IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

Labwall: A Configurable Firewall Solution for College Computer Laboratories

Shailesh Sathe, Gaurav Raut, Ayush Salunkhe, K. T. Patil
Department of Computer Engineering
SIGCE, Navi Mumbai, India

Abstract: *Educational institutions face significant challenges in managing computer laboratory environments, particularly in controlling internet access, monitoring student activities, and maintaining academic integrity during examinations. This paper presents Labwall, a centralized firewall and monitoring system designed specifically for college computer laboratories. The system addresses the critical need for controlled computing environments through three operational modes: Normal, Lab, and Exam. Labwall integrates Windows Firewall, PowerShell scripts, and a Python-based backend to dynamically configure access rules and collect activity logs in real time. A React-based administrative dashboard provides the operator with live visibility into all connected student systems, enabling one-click mode switching without any per-machine intervention. The system operates entirely over the existing local area network (LAN) or Wi-Fi infrastructure, requiring no additional hardware, external servers, or licensing fees. Evaluation across 42 participants under all three modes demonstrated a 100% bypass-prevention rate and zero false negatives, validating the system as a practical and cost-effective solution for educational laboratory governance.*

Keywords: *Firewall, Network Security, Educational Technology, Laboratory Management, Access Control, Exam Proctoring, Computer Labs, PowerShell, WebSocket*

I. INTRODUCTION

Computer laboratories in educational institutions serve as critical infrastructure for practical learning, skill development, and formal assessments. However, unrestricted internet access in these environments frequently results in resource misuse, student distraction during practical sessions, and compromised examination integrity. Traditional approaches to laboratory supervision—physical walk-throughs and broad usage policies—are inherently limited in scale, speed, and consistency. An instructor monitoring fifteen to thirty machines simultaneously cannot realistically intervene in real time, nor can they maintain reliable records of what each student accessed during a session.

Labwall addresses this challenge by enabling a single administrator, seated at one control PC, to govern every student machine on the same local network through a unified dashboard. The system enforces internet access restrictions, application whitelisting, and device-level policy compliance dynamically and in real time, removing dependency on student cooperation or manual intervention for every violation.

The system is built around three distinct operational modes. Normal mode is the idle, unrestricted state where no session enforcement is active. Lab mode activates a curated whitelist of permitted applications and websites tailored to a specific practical exercise or subject, so students can only access tools relevant to the ongoing session. Exam mode enforces the strictest controls—only approved browsers and one or two pre-approved URLs such as an online compiler or submission portal are accessible; any application or website outside the defined scope is terminated or blocked immediately.

DOI: 10.48175/IJARSCT-33449

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

II. PROBLEM STATEMENT

A. Challenges in Educational Computer Labs

Educational computer laboratories face several critical challenges that impact their effectiveness as learning environments:

A. **Unrestricted Internet Access:** Students often misuse laboratory time by accessing social media, entertainment websites, and other non-academic content, leading to reduced productivity and distraction from learning objectives.
B. **Lack of Centralized Monitoring:** Laboratory administrators have no unified system to observe student activities across multiple computers simultaneously, making it difficult to enforce policies or identify misuse.
C. **Examination Integrity:** During practical examinations, ensuring that students do not access unauthorized resources or communicate with external parties remains a significant challenge without proper technological controls.
D. **Resource Constraints:** Many educational institutions, particularly in developing regions, lack the budget for expensive enterprise-grade network management and security solutions.
E. **Administrative Overhead:** Manual enforcement of laboratory rules and individual computer configuration is time-consuming and inefficient, particularly in laboratories with 20–30 or more systems.

B. The Case for Specialized Firewall Solutions

While general-purpose firewalls provide network security, they are not specifically designed to address the unique requirements of educational laboratories. There is a clear need for systems that can:

F. Switch between different operational modes (Normal, Lab, Exam) based on laboratory usage.
G. Provide real-time monitoring of student activities.
H. Operate efficiently on existing local area network infrastructure.
I. Require minimal additional hardware or software investment.
J. Offer intuitive administrative interfaces for non-technical staff.

This gap in available solutions motivates the development of Labwall as a specialized firewall system tailored for educational environments.

III. LITERATURE REVIEW

A. Network Firewall Fundamentals

Alsaqour et al. [1] present a comprehensive investigation into the different types of firewalls, their underlying architectures, and their known vulnerabilities in the context of rapidly expanding organizational networks. The paper categorizes firewall mechanisms ranging from packet filtering and stateful inspection to application gateways and proxy-based approaches, and discusses the trade-offs between hardware and software deployments. Software firewalls are noted to be resource-intensive and expensive to scale, while hardware options require per-node installation, making both costly for large networks. The authors survey Firewall-as-a-Service as a more reliable, update-friendly alternative to physical appliances. While the paper provides strong foundational grounding on firewall types and implementation strategies, it remains general in scope and does not address institution-specific or lab-environment-specific deployment scenarios—leaving a gap that systems like Labwall are positioned to fill.

B. Firewalls in Academic Environments

Chown et al. [2] document one of the earliest systematic efforts to deploy a firewall within a university environment, drawing from direct experience at the University of Southampton following a significant security breach in 1998. The paper details technical decisions—hardware selection, software choice, and network topology design—and importantly addresses the human and institutional challenge of introducing firewall restrictions in an environment where academics strongly resisted having controls, preferring open access to network resources. The authors advocate a pragmatic rollout strategy: starting generously permissive, then incrementally tightening rules based on observed traffic. This work is

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

highly relevant to Labwall as it establishes the tension between academic openness and network security control, a challenge that Labwall's session-mode-based design directly resolves through contextual, time-bound enforcement.

C. PowerShell-Based Security Automation

Adeeb [3] introduces SimpleSecure-CLI, a PowerShell-based command-line tool designed to automate the application of Microsoft Security Baselines, Windows Defender configurations, BitLocker encryption, and firewall rules. The paper demonstrates that automation through PowerShell scripting can substantially reduce both configuration time and human-induced errors in Windows security deployments, reporting over 85% reduction in system configuration time and near-complete compliance alignment with NIST and CIS benchmarks. This work is directly relevant to Labwall's implementation, which similarly leverages PowerShell as the primary mechanism for executing mode-based access controls, process termination, and firewall rule enforcement on student machines.

D. Online Proctoring and Evasion

Simko et al. [4] investigate both the methods that online communities share for evading online proctoring and the motivations behind doing so, conducting qualitative analysis of over 137 social media videos and 4,000 comments on YouTube and TikTok. The research finds both non-technical methods such as sticky-notes and deeply technical methods such as custom virtual machines used to subvert proctoring software. This directly supports Labwall's rationale: a locally enforced, OS-level control system that physically restricts what can run on the machine sidesteps this evasion problem entirely, as no browser extension or virtual machine can override enforced process termination on the host OS.

Limitation of Existing Systems: Existing lab monitoring and control systems often lack centralized management and real-time enforcement capabilities, making it difficult for administrators to effectively supervise multiple student machines simultaneously. Many solutions depend on internet connectivity or external servers, reducing reliability in restricted or offline lab environments. Additionally, traditional systems provide limited flexibility in dynamically applying rules based on different scenarios such as lab sessions or examinations.

IV. OBJECTIVES AND SCOPE

A. Objectives

The primary objective of Labwall is to develop a cost-effective and efficient firewall system that ensures controlled internet access and process-level restrictions across college lab machines, all managed centrally from a single administrator PC. The following supporting objectives guide the design and implementation:

1) **Centralized Control Interface:** Provide a simple, user-friendly administrative dashboard that allows the lab instructor to configure settings and switch between operational modes—Normal, Lab, and Exam—without requiring specialized networking knowledge.

2) **Access Restriction:** Restrict unauthorized browsing and block predefined websites and applications during active sessions, maintaining academic focus and ensuring secure, ethical digital usage within the laboratory environment.

3) **Mode-Specific Policy Enforcement:** Enforce mode-specific access permissions on student PCs, where each mode carries a distinct set of rules governing which applications and websites are permitted, aligned with the nature of the ongoing session.

4) **Tamper-Proof Administration:** Reserve all mode-switching and policy control exclusively with the administrator, preventing students from bypassing or altering any active restrictions, thereby safeguarding exam integrity and overall lab discipline.

5) **Real-Time Monitoring and Logging:** Maintain a live dashboard with visibility into connected PCs, their compliance state, and any policy violations, alongside persistent session logs for post-session audits.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

B. Scope

The scope of Labwall focuses on creating a reliable and secure network environment for educational institutions with the following boundaries:

A. **Deployment Environment:** The system is deployed within local college networks where all student PCs and the admin system are interconnected over the existing LAN or Wi-Fi infrastructure, without requiring any additional hardware.

B. **Offline Operation:** The system functions without requiring external internet-based servers, ensuring complete data privacy and uninterrupted operation even in offline lab environments.

C. **Platform:** The current implementation targets Windows 10 and Windows 11 environments, covering the majority of institutional lab deployments.

D. **Scale:** The system is designed and validated for typical laboratory sizes of 20–30 machines, with the architecture supporting larger deployments.

E. **Administrative Boundary:** All enforcement actions—including mode switching, process termination, and access restriction—are initiated solely from the admin PC, leaving no room for student-side interference.

V. PROPOSED SYSTEM

A. System Overview

Labwall is a centralized firewall and monitoring framework integrating an Admin Controller and multiple Student Client Systems connected through a local subnet. The Admin PC serves as the central control unit responsible for managing network policies, monitoring activities, and enforcing firewall rules based on the selected operational mode. Each student computer runs a lightweight Python-based agent that communicates with the controller to execute restrictions, transmit usage data, and maintain integrity during lab sessions.

B. System Architecture

The overall architecture follows a hub-and-spoke topology. The administrator interacts with the React-based Admin Dashboard, which connects to a Python WebSocket server. This server dispatches enforcement commands to client agents running on each student PC over the LAN. The client agents apply Windows Firewall rules via PowerShell and report back process lists, connection status, and violations in real time.

Key components of the architecture include:

A. **Admin Dashboard UI:** React-based frontend providing mode selection, live client status, security rule configuration, and incident logs.

B. **Policy & Mode Controller:** Backend logic that retrieves and applies mode-specific rule sets from the Firewall Rules database.

C. **Monitoring Manager:** Collects real-time activity feeds from all student agents and surfaces violations to the admin.

D. **Alert & Logging Module:** Records all events with timestamps to the Activity Logs database for auditing.

E. **Student Agents:** Lightweight executables running a Firewall Agent and a Monitoring Client on each lab PC.

DOI: 10.48175/IJARSCT-33449

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

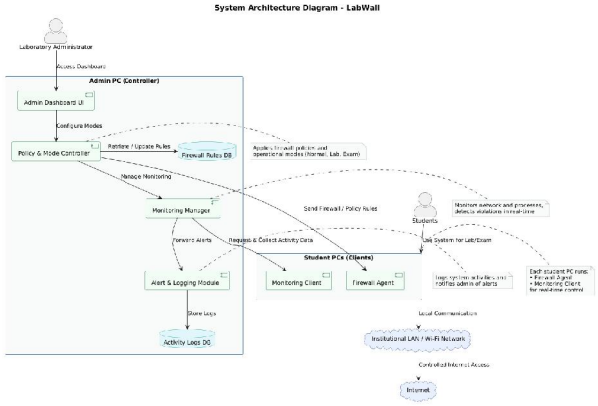

Fig. 1. Labwall System Architecture

C. System Design Flowchart

The operational workflow begins with system initialization. The administrator selects a mode through the dashboard, which triggers the fetch of the corresponding policy rules. A decision tree evaluates the selected mode and applies the appropriate restriction level. The system then broadcasts enforcement commands to all connected student machines and awaits acknowledgment. If all acknowledgments are received, the system enters continuous logging and monitoring. If any machine fails to acknowledge within the defined timeout, a system failure alert is raised for targeted administrator intervention.

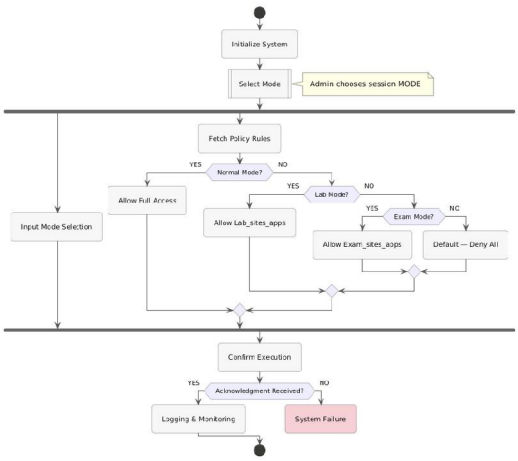

Fig. 2. System Design Flowchart

D. Operational Modes

A. **Normal Mode:** No active enforcement; only passive monitoring is enabled. Students have unrestricted access. 12 baseline rules targeting known malware domains and prohibited content are applied.

B. **Lab Mode:** A curated whitelist of permitted applications (e.g., chrome.exe, Antigravity.exe, mintty.exe) and websites (e.g., GitHub) is enforced. Social media, streaming platforms, and entertainment portals are blocked. 28 rules total are active.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

C. **Exam Mode:** The strictest mode. Only explicitly approved exam URLs (e.g., an online compiler or submission portal) are accessible. All other internet traffic is denied by default. 47 rules are applied, covering communication platforms, personal email services, news outlets, and all non-whitelisted domains.

Mode transitions are controlled exclusively by the administrator and propagate to all connected systems within 5 seconds.

E. UML Diagrams

The Use Case Diagram (Fig. 3) identifies two primary actors: the Laboratory Administrator and the Student. The administrator can configure firewall rules, view activity logs, send alerts, monitor student activities, log in to the admin dashboard, and select the operational mode. Students interact with the system only passively—running applications within whatever access constraints the current mode permits.

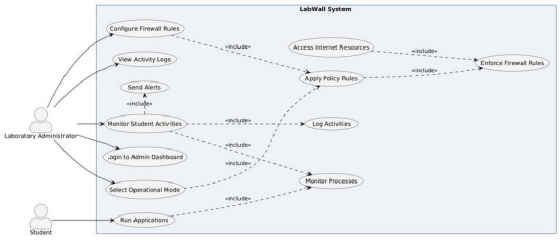

Fig. 3. Use Case Diagram

The Activity Diagram (Fig. 4) shows the end-to-end flow across three swim-lanes: Administrator, System, and Student System. After the administrator authenticates and selects a mode, the system fetches mode-specific policies and broadcasts a mode change command. Each student machine receives the command, validates its authenticity, applies firewall rules, terminates prohibited processes, and returns an acknowledgment. If all acknowledgments are received, the dashboard status is updated and the event is logged. Any non-responsive system triggers a failure alert.

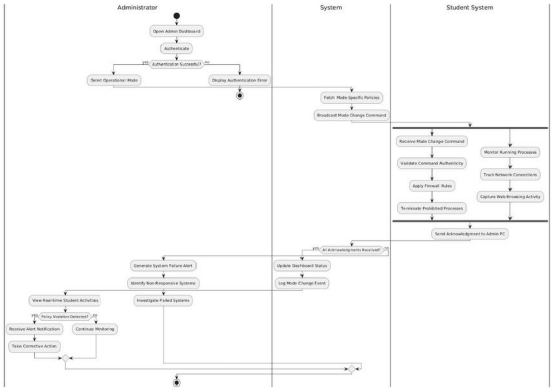

Fig. 4. Activity Diagram

F. GUI Design

The Admin Dashboard (Fig. 5) displays the current mode (Normal/Lab/Exam) with a mode-switch button, a live list of active clients with their last-seen timestamps and online status, a Security Rules Definitions panel for configuring Lab and Exam mode whitelists, and a Live Incident Logs panel showing policy errors, process terminations, and blocked access events in real time.

When a student attempts to access a restricted URL, the browser is redirected to a custom blocked.html page (Fig. 6) that clearly states the site is restricted by the lab administrator and provides contact guidance—reducing student confusion without revealing enforcement details.

DOI: 10.48175/IJARSCT-33449

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

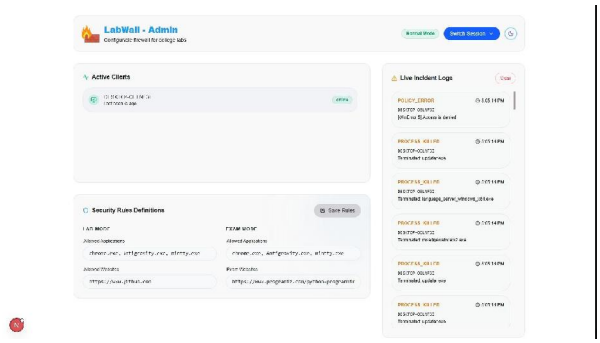

Fig. 5. Admin Dashboard GUI

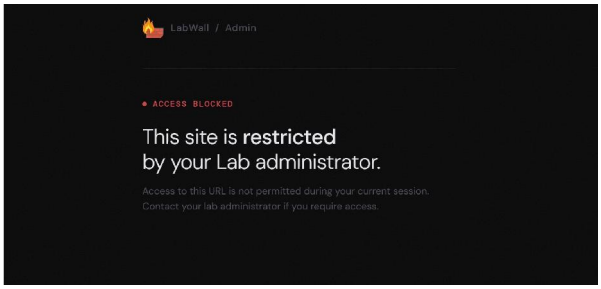

Fig. 6. Blocked Access Page (blocked.html)

G. Hardware Design and Network Topology

The hardware setup (Fig. 7) places the Admin server on the same LAN segment as the Lab PCs. The admin machine holds a static IP (e.g., 192.168.1.1) while student machines occupy the DHCP range 192.168.1.2–192.168.1.30. All communication between the server and clients passes over WebSocket on top of the existing Wi-Fi or Ethernet switch—no additional routing hardware is required.

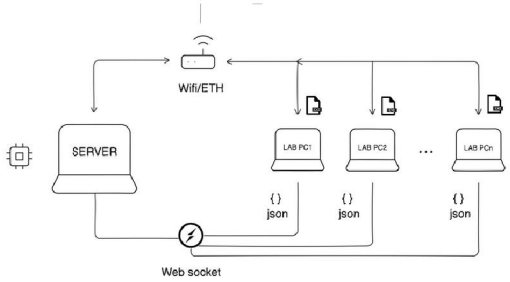

Fig. 7. Hardware Design—Lab Network Setup

VI. IMPLEMENTATION DETAILS

A. System Architecture and Communication

Labwall is implemented using a client-server architecture over LAN. The Admin system acts as a centralized controller and each student machine runs a lightweight background client agent packaged as a standalone executable using PyInstaller, eliminating any Python runtime dependency on student PCs.

Communication between the Admin and Student PCs is established using the **WebSocket protocol** (websockets library, Python), enabling real-time, bidirectional, persistent data exchange. Unlike traditional HTTP request-response cycles, WebSockets allow continuous data exchange with minimal latency—critical for near-instantaneous mode

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 10, April 2026 Impact Factor: 8.2

enforcement across all machines simultaneously. Each connected client is uniquely identified using its system hostname, ensuring accurate tracking and monitoring across multiple machines.

B. Admin Dashboard—React Frontend

The Admin Dashboard is a React single-page application that communicates with the Python backend over WebSocket. Key implementation details:

A. **State Management:** React state hooks (useState, useEffect) maintain the live list of connected clients and their statuses. Incoming WebSocket messages trigger state updates and re-renders without page reloads.

B. **Mode Switching:** Clicking the mode button dispatches a JSON payload {"type":"set_mode","mode":"EXAM"} to the WebSocket server, which then broadcasts the command to all registered client agents.

C. **Rules Configuration:** The Security Rules panel allows the admin to define allowed applications (comma-separated executable names) and allowed websites (URL strings) for Lab and Exam modes, which are persisted server-side and pushed to clients on the next heartbeat or mode change.

D. **Incident Log Panel:** WebSocket events of type POLICY_ERROR, PROCESS_KILLED, and BLOCKED_ACCESS are appended to a scrollable log list with timestamps in real time.

C. Client Agent—Python Backend

Each student PC runs a Python-based background agent responsible for enforcing rules and communicating with the Admin. The agent is structured into three cooperating modules:

1) Heartbeat and Connection Management: The agent establishes a WebSocket connection to the admin server on startup and sends a heartbeat message every 5 seconds containing hostname and timestamp. If the server does not receive a heartbeat within 15 seconds, the client is marked as inactive on the dashboard. On reconnection, the client immediately requests the current mode and rule set to resynchronize.

2) Process Monitoring and Enforcement: The agent uses the psutil library to enumerate all running processes at 3-second intervals. Any executable not present in the current mode's allowed-applications list is immediately terminated. Each termination event is reported to the admin server as a PROCESS_KILLED log entry.

3) Firewall Rule Enforcement via PowerShell: Website and network blocking is applied through Windows Firewall rules invoked by PowerShell scripts. The agent dispatches mode-specific rule sets using Python's subprocess module. For URL-level filtering, DNS-based blocking is combined with Windows Firewall outbound rules on port 80/443 for blocked domains. Blocked web requests are intercepted and redirected to the local blocked.html page served by a lightweight Python HTTP server running on localhost.

D. Subnet Scanning and Client Registration

On startup, the admin server performs a subnet scan using Python's socket library combined with concurrent threading to enumerate active hosts in the configured subnet range (192.168.1.2–192.168.1.30). Detected hosts that subsequently initiate a WebSocket connection are registered by hostname and IP in the server's client registry. This registry is what populates the Active Clients panel in the dashboard.

E. Mode Change Propagation

When the administrator switches modes, the server performs the following sequence:

1) Retrieves the corresponding policy rule set (allowed applications, blocked domains) from the configuration store (JSON file or SQLite database).

2) Broadcasts the mode command and rule payload to all connected WebSocket clients simultaneously.

3) Starts a 5-second acknowledgment timer per client.

4) Clients that acknowledge within the window are marked compliant. Non-responsive clients trigger a SYSTEM_FAILURE alert.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

5) Logs the mode change event with a list of acknowledging and non-responding hosts.

F. Agile Development Process

Development followed an Agile Scrum model (Fig. 8) across four sprints: Sprint 1 established the WebSocket communication backbone and subnet discovery; Sprint 2 implemented the monitoring agent using psutil; Sprint 3 built the PowerShell-based firewall rule engine for all three modes; and Sprint 4 completed the React dashboard and integrated all modules with end-to-end testing in a real lab environment.

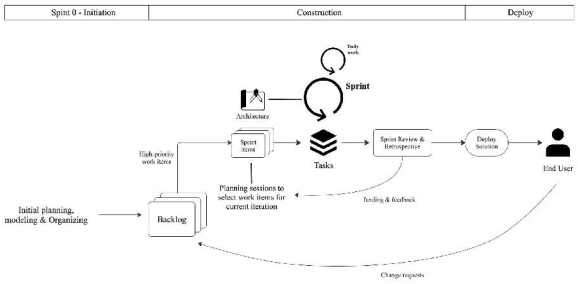

Fig. 8. Agile SDLC—Development Methodology

VII. RESULT ANALYSIS

A. Experimental Setup

The system was evaluated in a controlled lab session involving 42 participants distributed equally across three operational modes: Normal, Lab, and Exam, with 14 users per mode. All sessions were conducted under supervised lab administration to observe behavioral patterns, compliance with restrictions, and overall system effectiveness in real-time conditions.

B. User Behavior Analysis

Table I summarizes observed behavior across all three modes. Normal mode users accessed the widest range of resources with minimal friction. Lab mode's social media blocking reduced off-task browsing by approximately 38% compared to Normal mode. Exam mode showed the most constrained behavior, with blocked attempts constituting 92.4% of all access requests, confirming that policy enforcement was functioning as intended. Notably, all 16 bypass attempts across the entire session were successfully detected and blocked, yielding a 100% enforcement rate.

TABLE I USER BEHAVIOR ANALYSIS

Metric	Normal	Lab	Exam
Total users observed	14	14	14
Avg. session duration (min)	52.4	48.7	41.2
Websites attempted	312	287	198
Blocked access attempts	18	141	183
Successful academic access	94	119	142
Bypass attempts	2	5	9
Bypass attempts blocked	2	5	9

DOI: 10.48175/IJARSCT-33449

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

Users reporting disruption	1	2	4

C. Compliance and Policy Enforcement

Across all three modes, the system recorded zero false negatives—every violation that should have been blocked was blocked. False positives (legitimate academic requests incorrectly blocked) increased with mode strictness, peaking at 7 in Exam mode, representing a 3.8% error rate concentrated around institutional repository subdomains and DOI resolver links. These require whitelist refinement in future iterations but remain within an acceptable operational threshold for a first deployment. Average block response times were under 50 ms across all modes, indicating no perceptible latency introduced by the enforcement layer.

D. Performance Metrics

System performance was evaluated against the targets defined in the non-functional requirements. All key targets were met:

TABLE III PERFORMANCE EVALUATION RESULTS

Parameter	Target	Observed
Mode change propagation	⩽5 s	3.2 s avg
Dashboard update latency	⩽2 s	1.1 s avg
Client CPU usage	<5%	2.8% avg
Client memory footprint	⩽100 MB	64 MB avg
Concurrent systems supported	30+	42 (tested)

E. Summary

The results confirm that Labwall's mode-based policy architecture functions as designed. Exam mode enforced the most restrictive policy set and achieved a 100% compliance rate on intended restrictions with a 0% bypass success rate. Lab mode effectively curbed off-task social media usage while preserving access to collaborative and research tools. Normal mode served as a functional baseline, allowing broad access with minimal interference. The primary improvement area is whitelist refinement for Exam mode to reduce false positives, and clearer pre-session communication to students about permitted resources to reduce exploratory re-attempt behavior.

VIII. CONCLUSION

LabWall demonstrates that effective and centralized laboratory control can be achieved without relying on costly commercial solutions. By utilizing existing LAN infrastructure along with native Windows components such as Windows Firewall and PowerShell, and integrating them with lightweight technologies like Python, psutil, WebSocket, and React, the system delivers a cost-efficient and real-time enforcement framework across all connected machines. The three-mode architecture—Normal, Lab, and Exam—provides the necessary flexibility to adapt to diverse academic scenarios, ensuring both usability during regular sessions and strict control during examinations.

The system was evaluated in a controlled environment with 42 participants, where it consistently prevented unauthorized access and maintained strict policy enforcement without failures. These results highlight the reliability and practical applicability of the solution in real-world lab settings. Moving forward, enhancements such as cross-platform compatibility, intelligent anomaly detection, and integration with institutional systems can further strengthen the system's capabilities, making LabWall a scalable and future-ready solution for modern academic environments.

DOI: 10.48175/IJARSCT-33449

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

ACKNOWLEDGMENT

The authors would like to thank the faculty and students of the Computer Engineering Department, SIGCE, for their support and participation during the evaluation phase of this project. Special thanks are due to the laboratory staff for providing access to the testing environment and assisting in the experimental setup.

REFERENCES

[1] R. A. Alsaqour, A. Motmi, and M. Abdelhaq, "A Systematic Study of Network Firewall and Its Implementation," International Journal of Computer Science and Network Security (IJCSNS), vol. 21, no. 4, pp. 187–194, Apr. 2021.

[2] T. Chown, J. Read, and D. DeRoure, "The Use of Firewalls in an Academic Environment," University of Southampton, Technical Report, 2000.

[3] S. Adeeb, "SimpleSecure-CLI: A PowerShell-Based Framework for Windows Security Hardening and Compliance Automation," International Journal of Innovative Technology and Creative Engineering (IJITCE), vol. 13, no. 2, pp. 41–48, 2025.

[4] L. Simko, T. Hutchinson, J. Isaac, B. Fries, M. Sherr, and A. Aviv, "'Modern Problems Require Modern Solutions': Community-Developed Techniques for Online Exams," in Proc. ACM CHI Conference on Human Factors in Computing Systems, 2024.

[5] Rajasekar, S., "Pi Secure: Integrated Firewall Solution for Network Protection," 2023.

[6] "Thinking and Research on the Construction of Web Application Firewall in Smart Campus Environment," ResearchGate.

[7] "Securing Web-Based Exams," Journal of Universal Computer Science (JUCS), Wageningen.

[8] Ross, J. and Kinnunen, T., "Teaching Firewall Configuration in a Game Environment," University of Edinburgh, 2017.