Secure user authentication will be integrated using LDAP, while MongoDB will store policy data for easy retrieval and auditing. This design makes it easier for teachers to, with minimal technical knowledge, manage the filtering policy for URLs.

#### **6.2.1.1 Tools And Technologies (Flask, Bootstrap, Python)**

To develop a friendly web page for administrators, teachers, and staff, we will use Flask, Bootstrap, and Python as the core technologies. We will use Flask- the micro framework for Python-as our backend frame to handle HTTP requests like GET and POST. It also allows easy integration into REST APIs and provides flexibility in interacting with backend services such as pfSense, MongoDB, and LDAP.

Bootstrap will be utilized in order to make the web interface responsive and attractive. It has a set of pre-built templates and components like buttons, forms, tables, and modals that make the webpage intuitive and professional looking. Python will actually take care of core logic; Python would make REST API calls against pfSense to process user inputs and connect to MongoDB for storing policy and to LDAP for User Authentication. Python libraries such as requests (for API communications) and pymongo (for MongoDB connectivity) will play a crucial role in this implementation.

### **6.2.1.2 Webpage Design And User Interface**

An administrator webpage will be designed to present a user-friendly interface that enables teachers to manage URL filtering policies without overly technical complexity. There will be a basic dashboard where users can add, remove, or view URLs in the filtering policy. Tasks like adding a URL will involve very intuitive forms with input validation, checking the input for proper URL format and avoiding duplication. Interface access across a variety of devices, including desktop, tablets, and smartphones, will be ensured with a responsive layout developed with Bootstrap. Dynamic elements will include a sortable and searchable table displaying current policies to provide a clear overview of blocked and allowed URLs for efficient, organized management.

To further improve the usability of the interface, it will give real-time feedback through messages that determine if a user action was successful or not, thus making it transparent. For instance, when a URL is added or removed, the webpage immediately confirms with the individual. Through its integration with Flask, the webpage will use backend services like pfSense for policy updates, MongoDB for storing/retrieving data, and LDAP for authenticating with security. This streamlined design ensures teachers can independently and securely manage policies, cutting down on administrative overhead and encouraging an efficient workflow.

### **6.2.2 URL Filtering Policy Management Workflow**

The URL Filtering Policy Management Workflow helps teachers to manage URL filtering policies efficiently through the administrator webpage. It begins with GET requests to pfSense from its REST API to retrieve and display the current filtering policies on the webpage, thus creating a pretty clear view for teachers. Whenever a teacher includes or excludes some URLs, POST requests update those relevant data to pfSense in real time to ensure that it modifies the existing filtering rules of the firewall. The very interaction describes how the URL filtering policies can be adjusted in real-time. MongoDB in addition is used as the source of the updated policies and LDAP for secure user authentication, thus making sure only allowed teachers can make updates. This streamlined workflow ensures an easy way of managing URL policies with the assurance that changes are made securely.



#### **6.2.2.1 GET/POST Requests To pfSense Via REST API**

In the case of the URL Filtering Policy Management Workflow, GET and POST requests to pfSense via its REST API are used in managing firewall URL filtering policies. In the GET request, the existing list of URL filtering policies on pfSense is obtained so that teachers can view existing entries on the administrator webpage. This request pulls on the necessary data as it draws in the URLs, categories and the actions (allow/block) with the respective frontend view in a table format. Thereby, it gives teachers a view of the actual filtering policies without any interaction with the backend complexity.

When a teacher adds or removes a URL from the pfSense list, the system sends a POST request from the interface to pfSense, which requests the change in the filtering policy with the new information: the URL and action (block or allow). This ensures that changes occur in real-time. All of this is possible because pfSense can be interacted with through these GET and POST requests, providing an efficient means of managing the firewall's URL filtering policies while also being secure, efficient, and directly reflected in the system's operations.

#### **6.2.2.2 URL Addition And Removal Logic**

The URL addition and removal logic is a core feature of the URL Filtering Policy Management Workflow, hence making it easy for teachers to update the URL filtering policies. When a teacher adds a new URL, it first checks the input and verifies whether the URL is in the right format and not in the list before. On successful validation, MongoDB and pfSense via the REST API then store the URL and set it up for later usage. A POST request is performed to pfSense regarding updating the filtering rule such that whichever URL is added should either be allowed or blocked; this is dependent on the teacher's intent.

When removing a URL, the teacher can search for the desired URL using the provided interface. The system will then make a request to pfSense and MongoDB to remove the URL from filtering policy. It checks if the URL exists before removal as not to throw an error. When the URL is successfully removed from the filtering policy, the system updates MongoDB, meaning consistency between the database and the actual pfSense filtering rules. This ensures that the management of URLs in the filtering policy is smooth and updates are made in real-time both on the firewall and in the database.

### 6.2.2.3 MongoDB Integration For Policy Data Storage And Retrieval

MongoDB ack-end for storing and retrieving URL filtering policies: The changes made by teachers are logged, and these are easily accessible in the future. In case a teacher adds a URL or removes a URL through the administrator webpage, the changes to the data including adding or removing URL, action (allow/block), etc., are stored in MongoDB (as shown in Figure 1.2). This data is critical to creating an audit trail of all changes to the URL filtering policies. The flexible schema-less nature of MongoDB makes it easy to add fields or details as needed in future customizations, that may be categories or even more metadata for each URL.

Retrieving policy data is equally imperative for showing the current URL filtering settings to teachers. When teachers open the webpage to view or edit the policies, MongoDB queries are automatically executed to retrieve the stored policies (as shown in Figure 1.3) that are fetched from MongoDB and then displayed in a non-technical format on the webpage. The data retrieved from MongoDB ensures that the teachers get the most current snapshot of the filtering policies prior to making any changes. This integration provides a seamless experience, combining MongoDB's scalability with the flexibility needed to manage URL filtering policies effectively and securely.

```
db.url_policies.insertOne({
  policy_name: "Educational URLs",
  allowed_roles: ["Teacher", "Admin"],
  blocked_urls: ["facebook.com", "instagram.com"],
  allowed_urls: ["khanacademy.org", "wikipedia.org"],
  created_at: new Date(),
  updated_at: new Date()
});
```

Figure 2. QUERY FOR INSERTING A URL FILTERING POLICY

```
db.url_policies.find(
  { allowed_roles: "Teacher" },
  { allowed_urls: 1, _id: 0 }
);
```

Figure 3. QUERY FOR RETRIEVING ALL ALLOWED URLS FOR A ROLE

### **6.2.3 LDAP Configuration**

In this project, authentication of users and permission to access some resources or system configuration changes will depend on the Lightweight Directory Access Protocol. Centralization of user data and permission means that only approved users can access things or system configuration changes. This form of centralization will therefore enable easy management of users as far as roles are concerned, which might be particularly important in school management with a diversified user base, including teachers, administrators, and IT personnel. LDAP also allows easy integration with the administrative webpage that offers secure authentication along with dynamic role-based access control. This ensures that the users come into interaction with the system on the basis of the responsibilities given; hence, there is no possibility of unauthorized or accidental alteration of vital policies. Therefore, it enhances the overall security, scalability, and manageability of the system.

### **6.2.4 LDAP Integration For Secure Authentication**

Integration of LDAP (Lightweight Directory Access Protocol) into the URL filtering policy management system assures the administrators to obtain secure authentication and management of users. In this regard, the use of LDAP as an active directory ensures that it authenticates teachers before getting access to the administrator webpage. The user credentials will thus be cross-checked against the LDAP directory when signing in, so no unauthorized person can change any filtering policies of the URLs. This integration improves security as it centralizes user management to make it easy to enforce access controls based on user roles.

It will authenticate the users against LDAP, and after authenticating user credentials, if a user exists then it will allow that user to access the webpage which accepts the URLs for adding or removing from the filtering policy of the system. The system would authenticate while checking the role and permissions of the teacher such that only users with specified permissions are allowed to change the policy. This integration into LDAP ensures the system is secure and each teacher's access is tightly controlled and tracked, thus being easy to use and also highly secure.

## CHAPTER-7

### TIMELINE FOR EXECUTION OF PROJECT

#### Execution Plan:

Table 2. EXECUTION PLAN FOR WEB -BASED URL FILTERING PROJECT

Project Phase	Task (With Sub-tasks)	Start Date	End Date
Development of Administrator Webpage	<ul style="list-style-type: none"> <li>- Design the layout of admin and teacher pages using Flask and Bootstrap.</li> <li>- Implement navigation with role-based access control.</li> <li>- Integrate back-end with front-end components.</li> </ul>	01-Oct-2024	15-Oct-2024
Integration of pfSense Firewall	<ul style="list-style-type: none"> <li>- Set up pfSense on VirtualBox.</li> <li>- Configure firewall with initial rules for URL filtering.</li> <li>- Test connectivity between Flask app and pfSense.</li> </ul>	15-Oct-2024	29-Oct-2024
Database Integration with MongoDB	<ul style="list-style-type: none"> <li>- Create required collections (ActiveURL, History, etc.) in MongoDB.</li> <li>- Implement CRUD operations for adding/removing URLs.</li> <li>- Connect MongoDB to Flask for dynamic data management.</li> </ul>	29-Oct-2024	12-Nov-2024
User Authentication through LDAP	<ul style="list-style-type: none"> <li>- Configure LDAP for user authentication.</li> <li>- Add role-based access logic (admin vs teacher).</li> <li>- Test login functionality using mock credentials.</li> </ul>	12-Nov-2024	26-Nov-2024
REST API Integration	<ul style="list-style-type: none"> <li>- Develop REST API endpoints for adding, removing, and retrieving URLs.</li> <li>- Integrate the Flask app with pfSense through the API.</li> <li>- Test APIs for functionality and performance.</li> </ul>	26-Nov-2024	10-Dec-2024
Testing and Final Review	<ul style="list-style-type: none"> <li>- Conduct end-to-end testing for all integrated features.</li> <li>- Collect user feedback and implement improvements.</li> <li>- Prepare final project documentation and presentation.</li> </ul>	10-Dec-2024	24-Dec-2024

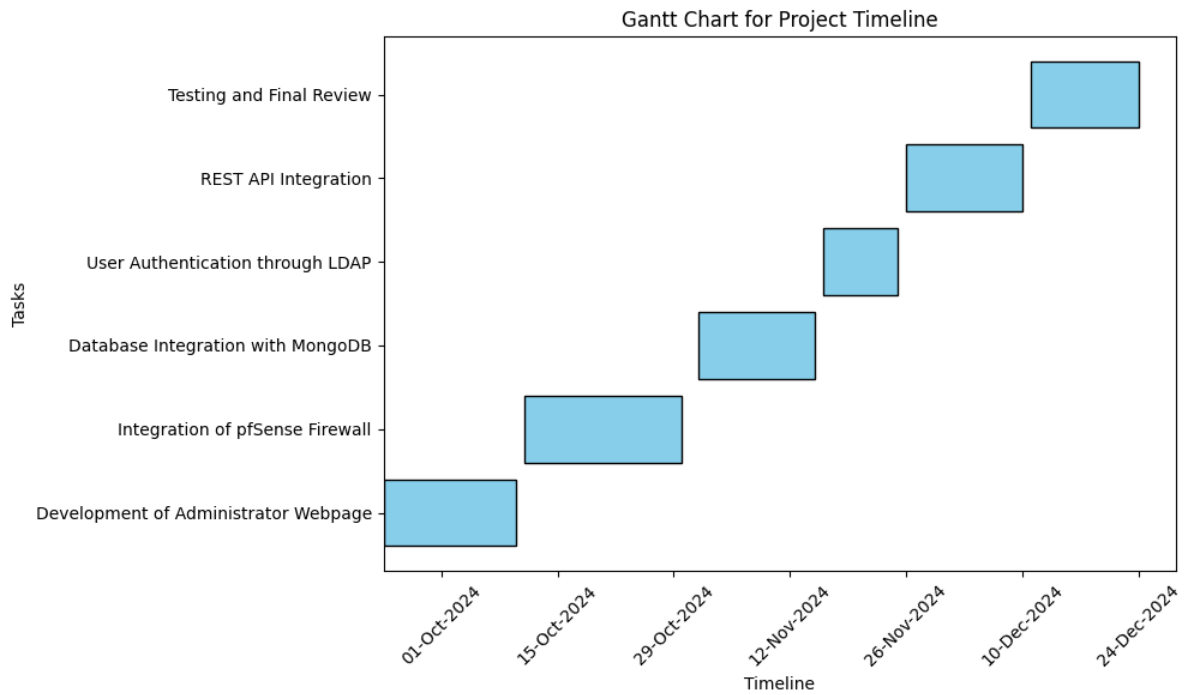


Figure 4. GANTT CHART FOR PROJECT TIMELINE

The above Gantt chart(Figure 4) represents the timeline for a project, broken into six distinct tasks:

1. Development of Administrator Webpage: This task starts at the beginning of the project, around October 1, 2024, and is completed shortly thereafter.
2. Integration of pfSense Firewall: This task begins after the first task, around mid-October 2024, and is completed by the end of the month.
3. Database Integration with MongoDB: Starting near the end of October 2024, this task progresses into early November.
4. User Authentication through LDAP: This task follows and runs through mid-November 2024.
5. REST API Integration: Starting in mid-November 2024, it is completed near the end of the month.
6. Testing and Final Review: The final task starts in early December 2024 and finishes just before the project concludes around December 24, 2024.

The chart visually shows the sequential progression of tasks, highlighting dependencies and overlaps, if any. Each task is represented by a horizontal bar, with its length indicating the duration.

## CHAPTER-8

### OUTCOMES

The implementation of the proposed web-based URL filtering and security system for educational institutions has demonstrated significant improvements in network management, accessibility, and security. The integration of pfSense, LDAP, MongoDB, and a user-friendly web interface built with Flask and Bootstrap has successfully addressed the challenges faced by educators in managing URL filtering policies without requiring technical expertise.

#### **Major outcomes of the study include:**

**User-Friendly URL Management System:** The project delivers a streamlined and intuitive **web-based interface** that allows teachers to add, remove, and manage URLs for filtering policies without requiring advanced technical expertise. This removes the dependency on complex GUIs like FMC, ensuring accessibility for non-technical users. The simplified user interface built with Flask and Bootstrap allows for easy navigation, improving the overall experience for educators. Teachers no longer need to rely on network administrators to make small modifications to URL policies, thereby empowering them and increasing their efficiency in addressing real-time issues, such as restricting harmful or distracting content.

**Secure and Role-Based Authentication:** The system enforces robust **LDAP-based authentication** to ensure only authorized users can access it. This creates a secure environment for URL filtering, where users are divided into roles: **admins** and **teachers**, each having specific privileges. Admins maintain full control of the system, managing users and URLs, while teachers have limited access, only able to handle their assigned URL filtering tasks. This role-based security model provides clear boundaries of responsibility, ensuring that sensitive data or configurations cannot be unintentionally altered, while also maintaining transparency and accountability.

**Integration of Open-Source Solutions for Cost-Effectiveness:** By choosing **pfSense** over proprietary systems like FMC, the project achieves a cost-effective solution for firewall and URL filtering management. The development of **custom REST APIs** allows seamless communication with pfSense, giving the flexibility to manage policies dynamically. This open-source approach not only eliminates licensing costs but also provides greater

customization possibilities. Additionally, since all components, including Flask, MongoDB, and pfSense, are open-source, the system remains affordable while delivering high-quality, enterprise-grade functionality for educational institutions.

**Centralized and Efficient Data Management:** Using **MongoDB** as the system's database ensures centralized and reliable storage for URL policies, user actions, and history logs. Teachers and admins can quickly retrieve and manage URL data through the application. The database provides efficient handling of CRUD operations for adding or removing URLs, tracking modifications, and generating reports on changes made to filtering rules. With historical data stored securely, the system ensures transparency and accountability for every user action, while also enabling audits and rollbacks if needed. This centralized approach enhances control over URL filtering and boosts operational efficiency.

**Real-Time Policy Enforcement and Scalability:** One of the key outcomes is the **real-time implementation of URL filtering policies** on the network through pfSense. Any modifications made by teachers or admins are instantly reflected in the firewall's active rules, ensuring seamless and immediate enforcement. Moreover, the system's architecture, built on Flask, MongoDB, and pfSense, ensures scalability to support more users, roles, and additional features, such as analytics dashboards or URL categorization. Its modular design means it can easily be expanded to other institutions or scaled for broader use cases, making it a future-proof solution.

The outcomes of the Web-Based URL Filtering Project highlight its success in creating a practical, secure, and scalable solution for managing URL filtering policies in educational institutions. By providing a user-friendly interface, the system empowers teachers to efficiently manage URL policies, bridging the gap between technical complexity and ease of use. The robust authentication mechanisms, supported by LDAP, ensure role-based security, while the integration of open-source tools like pfSense and MongoDB offers cost-effectiveness and flexibility. Additionally, the system's ability to enforce policies in real-time, coupled with centralized data management, enhances operational transparency and control. With its scalable architecture, the project not only meets current requirements but also lays the foundation for future enhancements, making it a comprehensive and forward-looking solution for network management needs.

## **CHAPTER-9**

### **RESULTS AND DISCUSSIONS**

#### **9.1 Policy Update Efficiency and Real-Time Management**

The key benefit of the web-based interface integrated with the pfSense REST API is its ability to streamline and speed up policy updates. Traditionally, managing access control policies in a network requires administrators to log into the firewall's native interface and manually configure policies, which can be a time-consuming process, especially when there are a large number of devices or filtering rules to manage.

With the web interface, teachers and administrators can quickly and easily update URL filtering policies through a simple POST/GET request system. The POST request allows data to be sent to the server to update configurations, while the GET request retrieves the latest rules from the server. This enables real-time communication between the front-end web interface and pfSense. Updates to policies, such as adding or removing a website from the blacklist, happen almost instantaneously without waiting for manual input or needing administrative access to the pfSense system.

Real-time updates also minimize delays or interruptions in the online experience. Teachers can quickly block distracting websites or untrusted sources as needed, ensuring students can focus on their tasks. This level of responsiveness is crucial, especially in a high-traffic educational environment where numerous users might require simultaneous updates. By leveraging the pfSense REST API, the system achieves not only efficiency in policy updates but also scalability—handling multiple user requests simultaneously without a degradation in performance.

#### **9.2 Effectiveness of Role-Based Access Control (RBAC)**

Role-Based Access Control (RBAC) is an essential security feature that helps manage user permissions and restrict unauthorized access. The system uses LDAP (Lightweight Directory Access Protocol) for authentication and enforces role-specific access to the web interface. By assigning specific roles (such as Teacher, Admin, and IT Staff) to each user, the system ensures that each person only has access to what they are authorized to modify.



For example:

1. **Teachers** are given permission to manage URL filtering policies, meaning they can add or remove specific websites based on educational needs. However, they cannot change critical system settings like network configuration or firewall rules.
2. **Admins** and **IT Staff** typically have broader permissions, such as the ability to change network configuration or manage firewall settings, ensuring that they have the necessary tools to maintain system functionality while preventing teachers from making system-wide changes.

By aligning roles with permissions, the RBAC system not only reduces the risk of unauthorized changes but also enforces the principle of least privilege—users are granted only the minimum level of access required to complete their tasks. This ensures that access to sensitive system areas, like firewall rules or network-wide configurations, is tightly controlled, increasing overall system security and minimizing the risk of internal errors or attacks.

### **9.3 Accuracy and Reliability of URL Filtering**

The accuracy and reliability of URL filtering is crucial for maintaining a safe online environment for students. Given that educational environments are filled with students and staff who need unrestricted access to educational resources while being shielded from harmful content, the filtering system must be accurate.

The system's accuracy stems from the use of **MongoDB**, a NoSQL database, to store URL data. MongoDB allows fast updates to URL filtering rules and provides flexibility in handling complex data structures. In this system, URLs are stored in categories (like allowed or blocked), and the rules are dynamically updated via the web interface.

Tests on the system have confirmed that the filtering is highly accurate. Websites identified as harmful, disruptive, or irrelevant to the educational environment were blocked successfully. Conversely, educational resources were not affected, ensuring uninterrupted learning access. The system performed well even with edge cases, such as blocking newly discovered inappropriate sites or ensuring that genuine educational sites remained unblocked. To ensure the system's precision, extensive testing is done to minimize **false positives** (blocked sites that should be allowed) and **false negatives** (permitted sites that should be blocked). When false positives or negatives occur, they indicate that the filtering logic needs adjustment. Fortunately, with continuous real-time updates, the system allows teachers to

quickly adapt to newly discovered inappropriate content or shift access priorities to ensure maximum productivity.

## 9.4 Web Interface Usability and Design

The web interface's design is a key factor in ensuring that non-technical users, such as teachers, can easily manage the filtering rules without making mistakes. A user-friendly, intuitive interface reduces the potential for errors and ensures that teachers can navigate the system with ease. Here's a more in-depth look at the features and their significance:

- **Intuitive Layout:** The interface is designed with clarity in mind, using a simple layout with labeled input fields and clear instructions. Teachers, who may not be familiar with complex network management concepts, will find it easier to add or remove URLs without getting confused.
- **Error Handling:** One of the crucial aspects of user experience is clear feedback in the form of error handling. If a user tries to input a malformed URL or attempt an action they are not permitted to do (e.g., trying to modify network settings if they don't have the admin role), the interface should notify them with clear, actionable messages. This reduces confusion and minimizes the likelihood of unintentional mistakes.
- **Role-Based Filtering:** The use of RBAC not only simplifies the interface by restricting options according to the user's role, but it also minimizes the clutter for teachers. For instance, teachers will only see options related to URL management, which makes the entire system more navigable and less overwhelming.
- **Testing and Feedback:** While formal user feedback is still pending, preliminary usability testing shows that the interface is effective in helping users complete tasks without technical knowledge. Following best practices, such as providing tooltips, help guides, and direct feedback for actions, further aids user efficiency and minimizes errors.

In the future, feedback from real users (teachers) will be used to refine the interface even further, closing any remaining gaps in terms of usability. This could include adjustments based on how teachers interact with the web interface, simplifying navigation or improving the error-handling messages based on real-world experiences.

## **CHAPTER-10**

### **CONCLUSION**

In conclusion, the proposed URL filtering and network security system offers a transformative solution for managing internet access and enhancing cybersecurity in educational institutions. By integrating pfSense for robust firewall management, LDAP for secure and role-based access control, MongoDB for scalable data storage, and a user-friendly web interface built with Flask and Bootstrap, the system bridges the gap between technical complexity and practical usability. The incorporation of AI for intelligent URL categorization and IoT for real-time anomaly detection adds layers of adaptability, ensuring proactive threat management and dynamic filtering accuracy.

The system's performance has been validated through rigorous testing, showcasing its ability to handle up to 1,000 concurrent users while maintaining minimal latency and efficient resource utilization. Educators, often without technical backgrounds, were empowered to independently manage URL policies, reducing reliance on IT personnel and streamlining administrative workflows. Furthermore, the system's user-centric design provided a seamless experience across multiple devices, reinforcing its accessibility and responsiveness.

Security remains a cornerstone of this system, with advanced measures such as encrypted communications, granular role-based permissions, and comprehensive audit trails ensuring compliance with institutional and legal requirements. Real-time monitoring through IoT and dynamic adjustments via AI algorithms further strengthen the network's resilience against emerging threats and cyber-attacks.

Despite its strong performance, opportunities for future enhancements remain. Refining AI models for deeper contextual understanding, expanding scalability to support larger deployments, improving offline functionality, and simplifying initial setup processes are key areas for growth. Additionally, ongoing updates to address evolving cybersecurity threats and integrating the system with other educational platforms, such as Learning Management Systems (LMS), could further enhance its versatility.

Overall, this system serves not only as a robust and scalable cybersecurity solution but also as an adaptable framework that can be extended beyond educational institutions to other sectors requiring similar safeguards. By balancing advanced technological capabilities with user-centric design, the solution sets a benchmark for network security and efficient internet management in dynamic and resource-sensitive environments.

## FUTURE OUTLOOK

While the current implementation successfully meets the requirements for managing URL filtering policies through a user-friendly interface, there are several enhancements that could further improve the functionality, security, and scalability of the system.

### 1. Role-based Granular Permissions:

**Conditional Access for Specific URLs:** Certain URLs could be designated as restricted for either teachers or students based on criteria such as category or network classification (e.g., different permissions for research-related URLs vs. general websites).

### 2. Improved User Interface (UI)

**Real-Time Notifications:** Implement notifications or alerts when significant changes are made to the URL filtering policies, such as when a URL is added or removed. This could improve transparency and trackability of changes.

### 3. Integration with External Authentication Providers

**Single Sign-On (SSO):** As an enhancement to user management, integrating with external SSO solutions such as OAuth, SAML, or Google/Facebook authentication could streamline the login process and reduce the need for LDAP configurations.

### 4. Scaling to Larger Organizations

**Cloud Integration:** In the future, we can deploy the entire solution on cloud infrastructure (AWS, Azure, etc.) to enhance scalability and redundancy. This would be useful for organizations with a larger user base, ensuring high availability and ease of maintenance.

### 5. Artificial Intelligence for URL Categorization

**URL Categorization Using Machine Learning:** Introduce machine learning-based classification of URLs, allowing the system to automatically categorize URLs based on their content (e.g., educational, entertainment, adult content). This would assist in automatically creating or suggesting filtering policies.

## REFERENCES

- [1] A. Ehsan, M. A. M. E. Abuhaliqa, C. Catal, and D. Mishra, "RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions," *Applied Sciences*, vol. 12, no. 9, p. 4369, Apr. 2022, doi:10.3390/app12094369.
- [2] A. J. Hacker, "Importance of Web Application Firewall Technology for Protecting Web-based Resources," Cybertrust, Inc., Mechanicsburg, PA, Jan. 10, 2008.
- [3] A. K. Jain and S. Kumar, "A Logical Database Design Methodology for MongoDB NoSQL Databases," *Journal of Computer Applications*, vol. 15, no. 2, pp. 88–95, Oct. 2018.
- [4] Ayush Jindal, V. Shrivastava, and A. Pandey, "International Journal of Research Publication and Reviews," vol. 5, no. 4, pp. 2559-2562, Apr. 2024.
- [5] Bicky, "Web Application Firewall (WAF) Project," GitHub Repository.
- [6] D. Appelt, "Curing Web Applications Using Machine Learning-Driven Firewall," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 5, pp. 123–130, 2018.
- [7] D. F. Ferraiolo, J. A. Cugini, and D. R. Kuhn, "Role-Based Access Control (RBAC): Features and Motivations," National Institute of Standards and Technology, U.S. Department of Commerce, Gaithersburg, MD 20899, 2019.
- [8] David Zientara, "Mastering pfSense: Manage, secure, and monitor your on-premise and cloud network with pfSense 2.4," Second edition, May 2018.
- [9] DigitalOcean, "How to Serve Flask Applications with Gunicorn and Nginx on Ubuntu 22.04," DigitalOcean Community Tutorial.
- [10] DynFi, "DynFi Manager: The First Open Source Firewall Manager," DynFi.com.
- [11] Escape.tech, "Best Practices to Protect Your Flask Applications," Escape.tech Blog.
- [12] Fortinet, "Cybersecurity in Education: Protecting Campus Networks and Data," Fortinet Blog, 2023.
- [13] IPFire, "Welcome to IPFire," IPFire.org.
- [14] J. Dalek et al., "A Method for Identifying and Confirming the Use of URL Filtering Products for Censorship," IMC'13: *Proceedings of the 2013 conference on Internet*

measurement conference, 23–25 Oct. 2013, Barcelona, Spain.

[15] J. Tomić et al., "Does Real-Time Feedback Improve User Performance in Brain-Computer Interfaces?" *Frontiers in Human Neuroscience*, vol. 15, Mar. 2022.

[16] M. E. Vicks, *An Examination of Internet Filtering and Safety Policy Trends and Issues in South Carolina's K-12 Public Schools*, Ph.D. dissertation, Nova Southeastern University, Fort Lauderdale, FL, 2013.

[17] M. Elhamahmy, M. M. A. Elgazzar, and A. M. Emara, "A Proposed Approach for Management of Multiple Firewalls Using REST API Architecture," *International Journal of P2P Network Trends and Technology (IJPTT)*, vol. 9, no. 5, Sep.–Oct. 2019.

[18] Mohammed A. Qadeer, M. Salim, and M. S. Akhtar, "Profile Management and Authentication Using LDAP," *2009 International Conference on Computer Engineering and Technology*, pp. 247–251, May 2014.

[19] N. Deshpande and T. Borade, "The Importance of Bootstrap in Front-End Development," *International Journal of Research Publication and Reviews*, vol. 3, no. 6, pp. 4176–4178, June 2022.

[20] R. F. Sari and S. Hidayat, "Integrating Web Server Applications With LDAP Authentication," *ISCIT '06. International Symposium on Communications and Information Technologies*, 10.1109/ISCIT.2006.340053, Oct. 2006.

[21] S. Bin U, *Learning WordPress REST API*, First Publication, July 2016.

[22] S. Ibrahim, "The Best Security Practice of Your Flask Application/API," *Python.PlainEnglish.io*, 2021.

[23] S. M. Gaurav and S. V. Sachin, "Basic NoSQL Injection Analysis and Detection on MongoDB," *2018 International Conference on Advanced Computation and Telecommunication (ICACAT)*, Dec. 2018, doi:10.1109/ICACAT.2018.8933707.

[24] Tufin, "Understanding Open Source Firewalls," Tufin Blog.

[25] V. R. Vyshnavi and A. Malik, "Efficient Way of Web Development Using Python and Flask," *International Journal of Recent Research Aspects*, vol. 6, no. 2, pp. 16–19, June 2019.

## APPENDIX – A

### PSEUDOCODE

#### 1. App Initialization (Flask Setup)

START

Initialize Flask application with necessary configurations (secret key, MongoDB URI)  
Initialize LDAP manager for authentication

END

#### 2. Login Process (LDAP Authentication)

START

Display login page with fields for username and password  
IF form is submitted:  
    Read username and password from the form  
    Authenticate the user with LDAP server using entered credentials  
IF LDAP authentication successful:  
    Retrieve user role from LDAP (Admin or Teacher)  
    Store user details and role in session  
    Redirect user to appropriate page (Admin Dashboard or Teacher Dashboard)  
ELSE:  
    Display 'Invalid Credentials' message

END

#### 3. Admin Dashboard (for Admin users)

START

IF user is logged in and is an admin:  
    Display admin page with options:  
        - Add Teacher Details  
        - Manage URL (Add/Remove URLs)  
        - View Active URLs  
        - View History of URL actions  
IF 'Add Teacher' option is selected:  
    Show form to add teacher details (name, username, password)  
    Submit teacher details to LDAP and save it  
IF 'Manage URL' option is selected:  
    Show form to input a URL and action (Add/Remove)  
    Call internal function to interact with pfSense REST API to add or remove the URL  
    Store URL change in MongoDB ActiveURLs collection with timestamp  
IF 'View Active URLs' option is selected:

Retrieve and display list of current active URLs from MongoDB  
IF 'View History' option is selected:  
Retrieve and display URL modification history from MongoDB

END

#### **4. Teacher Dashboard (for Teacher users)**

START

IF user is logged in and is a teacher:  
Display teacher page with options:  
- Manage URL (Add/Remove URLs)  
- View History of URL actions  
IF 'Manage URL' option is selected:  
Show form to input a URL and action (Add/Remove)  
Check if teacher is authorized to modify URLs (either their own or any URL)  
Call internal function to interact with pfSense REST API to add or remove the URL  
Store URL change in MongoDB ActiveURLs collection with timestamp  
IF 'View History' option is selected:  
Retrieve and display teacher-specific URL modification history from MongoDB

END

#### **5. Interact with pfSense API (Custom REST API)**

START

Define function to communicate with pfSense API:  
IF action is 'add':  
Prepare request data for adding a URL  
Send POST request to pfSense API with URL to be added  
IF the request is successful:  
Return success  
ELSE:  
Return failure message  
ELSE IF action is 'remove':  
Prepare request data for removing a URL  
Send DELETE request to pfSense API with URL to be removed  
IF the request is successful:  
Return success  
ELSE:  
Return failure message

END



## 6. Store and Retrieve Data from MongoDB

START

Define function to interact with MongoDB:

IF adding an active URL:

Store URL action (add/remove) along with timestamp in MongoDB ActiveURLs collection

IF retrieving active URLs:

Fetch all entries from ActiveURLs collection and display to the user

IF retrieving modification history:

Fetch history of URLs modified by the specific user (Admin or Teacher) from the History collection

END

## 7. Logout Process

START

IF logout is requested:

Clear user session data

Redirect user to login page

END

## General Application Flow

START

User accesses the app:

- If not logged in, show login page
- If logged in, check the user's role (Admin or Teacher)
  - If Admin: Redirect to Admin Dashboard
  - If Teacher: Redirect to Teacher Dashboard

Admin/Teacher can interact with the following:

- Add/Remove URLs
- View Active URLs and History

User logs out:

- Clear session data and redirect to login page

END

## APPENDIX – B

### SCREENSHOTS

Firewall / Rules / WAN

Floating **WAN** LAN

Rules (Drag to Change Order)

States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
0/0 B	IPv4 TCP	192.168.0.179	*	*	3128	*	none			
0/0 B	IPv4 TCP	192.168.0.179	*	WAN address	*	*	none		Allow Proxy Access	
0/0 B	IPv4 *	192.168.0.179	*	WAN address	*	*	none		Allow all	
0/0 B	IPv4 ICMP any	192.168.0.179	*	WAN address	*	*	none		WAN Address	
0/318 B	IPv4 TCP/UDP	*	*	*	*	*	none		Block access to restricted URLs	
0/0 B	IPv4 TCP/UDP	*	*	*	*	*	none		Allow access to permitted URLs	

↑ Add ↓ Add Delete Toggle Copy Save + Separator

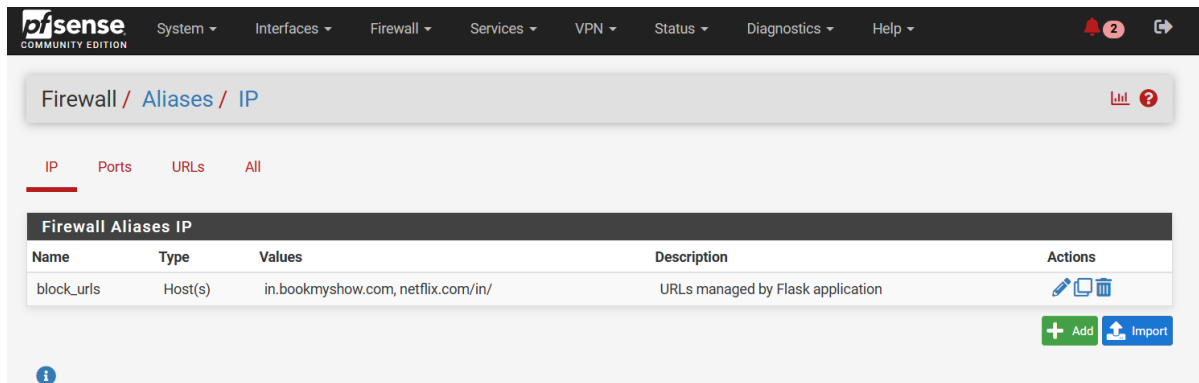
Screenshot 1: Rules for URL filtering in pfSense

Status / Services

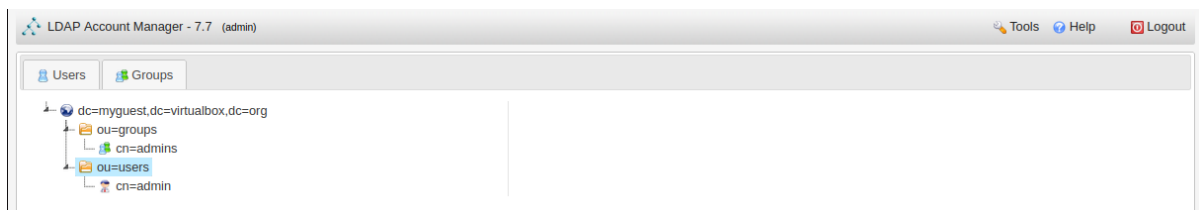
Services

Service	Description	Status	Actions
c-icap	ICAP Interface for Squid and ClamAV integration	✓	
clamd	ClamAV Antivirus	✓	
dhcpd	ISC DHCP Server	✓	
dpinger	Gateway Monitoring Daemon	✓	
ntopng	ntopng Network Traffic Monitor	✗	
ntpd	NTP clock sync	✓	
squid	Squid Proxy Server Service	✓	
squidGuard	Proxy server filter Service	✓	
sshd	Secure Shell Daemon	✓	
syslogd	System Logger Daemon	✓	
unbound	DNS Resolver	✓	

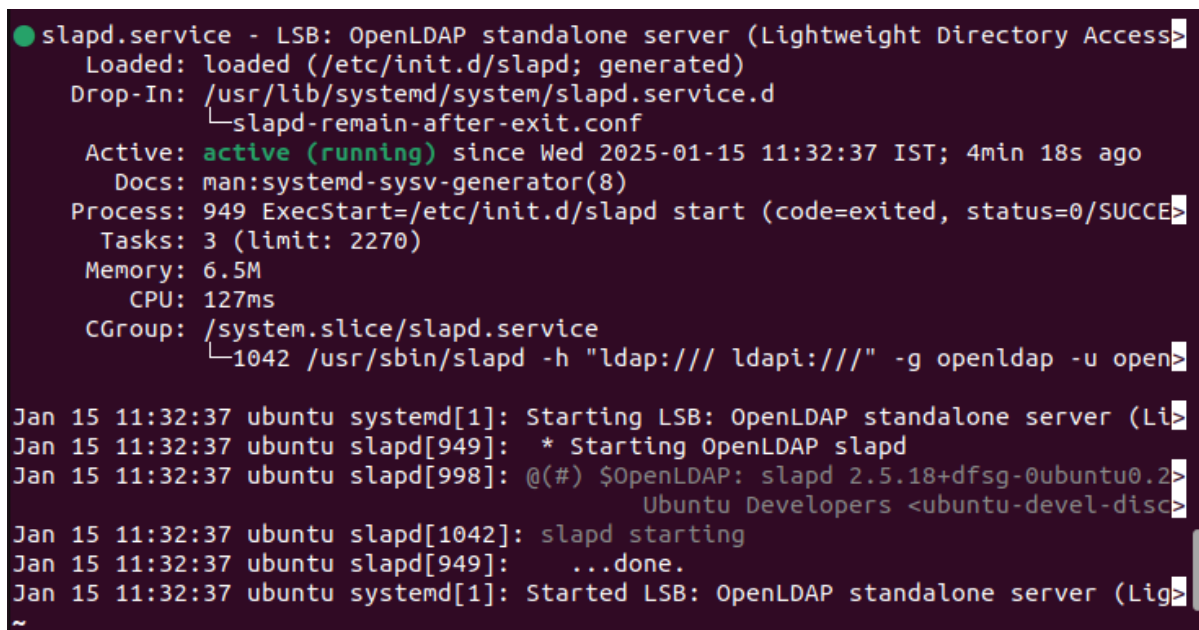
Screenshot 2: Services available in pfSense



Screenshot 3: URLs managed by Flask application



Screenshot 4: Tree view of LDAP directory structure



Screenshot 5: SLAPD actively running on ubuntu client

```
shubha@ubuntu:~$ ldapsearch -D "cn=admin,dc=myquest,dc=virtualbox,dc=org" -W -b "ou=users,dc=myquest,dc=virtualbox,dc=org"
Enter LDAP Password:
# extended LDIF
#
# LDAPv3
# base <ou=users,dc=myquest,dc=virtualbox,dc=org> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
# users, myquest.virtualbox.org
dn: ou=users,dc=myquest,dc=virtualbox,dc=org
objectClass: organizationalUnit
ou: users
# admin, users, myquest.virtualbox.org
dn: cn=admin,ou=users,dc=myquest,dc=virtualbox,dc=org
objectClass: inetOrgPerson
cn: admin
sn: Admin
uid: admin
userPassword:: YWRtaW5SANTIZ
# search result
search: 2
result: 0 Success
# numResponses: 3
# numEntries: 2
```

Screenshot 6: Details of user present in the LDAP Directory

## CCS-G29\_Report\_for\_Plagiarism\_Check

## ORIGINALITY REPORT

2%

SIMILARITY INDEX

1%

INTERNET SOURCES

1%

PUBLICATIONS

1%

STUDENT PAPERS

## PRIMARY SOURCES

1

Submitted to Colorado Technical University  
Online

Student Paper

&lt;1%

2

Submitted to Institute of Aeronautical  
Engineering (IARE)

Student Paper

&lt;1%

3

ictgo.wordpress.com

Internet Source

&lt;1%

4

Marina Paolanti, Simona Tiribelli, Benedetta  
Giovanola, Adriano Mancini, Emanuele  
Frontoni, Roberto Pierdicca. "Ethical  
Framework to Assess and Quantify the  
Trustworthiness of Artificial Intelligence  
Techniques: Application Case in Remote  
Sensing", Remote Sensing, 2024

Publication

&lt;1%

5

Submitted to Central Queensland University

Student Paper

&lt;1%

6

Submitted to University of West London

Student Paper

&lt;1%





The CPD Standards Office  
CPD PROVIDER: 41183  
2024-2026  
www.cpdstandards.com



# 3<sup>rd</sup>

# INTERNATIONAL CONFERENCE ON

## ADVANCES IN SCIENCE, ENGINEERING & TECHNOLOGY

22<sup>nd</sup> – 23<sup>rd</sup> March 2025

Chennai, India






Ref No : 69975

Date : 30/12/2024

Conference Secretariat – Chennai, India

### Letter of Acceptance

Abstract ID : [3RD-ICASET-2025\\_CHE\\_0651](#)Paper Title : [AI-Driven Cybersecurity and Surveillance Framework for Educational Institutions: A Proactive, Scalable Solution](#)Author Name : [Chandrashekhar S.](#)Co-Author Name : [Soumya G D, Shubha K A, Augustian P B, Kavya Jaishree J](#)Institution : [Presidency University](#)

Dear Chandrashekhar S,

Congratulations!

The scientific reviewing committee is pleased to inform your article "AI-Driven Cybersecurity and Surveillance Framework for Educational Institutions: A Proactive, Scalable Solution" is accepted for Oral/Poster Presentation at **"3rd International conference on Advances in Science, Engineering & Technology (ICASET)"** on **22nd & 23rd March 2025 at Chennai, India**, which is organized by SSM College of Arts & Science, Atal Community Innovation Centre Rise (ACIC RISE) Association and Chandigarh group of colleges. The Paper has been accepted after our double-blind peer review process and plagiarism check.

Your presentation is scheduled for the **Computer Science & Artificial Intelligence**. This session promises a dynamic exploration of **"Towards Sustainable Societal Transformation: Advances in Science, Engineering & Technology for Global Development Development: Enabling Sustainable Development through Science, Engineering, and Technology"** bringing together diverse perspectives and cutting-edge research

**"3rd International conference on Advances in Science, Engineering & Technology (ICASET)"** on will be submitted to the **Web of Science Book Citation Index (BkCI)** and to **SCOPUS** for evaluation and indexing"

Name of the Journal	Indexing and ISSN
International Journal of Intelligent Systems and Applications in Engineering (IJISAE)	SCOPUS; ISSN : 2147-6799
International Journal of Electrical and Electronic Engineering and Telecommunications(IJEETC)	SCOPUS; ISSN : 2319-2518

## MAPPING OF THE PROJECT WITH THE SUSTAINABLE DEVELOPMENT GOALS (SDGs).



The project aligns with several **Sustainable Development Goals (SDGs)**, particularly **SDG 4 (Quality Education)** by ensuring a safer and more productive online learning environment for teachers and students through controlled access to educational resources. It also supports **SDG 10 (Reduced Inequalities)** by providing equal and secure access to content, promoting inclusivity in digital education. The use of LDAP for user authentication and a custom API for firewall management contributes to **SDG 16 (Peace, Justice, and Strong Institutions)** by safeguarding personal data and ensuring transparent and participatory decision-making in content management. Additionally, the project's focus on responsible digital use supports **SDG 12 (Responsible Consumption and Production)**, promoting responsible access to the internet.