GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

PRESIDENCY UNIVERSITY

Private University Estd. in Karnataka State by Act No. 41 of 2013

BANGALORE

PRESIDENCY GROUP
OVER 40 YEARS OF ACADEMIC WISDOM

A Project Report

On

“SECURITY & SURVEILLANCE FOR TEACHERS AND STUDENTS”

Batch No: CCS – G29

Sl. No.	Roll Number	Student Name
1	20211CCS0065	Chandrashekhar S
2	20211CCS0067	Shubha K A
3	20211CCS0104	Augustian P B
4	20211CCS0131	Kavya Jaishree

School of Computer Science and Engineering,

Presidency University, Bengaluru.

Under the guidance of,

Ms. Soumya

Assistant Professor

School of Computer Science,

Presidency University, Bengaluru

CONTENTS

1. INTRODUCTION

The rapid advancement of technology and its subsequent integration into various fields has transformed the way we operate in every aspect of our lives. One such field where technology has had a profound impact is education. With the widespread availability of the internet and the increasing reliance on digital resources, classrooms have evolved from traditional brick-and-mortar spaces to hybrid environments where students and teachers engage with information and each other online. While the internet provides a vast array of educational resources, it also poses numerous challenges, particularly in terms of managing and securing access to web content. The proper management of this access is essential in creating a safe and productive learning environment. This is especially relevant in educational institutions where both students and teachers require access to specific online materials, but uncontrolled access can lead to disruptions, distractions, or exposure to harmful content.

One of the most effective ways to manage this access is through Access Control Policies (ACPs), which include URL filtering as a critical component. URL filtering is a method used to block or permit access to websites based on predefined criteria, ensuring that users can only visit sites that align with the institution's guidelines. While these policies play a vital role in maintaining network security and enhancing productivity, their management can often be a complex and time-consuming process, particularly for non-technical users such as teachers. The problem is compounded by the fact that the tools available for managing these policies, such as the Firepower Management Center (FMC), are primarily designed for network administrators and IT professionals with a high level of technical expertise.

The Importance of Web Filtering in Educational Institutions

The internet has become an essential tool in education, providing students with access to a wealth of information that would otherwise be unavailable. However, it also presents risks, particularly when students, many of whom are minors, have unrestricted access. Inappropriate content, such as violence, adult material, or harmful ideologies, can be easily accessed if proper filtering is not in place. Moreover, certain websites, while not harmful, can serve as distractions, leading to a loss of productivity during school hours. Social media, online gaming, and streaming platforms are just a few examples of content that might need to be restricted during learning hours to ensure students focus on their studies.

For educational institutions, web filtering is not just about blocking inappropriate content but also about promoting a safe and controlled learning environment. URL filtering policies ensure that students have access to educational resources while being shielded from content that could be harmful or distracting. In many countries, such filtering is not just recommended but is required by law. For example, in the United States, the Children’s Internet Protection Act (CIPA) mandates that schools and libraries receiving government funding must implement measures to block access to inappropriate online content.

While web filtering serves an important function, it must be flexible enough to adapt to the evolving needs of students and teachers. Educational content is constantly changing, and websites that were once deemed useful may no longer be relevant. Conversely, new resources become available every day, and it’s essential that teachers have the ability to quickly and easily update the policies that control internet access in their classrooms.

The Complexity of Managing Access Control Policies (ACPs)

Access Control Policies (ACPs) are an integral part of any institution’s cybersecurity strategy. ACPs dictate what users are allowed to do within a network, controlling access to files, applications, and websites based on predefined rules. These policies are enforced to protect the network from potential threats and ensure that resources are being used appropriately. In educational settings, ACPs are particularly important because they govern access to websites and online resources. URL filtering, a component of ACPs, enables institutions to define which websites are accessible based on categories such as “education,” “entertainment,” or “adult content,” and to either permit or block access based on these categories.

While ACPs provide a robust mechanism for managing access, the tools used to create and update these policies are often complex. Most of the available solutions are designed for network administrators and IT professionals who have a deep understanding of the technical aspects of network security. These tools often involve multiple steps, require knowledge of how different rules interact with each other, and necessitate a clear understanding of the overall network architecture. Firepower Management Center (FMC), for instance, is a powerful platform for managing security policies, but its complexity can make it difficult for non-technical users to navigate.

The FMC GUI allows administrators to manage and monitor all security policies across the network, including ACPs. It provides a detailed interface where administrators can define rules for traffic inspection, malware protection, URL filtering, and other security measures. While this level of control is necessary for managing large networks, it can be overwhelming for users without technical expertise. Teachers, for example, who may need to add or remove URLs from a filtering policy, would need to navigate through multiple menus and configurations to make even the simplest changes. This complexity can lead to errors, delays, or the need for constant assistance from the IT department, which in turn affects the institution's ability to quickly respond to new educational needs.

The Gap Between Technical and Non-Technical Users

The key issue in many educational institutions is the gap between those who manage network security (typically IT professionals) and those who use the network on a daily basis (teachers and students). While IT administrators are responsible for setting up and maintaining ACPs, it is the teachers who interact with students and understand their day-to-day educational needs. Teachers may need to quickly modify URL filtering policies to allow access to new educational resources or block sites that have become distracting or irrelevant. However, because of the complexity of existing tools like FMC, teachers often have to rely on IT staff to make these changes. This can create a bottleneck, as IT departments are often busy with other tasks, leading to delays in updating policies.

Additionally, giving teachers direct access to FMC can pose security risks. The FMC GUI provides access to a wide range of settings and configurations beyond URL filtering, many of which should only be managed by trained IT professionals. Allowing teachers to navigate this interface could lead to accidental changes in critical security settings, compromising the safety and integrity of the institution's network. Therefore, while teachers need a way to manage URL filtering, they also need a system that limits their access to only the relevant functions, ensuring that they can perform their tasks without the risk of damaging the overall network configuration.

The Impact of Delays in Updating URL Filtering Policies

In an educational environment, timing is critical. A delay in updating URL filtering policies can have a significant impact on both teachers and students. For instance, if a teacher discovers a new online resource that would benefit their students, but the site is blocked due to current filtering policies, they may not be able to use it in their lessons. Waiting for IT staff to update the policy could take hours or even days, during which time the opportunity to use the resource may have passed. Similarly, if students find a way to access distracting or inappropriate content that should be blocked, the longer it takes to update the filtering policy, the more time students have to engage with that content, potentially causing disruptions in the classroom.

These delays can also affect the reputation of the institution. Parents and students expect a certain level of efficiency and responsiveness when it comes to managing technology in schools. If it becomes known that the school is slow to respond to issues with internet access or web filtering, it could lead to dissatisfaction and complaints. In an era where educational institutions are increasingly judged by their ability to integrate technology into the learning process, the ability to quickly and effectively manage URL filtering policies is a key factor in maintaining a positive reputation.

Security Concerns and the Need for a Controlled Interface

While teachers need the ability to manage URL filtering policies, this must be done in a way that does not compromise network security. Security in educational institutions is critical, not just to protect the personal information of students and staff, but also to ensure the integrity of the learning environment. A breach in security could lead to unauthorized access to sensitive information or disruption of services, both of which could have serious consequences for the institution.

Giving teachers access to a platform like FMC, which manages all aspects of network security, introduces a number of risks. Even well-meaning users could inadvertently make changes to settings that affect the entire network. For example, a teacher might accidentally disable a critical security rule while trying to update a URL filtering policy, leaving the network vulnerable to attacks. Additionally, FMC provides access to logs and reports that may contain sensitive information, such as the browsing history of students or details about network traffic. This information should be restricted to IT administrators and should not be accessible to teachers or other staff members who do not have a direct role in managing network security.
Therefore, it is essential to provide teachers with a controlled interface that limits their access to only the functions they need to perform their tasks. This interface should allow them to add or remove URLs from the filtering policy without giving them access to other security settings or sensitive information. By doing so, the institution can ensure that teachers can manage internet access in their classrooms while maintaining the overall security of the network.

The Role of Automation in Managing URL Filtering

Automation can play a significant role in reducing the complexity of managing URL filtering policies. By automating certain processes, such as the addition of new educational resources or the blocking of distracting sites, the system can be made more efficient and less reliant on manual updates. For example, the system could be programmed to automatically allow access to certain categories of websites during specific hours, or to block social media sites when class is in session. Automation can also help to reduce the risk of human error, as policies can be updated based on predefined rules rather than relying on individual users to make manual changes.

However, automation alone is not enough. There still needs to be a way for teachers to manually update the URL filtering policy when necessary, particularly in cases where new resources are discovered or when specific content needs to be blocked quickly. Therefore, the ideal solution is one that combines automation with manual control, allowing teachers to make changes as needed while ensuring that the system remains efficient and secure.

2. LITERATURE REVIEW

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. 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.

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.

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.

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.

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.

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.

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.

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

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.

Each of these methods provides varying levels of control and usability, with their own sets of advantages and limitations. The choice of method largely depends on the specific requirements of the institution or organization, including factors such as technical expertise, budget, and network size. Understanding these options helps in formulating a more effective solution to URL filtering challenges in educational settings.

3. OBJECTIVES

1. Simplifying URL Filtering for Teachers

The existing Firepower Management Center (FMC) GUI offers extensive control but is highly technical and challenging for non-technical users. Teachers, who are not trained in network security, often find it difficult to navigate and modify URL filtering policies. The project aims to create an intuitive and simplified interface that enables teachers to modify the URL filtering rules without requiring advanced technical knowledge. The interface should abstract the complexities involved in URL management while maintaining the necessary security standards.

- **Rationale**: Teachers may need to block or allow specific URLs to align with the educational needs of their students. For example, a teacher may wish to block social media websites during class hours but allow them afterward. By creating a user-friendly interface, the project eliminates the need for teachers to consult network administrators for minor adjustments, saving time and improving workflow.

2. Maintaining Security Through Role-Based Access Control

While simplifying URL management for teachers, it is critical to maintain strict access control to ensure that sensitive or critical network settings cannot be modified by unauthorized users. The project will incorporate role-based access control (RBAC) to ensure that only authorized users, such as teachers, have access to URL filtering functions, while system administrators retain control over more advanced settings.

- **Rationale**: Security must be a priority when implementing any changes to network control policies. By using role-based access control, the project ensures that teachers only have the permissions necessary to modify URLs, preventing accidental or malicious alterations to the overall network configuration. Furthermore, the interface will be designed to automatically authenticate users against a directory server, powered by Microsoft Active Directory (AD), ensuring that only authorized personnel are granted access.

3. Integration with Firepower REST API for Policy Updates

The project will leverage Firepower's REST API to interact with the FMC for making real-time policy changes. Instead of manually navigating through the FMC interface, the new system will send GET/POST requests to modify URL policies directly.

- **Rationale**: The FMC already has a robust system for managing access control policies, but accessing it through the GUI can be cumbersome. By integrating the Firepower REST API, the project allows teachers to make immediate changes to URL filtering policies through a more intuitive and user-friendly platform. The use of REST API also ensures that policy changes are implemented seamlessly in real-time, reducing the administrative burden on network administrators.

4. Improving User Experience with a Web-Based Interface

The project proposes a web-based interface using Flask (Python framework) and Bootstrap (CSS framework) to provide a responsive and easy-to-navigate platform for URL filtering management. The web-based approach ensures that teachers can access the platform from any location with internet access, without needing to install specialized software.

- **Rationale**: The ease of use is paramount in ensuring adoption by non-technical users. Teachers should be able to log in, view current URL policies, and make necessary adjustments quickly. The use of modern web technologies like Flask and Bootstrap will enhance the user experience by providing a visually appealing and responsive interface that adjusts to different screen sizes and devices.

5. Ensuring Compatibility with Microsoft Active Directory for Authentication

Authentication is a crucial part of maintaining the security and integrity of the system. The project will integrate with Microsoft Active Directory (AD) to authenticate users based on their credentials and group memberships. This ensures that only authorized teachers and administrators can access the system and modify URL filtering policies.

- **Rationale**: Microsoft Active Directory is a widely used authentication system in educational and corporate environments. By leveraging AD, the project can authenticate users seamlessly based on their existing credentials without needing to create new login mechanisms. This not only simplifies the user experience but also provides a higher level of security by utilizing established authentication practices. Group-based authentication will further ensure that only specific users, such as teachers, are allowed access to the URL management system.

4. METHODOLOGY

Technical Implementation

The project involves several key methods and techniques to handle various tasks related to security, user authentication, data flow, and URL filtering management. Below is a breakdown of the methods being used in the project:

1. User Authentication (Microsoft Active Directory)

- **Method Used:** Identity Policy via Active Directory Authentication
    - **Description:** User authentication is handled using Microsoft Active Directory (AD), which stores user credentials and their respective groups. This allows for role-based access control (RBAC). The system verifies the identity of teachers and administrators against the directory, ensuring only authorized users can modify URL filtering policies.
    - **How it works:** The Flask application sends the user's credentials (username and password) to Active Directory via an API or LDAP (Lightweight Directory Access Protocol) query. Active Directory checks the credentials and responds with an authentication result.

2. Role-Based Access Control (RBAC)

- **Method Used:** Access Control via Role Verification
    - **Description:** After authentication, the system checks the user's role (teacher or admin) to determine their access level. Admins can add/remove teachers or manage URL policies, while teachers can only manage URL policies.
    - **How it works:** User roles are verified using the user's group information stored in Active Directory. Based on the role, specific access rights are granted to different sections of the web application.

3. URL Filtering Management

- **Method Used:** POST/GET Requests to Firepower Management Center (FMC) REST API
    - **Description:** URL filtering is managed by interacting with Cisco Firepower Management Center's (FMC) REST API. Teachers or admins can add or remove URLs from the filtering policy by sending requests to this API.
    - **How it works:** The Flask backend sends HTTP POST/GET requests to Firepower's REST API. These requests contain the necessary parameters (such as URL, action type—block or allow) to edit the URL filtering policy.

4. Frontend-Backend Interaction (Flask Framework)

- **Method Used:** Flask Framework for Web Application
    - **Description:** The Flask framework is used to build the web application that allows teachers and admins to interact with the system. It provides the necessary routing for handling user requests, managing sessions, and interacting with the backend processes.
    - **How it works:** Flask handles the user interface, processing form submissions (like login and URL changes), and forwards these requests to the backend Python scripts for further processing. It also manages session data (user login state) to ensure secure interaction.

5. Database Management (MongoDB)

- **Method Used:** MongoDB for Credential and Log Storage

    - **Description:** MongoDB, a NoSQL database, is used to store admin and teacher credentials (in cases where Active Directory is not directly linked) and log actions (such as URL additions/removals) performed by users.

    - **How it works:** When a URL is added or removed from the filtering policy, the system logs this change in MongoDB, storing the user's identity and the timestamp. MongoDB provides fast, scalable data storage for such use cases.

6. Data Validation and Error Handling

- **Method Used:** Input Validation in Frontend and Backend

    - **Description:** Input validation is performed both at the frontend (using JavaScript) and backend (using Python) to ensure the integrity and security of the system. This prevents unauthorized actions, SQL injection, and other security vulnerabilities.

    - **How it works:** When a user inputs data (such as URLs to block or login credentials), the system validates these inputs to ensure they are in the correct format and meet required security criteria. If validation fails, an error is returned to the user.

7. HTTP Session Management

- **Method Used:** Flask Session Management

    - **Description:** The project uses HTTP session management to track the user's login state. Flask's session feature securely stores session data, such as user roles and authentication status, to ensure seamless navigation through the web pages.

    - **How it works:** After a user log in, Flask stores a session ID on the client side, and the user's session data (such as role and login status) is tracked on the server. Sessions are encrypted and secured to protect against session hijacking.

8. API Integration and Communication

- **Method Used:** REST API Integration

    - **Description:** The project uses RESTful APIs to communicate between the web application (Flask) and external services such as Firepower Management Center (for URL filtering) and Active Directory (for authentication). These APIs allow the application to send and receive data in JSON format.

    - **How it works:** The web application sends HTTP requests (either GET or POST) to the API endpoints of FMC and AD. These requests are processed, and the response is returned to the web application for further action.

9. User Interface Design (Bootstrap)

- **Method Used:** Bootstrap Framework for UI Design

    - **Description:** Bootstrap is used to create a responsive, user-friendly interface for the web application. This ensures the application is accessible on various devices and provides a clean, structured layout.

    - **How it works:** Bootstrap components are integrated into HTML templates to create a cohesive user interface. Custom styles are applied to ensure the application meets the project's design requirements.

10. Testing and Validation

- **Method Used:** Unit Testing, Integration Testing, and User Acceptance Testing (UAT)

    - **Description:** Various testing methods are employed to ensure the project runs smoothly and meets user requirements. Unit testing is used to test individual components, integration testing verifies that all modules work together, and user acceptance testing ensures the application is user-friendly and functions as expected.

    - **How it works:** Each component (frontend, backend, API integration) is tested individually during unit testing. In integration testing, the interaction between components is tested to identify any issues in data flow. Finally, user acceptance testing is performed by allowing teachers and admins to use the system and provide feedback.

Project Development Phases:

1. Project Planning and Requirements Gathering

The first step in the methodology is thorough **planning and requirements gathering**. This phase ensures the development team understands the problem, objectives, and scope of the project.

1.1 Problem Identification and Stakeholder Analysis

The project stems from the challenge teachers face when needing to modify URL filtering policies via the Firepower Management Center (FMC) graphical interface, which is complex and requires technical expertise. The stakeholders, including teachers, IT staff, and administrators, are identified, and their needs are understood.

- **Teachers** need a simplified, intuitive way to add/remove URLs without technical know-how.
- **IT administrators** want secure role-based access control to ensure that only authorized users can make changes to the URL filtering policies.

1.2 Requirements Specification

The following requirements were specified:

- **Functional Requirements**: The platform should allow teachers to log in, view current URL filtering policies, and add/remove URLs with ease.

- **Non-Functional Requirements**: The platform should be secure, scalable, and responsive, with efficient user authentication through Microsoft Active Directory (AD).

2. System Design

The second phase involves **system design**, where the architecture, data flow, and key components of the system are outlined. The design focuses on creating a robust, secure, and user-friendly system.

2.1 High-Level Architecture Design

The architecture of the system is divided into three main layers:

1. **Frontend (User Interface)**: Built with Flask, HTML, CSS, and Bootstrap, the frontend offers a simple and responsive interface for teachers to interact with the system.

2. **Backend (Application Logic)**: The backend, also built with Flask, interacts with the Firepower REST API, processes requests from the frontend, and manages user authentication with Active Directory.

3. **Database**: MongoDB is used for storing logs of policy changes and managing user sessions if needed.

This layered architecture ensures separation of concerns, making the system easier to maintain and scale.

2.2 Data Flow Diagram (DFD)

A data flow diagram is created to illustrate how information flows through the system. The main interactions include:

- Teachers logging in through the web interface (authenticated via Active Directory).
- Teachers viewing and modifying the URL filtering policies.
- The backend sending API requests to Firepower's REST API to apply changes.
- The system logging policy modifications in the database for audit purposes.

2.3 User Interface Design

The interface is designed to be simple and intuitive for non-technical users. Key screens include:

- **Login Page**: Users authenticate using Active Directory credentials.
- **Dashboard**: Displays current URL filtering policies with options to add or remove URLs.
- **Success/Failure Notifications**: Users receive real-time feedback on policy changes.

3. Development Phase

After the design phase, the project moves into **development**, where the actual code for the system is written. The development is divided into two main parts: frontend and backend development.

3.1 Frontend Development

The frontend development focuses on creating the user interface (UI) using HTML, CSS, and JavaScript, with Flask as the web framework.

- **HTML/CSS**: Basic structure and styling for the user interface are built using Bootstrap to ensure the pages are responsive and accessible on different devices (desktops,tablets, mobile).
- **JavaScript**: Used to handle dynamic elements and client-side validations. For instance, when a teacher inputs a URL, the system checks for valid URL formats before sending the request to the backend.

3.2 Backend Development

The backend handles the application logic, REST API interactions, and data management.

- **Flask**: Flask is used as the web framework to handle HTTP requests, render HTML templates, and serve content dynamically.
- **Integration with Firepower REST API**: The system uses Flask's built-in functionality to send GET/POST requests to Firepower's REST API. The GET request retrieves the current URL filtering policies, while the POST request sends the updated list to FMC.

- **Role-Based Access Control**: Using Flask-Login and integration with Microsoft Active Directory, the system ensures only authorized users (teachers) can log in and modify URL filtering policies.

4. Integration with Firepower’s REST API

One of the most critical aspects of the project is **integrating the system with Firepower's REST API**. This enables the platform to directly modify URL filtering policies in the Firepower Management Center (FMC).

4.1 API Authentication

Before making API requests, the system authenticates itself with Firepower’s REST API using pre-configured credentials. This ensures secure communication between the system and the Firepower Management Center.

4.2 API Requests

The system performs two types of API requests:

- **GET Requests**: Used to retrieve the current URL filtering policies from FMC, which are displayed on the teacher’s dashboard.
- **POST Requests**: Used to send data (e.g., added or removed URLs) back to FMC to update the URL filtering policies.

4.3 Error Handling

Proper error handling is implemented to ensure the system responds gracefully to API failures. If an API request fails, the teacher is notified with an error message, and the system retries the operation or logs the issue for IT administrators.

5. Security Implementation

Security is a critical consideration in this project since it deals with network access control policies. The following security measures are implemented to protect the system and its users:

5.1 User Authentication and Authorization

The system integrates with **Microsoft Active Directory** to authenticate users. This ensures that only authorized users (teachers and administrators) can log in. Role-based access control (RBAC) ensures that teachers can only modify URL policies, while higher-level access (such as firewall settings) is restricted to administrators.

5.2 Secure Communication (HTTPS)

All communication between the user’s browser and the web application is encrypted using **HTTPS** to prevent interception of sensitive data like login credentials and policy changes.

5.3 Session Management

The system uses **Flask-Login** to manage user sessions securely. User sessions expire after a set period of inactivity, reducing the risk of unauthorized access if a user forgets to log out.

6. Testing Phase

Before deployment, the system undergoes rigorous **testing** to ensure it functions as expected. The testing phase involves several different types of testing:

6.1 Unit Testing

Each individual component of the system, both on the frontend and backend, is tested independently. For example, the URL input form is tested for correct validation, while the API interaction is tested to ensure policies are retrieved and updated correctly.

6.2 Integration Testing

After unit testing, the frontend and backend components are integrated and tested together to ensure seamless interaction between the two. This testing ensures that teachers can successfully log in, view policies, and update them without errors.

6.3 User Acceptance Testing (UAT)

Teachers and IT administrators participate in user acceptance testing to ensure the system meets their requirements. Feedback is gathered and any necessary changes are made to improve usability or fix bugs.

6.4 Security Testing

Penetration testing is performed to identify any vulnerabilities in the system. This includes testing for common web vulnerabilities such as **SQL injection**, **cross-site scripting (XSS)**, and **cross-site request forgery (CSRF)**.

7. Deployment and Integration

Once testing is complete, the system is deployed in the institution’s environment. Deployment includes setting up the application on a secure web server, configuring the domain and SSL certificates, and integrating with Active Directory and the Firepower Management Center.

7.1 Server Configuration

The web application is deployed on a secure server (either on-premise or cloud-based) with appropriate firewall and security configurations. The domain is set up with SSL certificates to enable secure HTTPS communication.

7.2 Integration with Active Directory

Active Directory is configured to handle authentication for teachers, administrators, and any other users accessing the system. This eliminates the need for users to create new accounts or manage additional passwords.

8. Post-Deployment Maintenance

After deployment, the system enters a **maintenance phase** where it is monitored and updated regularly to ensure ongoing functionality and security.

8.1 Monitoring

System logs and performance metrics are monitored to identify any potential issues, such as slow response times or API errors. The logs also help in tracking user activity, such as who added or removed URLs.

8.2 Software Updates

Regular software updates are applied to the web application, API integration, and dependencies to ensure security and performance improvements.

8.3 Support and Feedback

User feedback is collected to address any challenges teachers face while using the platform. Technical support is provided for troubleshooting, and new features may be added based on feedback and evolving requirements.

5. ARCHITECTURE

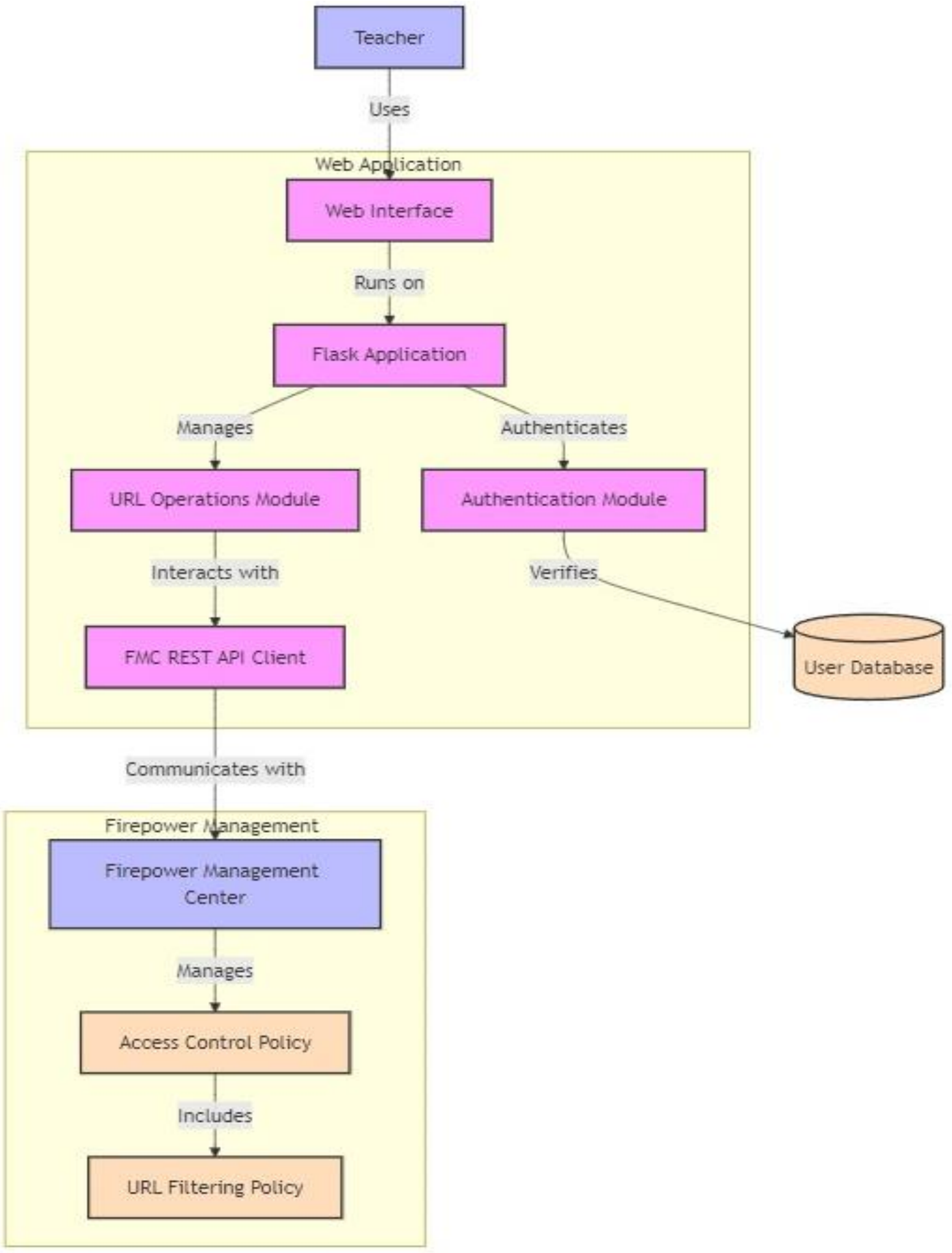


Figure 6.1 FMC URL Filtering Management Architecture

The figure 6.1 illustrates the architecture of a web application designed for managing URL filtering policies in a Firepower Management Center (FMC) environment, with the primary user being a teacher. Here's a stepwise explanation of the architecture:

1. Teacher Interacts with the Web Application:

- **Role**: The teacher is the primary user of the system, accessing it through a web interface.
- **Interface**: They interact with the application via a web browser, using the web interface.

2. Web Application Runs on Flask Framework:

- **Technology**: The web application is built using the Flask framework, a Python-based web development framework.
- **Components Managed**: The Flask application serves as the core of the system, managing different modules (authentication and URL operations).

3. Authentication Module:

- **Function**: The Authentication Module is responsible for verifying the credentials of the teacher. It handles the login process and determines access permissions.

- **Interaction with Database**: This module verifies the teacher's credentials (such as username and password) by querying the User Database. If authentication is successful, the user gains access to further functionalities.

4. URL Operations Module:

- **Function**: After successful authentication, the URL Operations Module is responsible for managing URL filtering policies. It allows the teacher to add, remove, or modify URL lists that are part of the access control policies.

- **Interaction with FMC REST API Client**: This module interacts with the FMC REST API Client, sending requests to modify policies in the Firepower Management Center.

5. FMC REST API Client:

- **Function**: The FMC REST API Client is a crucial component that communicates with the Firepower Management Center (FMC). It uses REST API calls to make changes to access control policies.

- **Modules Managed:** This client enables the web application to interact with the underlying Firepower Management Center, which manages network security policies.

6. Firepower Management Center (FMC):

- **Role**: The FMC is a centralized platform for managing network security policies.

- **Function**: It controls the Access Control Policy, which includes URL Filtering Policies that control which URLs are accessible on the network.

- **Communication**: The FMC receives commands from the FMC REST API Client and applies changes to the access control policies based on the teacher's actions within the web application.

7. Final Outcome:

- The teacher uses the web interface to authenticate and manage URL policies, and their changes are pushed to the FMC, which enforces these policies across the network.

Key Components of the Architecture:

1.User Interface (Frontend):

- The web-based interface (built using Flask, Bootstrap, HTML/CSS, JavaScript) that teachers and administrators use to log in and manage URL filtering policies.

2.Authentication System:

- The system authenticates users through Microsoft Active Directory (AD), verifying their credentials and roles (teacher, admin, or student).

3.Access Control:

- The logic that defines whether a user is authorized to modify URL filtering policies based on their role (admin vs. teacher).

4.Backend Logic (Flask, Python):

- The core logic of the application that processes user requests (adding/removing URLs) and interacts with the Firepower REST API.

5.Database (MongoDB):

- Stores teacher/admin credentials and logs of changes to URL filtering policies.

6.Firepower Management Center (FMC) API:

- Sends GET and POST requests to Firepower's REST API, which manages the actual Access Control Policies for URL filtering.

Data Flow Diagram (DFD):

1. User Interaction:

- Step 1:
  - A teacher or admin opens the web application.
  - Inputs: The user enters credentials (username, password) for authentication.
- Step 2:
  - The system sends the credentials to Microsoft Active Directory for authentication.
  - Output: The Active Directory responds with an authentication result (valid or invalid).

2. Role Verification and Access Control:

- Step 3:
  - Once authenticated, the system verifies the user's role (admin or teacher).
  - Admins have full access to both the "Add/Remove Teacher" page and the URL policy management page.
  - Teachers can only access the URL policy management page.
- Step 4:
  - If the user is unauthorized or a student, they are denied access and redirected to a failure page.

3. URL Management Actions:

- Step 5:
  - An authenticated teacher/admin enters a URL (to be blocked or unblocked) in the web interface.
  - Inputs: The teacher/admin provides the URL and selects whether to block or unblock it.
- Step 6:
  - The application sends this URL request to the backend logic (Flask + Python).

4. Interfacing with Firepower REST API:

- Step 7:
  - The backend logic formats the request and sends a POST or GET request to the Firepower Management Center (FMC) REST API to apply the change to the URL filtering policy.
- Step 8:
  - The FMC updates its URL filtering policy accordingly and returns a response indicating success or failure.
- Step 9:
  - The response is displayed on the web interface as a success or failure message (e.g., "URL blocked successfully").

5. Logging and Storing Changes:

- Step 10:
  - The backend logs the action (e.g., URL addition or removal) and stores this information in the MongoDB database. The logs can be reviewed by the admin to monitor changes.

6. EXPECTED OUTCOMES

1. User-Friendly Interface for Policy Management

- One of the key expected outcomes is the development of a **simple and intuitive web interface** that allows teachers and authorized staff to easily manage URL filtering policies. This will eliminate the need for them to interact with the complex Firepower Management Center (FMC) GUI or the command-line interface (CLI).
- Teachers will be able to **add** or **remove URLs** from the filtering policy with minimal technical knowledge, empowering them to adapt to changing educational needs (such as blocking or unblocking certain websites) without relying on IT specialists.

2. Improved Workflow Efficiency

- By providing a **streamlined process** for teachers to manage URL policies, the project will reduce the **time and effort** required for managing internet access within the institution.
- This will enable quicker responses to requests for website access or blocking, improving overall network security while accommodating academic needs.
- IT staff will no longer have to handle routine requests for URL changes, freeing them to focus on more critical tasks.

3. Role-Based Access Control and Security

- The system will integrate **Microsoft Active Directory** to authenticate users and implement **role-based access control (RBAC)**. Only authorized users (teachers and administrators) will be able to modify URL filtering policies, ensuring that **student users** and **unauthorized personnel** cannot tamper with the network's security settings.
- This will create a secure environment where access to sensitive policy settings is restricted and managed appropriately, maintaining the integrity of the institution's internet filtering policies.

4. Real-Time Integration with Firepower's REST API

- The solution will effectively **communicate with Firepower's REST API**, sending **POST** and **GET** requests to create or edit access control policies in real time. This means that any changes made by teachers on the web interface will be reflected in the Firepower Management Center almost instantaneously.
- This will help ensure that security policies are enforced **immediately** after changes are made, minimizing delays in adjusting internet access restrictions for students.

5. Enhanced Security and Surveillance of Student Internet Activity

- Through efficient URL filtering, the project will enable the institution to enforce **internet usage policies** that safeguard students from harmful content and ensure compliance with academic guidelines.
- This will also support the **surveillance and monitoring** of students' online activities, as inappropriate websites can be quickly identified and blocked, promoting a safer digital learning environment.

6. Scalability and Flexibility

- The system will be **scalable**, meaning that it can accommodate a growing number of users (teachers, administrators, etc.) and an increasing number of URLs in the filtering policy as the institution expands or its needs change.
- The **flexibility** of the system will allow it to adapt to future technological changes and policy updates, such as integration with additional APIs or support for new security features.

7. Reduced Training Requirements for Non-Technical Users

- Since the web interface will be designed to be **user-friendly** and accessible even to non-technical users, teachers and staff will require minimal training to use the platform effectively.
- This will result in **lower costs** associated with training personnel, as well as fewer errors and technical issues, since the interface abstracts the complexity of URL policy management.

8. Centralized Management with Activity Logs

- The system will provide a **centralized platform** for managing all URL filtering policies. This will ensure that all changes made by teachers are tracked and logged for auditing purposes, allowing IT staff and administrators to monitor activity and ensure compliance with institutional guidelines.
- This **logging and auditing** capability will also help in identifying and addressing any unauthorized changes or suspicious behaviour, contributing to overall network security.

9. Seamless Integration into the Educational Environment

- The project will integrate seamlessly into the existing educational infrastructure, providing a tool that supports teachers in creating a safe, educationally appropriate internet environment for students.
- The URL filtering system will align with the institution's academic policies and objectives, ensuring that students can access the online resources they need for learning while being protected from inappropriate or harmful content.

10. Reduction in Human Error and IT Load

- By offering a more **automated and user-friendly solution**, the project will significantly reduce the potential for human error, which is more likely to occur with manual URL filtering through the FMC GUI or command line.
- IT departments will no longer be the **primary point of contact** for managing URL lists, which will reduce their workload and enable them to focus on more critical security tasks.

11. Faster Response to URL Filtering Needs

- The system will allow for **instant changes** to the URL filtering policies, so teachers can quickly block or unblock websites as needed without delays. This will be especially beneficial in scenarios where websites need to be adjusted in real-time, such as during online classes or examinations.
- This faster response will contribute to a more agile and secure network environment that can adapt to the dynamic needs of both students and teachers.

12. Increased Awareness of Network Security Among Teachers

- By involving teachers directly in the management of URL filtering policies, this project will promote greater **awareness** of network security practices among non-technical staff.
- Teachers will better understand the importance of maintaining a secure network environment and will be more engaged in enforcing internet usage policies, thereby contributing to the overall security culture of the institution.

13. Cost-Effective Solution for Policy Management

- Compared to more complex third-party network management solutions, this project will provide a **cost-effective alternative** by using open-source frameworks such as Flask, Python, and MongoDB. This will reduce the need for expensive software licenses or external services, making the project budget-friendly.
- Additionally, the system will rely on existing infrastructure (e.g., Microsoft Active Directory and Firepower's REST API), minimizing additional costs for hardware or software procurement.

14. Adaptable for Other Educational Institutions

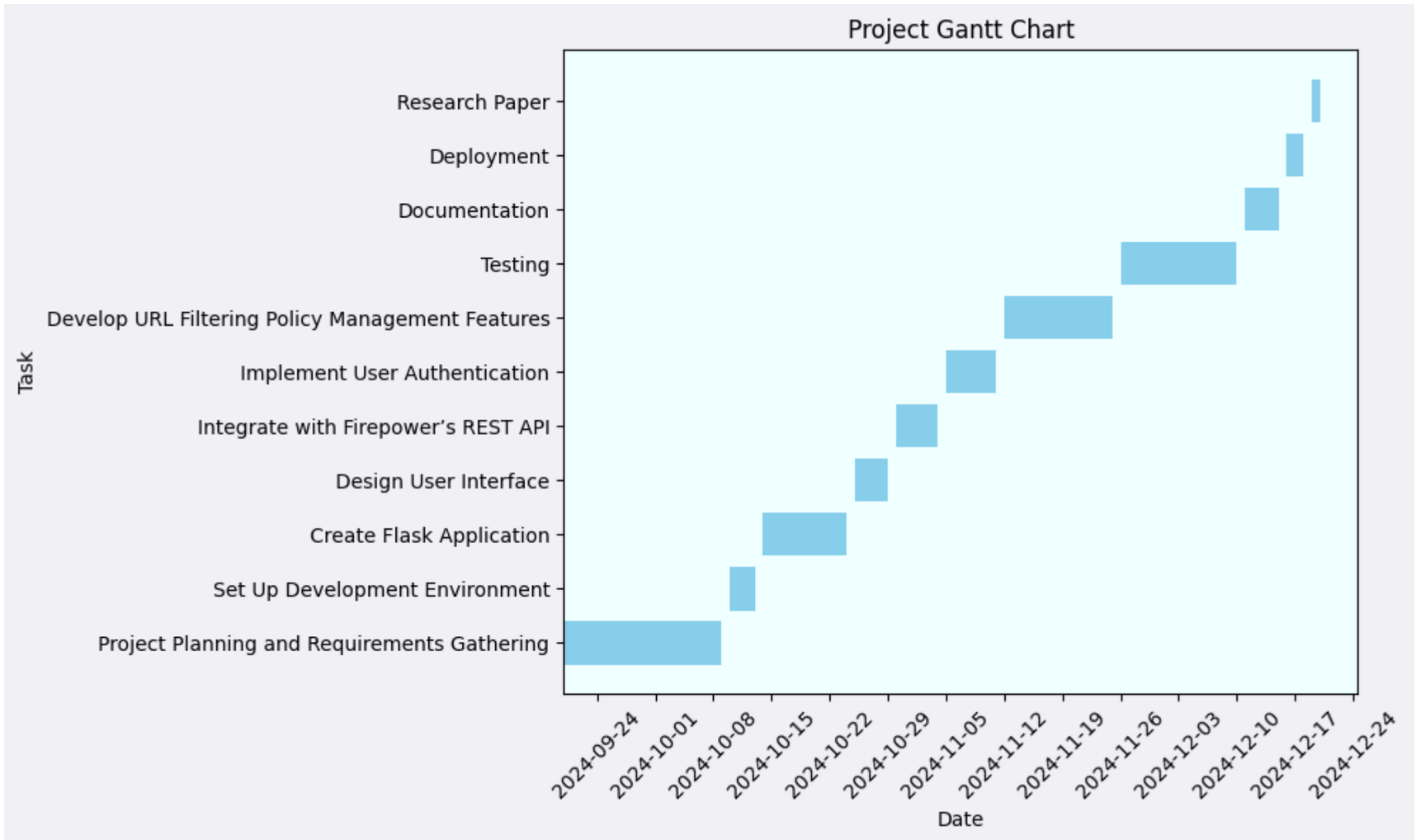
- The final product will not be limited to just the current institution but can be **adapted for use** in other educational environments with similar URL filtering and access control needs.
- This adaptability could allow for wider application of the project in other schools, colleges, or universities, contributing to improved security and internet management across multiple educational sectors.

7. TIMELINE OF THE PROJECT

Execution Plan:

Project Phase	Task (with Sub-tasks)	Start Date	End Date
1. Project Planning and Requirements Gathering	1.1 Identify problems and gather stakeholder requirements. 1.2 Analyze teacher and IT administrator needs. 1.3 Define user roles and access levels.	September 24, 2024	October 10, 2024
2. System Design	2.1 Design high-level system architecture. 2.2 Create data flow diagram (DFD). 2.3 Design user interface (UI) with mockups.	October 13, 2024	October 21, 2024
3. Development Phase	3.1 Develop the frontend using Flask, HTML, CSS. 3.2 Implement dynamic elements with JavaScript. 3.3 Develop backend integration with Firepower API.	October 25, 2024	November 3, 2024
4. Integration with Firepower's REST API	4.1 Authenticate with Firepower's REST API. 4.2 Retrieve current URL filtering policies. 4.3 Implement POST request to update URL policies.	November 4, 2024	November 11, 2024
5. Security Implementation	5.1 Implement Active Directory-based authentication. 5.2 Secure communication using HTTPS. 5.3 Implement role-based access control (RBAC).	November 12, 2024	November 18, 2024
6. Testing Phase	6.1 Conduct unit testing for frontend and backend. 6.2 Perform integration testing with Firepower API. 6.3 Perform security penetration testing.	November 19, 2024	November 27, 2024
7. Deployment and Integration	7.1 Deploy the system on a secure web server. 7.2 Configure Active Directory for user access. 7.3 Set up HTTPS and firewall configurations.	November 28, 2024	December 1, 2024
8. Post-Deployment Maintenance	8.1 Monitor system logs and user activity. 8.2 Apply regular software updates. 8.3 Provide technical support and gather feedback.	December 2, 2024	December 5, 2024

Gantt Chart:


Project Gantt Chart
Research Paper
Deployment
Documentation
Testing
Develop URL Filtering Policy Management Features
Implement User Authentication
Integrate with Firepower’s REST API
Design User Interface
Create Flask Application
Set Up Development Environment
Project Planning and Requirements Gathering
Task
2024-09-24
2024-10-01
2024-10-08
2024-10-15
2024-10-22
2024-10-29
2024-11-05
2024-11-12
2024-11-19
2024-11-26
2024-12-03
2024-12-10
2024-12-17
2024-12-24
Date

CONCLUSION

This project demonstrates a robust, scalable, and user-friendly solution for managing URL filtering policies in a network environment using Cisco's Firepower Management Center (FMC). By integrating the power of Flask for web application development and FMC's REST API for network policy management, the system effectively empowers authorized users—such as teachers—to dynamically manage and modify URL filtering rules.

The architecture prioritizes security through its **Authentication Module**, ensuring that only verified users can access the sensitive operations of URL management. This access control is strengthened by using a dedicated **User Database**, which stores teacher credentials and verifies identity before permitting policy modifications.

The core of the application, the **URL Operations Module**, simplifies the complexity of interacting with the Firepower Management Center, allowing teachers to manage URL lists effortlessly. Through this interface, they can add, remove, or update URLs, which are directly applied to the FMC via the **REST API Client**. The seamless communication between the **FMC REST API Client** and the FMC ensures that the teacher's changes are immediately reflected in the network's security policies.

The overall system contributes significantly to network security by enabling granular control over URL filtering, a critical feature for environments like educational institutions where access to content needs to be regulated. Additionally, by placing the control in the hands of trusted individuals (teachers), the system enhances flexibility and responsiveness in policy management, reducing the overhead of manual intervention by network administrators.

Key Highlights:

1. **Scalability**: The system can be easily extended to accommodate multiple users or more complex policy management tasks.

2. **Security**: The robust authentication system prevents unauthorized access to sensitive policy controls.

3. **Real-time Policy Updates**: Changes made by the user are quickly propagated to the network through the FMC, ensuring that security rules are always up to date.

4. **User-friendly Interface**: The web application provides a simple and intuitive interface for non-technical users like teachers to manage complex security policies without needing deep technical expertise.

Future Prospects:

To further enhance this system, several improvements can be made:

- **Role-based Access Control (RBAC)**: Implementing RBAC could allow for different permission levels, with admins having broader control over access policies and teachers restricted to only specific URL filtering.

- **Enhanced Logging and Monitoring**: Integrating logging of user actions (such as URL modifications) would improve traceability and accountability within the system.

- **Improved UI/UX**: While the current interface is functional, investing in a more interactive and visually appealing user experience could enhance usability.

Overall, this project delivers a practical solution to URL filtering management, enhancing both security and usability within a controlled environment. The collaboration between Flask, FMC, and the REST API ensures that this system can be a powerful tool in institutional and enterprise settings.

REFERENCES

- https://www.cisco.com/c/en/us/td/docs/security/firepower/610/configuration/guide/fpmc-config-guide-v61/fpmc-configguide-v61_chapter_01110011.html

- https://www.cisco.com/c/en/us/td/docs/routers/sdwan/configuration/security/ios-xe-16/security-book-xe/url-filtering.pdf

- https://ieeexplore.ieee.org/document/9501961

- https://doi.org/10.13052/jwe1540-9589.2345

- https://www.cisco.com/c/en/us/td/docs/security/firepower/630/configuration/guide/fpmc-config-guidev63/create_and_manage_identity_policies.html

- https://www.cisco.com/c/en/us/td/docs/security/firepower/710/fdm/fptd-fdm-config-guide-710/fptd-fdm-identity.html

- https://www.cisco.com/c/en/us/td/docs/security/firepower/650/api/REST/Firepower_Management_Center_REST_API_Quic k_Start_Guide_650.pdf

- https://www.cisco.com/c/en/us/td/docs/security/firepower/710/fdm/fptd-fdm-config-guide-710/fptd-fdm-identity.html