Volume 11, Issue 4, April – 2026 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr1306

Smart Infrastructure Management

Pragati Patil[1]; Siddhi Phatkare[2]; Pranali Rane[3]; Shreya Janathe[4]

[1]Pragati Patil, Vidyavardhini's College of Engineering and Technology
[2]Siddhi Phatkare, Vidyavardhini's College of Engineering and Technology
[3]Pranali Rane, Vidyavardhini's College of Engineering and Technology
[4]Shreya Janathe, Vidyavardhini's College of Engineering and Technology

Publication Date: 2026/05/04

Abstract: Managing distributed computer systems in academic and corporate environments has become increasingly demanding as infrastructures grow in both size and complexity. Traditional administration practices that depend heavily on manual intervention are slow, difficult to scale, and prone to mistakes. To address these limitations, this paper presents Smart Infrastructure Management, implemented as a centralized, web-based platform that streamlines and automates routine administrative tasks. The system integrates Ansible-based automation with a straightforward administrative dashboard, enabling administrators to manage multiple systems from a single interface. Communication with Windows machines is handled through Windows Remote Management, providing agentless operation and eliminating the need for additional software installation on target hosts. Key capabilities include dynamic inventory management, execution of predefined playbooks, and detailed logging of administrative actions. The system was evaluated across multiple laboratory environments, where it significantly reduced manual effort and improved configuration consistency across systems. The results suggest that the solution is scalable and can be extended to meet future requirements. Possible extensions include support for additional operating systems, AI-assisted predictive maintenance, real-time monitoring, and cloud-based deployment.

Keywords: *Ansible Automation, Centralized Dashboard, Infrastructure Management, IT Automation, Lab Management, Playbooks, Software Deployment, WinRM.*

How to Cite: Pragati Patil; Siddhi Phatkare; Pranali Rane; Shreya Janathe (2026) Smart Infrastructure Management. *International Journal of Innovative Science and Research Technology*, 11(4), 3137-3154. https://doi.org/10.38124/ijisrt/26apr1306

I. INTRODUCTION

In IT administration, maintaining multiple machines in a reliable and secure state is a challenging task. Whether the environment is a university computer lab, a corporate office network, or a research facility, the day-to-day requirements of software deployment, security patching, configuration changes, and system monitoring can escalate quickly. When administrators manage all of this manually, critical details are often overlooked. For instance, a single machine running an outdated version or a missed security update can lead to minor discrepancies that can develop into serious operational issues.

Manual system administration is not scalable. Practices that are effective for managing a small number of machines become unsustainable as the scale increases to dozens or hundreds of systems. These challenges are practical real risks, including inconsistent software environments that create compatibility problems, unpatched systems that become security liabilities, and the sheer time lost to repetitive tasks pulls administrators away from work that truly requires human judgment. There is a need for more efficient tools, and system administrators require reliable solutions to do their jobs.

Smart Infrastructure Management is proposed to address this need. At its core, it is designed as a centralized, web-based platform that brings the power of Ansible automation within reach of anyone responsible for managing a networked environment, not just those with deep command-line or DevOps expertise. Ansible has long been respected in the infrastructure world for its flexibility and agentless architecture, but its learning curve can be a barrier for many teams.

This system simplifies operations by integrating Ansible into an intuitive dashboard, enabling routine tasks without coding.

The main contributions of this work are summarized as follows:

- The system demonstrates several key technical contributions that together make it a well-rounded solution for real-world infrastructure management.
- A fully functional, centralized web platform was built from the ground up, providing administrators with a centralized point of control for their entire infrastructure, eliminating reliance on fragmented tools and processes.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr1306

- Ansible was integrated directly into the platform's core, connecting its powerful automation engine to an intuitive, interactive dashboard in a manner that does not require extensive technical expertise for routine operations.
- Remote system management was implemented using the Windows Remote Management (WinRM) protocol, which removes the need to install additional software agents on managed machines. This approach is advantageous in environments where deploying and maintaining agents would introduce additional overhead.
- The platform supports dynamic inventory creation, allowing administrators to avoid reliance on static machine lists. Systems can be organized, grouped, and updated as the environment evolves, ensuring that management remains accurate and up to date.
- Comprehensive logging and error reporting were built into the system from the outset, as an integral part of the system design. Every action performed through the platform is tracked, and failures are reported with sufficient detail to be useful during troubleshooting.
- Finally, the system was evaluated in real lab environments rather than purely theoretical settings. The results showed measurable improvements in the execution time of administrative tasks and a reduction in the manual effort required for system maintenance.

II. LITERATURE REVIEW

Several research papers were reviewed to understand existing approaches in automation and infrastructure management. The key findings from these studies are summarized below.

Jones [1] provides a clear introduction to Linux-based automation using Ansible, emphasizing its agentless nature and ease of setup. The study explains how multiple systems can be managed without installing additional software on each machine, which significantly simplifies deployment. It also discusses important features such as playbooks, YAML-based configurations, and parallel task execution. These aspects help reduce manual work and maintain consistency across systems. This work significantly influenced the design of the proposed system, where Ansible is used as the main automation engine for centralized system control.

Lee and Lee [3] examine how automation and monitoring can be combined in smart environments, particularly in smart building systems. Their work shows that integrating real-time monitoring with automated control improves overall efficiency and system responsiveness. They also highlight the importance of having a single interface where administrators can both monitor and control systems. This idea is reflected in the proposed platform through its dashboard, which brings together monitoring data and automation features in one place.

Kumar and Singh [5] present an IoT-based infrastructure monitoring system aimed at institutional use. Their research focuses on how centralized monitoring helps improve system performance, reduce downtime, and make better use of available resources. By collecting real-time data from connected devices, administrators can track system health more effectively. This approach aligns with the monitoring component of the proposed system, which continuously gathers and displays system information to help administrators take timely action.

Sharma [7] evaluates the performance of Ansible and compares it with other configuration management tools. The study finds that Ansible performs well in terms of efficiency, scalability, and ease of use. Its agentless architecture and simple configuration format make it suitable for managing large-scale environments with minimal overhead. These findings support the decision to use Ansible in this work, as it can handle multiple systems efficiently while keeping the setup straightforward.

Srivastava [9] describes a real-time monitoring and control system for lab networks using Flask and MongoDB, centered on a web-based dashboard enabling remote monitoring and control. The study illustrates how combining backend services with an interactive interface enhances the practicality of infrastructure management. This is consistent with the proposed system, which integrates a web-based frontend with backend services to deliver an accessible management platform.

Overall, the reviewed studies highlight the growing importance of automation, centralized monitoring, and user-friendly interfaces in modern infrastructure management. Together, these studies support the approach adopted in this work and show that combining tools like Ansible with web technologies can result in a scalable and efficient solution for managing complex IT environments.

III. METHODOLOGY

Building a platform that manages multiple systems across a network requires a well-defined architectural foundation. The Smart Infrastructure Management system is designed around a three-tier architecture using the MERN stack, combining a React frontend, a Node.js and Express.js backend, and MongoDB for data storage. On top of this, Ansible sits as an additional layer dedicated entirely to carrying out automation tasks on target Windows machines. Each component of the system has a clearly defined responsibility, which makes the overall platform easier to maintain, scale, and improve over time. Because the layers operate independently, updating or modifying one section rarely introduces issues in other components, thereby enhancing long-term flexibility to the system.

The frontend is built with React and serves as the primary user interface through which administrators interact with the system. It is designed to be clean, responsive, and easy to navigate, allowing users to check device status, initiate automation tasks, and manage configurations across multiple machines without friction. Communication with the backend happens through RESTful APIs over HTTP, which provides a reliable and consistent channel for sending and receiving data. The React component-based structure also enables dynamic updates, so the interface reflects changes in near real-time

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr1306

rather than requiring full page reloads, making the experience feel considerably more fluid and modern.

The backend is where the core logic of the application is implemented. Built with Node.js and Express.js, it receives incoming requests from the frontend, processes them, and coordinates with the other components to execute required operations. One of its more significant responsibilities is generating Ansible inventory files and playbooks dynamically, based on the configurations defined by the administrator through the interface. This means users can trigger complex multi-step operations without requiring manual script development. Once the necessary files are prepared, the backend launches Ansible as a child process, handing off execution to the automation layer and waiting for the results to come back.

Ansible handles everything related to executing tasks on remote machines. Its agentless design is a practical advantage here; no additional software installation or maintenance is required on the client's systems themselves. It connects to Windows machines using the WinRM protocol and carries out the required operations, whether that involves installing software, applying configurations, or running system checks. Once a task is complete, Ansible returns a detailed report covering what succeeded, what failed, and any additional information that might be relevant. This output is passed back to the backend for processing and storage.

MongoDB handles all persistent data in the system, including device records, user information, task configurations, and the full history of execution logs. Its flexible, document-oriented structure suits this kind of application well, since the data produced by automation workflows can vary significantly in shape and detail. Every operation performed through the platform is recorded, giving administrators a reliable reference for tracking activity, troubleshooting unexpected behavior, and reviewing system performance over time.

The overall data flow follows a request-response pattern. When an administrator acts on the frontend, a request is sent to the backend, which processes it and triggers Ansible for automation tasks when required. For operations with longer execution times, the frontend periodically retrieves progress updates from the backend, eliminating the need for continuous manual monitoring. This approach ensures a responsive and transparent user experience.

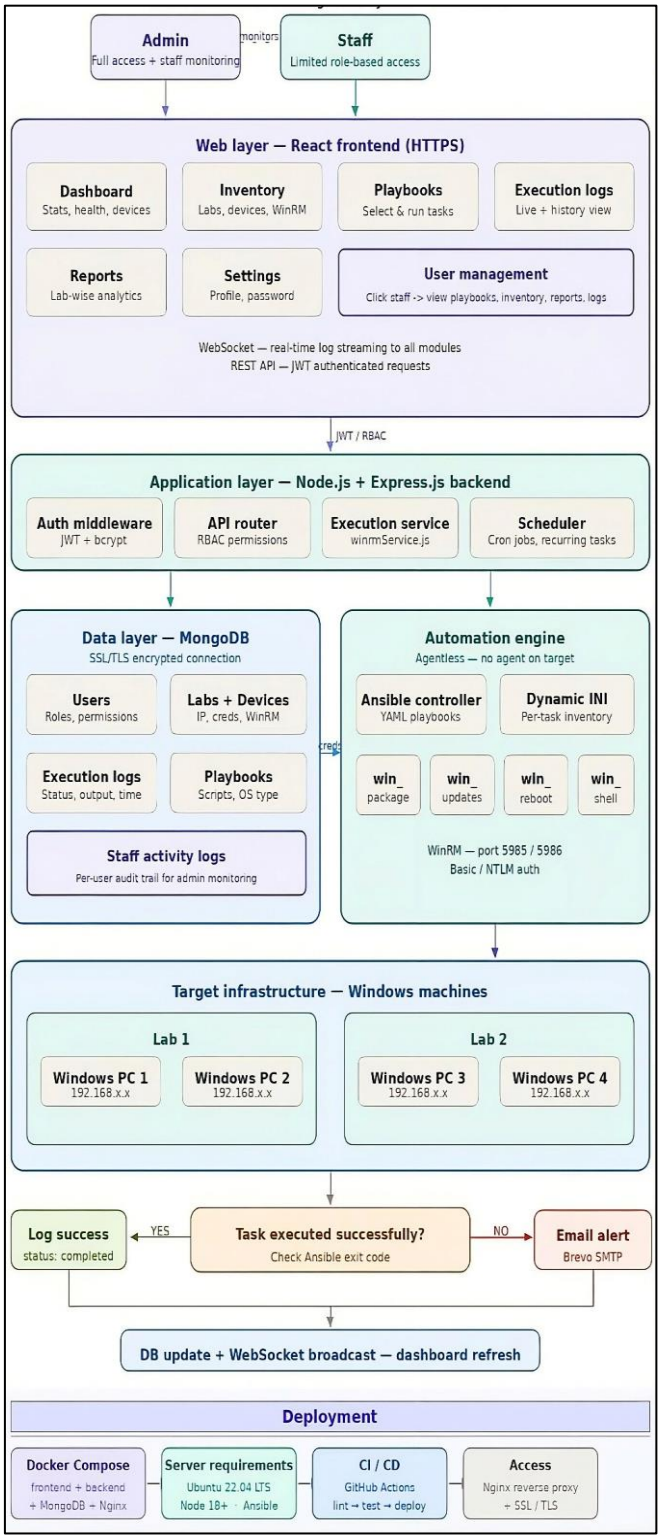

Fig 1 System Architecture

➢ *Data Acquisition and Preprocessing*

In the Smart Infrastructure Management system, data acquisition begins with collecting essential information about target machines and the tasks administrators intend to perform. This information is primarily obtained through the web-based dashboard, where administrators provide details such as IP addresses, hostnames, operating system types, login credentials, and required software installations or configuration updates. In addition to manual input, the system

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr1306

verifies whether devices are online, reachable, and properly connected to the network before initiating any operation, ensuring that tasks are directed only to valid and accessible machines. All collected data is securely stored in the database for future use.

Before execution, the data undergoes a preprocessing stage to ensure accuracy and reliability. IP addresses and credentials are validated, duplicate entries are removed, and inactive or unreachable devices are filtered out. The processed data is then structured into a format compatible with Ansible. The backend converts this refined data into an inventory file and maps the requested operations to corresponding YAML playbooks. If any inconsistencies or missing inputs are detected, the system immediately notifies the administrator to resolve the issue prior to execution. This validation process significantly reduces runtime errors and enhances system reliability.

- *Communication Protocol*

For communication with remote Windows systems, the platform utilizes Windows Remote Management (WinRM). This protocol is based on Microsoft's implementation of the WS-Management standard, which provides a standardized mechanism for exchanging management data across networked systems. In this architecture, WinRM operates over HTTPS on port 5986 by default, ensuring that all transmitted data remains encrypted and secure.

A key advantage of WinRM is its compatibility with agentless management, aligning with Ansible's design principles. No additional software installation is required on the target machines. Authentication is handled using Kerberos in domain-based environments, while NTLM or Basic authentication is used in non-domain setups. Initial configuration of WinRM requires a one-time PowerShell setup on the target system; thereafter, all operations can be managed remotely through the platform without requiring direct system access. This approach ensures secure, centralized, and scalable administration of Windows hosts while maintaining compliance with enterprise security policies and standards across environments.

- *Technology Stack*

The frontend is developed using React.js and organized into dedicated modules, each handling a specific area of the system.

The Dashboard provides a real-time overview of all registered devices, showing briefly which ones are currently online and which are offline. Device availability is typically determined through a straightforward Ansible ping check. Alongside connectivity status, the Dashboard displays summary information including the total device count, how many are currently active, and a snapshot of the most recently executed task.

The Inventory module is where administrators manage the actual devices in the system. Machines are grouped into labs or departments, and each entry holds key details like IP address, username, and password. Adding new devices, updating existing records, or removing machines that are no longer in use can all be done directly from this module. Everything stored here feeds into the inventory files generated when tasks are executed.

The Playbooks module gives users access to a range of automation tasks, installing software, running system updates, restarting or shutting down machines, and removing applications. For installations, administrators simply provide the name of the software they want deployed. Once the request is submitted, it moves to the backend for processing without requiring the user to interact with Ansible directly.

The Logs module maintains a chronological record of every task that has been run through the system. Each entry captures when the task was executed, which devices were involved, what type of operation was performed, and whether it completed successfully. Detailed Ansible output is also available within each log entry, giving administrators the information they need to investigate anything that did not go as expected.

The Report module takes a broader look at system activity over time, drawing on log data and monitoring information to surface meaningful patterns. This includes task success rates, how frequently different operations are performed, device uptime trends, and overall usage patterns. Reports can be exported as CSV files, making them straightforward to use for documentation, audits, or further analysis in routine operational review cycles and governance.

The Settings module handles system-level configuration, allowing administrators to adjust options such as the default WinRM port and the file path to the Ansible executable.

The User Management module is an administrator-only feature that serves as the control centre for managing who can access the system and what they can do within it. As seen in the platform, administrators can view all registered users in a single table, including their name, email address, assigned role, department, status, and last login date. Each user's entry also comes with action controls for editing details, managing role assignments, or removing accounts entirely. From here, an administrator can oversee staff accounts, ensuring that roles are correctly assigned, access remains appropriate, and the overall user base stays organized and up to date. Staff members do not have access to this module at all; their experience is deliberately focused on the operational tasks within their own scope, keeping the interface clean, and their responsibilities clearly defined.

On the backend side, the system runs on Node.js with the Express.js framework, exposing a set of RESTful APIs that the frontend communicates with. Core endpoints cover retrieving and updating device information, adding or removing machines, triggering task execution, and pulling log data. When a task is initiated, the backend dynamically builds an Ansible inventory file using the relevant device records pulled from MongoDB, selects or constructs the appropriate YAML playbook for the operation, and launches Ansible through Node.js's child process module. All output produced during

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr1306

execution, including standard results and any error messages, is captured and written back to the database as part of the permanent execution log. The backend also exposes endpoints to view historical execution logs and filter them by device, task type, or time range, enabling detailed auditing and troubleshooting. Robust error handling and structured logging ensure failed Ansible runs are surfaced to the frontend with sufficient context for rapid diagnosis and resolution.

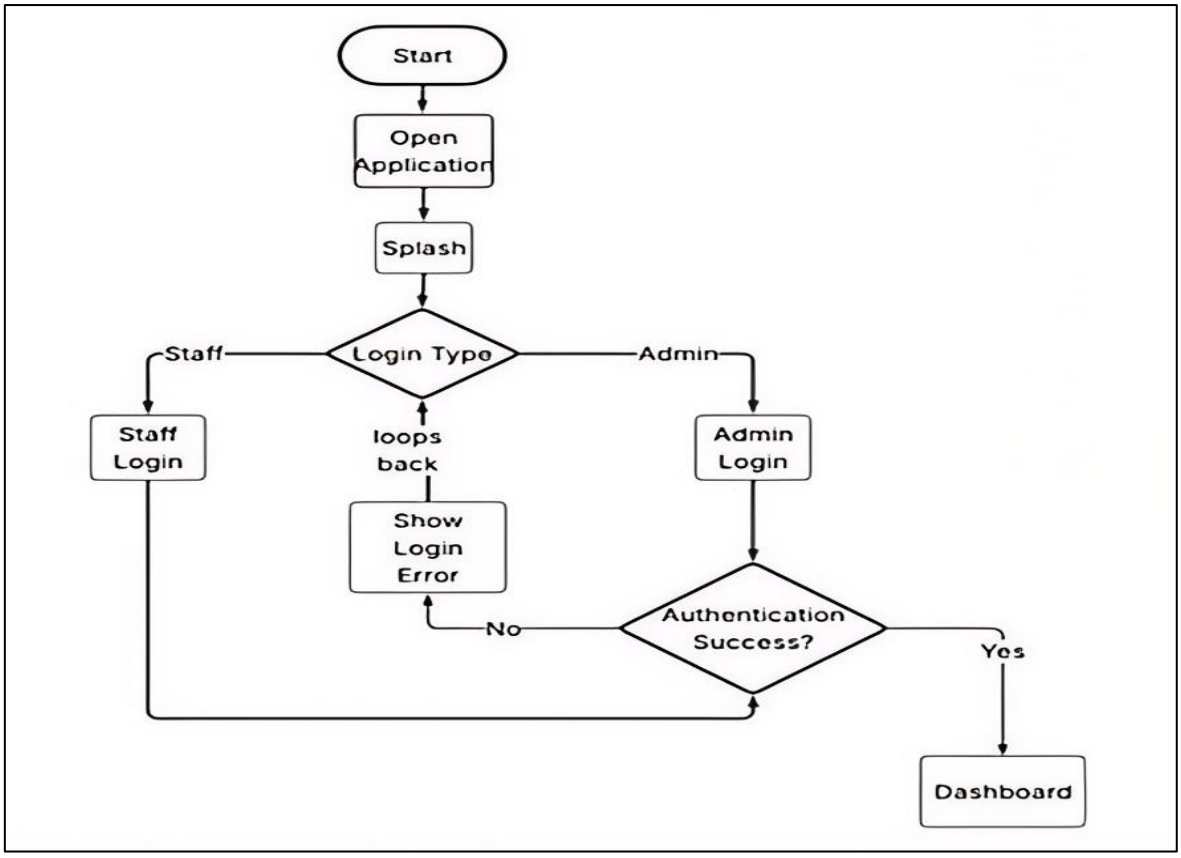

Fig 2 System Flow (a)

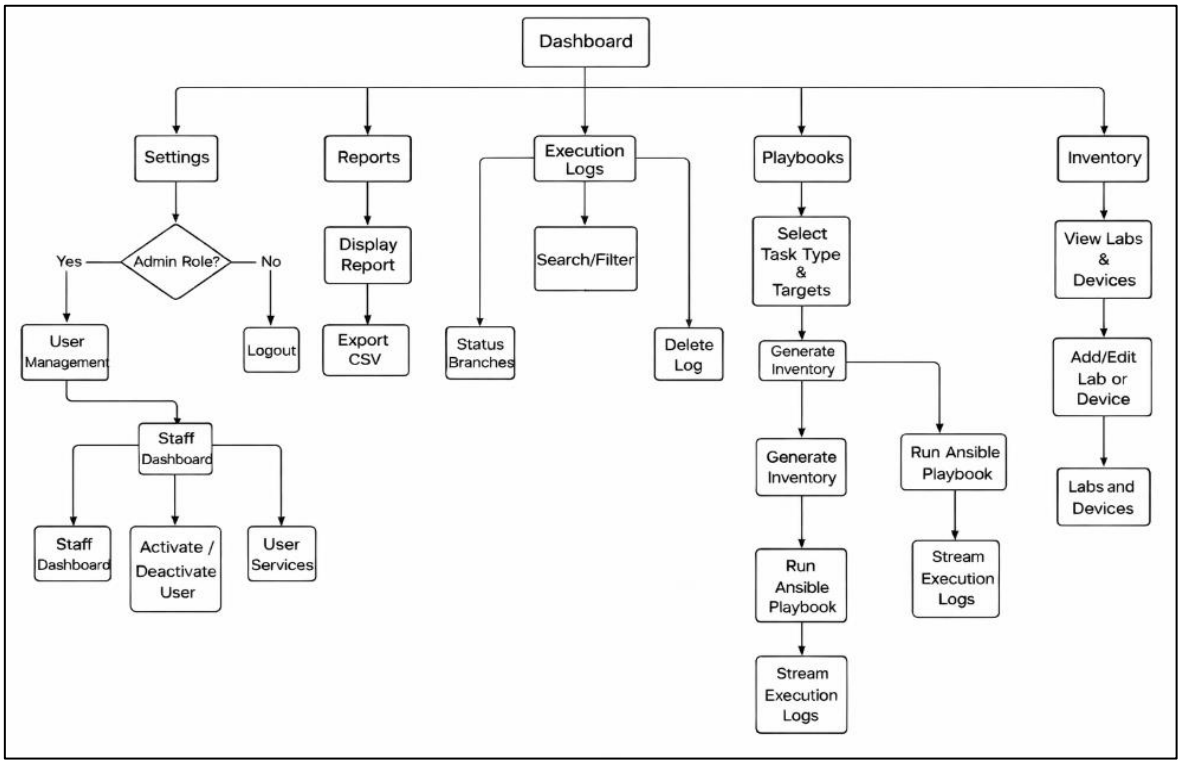

Fig 3 System Flow (b)

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr1306

IV. IMPLEMENTATION AND RESULTS

Setting up remote access to multiple Windows machines using IP addresses and login credentials went smoothly, with no significant obstacles encountered during the process. To confirm that communication between the controller and the target systems was working correctly, Ansible's ping module was used to run connectivity checks across all configured machines, and every properly set up device responded exactly as expected.

Where things did not go according to plan, the system handled it gracefully. Machines that were offline or had WinRM configured incorrectly were detected automatically and marked as unreachable rather than causing the entire process to stall. The relevant error details were captured and written into the execution logs, giving administrators a clear starting point for diagnosing what went wrong and getting those machines back into working order without much guesswork involved.

With connectivity confirmed, a range of remote operations was put through their paces to verify that the system's core functionality held up under real conditions. Software installation tasks were among the first to be tested. Using the appropriate Ansible modules, applications were selected through the interface and deployed to the target machines. By the time execution completed, the software was installed and ready to use on each of the selected systems, exactly the outcome that was expected.

System update tasks were tested next. Windows Update was triggered remotely through the platform, and once the process finished, the system reported back with the number of updates that had been applied. This gave administrators a straightforward way to confirm that machines were being kept current without needing to log into each one individually. Restart and shutdown operations rounded out the testing. Both commands were sent to target machines through the platform, and in each case, the machines responded correctly, restarting or powering down as instructed. The results across all of these tests pointed to a system that behaves reliably and delivers on what it sets out to do.

The system also showed strong performance when handling multiple machines at once. It maintained stability and consistency throughout the process. Tasks were applied the same way across all devices, which lowered the risk of configuration differences. Detailed logs and organized outputs made it easy to track and verify each operation. Overall, the platform was efficient, reliable, and well-suited for centralized remote management. It also scaled well as the number of concurrent tasks increased, without noticeable slowdowns. The clear separation of roles and permissions helped maintain control and accountability for each action taken. This combination of stability, transparency, and scalability makes the system a strong fit for large, distributed environments. It also provided consistent results across repeated runs, reinforcing confidence in its use for ongoing operational workflows.

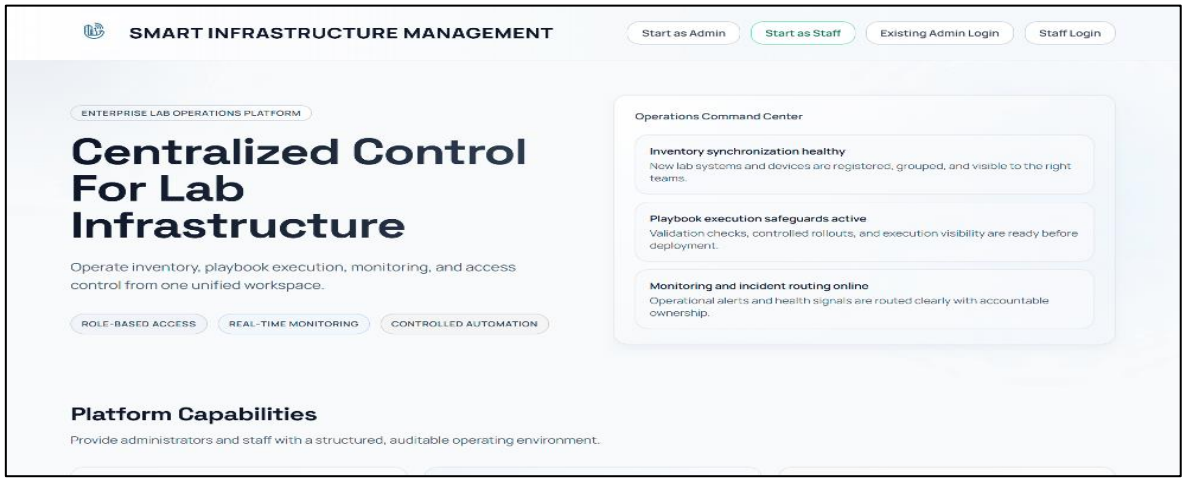

Fig 4 Splash Page

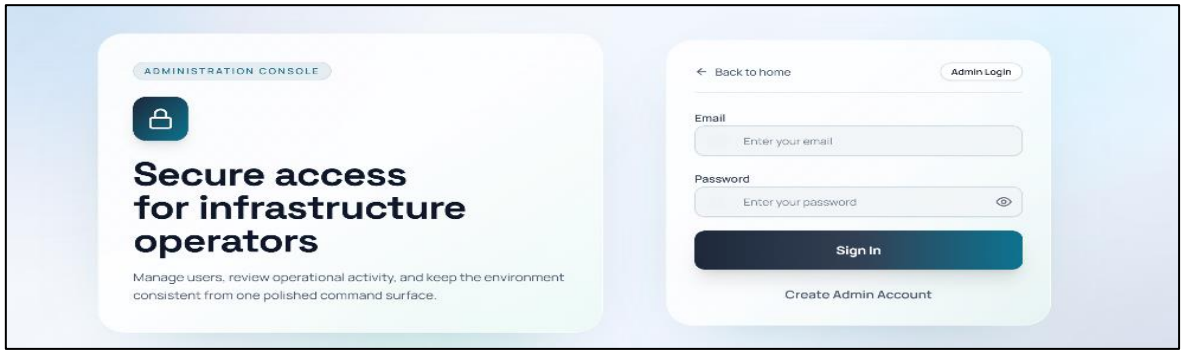

Fig 5 Admin Login

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr1306

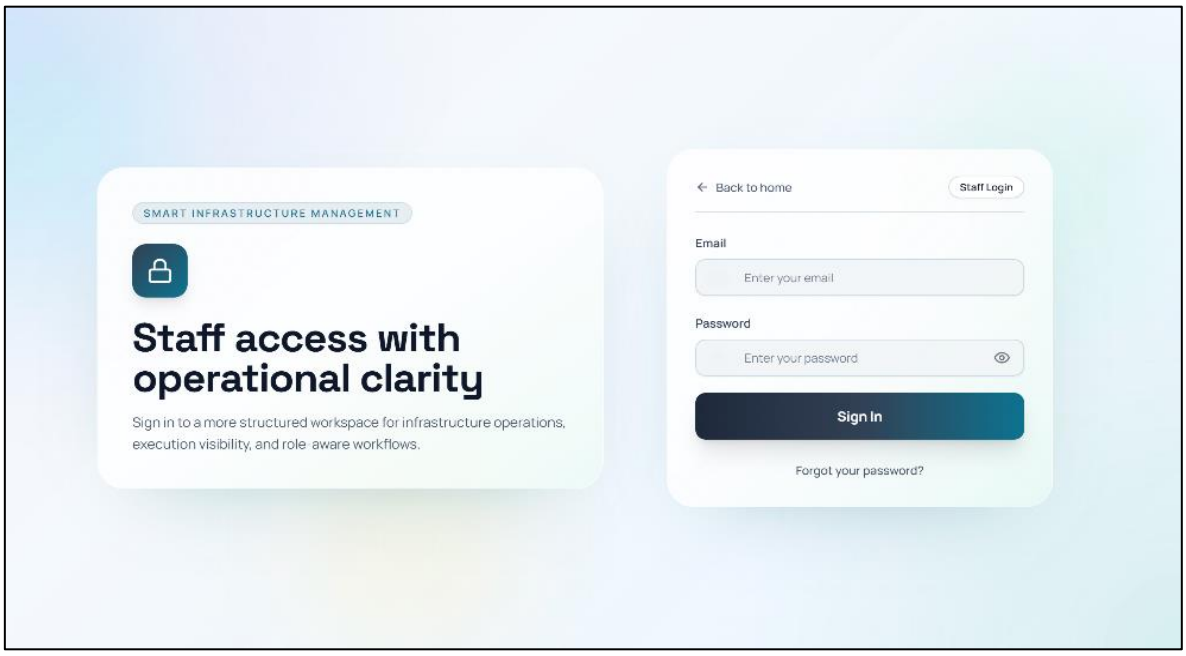

Fig 6 Staff Login

A. System Output

- Dashboard Interface: The Dashboard is the first thing you see when you log into the Smart

Infrastructure Management system, and for good reason. It brings everything that matters into one focused view: your labs, your devices, your users, and the overall health of your infrastructure, without making you jump between screens to piece the picture together.

At the top, you'll immediately notice a personalized welcome that tells you which user is currently logged in and how many devices are under their management. This small detail makes a big difference in multi-user environments where different people oversee different parts of the system.

- Device Status at a Glance: The three headline cards — Total Devices, Online Devices, and Offline Devices- cut straight to what administrators care about most. Online devices are confirmed as healthy and reachable, while offline devices are flagged clearly so nothing slips through unnoticed. If something needs attention, you'll know the moment you open the dashboard. This immediate visibility helps teams prioritize triage efforts and maintain overall lab stability with minimal effort.
- Lab Availability Overview: Below the summary cards, the Lab Availability Overview breaks down device status by lab: Dev Lab, Test Labs, and Unassigned. Each lab shows its own online and offline count with a visual progress bar, making it immediately clear which environments are running smoothly and which ones need a closer look. Users can quickly compare capacity across labs at a glance, helping them decide where to schedule new tasks or redistribute workloads. This high-level view reduces the need to dig into individual devices unless a specific lab shows signs of issues.
- Role-Based Views — What Admins and Staff Actually See: This is where the system gets thoughtful about access. An administrator sees the complete picture: all devices across every lab, total and active user counts, and a dedicated Administration section in the navigation that includes User Management. From there, admins can manage who has access to the system, assign roles, and monitor staff activity, as seen in the User Management screen showing users with their respective roles, statuses, and last login details. A staff member, on the other hand, sees only what falls within the scope. Navigation is streamlined, showing Dashboard, Inventory, Playbooks, Execution Logs, and Settings, without the administrative controls that aren't relevant to the role. This keeps the interface clean and ensures people aren't overwhelmed with information outside their responsibilities.
- Navigation Panel: The left-hand navigation panel stays consistent across both roles but adapts based on permissions. Staff members work within their assigned area, while administrators carry the additional responsibility of overseeing users and system-wide settings. Everything is logically grouped so that switching between Inventory, Playbooks, Execution Logs, and Reports feels natural rather than like hunting through menus.

The Dashboard, in short, doesn't just display data; it gives each person exactly the view they need to do their job well.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr1306

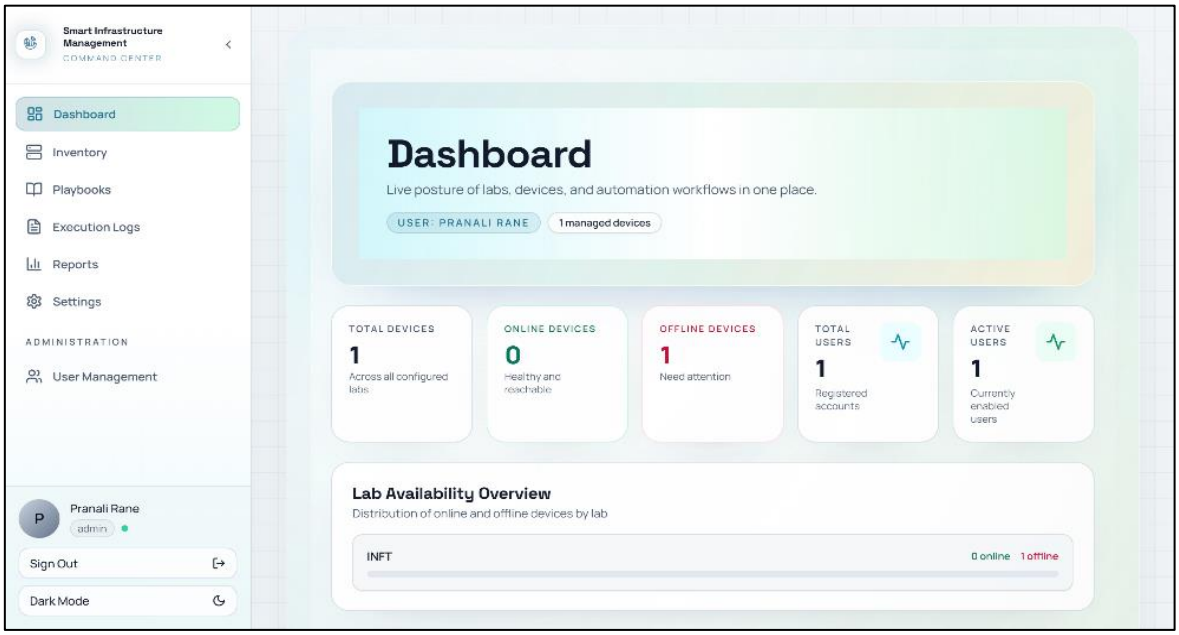

Fig 7 Admin Dashboard

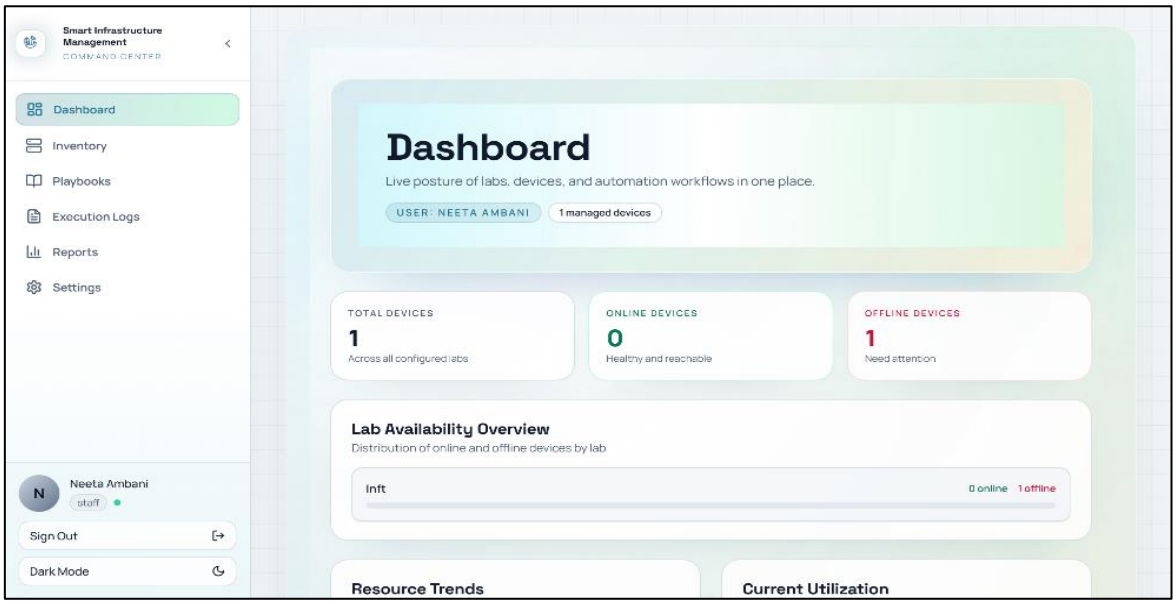

Fig 8 Staff Dashboard

- Inventory Management: The Inventory page brings everything administrators need to know about their managed environments into one well-organized view. At a glance, it shows the total number of configured labs, the full list of connected devices, overall system uptime, and any active alerts that warrant attention. Rather than hunting through separate screens to piece this information together, administrators can take in the current state of their infrastructure from a single location.

For each lab, the page goes a level deeper, showing how many devices are currently online or offline, whether any maintenance work is in progress, and a general sense of how resources are being used. This gives administrators a quick and reliable read on the condition of a particular environment without needing to dig into granular details unless something specific comes up.

Filtering options are also built into the page, letting users sort and narrow down data by lab or device type. When the number of systems being managed grows large, this kind of focused view becomes less of a convenience and more of a necessity, making monitoring and day-to-day management considerably more manageable. From this same view, administrators can also spot trends over time, such as labs that frequently report issues or devices that repeatedly drop offline. By centralizing both real-time status and historical context, the Inventory page supports faster decision-making and more proactive infrastructure planning.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr1306

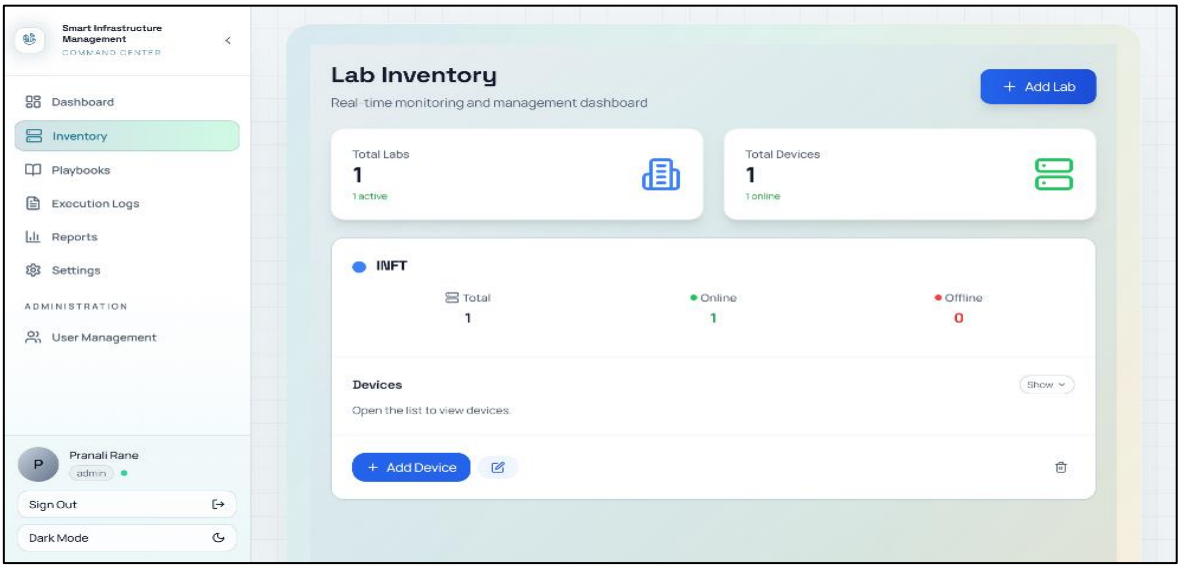

Fig 9 Lab Inventory

- Playbook Execution: The Playbooks interface is where automation tasks are created and launched directly from the dashboard, removing the need to drop into a command line for routine operations. Through a clean and structured layout, administrators can handle tasks that would otherwise require careful manual steps, cutting down both the time involved and the margin for error during execution.

The module is built around four key components that work together to make automation straightforward:

✓ The Task Configuration Panel is where administrators define what they want the system to do, whether that is installing a piece of software, cleaning up logs, or adjusting access settings. The process is guided and intuitive, with no requirement to write scripts from scratch.

✓ The Lab and Device Selection Menu gives users the flexibility to target either a specific lab or a chosen set of individual devices. This means tasks can be applied precisely where they are needed without unintentionally affecting systems that should be left alone.

✓ Automated Script Generation takes care of the technical preparation behind the scenes. Once the task details are filled in, the system automatically builds the required Ansible playbook, ensuring that every execution follows a consistent structure regardless of who initiated it or when.

✓ The Execution Console provides live visibility into what is happening as a task runs. Progress updates appear in real time, and once execution wraps up, a summary of the results is displayed, making it easy to confirm that everything went as planned or to spot where something needs a closer look.

The practical value of this interface was put to the test by running multiple tasks simultaneously across ten client machines. The system handled the load without any noticeable issues, reinforcing that the design holds up under real working conditions and not just controlled scenarios in everyday operational use cases.

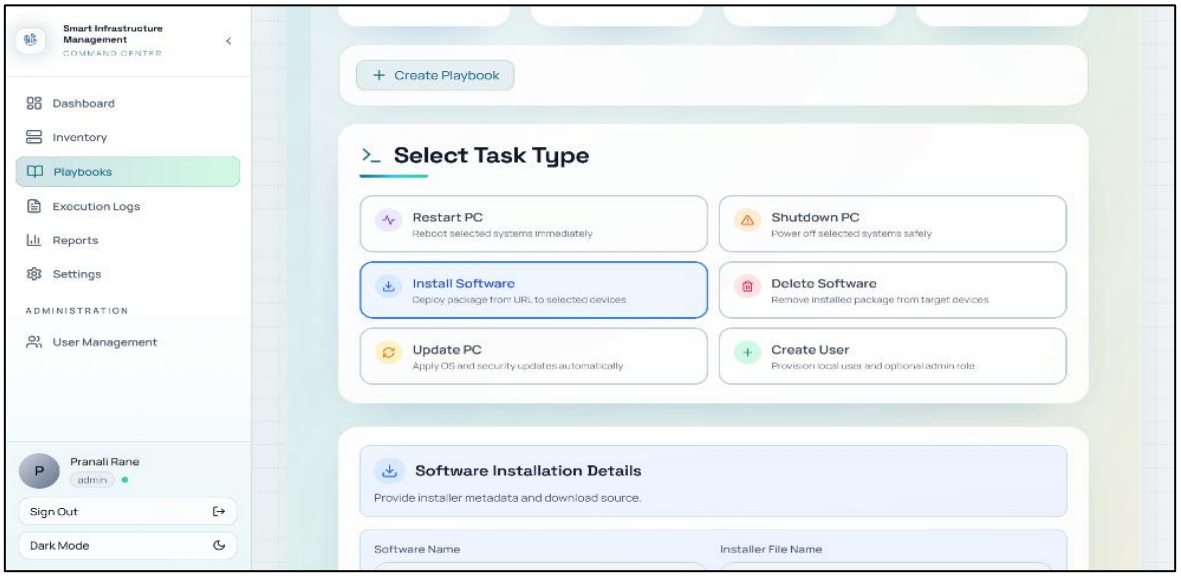

Fig 10 Playbooks

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr1306

How automation tasks are visually structured and executed within the playbooks module:

- User Account Creation Interface: The interface provides dedicated fields for entering a username and password, along with role-based configuration. Administrators can create local user accounts across selected devices with consistent credentials. The inclusion of account scope and role settings ensures controlled access management without manual configuration on each machine.

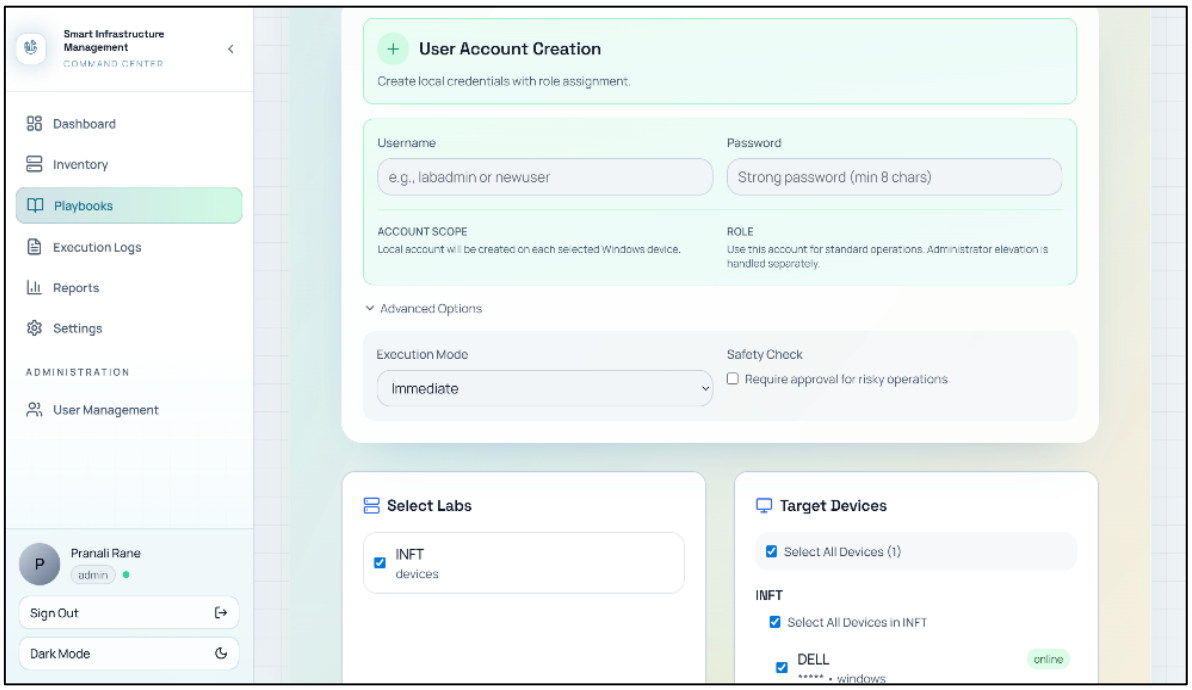

Fig 10 (a) User Account Creation

- System Restart Module: The restart playbook clearly outlines its impact before execution. It highlights that the system will immediately restart devices, force-close applications if necessary, and notify users via system alerts. This transparency helps administrators make informed decisions before triggering the task. Together, these safeguards ensure that restart operations are carried out in a controlled, predictable manner that aligns with operational policies.

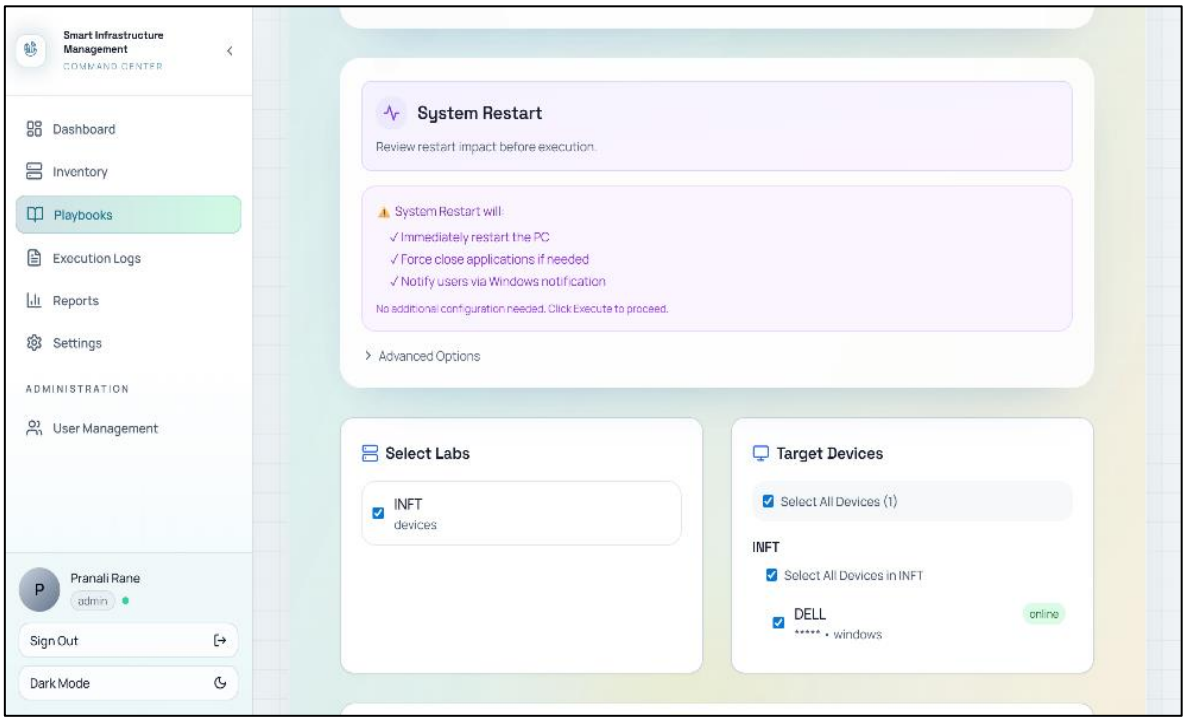

Fig 10 (b) System Restart

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr1306

- System Shutdown Module: Much like the restart option, the shutdown interface walks administrators through a confirmation summary beforehand. It makes sure they fully understand that systems will power off right away, any running applications will be closed, and active users will be disconnected. Having this step in place significantly lowers the chances of something getting disrupted by accident. The shutdown module has a confirmation step that makes it clear to administrators that the power will be turned off right away, applications will be closed, and users will be

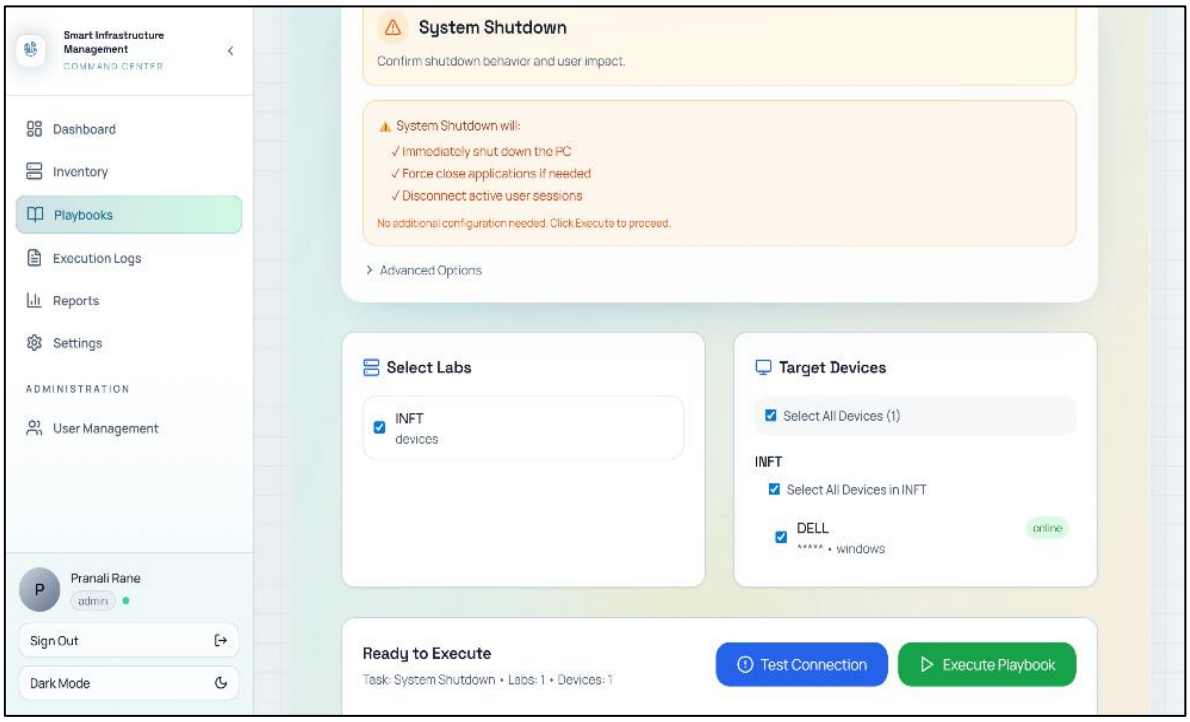

Fig 10 (c) System Shutdown

- Software Installation Panel: This section lets administrators roll out software remotely by specifying the application name, the installer file, and a download URL. The system works with several file formats, including .msi and .exe, so it can handle a wide range of applications without any compatibility headaches. This takes away the need to go device by device and install things manually. Once configured, the installation job can be pushed to multiple target machines at once, ensuring consistent deployments across the environment in a fraction of the time.

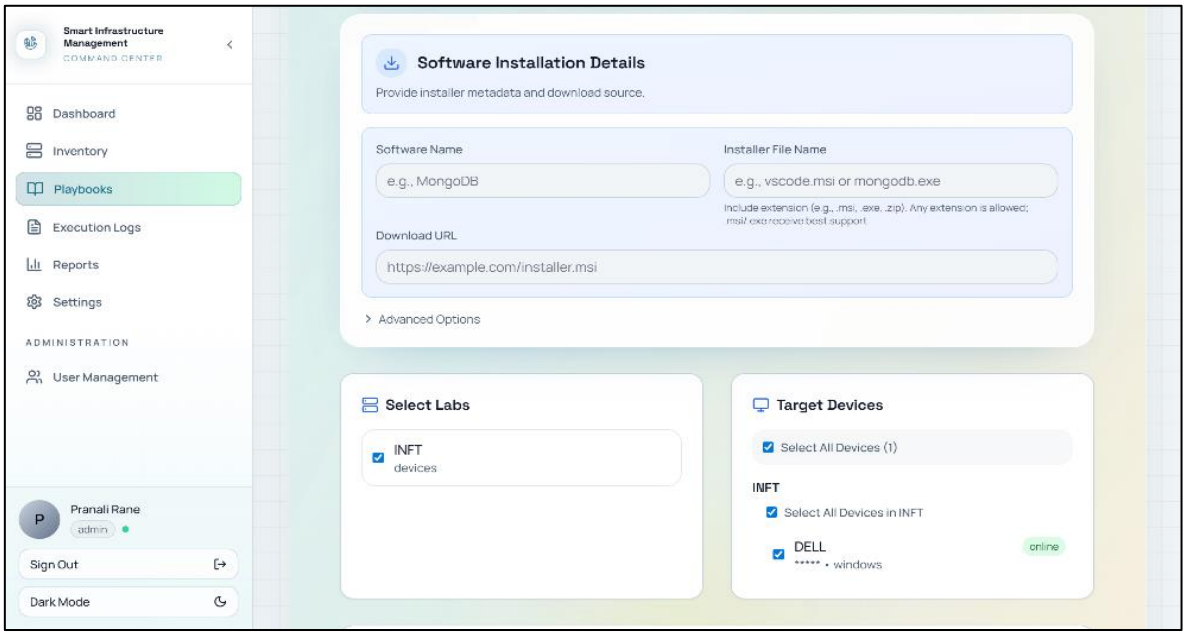

Fig 10 (d): Software Installation

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr1306

- Software Deletion Panel: The deletion interface makes removing installed software straightforward; administrators simply enter the application name exactly as it appears in the system. This keeps the process accurate and prevents issues like leftover files or mismatched uninstalls that can sometimes occur with less precise methods. The deletion panel makes it easy and simple to get rid of apps you don't want from target systems. It makes sure that the right software is removed by requiring the exact name of the application. This method lowers the chance of mistakes and keeps the system stable while the software is being uninstalled. It also lowers the risk of leftover files or incomplete removals, which keeps systems clean and well-organized.

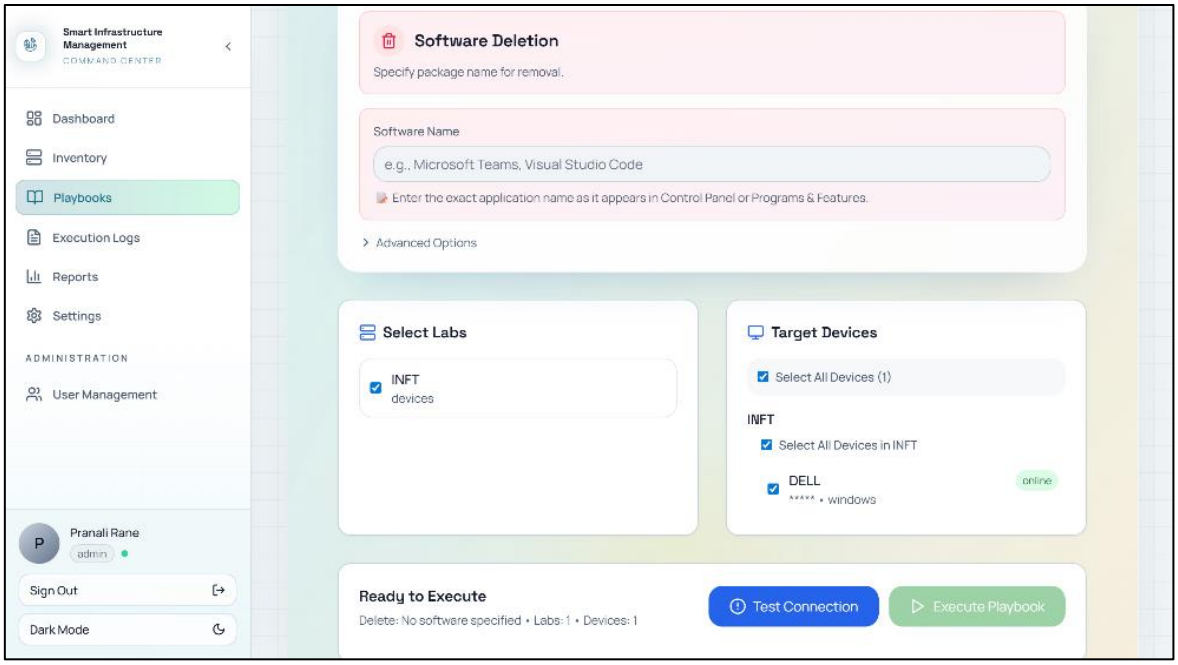

Fig 10 (e) Software Deletion

- Windows Update Module: The Playbooks interface also comes with built-in Windows Update functionality, giving administrators a way to push operating system and security updates to multiple machines at the same time. Rather than checking each device separately, the system handles everything from detecting available updates to downloading and installing patches on its own. Keeping updates centralized like this helps organizations stay on top of security, maintain a consistent environment across all devices, and avoid the kind of downtime that tends to come with outdated software.

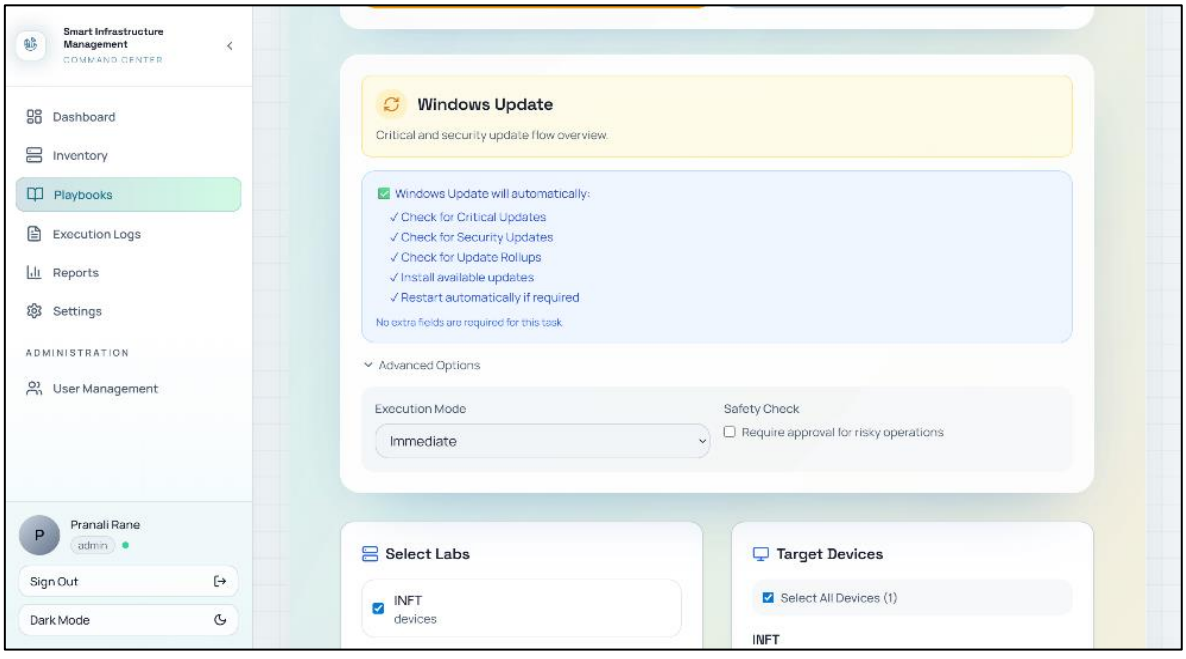

Fig 10 (f) Windows Update

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr1306

- Execution Logging: Every task carried out through the system leaves a detailed trail. Log entries record the time a task was executed, the devices it touched, the type of operation performed, and any parameters that were part of the request. The complete output generated during execution is stored alongside this information, making it straightforward to see at a glance whether things went smoothly or whether specific devices ran into trouble. All of this log data lives in MongoDB, which handles large volumes of records efficiently and keeps retrieval fast even as the history grows. Administrators access these records through the Logs module in the web interface, where entries are presented in a clean and readable format. Filtering by time, device, or task type means that finding a particular record does not require scrolling through everything; relevant entries surface quickly.

Where the logging feature proves most valuable is in troubleshooting. When a task does not complete as expected, the stored output points directly to what went wrong, whether the cause turns out to be a connectivity problem, a misconfigured setting, or an error that occurred partway through execution. Having that level of detail available cuts down the time spent diagnosing issues and allows administrators to act on problems faster. Beyond troubleshooting, the logs serve an important accountability function. Because every action is recorded, there is always a reliable reference for reviewing what has been done, when it happened, and which systems were involved. This kind of audit trail supports better oversight of the infrastructure and gives administrators confidence that nothing is happening in the system without a record of it. Taken together, the logging mechanism strengthens reliability, simplifies how errors are handled, and contributes to smoother operations on an ongoing basis.

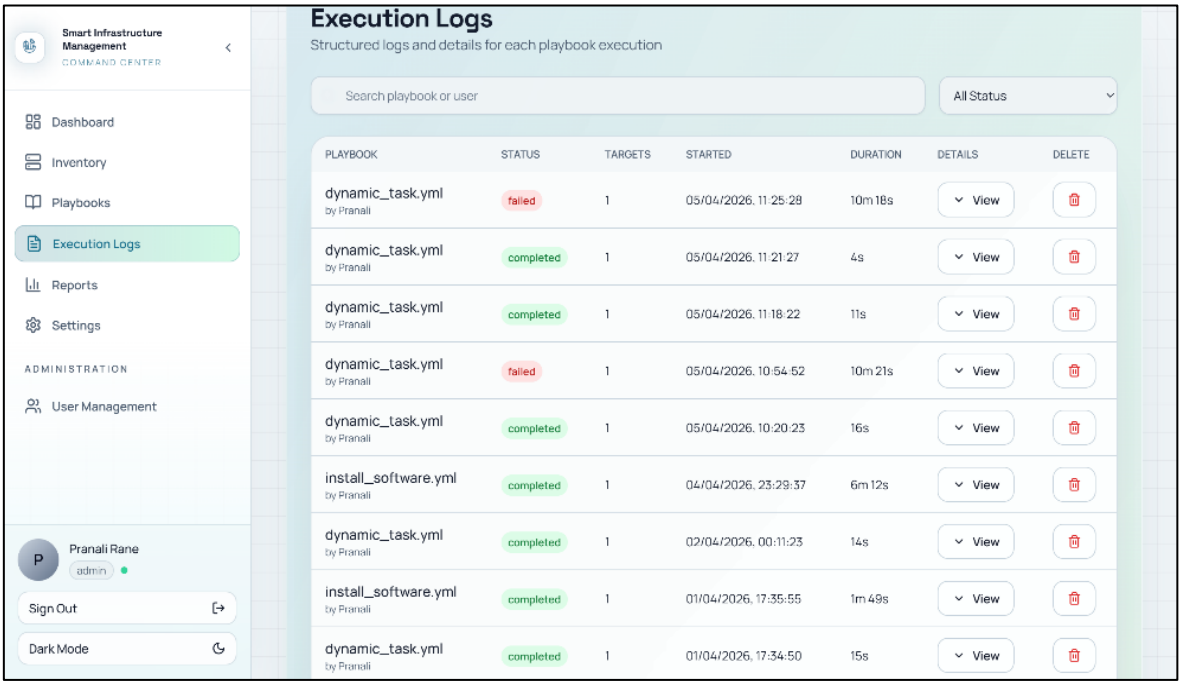

Fig 11 Execution Logs

- Report Module: The Report module gives administrators a way to step back from individual tasks and look at how the system is performing as a whole. It draws on data from across the platform, particularly logs and monitoring activity and presents it in a format that is organized enough to actually be useful rather than just overwhelming.

Through this module, administrators can examine how often tasks succeed versus fail, track which types of operations are being run most frequently, and review device performance in terms of uptime and downtime over a given period. Usage trends across the system also become visible here, making it easier to notice patterns that might not be obvious when looking at day-to-day activity in isolation.

Repeated issues that show up in the data can be identified and addressed before they become larger problems, and the overall picture helps in assessing whether the current infrastructure setup is delivering the efficiency it should be.

Reports can be exported in CSV format whenever the data needs to go further, whether that means running additional analysis, preparing documentation, or sharing findings with others who need visibility into how the system is operating. Access to historical records also supports longer-term planning, helping administrators think ahead about capacity, anticipate where improvements are needed, and make more informed decisions about the infrastructure over time.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr1306

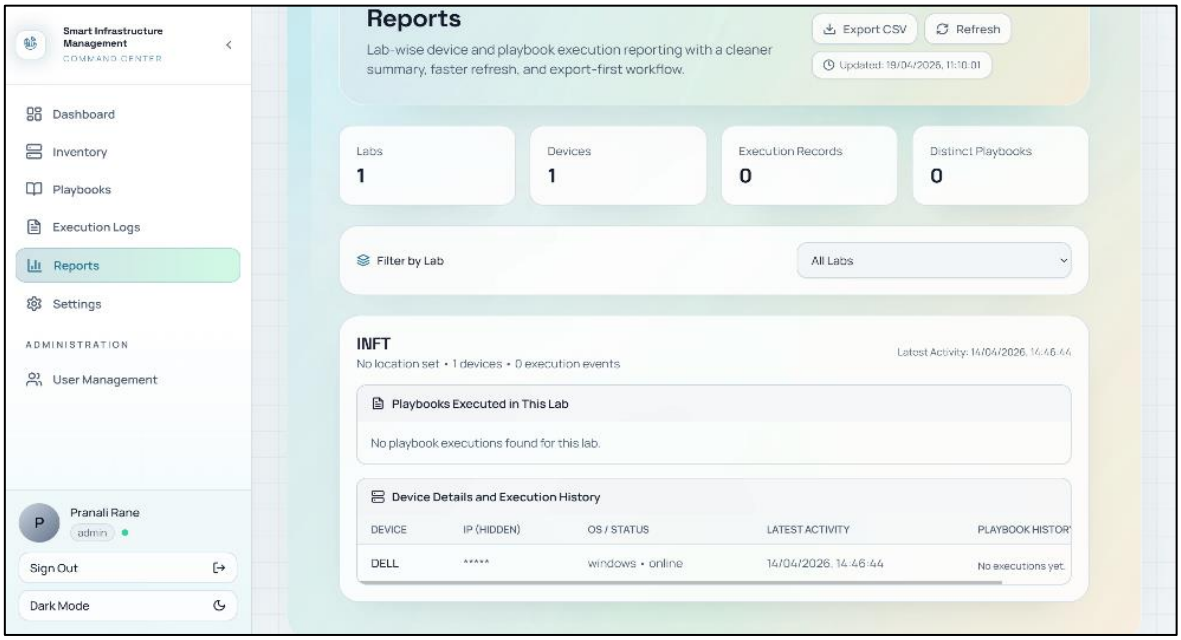

Fig 12 Reports

- Settings and User Management: The Settings page acts as a central hub where administrators can tailor the system to fit their specific operational needs. From here, they can update profile information, manage account credentials, and configure notification preferences for alerts and system updates. Options to customize certain interface behaviors and adjust infrastructure-related parameters are also available, along with basic security settings that help keep the platform operating safely. Since access to this page is restricted to administrators, staff members cannot modify any system-level configurations, keeping the broader setup stable and consistent regardless of how many people are using the platform day to day across different operational contexts. Clear role-based access controls and auditability further support governance, compliance, and responsible platform stewardship.

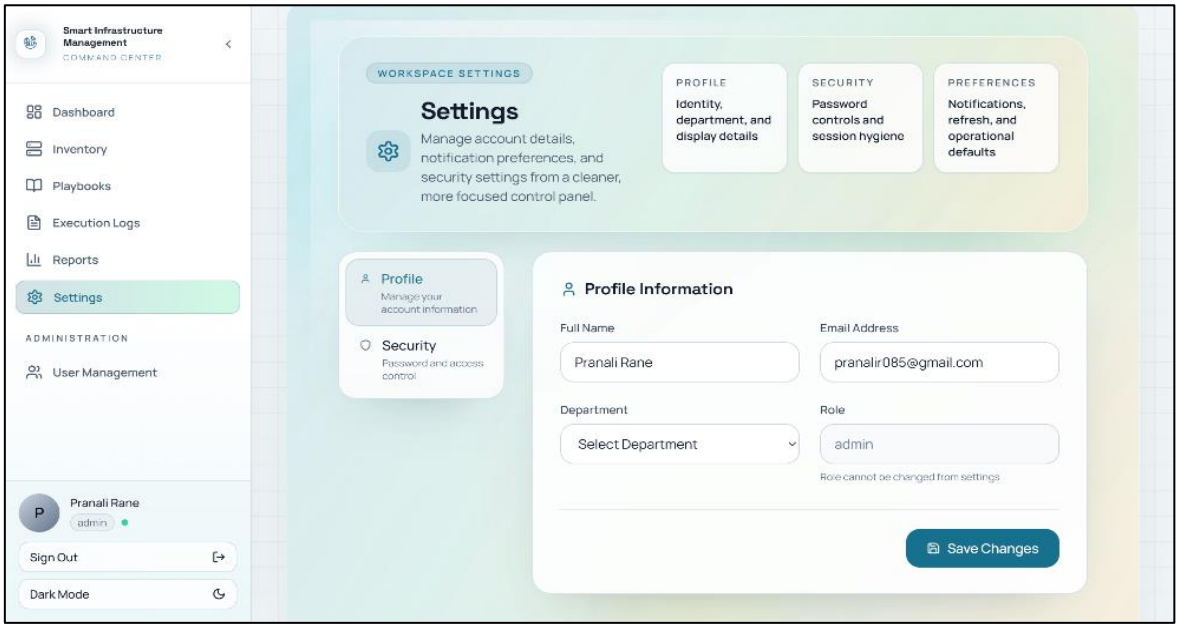

Fig 13 Settings

The User Management module gives administrators a clear and organized view of everyone who has access to the platform. As reflected in the interface, each user entry displays the person's name, email address, assigned role, department, current account status, and the date they last logged in. This makes it straightforward for an administrator to understand at a glance who is active in the system and what level of access each person holds. Administrators have full control within this module. They can bring new users onto the platform, assign roles based on each person's responsibilities, and adjust or revoke access whenever circumstances change. The role-based structure means that what a user can see and do within the system is directly tied to the role they have been given,

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr1306

keeping sensitive operations out of reach for those who do not need them.

In the current setup, two distinct roles shape how users experience the platform:

- Administrators — Administrators have unrestricted access across the entire system. They can manage user accounts, configure settings, run playbooks across any lab or device, and view all logs and reports, all from a single interface. A dedicated Administration section appears exclusively in their navigation panel, reflecting the broader scope of responsibility they carry compared to other roles. This serves as their control hub for higher-level system management tasks.

Additionally, staff activity is fully visible to administrators, allowing them to monitor and track every action performed across the platform. This ensures complete accountability, helping administrators stay informed and maintain a secure, well-managed environment always.

- Staff members —work within a more focused scope. Their dashboard shows only the devices and labs assigned to them, and their navigation is streamlined to include the tools relevant to their day-to-day work: Dashboard, Inventory, Playbooks, Execution Logs, and Settings at a personal level. They can carry out automation tasks and monitor their assigned systems, but they do not have access to system-wide configuration, user management, or platform-level reports. This keeps their interface clean and their responsibilities clearly defined without unnecessary complexity getting in the way. This makes it more reliable.

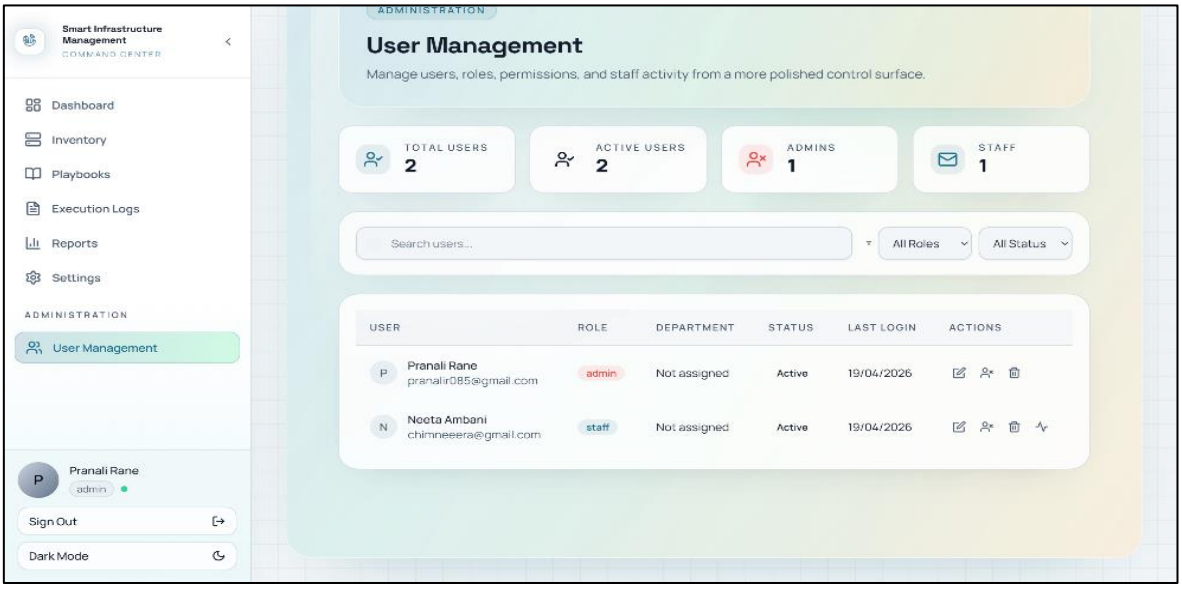

Fig 14 Admin User Management

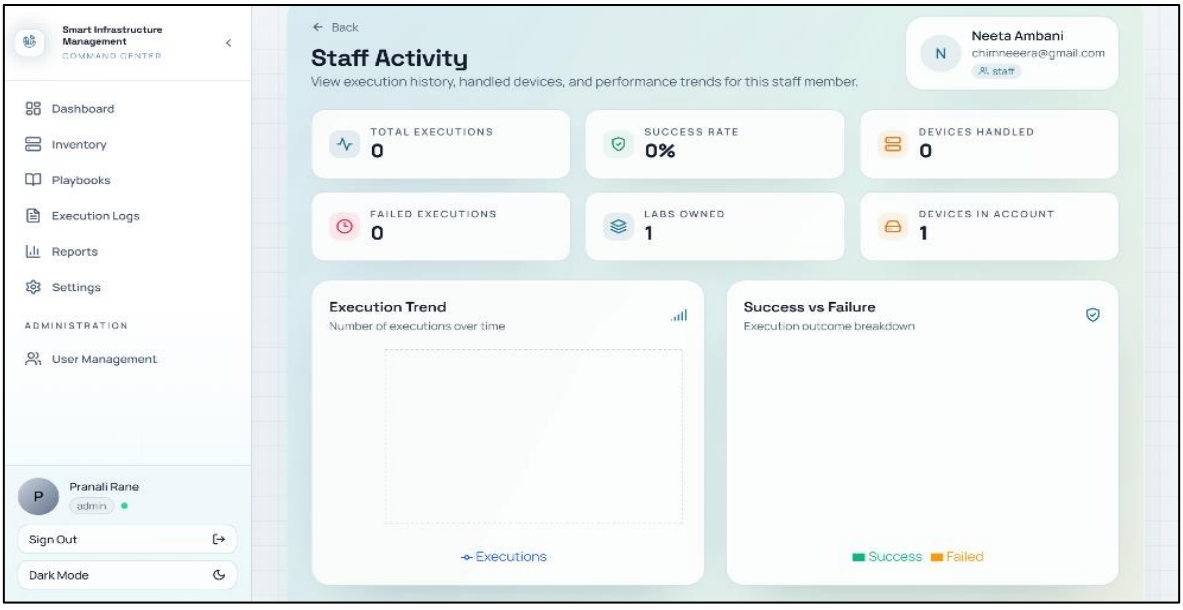

Fig 14 (a) Admin Monitoring Staff Activity

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr1306

B. Performance Observations

The performance evaluation carried out during testing painted a genuinely encouraging picture of how the Smart Infrastructure Management System holds up when put to work across multiple devices. Execution times consistently fell within a reasonable range, meaning administrators can carry out a variety of operations without sitting through frustrating delays. This level of responsiveness makes the system a practical fit for real-world environments, academic computer labs, small-scale organizational networks, and similar settings where reliability matters as much as raw capability.

A significant part of why the system performs as well as it does comes down to how Ansible handles task execution. Rather than working through devices one at a time in a sequential queue, the system runs operations across multiple machines simultaneously. The practical impact of this is meaningful; the total time needed to complete a job shrinks considerably, and the system stays responsive throughout the process even when a substantial number of machines are involved. For administrators who regularly need to push changes or run checks across an entire lab at once, this parallel approach represents a genuine step forward compared to traditional methods, where every device has to wait for its turn.

Table 1 Performance Evaluation of Operations

Operation	Avg. Time	Result
Ansible Ping (connectivity check)	~8 seconds	All devices responded
Software Installation	~45 seconds	Successful on all targets
System Update	~90 seconds	Updates applied successfully
System Restart	~12 seconds	All systems restarted
System Shutdown	~10 seconds	All systems shut down
Software Deletion	~30 seconds	Software removed cleanly

Another observation worth noting from the testing phase was how uniformly tasks were executed across different machines. Connectivity checks, software installations, and system-level commands all behaved the same way regardless of which device was on the receiving end, with every correctly configured target machine responding as anticipated. This kind of consistency is not something to take for granted; it reflects a well-structured automation setup that can be trusted to apply configurations in the same reliable way across the entire infrastructure, rather than producing slightly different outcomes depending on the machine.

The system was also put through its paces with multiple tasks running at the same time, and it held steady throughout. There was no measurable drop in performance and no unexpected failures, even when simultaneous operations were being handled across several devices at once. This points to an overall architecture that is genuinely well-suited for concurrent workloads, one that distributes the demands of parallel execution without placing excessive strain on the underlying resources. For environments where administrators cannot afford to run operations in isolation or wait for one task to finish before starting the next, this stability under load is exactly the kind of characteristic that makes a system dependable in practice rather than just in theory.

Another important thing to remember is that the system could smoothly recover from small mistakes in execution without affecting the flow of the whole task. Even though some nodes were a little late, the orchestration kept going without stopping. This made sure that big operations stayed efficient and predictable. Such resilience makes people even more sure that the system can handle real-world, changing situations well. Over time, this consistent behavior builds trust, encouraging teams to rely on the platform for increasingly critical and large-scale automation workflows in day-to-day operations across all teams. In practice, this reduces the need for constant supervision and emergency intervention during complex rollouts or maintenance windows. Administrators can plan and execute larger change sets with greater confidence, knowing that isolated delays or minor issues will not derail the entire process. This combination of stability, fault tolerance, and predictable behavior forms a strong operational backbone for long-term infrastructure management. This makes it more reliable for sustained, large-scale use. This foundation positions the platform well for continued growth. As environments evolve and new services are added, the platform can adapt without major redesign. This future-ready posture helps organizations modernize steadily while minimizing operational risk. It also supports consistent governance practices across diverse teams and environments. In turn, organizations can scale their automation footprint with clearer oversight and reduced operational overhead. Collectively, these characteristics position the platform as a dependable choice for mission-critical infrastructure operations in modern enterprises and complex hybrid environments over extended planning horizons.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr1306

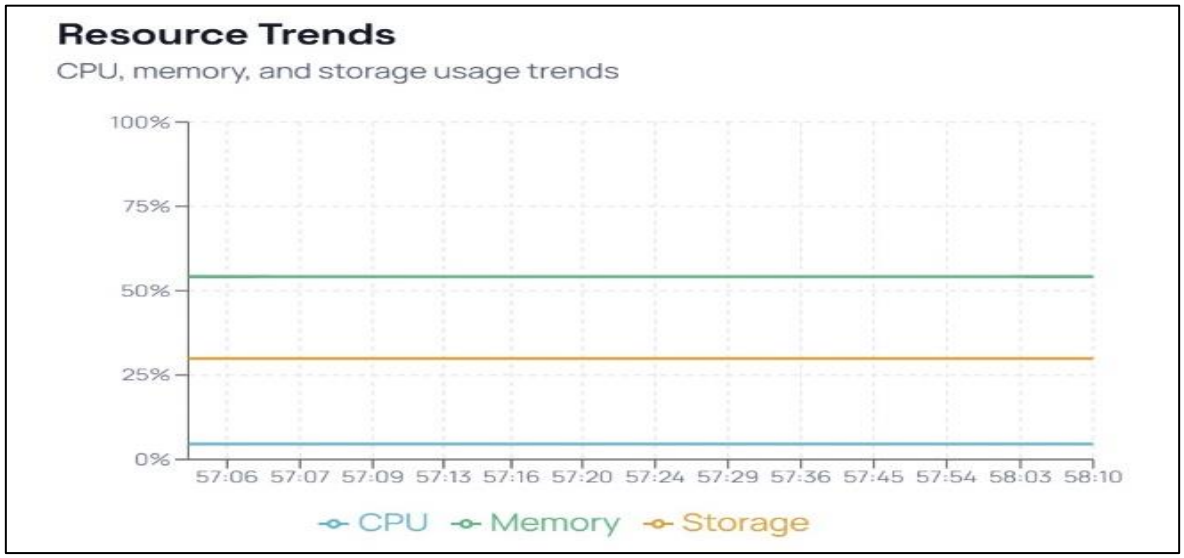

Fig 15 Resource Trends

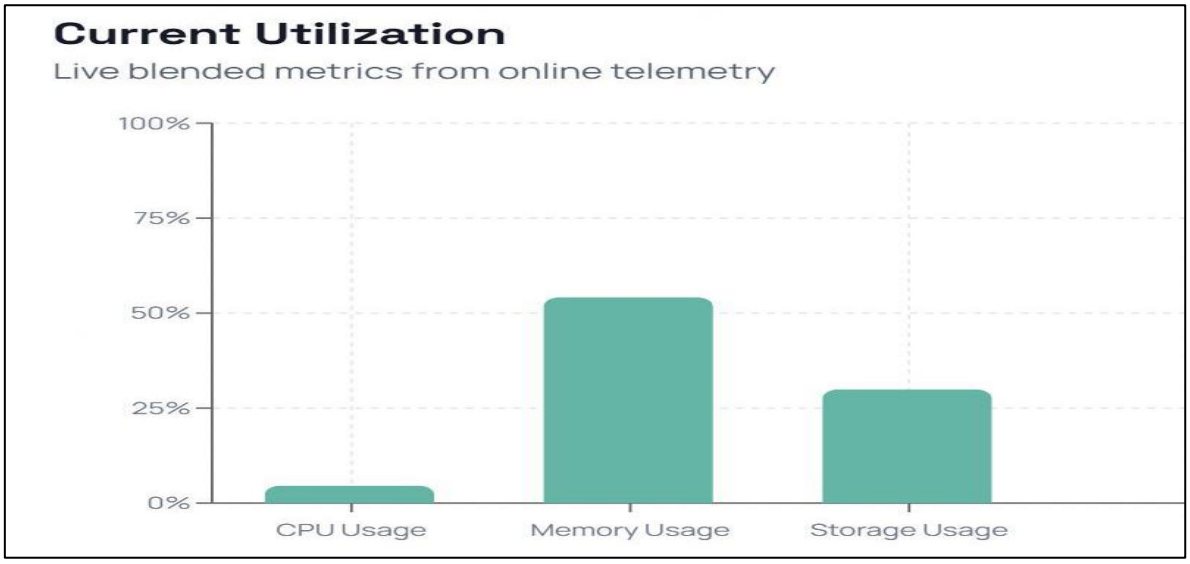

Fig 16 Current Utilization

V. CONCLUSION

The Smart Infrastructure Management System delivers on what it set out to accomplish: a practical, web-based platform that brings multiple systems under centralized control without making the process harder than it needs to be. By pairing Ansible's automation capabilities with a MERN-based interface, the system takes the heavy lifting out of routine administrative work while ensuring that devices across the infrastructure stay configured consistently. Tasks that would otherwise require an administrator to touch each machine individually, such as software installations, system updates, restarts, and shutdowns, can now be handled across an entire fleet of devices in a single operation. The decision to use WinRM for remote communication adds another layer of practicality to the setup, since its agentless nature means the initial one-time configuration on each target machine is all that is needed to get things running. There is nothing to install, nothing to maintain on the client side, and nothing standing in the way of scaling the system as the number of managed devices grows.

Beyond the technical foundations, one of the qualities that sets this system apart is how approachable it feels to use. The complexity that normally comes with automation tools like Ansible has been absorbed into the interface, so administrators who are newer to this kind of tooling are not left struggling to get results. The learning curve that has historically kept infrastructure automation out of reach for smaller teams or less specialized administrators simply is not the barrier it once was. At the same time, the underlying architecture has been kept modular and flexible, so the system does not hit a wall when requirements change. New devices can be brought in without structural overhauls, and the platform is well-positioned to grow in several meaningful directions, expanding support to cover additional operating systems, layering in cloud-based capabilities, or introducing predictive analysis features that could give administrators earlier visibility into potential issues before they develop into real problems.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26apr1306

REFERENCES

[1]. M. T. Jones, “An introduction to Linux-based automation with Ansible,” https://developer.ibm.com/articles/automation-with-ansible/, IBM Developer, 2021.

[2]. M. S. Mahmoud, “IoT-based smart monitoring and controlling of electrical power system,” IEEE Access, vol. 7, pp. 164 238–164 250, 2019.

[3]. K. Lee and S. Lee, “Integration of automation and monitoring systems in smart buildings,” Sensors, vol. 20, no. 16, pp. 1–15, 2020.

[4]. D. P. Bertsekas and J. N. Tsitsiklis, Introduction to Probability, 2nd ed. Athena Scientific, 2008.

[5]. P. Kumar and R. Singh, “IoT-based infrastructure monitoring for institutional automation,” in Proc. IEEE Int. Conf. on Smart Tech, Pune, India, 2021, pp. 78–84.

[6]. M. Rahman and M. A. Hossain, “Cloud-based infrastructure management using automation tools,” IEEE Cloud Computing, vol. 8, no. 2, pp. 56–65, 2021.

[7]. P. Sharma, “Performance evaluation of Ansible for configuration management,” in Proc. 2020 Int. Conf. on Computational Intelligence, Jaipur, India, 2020, pp. 219–224.

[8]. M. T. Nguyen et al., “A lightweight framework for networked infrastructure monitoring using Python,” IEEE Internet of Things Journal, vol. 8, no. 9, pp. 7321–7332, 2021.

[9]. A. K. Srivastava, “Real-time monitoring and control of lab networks using Flask and MongoDB,” in Proc. IEEE ICCC, 2022, pp. 334–340

[10]. P. Verma and A. Kumar, “Energy-efficient smart campus using IoT and data analytics,” IEEE Transactions on Industrial Informatics, vol. 16, no. 9, pp.6079–6087, 2020.

[11]. M. Kaur and V. Gupta, “Design of centralized monitoring system using Python and Node.js,” International Journal of Computer Applications, vol. 177, no. 18, pp. 45–50, 2019.

[12]. R. Raj, “Performance comparison of automation tools: Ansible, Puppet, Chef, and SaltStack,” International Journal of Advanced Research in Computer Science, vol. 10, no. 4, pp. 56–62, 2019.

[13]. H. Zhang et al., “Automated system administration through declarative configuration management,” IEEE Software, vol. 37, no. 6, pp. 71–78, 2020.

[14]. P. Venkatesh and M. Thomas, “Smart infrastructure automation using Python scripting,” in Proc. IEEE ICIOT, 2020, pp. 245–251.

[15]. M. H. Hassan and S. Rahman, “IoT-based smart facility management system,” IEEE Access, vol. 9, pp. 119 281–119 295, 2021.

[16]. A. Gupta and R. S. Dey, “Centralized dashboard for IT infrastructure monitoring using Streamlit” in Proc. Int. Conf. on Data Science, 2022, pp.109–114.

[17]. S. Choudhary and M. Bhatia, “Real-time system health tracking using psutil and Flask,” International Journal of Emerging Technologies in Computer Science, vol. 13, no. 2, pp. 88–94, 2021.

[18]. M. A. Khan, “Automation frameworks for multi-platform device management,” IEEE Systems Journal, vol. 15, no. 5, pp. 7201–7210, 2021.

[19]. M. Patel and K. Jain, “Secure SSH communication and authentication in multi-node networks,” IEEE Transactions on Network Security, vol. 8, no. 3, pp. 245–253, 2020.

[20]. A. Banerjee, “Role of open-source automation tools in modern infrastructure,” Open Source For You Journal, vol. 9, no. 5, pp. 12–18, 2022.

[21]. Pragati A. Patil and Dr. Anil Rao Pimplapure “Study On Integration Of Blockchain And Big Data Challenges Cloud Computing”, Journal of Survey in Fisheries Sciences, Volume 10 - Issue 1 (2023), pp.10(1)16887-16891 doi: https://doi.org/10.53555/sfs.v10i1.3091

[22]. P. Patil, S. Singh, S. Hate and A. Parekh, "Thynk : Your Purpose-Driven Workplace Intelligence," 2026 International Conference on Communication, Computing and Emerging Technologies (IC3ET), Vasai, India, 2026, pp. 1-8, doi: 10.1109/IC3ET64989.2026.11467556

[23]. Tejal Mendhe,Amruta Jadhav, Pragati Patil, "VOLTVISION: YOUR ELECTRICAL EXPERTISE HUB", International Journal of Creative Research Thoughts (IJCRT), ISSN:2320-2882, Volume.13, Issue 3, pp. i542-i549, March 2025

[24]. Ms. Pragati Patil and Dr. Anil Pimpalapure, “Decentralized Security Architecture Based on Software Defined Networking (SDN) in Blockchain for IOT Network”, International Journal of Engineering Research and Science, vol. 10, no. 6, pp. 23–31, Jun. 2024, doi: 10.5281/zenodo.15203773.

[25]. Vaishali Shirsath, Pragati Patil, Manaswi Jadhav, Omkar Jadhav, Sanskruti Kokare, Shreya Bokare (2024), Design and Development of Billing Module for DRM. International Journal of Innovative Science and Research Technology (IJISRT) IJISRT24APR2068, 2529-2537. DOI: 10.38124/ijisrt/IJISRT24APR2068.