NexusGrid:System Monitoring and Control

Submitted in partial fulfillment of the requirements

of the degree of

Bachelor of Engineering

by

Chetan Chaudhari	**10**
Parth Shikhare	**53**
Nischay Chavan	**14**

Supervisor:

Prof. Shiv Negi

UNIVERSITY OF MUMBAI

NexusGrid:System Monitoring and Control

Submitted in partial fulfillment of the requirements

of the degree of

Bachelor of Engineering

by

Chetan Chaudhari	**10**
Parth Shikhare	**53**
Nischay Chavan	**14**

Supervisor:

Prof. Shiv Negi

Department of Information Technology

Don Bosco Institute of Technology
Vidyavihar Station Road, Mumbai - 400070

2024-2025

DON BOSCO INSTITUTE OF TECHNOLOGY

Vidyavihar Station Road, Mumbai - 400070

Department of Information Technology

CERTIFICATE

This is to certify that the project entitled **"NexusGrid:System Monitoring and Control"** is a bonafide work of

Chetan Chaudhari	**10**
Parth Shikhare	**53**
Nischay Chavan	**14**

submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **Undergraduate** in **Bachelor of Information Technology**

Date: / /

(Prof. Shiv Negi)
Supervisor

(Prof. Sunantha Guruswamy)
HOD, IT Department

(Dr. Sudhakar Mande)
Principal

DON BOSCO INSTITUTE OF TECHNOLOGY

Vidyavihar Station Road, Mumbai - 400070

Department of Information Technology

Project Report Approval for B.E.

This project report entitled **"NexusGrid:System Monitoring and Control"** is approved for the degree of **Second Year Semester 4 in Information Technology**

(Examiner's Name and Signature)

1. ——————————————

2. ——————————————

(Supervisor's Name and Signature)

1. ——————————————

(Chairman)

1. ——————————————

Date:

Place:

DON BOSCO INSTITUTE OF TECHNOLOGY
Vidyavihar Station Road, Mumbai - 400070

Department of Information Technology

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Name	**Roll No.**	**Signature**
Parth Shikhare	53	
Chetan Chaudhari	10	
Nischay Chavan	14	

Date:

Abstract

In modern digital workspaces, the efficiency and reliability of computer systems are critical to maintaining seamless operations. However, issues such as delayed fault reporting, inefficient resource allocation, and lack of real-time monitoring often disrupt productivity and increase operational costs. To address these challenges, NexusGrid is developed as an automated system monitoring and control platform. It offers role-based dashboards for Admins, Supervisors, and Users, allowing streamlined communication and efficient fault management. Key features include real-time health checks, QR code-based system identification, interactive workspace layouts, automated energy management, and report generation with analytics. By integrating these functionalities, NexusGrid aims to enhance system reliability, reduce downtime, and optimize resource utilization in a proactive and user-friendly manner.

The system is developed using the Django framework for the backend, offering a secure and scalable infrastructure, while the frontend employs HTML, CSS, Bootstrap, and JavaScript to deliver a responsive and intuitive user interface. The platform is powered by a suite of Python libraries such as Celery for task scheduling, psutil and pySMART for system health monitoring, and Plotly and Matplotlib for real-time data visualization. Features like automated software deployment via Ansible, dynamic QR code generation for system identification, and interactive workspace layouts built using PyGame add functional depth to the project. Additionally, the use of SQLite ensures lightweight yet efficient database operations. NexusGrid thus not only demonstrates full-stack development and system integration skills but also presents a practical, real-world solution to recurring infrastructure management problems in academic institutions and corporate IT environments.

Keywords:System Monitoring, Fault Reporting, Resource Allocation, Real-Time Health Checks, QR Code Identification, Interactive Dashboard, psutil, Data Visualization, Energy Management.

Contents

List of Figures

Chapter 1

Introduction

In the evolving landscape of modern workplaces, educational institutions, and IT-driven environments, the reliance on digital infrastructure has become fundamental to day-to-day operations. As organizations grow in scale and complexity, the number of systems and devices they manage also increases—making efficient system administration a necessity rather than a luxury. However, many organizations still depend on outdated, manual processes for managing faults, scheduling maintenance, and allocating resources. Faults are typically reported through emails or verbal communication, often leading to miscommunication, delayed responses, and prolonged system downtimes. Moreover, the absence of real-time monitoring means that potential system failures go undetected until they cause disruptions, leading to reduced productivity, increased maintenance costs, and compromised system reliability. These issues highlight the urgent need for a robust, automated platform capable of handling monitoring, fault reporting, and resource management with precision and speed

To bridge this gap, NexusGrid has been conceptualized and developed as a comprehensive solution for system monitoring and control. It is a full-stack web application that enables real-time health monitoring of computer systems, fault detection and reporting, energy management, and intelligent resource handling—all through interactive, role-based dashboards tailored for Admins, Managers, and Employees. The platform leverages Django for the backend, coupled

with frontend technologies like HTML, CSS, Bootstrap, and JavaScript to provide a seamless user interface. Key technical features include QR code-based system identification, automated task management with Celery, and graphical analytics through libraries like Plotly and Matplotlib. By automating time-consuming processes and presenting data in a visual, actionable manner, NexusGrid improves both system reliability and administrative efficiency. It not only addresses present-day challenges in infrastructure management but also offers a scalable foundation for smart IT environments of the future.

Chapter 2

Review

NexusGrid presents a robust and innovative solution to the persistent challenges faced in IT infrastructure management, particularly within academic and organizational settings. By automating fault reporting, system health monitoring, and resource allocation, it eliminates the inefficiencies associated with manual processes such as verbal communication or delayed emails. The platform is built using Django for the backend and modern web technologies for the frontend, ensuring both functionality and ease of use. It incorporates advanced features like QR code tagging for rapid fault identification, real-time performance monitoring using libraries like psutil and Django Channels, and automated task handling through Celery. Additionally, it supports analytics and reporting capabilities, enabling data-driven decision-making. The interactive, role-based dashboards enhance user engagement and streamline operations for Admins, Managers, and Employees alike. Overall, NexusGrid not only meets the immediate requirements of a smart monitoring system but also provides a scalable framework adaptable to larger, more complex environments in the future.

Chapter 3

Analysis and Design

3.1 Entity Relationship Diagram

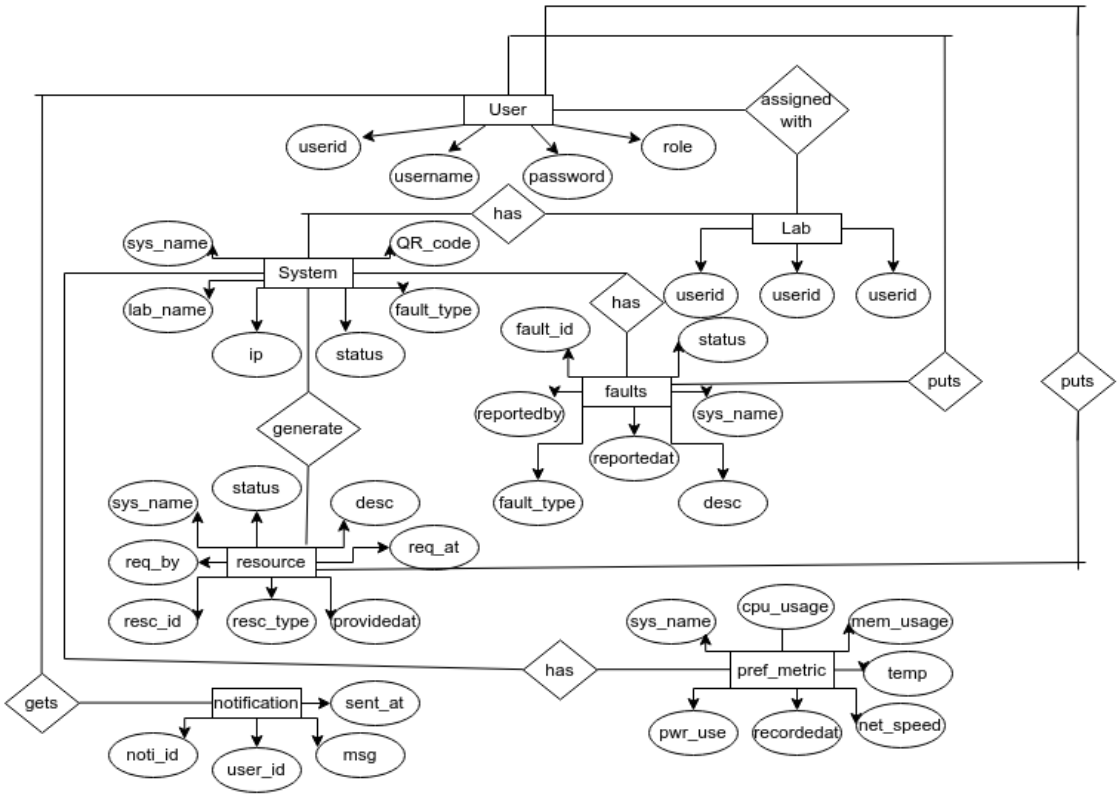
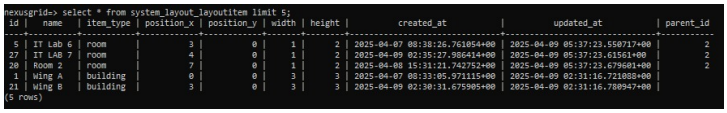


Figure 3.1: ER Diagram

The ER diagram represents a database structure for managing users, labs, systems, faults, resources, and related components in a lab environment. It includes entities like User, Lab, System, Faults, Resource, Notification, and Prefmetric, each with specific attributes. The relationships show that users are assigned to labs, labs contain systems, and systems generate faults and hold resources. Users are linked to faults and resources they report or request and also receive notifications. Additionally, systems have associated performance metrics through Prefmetric. The diagram provides a clear conceptual overview of how these entities interact within the system.

3.2 Database Design

3.2.1 System Layout Table

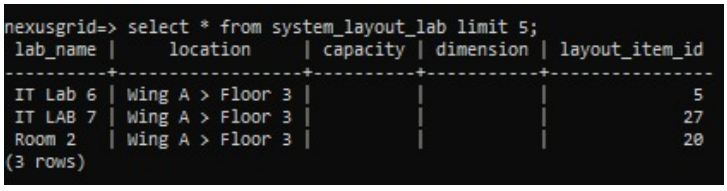


```
nexusgrid=> select * from system_layout_layoutitem limit 5;
 id |   name   | item_type | position_x | position_y | width | height |          created_at           |          updated_at           | parent_id
----+----------+-----------+------------+------------+-------+--------+-------------------------------+-------------------------------+-----------
  5 | IT Lab 6 | room      |          3 |          0 |     1 |      2 | 2025-04-07 08:38:26.761054+00 | 2025-04-09 05:37:23.550717+00 |         2
 27 | IT LAB 7 | room      |          4 |          0 |     1 |      2 | 2025-04-09 02:35:27.986414+00 | 2025-04-09 05:37:23.61561+00  |         2
 20 | Room 2   | room      |          7 |          0 |     1 |      2 | 2025-04-08 15:31:21.742752+00 | 2025-04-09 05:37:23.679681+00 |         2
  1 | Wing A   | building  |          0 |          0 |     3 |      3 | 2025-04-07 08:33:05.971115+00 | 2025-04-09 02:31:16.721088+00 |
 21 | Wing B   | building  |          3 |          0 |     3 |      3 | 2025-04-09 02:30:31.675905+00 | 2025-04-09 02:31:16.780947+00 |
(5 rows)
```

Figure 3.2: System Layout Table

This image shows data directly from the above database table, displaying rows that represent layout elements like rooms and buildings with their properties such as name, type, position, and timestamps. It's a direct view of the underlying data structure for the system layout.

3.2.2 System Layout Lab



```
nexusgrid=> select * from system_layout_lab limit 5;
 lab_name |     location      | capacity | dimension | layout_item_id
----------+-------------------+----------+-----------+----------------
 IT Lab 6 | Wing A > Floor 3 |          |           |              5
 IT LAB 7 | Wing A > Floor 3 |          |           |             27
 Room 2   | Wing A > Floor 3 |          |           |             20
(3 rows)
```

Figure 3.3: System Layout Lab

This image displays the contents of the above database table, retrieved by a MySQL query. It lists labs by name and their location within the building structure, along with a corresponding layout item ID linking them to the overall system layout.

3.2.3 Fault Report

```
nexusgrid=> select * from faults_faultreport limit 5;
 fault_id | fault_type |        description        |   status    |          reported_at          |          resolved_at          | reported_by_id | system_name_id
----------+------------+---------------------------+-------------+-------------------------------+-------------------------------+----------------+---------------
        9 | Software   | ads                       | resolved    | 2025-04-07 11:44:51.635095+00 | 2025-04-07 12:35:05.560579+00 |              2 |              9
       10 | Software   | Title: Test Repot\r      +| Resolved    | 2025-04-07 12:30:33.328721+00 | 2025-04-07 12:52:17+00        |              2 |              8
          |            | \r                       +|             |                               |                               |                |
          |            | Test Desc                 |             |                               |                               |                |
       11 | Hardware   | desc                      | unaddressed | 2025-04-08 05:34:31.94763+00  |                               |              2 |              8
       13 | Hardware   | Motherboard is not working | in-progress | 2025-04-08 07:46:58.190868+00 |                               |              2 |              9
       12 | Software   | faultt                    | resolved    | 2025-04-08 07:25:06.138575+00 | 2025-04-08 09:15:21.035325+00 |              2 |              9
(5 rows)
```

Figure 3.4: Fault Report

This image displays the contents of the above database table, showing records of system faults. It includes details such as the fault type, description, current status (resolved, unaddressed, in-progress), and timestamps for when the fault was reported and resolved.

3.2.4 Login Manager

```
nexusgrid=> select * from login_manager_user limit 5;
 id |                              password                              |          last_login           | is_superuser | username | first_name | last_name |         email
          | is_staff | is_active |          date_joined          |   role
----+---------------------------------------------------------------------+-------------------------------+--------------+----------+------------+-----------+--------------
----------+----------+-----------+-------------------------------+----------
 12 | pbkdf2_sha256$870000$dummy4$abcdefghijklmnopqrstuv1234567890== | 2025-04-08 12:39:11.687882+00 | f            | PriyaK   |            |           | priya.kapoor@e
xample.com | f        | t         | 2025-04-08 12:39:11.687882+00 | No Roles
 14 | pbkdf2_sha256$870000$dummy6$abcdefghijklmnopqrstuv1234567890== | 2025-04-08 12:39:11.687882+00 | f            | IshitaM  |            |           | ishita.mehra@e
xample.com | f        | t         | 2025-04-08 12:39:11.687882+00 | No Roles
 15 | pbkdf2_sha256$870000$dummy7$abcdefghijklmnopqrstuv1234567890== | 2025-04-08 12:39:11.687882+00 | f            | RohanP   |            |           | rohan.patel@ex
ample.com  | f        | t         | 2025-04-08 12:39:11.687882+00 | No Roles
 16 | pbkdf2_sha256$870000$dummy8$abcdefghijklmnopqrstuv1234567890== | 2025-04-08 12:39:11.687882+00 | f            | MeeraN   |            |           | meera.nair@exa
mple.com   | f        | t         | 2025-04-08 12:39:11.687882+00 | No Roles
 17 | pbkdf2_sha256$870000$dummy9$abcdefghijklmnopqrstuv1234567890== | 2025-04-08 12:39:11.687882+00 | f            | YashS    |            |           | yash.singh@exa
mple.com   | f        | t         | 2025-04-08 12:39:11.687882+00 | No Roles
(5 rows)
```

Figure 3.5: Login Manager

This image shows the contents of the above database table, retrieved via a MySQL query. It displays user account information including IDs, status flags (staff, active, superuser), join and login dates, assigned roles, usernames, names, and email addresses.

3.2.5 System Monitoring

```
nexusgrid=> select * from monitoring_systeminfo limit 5;
 id |     hostname     | system  |                          version                          |     release      | machine |                processor
                | architecture | cpu_physical_cores | cpu_total_cores | cpu_max_freq | cpu_min_freq | cpu_current_freq | cpu_usage | memory_total | memory_available | m
emory_used | memory_usage_percent | disk_total | disk_used | disk_free | disk_usage_percent |   ip_address    | bytes_sent | bytes_received | users_count | logged_in_us
ers |           timestamp
----+------------------+---------+-----------------------------------------------------------+------------------+---------+------------------------------------
---------------+--------------+--------------------+-----------------+--------------+--------------+------------------+-----------+--------------+------------------+--
----------+----------------------+------------+-----------+-----------+--------------------+-----------------+------------+----------------+-------------+------------
----+-------------------------------
  1 | LAPTOP-I8S56VV9  | Windows | 10.0.26100                                                | 11               | AMD64   | Intel64 Family 6 Model 154 Stepping
3, GenuineIntel | 64bit        |                  8 |              12 |         2000 |            0 |             2000 |        19 |        15.73 |             6.74 |
      8.99 |                 57.2 |        200 |      9.11 |    190.89 |                4.6 | 192.168.0.103   |  162924432 |     1864448223 |           1 | Parth
    | 2025-04-06 05:13:56.314247+00
  3 | parth-VirtualBox | Linux   | #19~24.04.1-Ubuntu SMP PREEMPT_DYNAMIC Mon Feb 17 11:51:52 UTC 2 | 6.11.0-19-generic | x86_64  | x86_64
                | 64bit        |                  8 |               8 |            0 |            0 |             2496 |       0.7 |         8.42 |             7.15 |
         1 |                 15.1 |      39.41 |     15.17 |     22.21 |               40.6 | 127.0.1.1       |      33748 |          25971 |           2 | parth, parth
    | 2025-04-06 05:17:57.010894+00
  4 | LAPTOP-I8S56VV9  | Windows | 10.0.26100                                                | 11               | AMD64   | Intel64 Family 6 Model 154 Stepping
3, GenuineIntel | 64bit        |                  8 |              12 |         2000 |            0 |             1520 |       5.5 |        15.73 |             6.16 |
      9.57 |                 60.8 |        200 |      9.11 |    190.89 |                4.6 | 192.168.0.104   |   10264456 |       23136355 |           1 | Parth
    | 2025-04-07 04:12:31.235333+00
  5 | LAPTOP-295DUFOU  | Windows | 10.0.26100                                                | 11               | AMD64   | Intel64 Family 6 Model 151 Stepping
2, GenuineIntel | 64bit        |                  8 |              12 |         2400 |            0 |             1520 |      11.9 |        11.71 |             1.65 |
     10.07 |                 85.9 |      28.64 |      0.15 |     28.49 |                0.5 | 192.168.102.216 |   28064681 |      407730351 |           1 | OWAIS
    | 2025-04-07 04:46:43.86098+00
  6 | LAPTOP-I8S56VV9  | Windows | 10.0.26100                                                | 11               | AMD64   | Intel64 Family 6 Model 154 Stepping
3, GenuineIntel | 64bit        |                  8 |              12 |         2000 |            0 |             1520 |       4.9 |        15.73 |             5.88 |
      9.86 |                 62.7 |        200 |      9.11 |    190.89 |                4.6 | 192.168.102.214 |   10244295 |       10754113 |           1 | Parth
    | 2025-04-07 04:57:49.142072+00
(5 rows)
```

Figure 3.6: System Monitoring Info

This image displays data from the above database table, showing detailed monitoring metrics for various systems. It includes information on system specifications, CPU, memory, and disk usage, network activity, connected users, and timestamps for reporting system status

3.2.6 Tables List

```
nexusgrid=> \dt
                    List of relations
 Schema |               Name                | Type  |     Owner
--------+-----------------------------------+-------+---------------
 public | account_emailaddress              | table | nexusgrid_user
 public | account_emailconfirmation         | table | nexusgrid_user
 public | auth_group                        | table | nexusgrid_user
 public | auth_group_permissions            | table | nexusgrid_user
 public | auth_permission                   | table | nexusgrid_user
 public | django_admin_log                  | table | nexusgrid_user
 public | django_content_type               | table | nexusgrid_user
 public | django_migrations                 | table | nexusgrid_user
 public | django_session                    | table | nexusgrid_user
 public | django_site                       | table | nexusgrid_user
 public | faults_faultreport                | table | nexusgrid_user
 public | faults_resolved                   | table | nexusgrid_user
 public | login_manager_user                | table | nexusgrid_user
 public | login_manager_user_groups         | table | nexusgrid_user
 public | login_manager_user_user_permissions | table | nexusgrid_user
 public | monitoring_systeminfo             | table | nexusgrid_user
 public | resources_resourcerequest         | table | nexusgrid_user
 public | socialaccount_socialaccount       | table | nexusgrid_user
 public | socialaccount_socialapp           | table | nexusgrid_user
 public | socialaccount_socialapp_sites     | table | nexusgrid_user
 public | socialaccount_socialtoken         | table | nexusgrid_user
 public | system_layout_lab                 | table | nexusgrid_user
 public | system_layout_lab_assistants      | table | nexusgrid_user
 public | system_layout_lab_instructors     | table | nexusgrid_user
 public | system_layout_layoutitem          | table | nexusgrid_user
 public | system_layout_system              | table | nexusgrid_user
(26 rows)
```

Figure 3.7: List Of Tables

This image displays the output of a command listing relations () in a database schema. It shows a list of table names within the ”public” schema, providing an overview of the different data tables present in the database.

Chapter 4

Implementation

4.1 Login

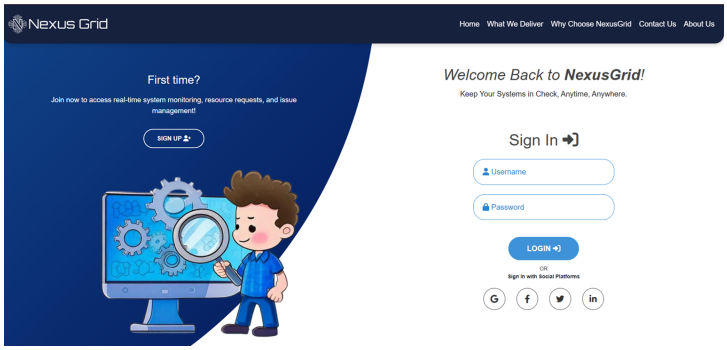


Figure 4.1: Login page

The NexusGrid landing page serves as the entry point for both new and returning users. For first-time visitors, the left side of the screen encourages them to sign up, offering access to real-time system monitoring, resource management, and fault reporting features. It includes a visually engaging cartoon illustration depicting system analysis, with a prominent ”Sign Up” button to begin the registration process.

For returning users, the right side greets them with a ”Welcome Back to NexusGrid!” message, emphasizing the platform’s purpose of ensuring systems

remain operational anytime and anywhere. The sign-in panel includes fields for entering a username and password, alongside a clearly visible ”Login” button for secure access. Additionally, users can log in via social platforms like Google, Facebook, Twitter, and LinkedIn, providing more convenience. The overall interface is modern, clean, and responsive, designed for user-friendliness while maintaining clarity in functionality.

4.2 Registration

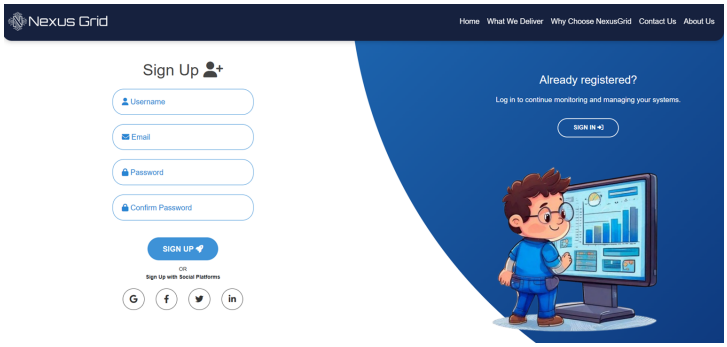


Figure 4.2: Registration page

This webpage for ”Nexus Grid” serves as a user authentication portal, featuring a prominent ”Sign Up” form on the left side for new users to register with username, email, and password fields, along with options for social media sign-up via Google, Facebook, Twitter, and LinkedIn. The right side is dedicated to existing users, prompting them to ”SIGN IN” and featuring an illustration of a boy interacting with a computer displaying data, suggesting the service involves monitoring and managing systems. A header across the top displays the ”Nexus Grid” logo and navigation links to other parts of the website like Home, What We Deliver, Why Choose NexusGrid, Contact Us, and About Us.

4.3 Landing Pages

These images showcase different facets of the ”Nexus Grid” online platform, which is designed for intelligent system monitoring and management.Another image appears to be a central landing page, presenting the ”Nexus Grid” name and its core purpose: ”Intelligent System Monitoring and Management for Seamless Workspace Operations!”, set against a backdrop of icons representing various technological concepts relevant to the service.

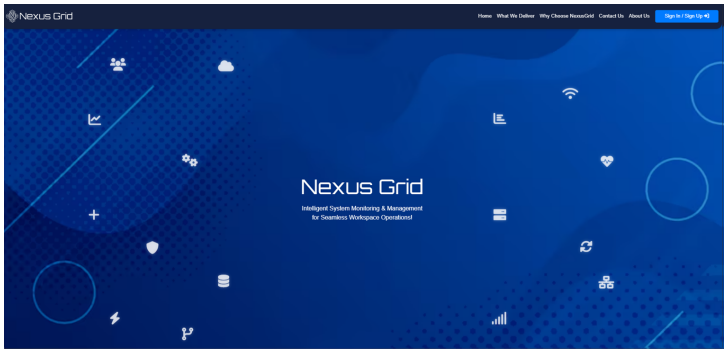


Figure 4.3: Home Page

4.3.1 Services and Features

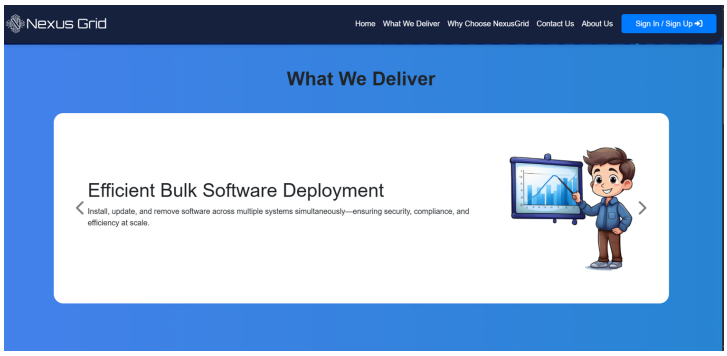


Figure 4.4: Features

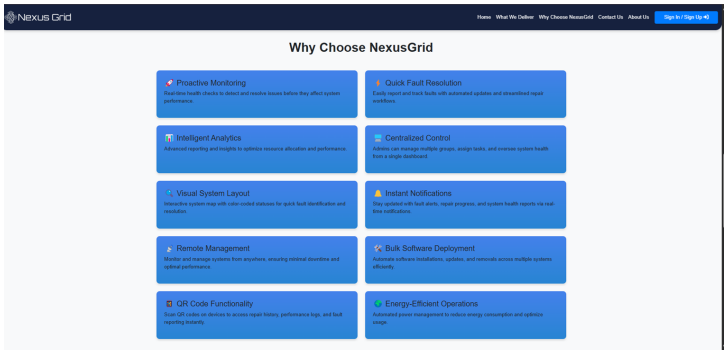


Figure 4.5: Services

A fourth image delves into specific offerings, highlighting ”Efficient Bulk Software Deployment” with a description of its capabilities for streamlined software management across multiple systems, accompanied by a visual aid. Finally, a page titled ”Why Choose NexusGrid” lists numerous advantages in a clear grid layout, such as Proactive Monitoring, Quick Fault Resolution, Intelligent Analytics, and Centralized Control, providing users with compelling

reasons to utilize the platform. All these pages maintain a consistent visual identity with the Nexus Grid logo and navigation links in the header.

4.3.2 How To Communicate?

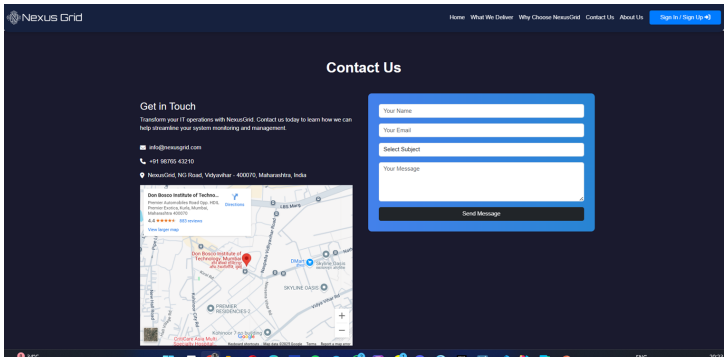


Figure 4.6: Contact Us

This section allows users to get in touch with the NexusGrid team via email, phone, or a contact form. It includes a Google Map for location reference and encourages users to reach out for support or inquiries.

4.3.3 About NexusGrid

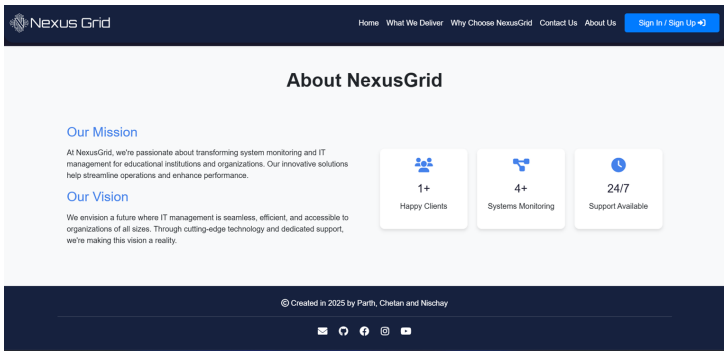


Figure 4.7: About NexusGrid

This section outlines NexusGrid's mission and vision for transforming IT management. It also highlights key achievements like system monitoring count, 24/7 support availability, and client satisfaction.

4.4 Dashboard

The NexusGrid Dashboard offers a visually rich and intuitive interface that provides real-time insights into system performance, fault trends, and user ac-

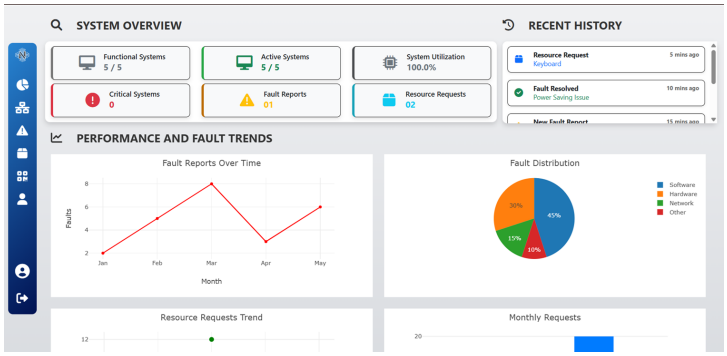


Figure 4.8: Dashboard

tivity. At the top, the System Overview panel summarizes the current status of all monitored systems, including the number of functional and active systems, critical system alerts (if any), system utilization percentage, fault reports, and pending resource requests. This allows administrators to quickly assess the health of the entire infrastructure at a glance.

The Recent History panel on the right side highlights time-stamped activity logs such as resource requests, fault resolutions, and new fault entries, promoting transparency and traceability in operations.

The Performance and Fault Trends section includes analytical graphs:

Fault Reports Over Time displays monthly fault counts, helping track recurring or increasing issues.

Fault Distribution is shown via a pie chart, categorizing faults into software (45percent) (30percent), network (15percent), and other (10percent) segments for better diagnosis and prioritization.

Resource Requests Trend and Monthly Requests (partially visible) provide visual insights into how often users request resources and in what volume.

Together, these components create a comprehensive and interactive command center for efficient system monitoring, maintenance, and decision-making.

4.5 Resource Requests

This interface showcases the Resource Request management module within the NexusGrid system. Users can search and filter requests using criteria such as time range, status, and sorting preferences. The page displays a list of submitted resource requests, each showing the requester's name, location (e.g., IT Lab 6), type of resource (e.g., Peripheral), specific item (e.g., Mouse), and the date/time

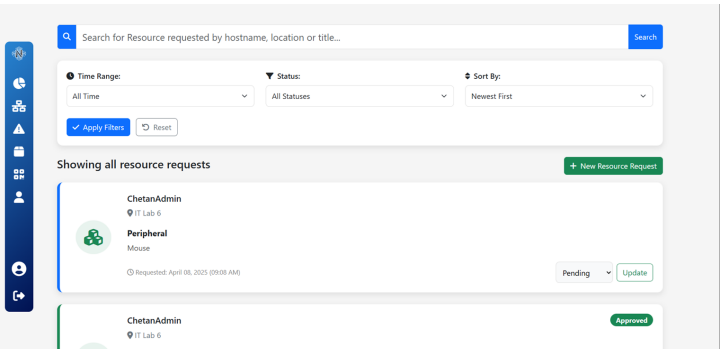


Figure 4.9: Resource Requests

of the request. Each entry includes status options (e.g., Pending, Approved) with the ability to update request states. The clear structure and intuitive controls help administrators and users manage lab equipment or IT asset requests efficiently. A prominent "New Resource Request" button encourages quick and easy submission of additional requests, promoting streamlined operations. The left-side navigation bar provides access to different dashboard modules for a seamless user experience.

4.6 Fault Reports

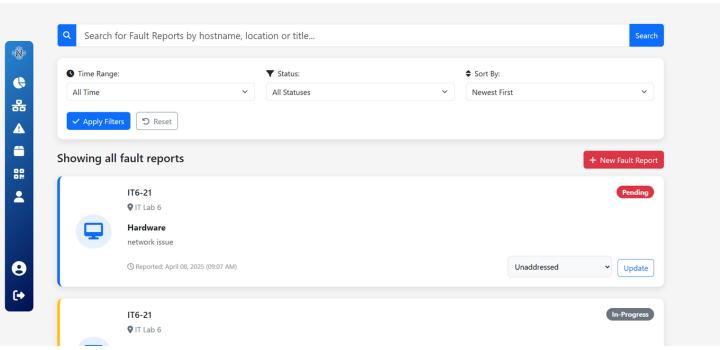


Figure 4.10: Fault Reports

This screen presents the *Fault Reporting* module of NexusGrid, designed for logging, tracking, and managing infrastructure issues within a lab or institution. The user can search faults by hostname, location, or title using a global search bar at the top. Dynamic filters allow narrowing results based on time range, status (e.g., Pending, In-Progress, Resolved), and sorting preferences.

The dashboard lists individual fault reports, displaying information such as system ID (e.g., IT6-21), lab location (e.g., IT Lab 6), type of issue (e.g., Hardware), and a brief description (e.g., network issue). Each report includes a

timestamp indicating when it was submitted and a dropdown to update the fault status (e.g., Unaddressed to In-Progress or Resolved), offering interactive control over incident resolution.

A red **“+ New Fault Report”** button in the top-right enables users to swiftly log new issues, making the reporting process seamless and accessible. Color-coded status labels—such as **Pending** in red and **In-Progress** in grey—enhance visual tracking and prioritization. The left sidebar provides consistent access to different modules, maintaining UI continuity throughout the NexusGrid system.

4.7 QR Code Scanner

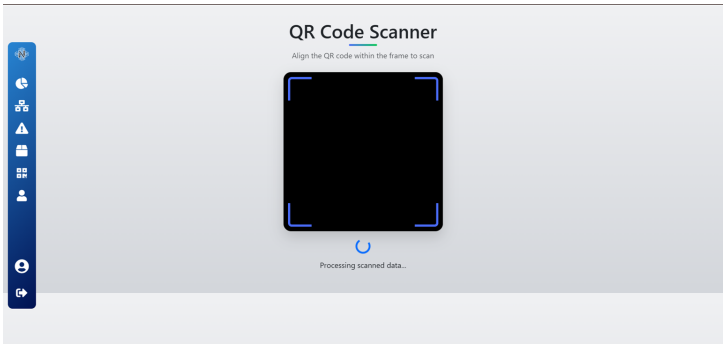


Figure 4.11: QR Code Scanner

This screen represents the QR Code Scanner module of NexusGrid, used for quick identification of assets or reports. Users simply align the QR code within the frame, and the system processes the scanned data automatically.

4.8 System Layouts

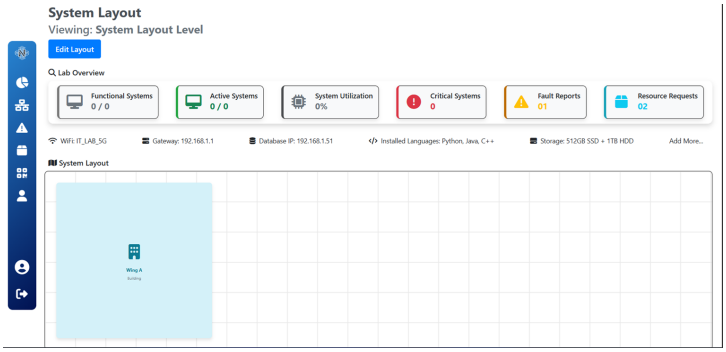


Figure 4.12: System Layout Wing:A

This image shows a system layout dashboard providing an overview of a lab environment. Titled ”System Layout,” it displays key metrics such as functional and active systems, utilization, critical systems, fault reports, and resource requests. Connectivity details like WiFi, Gateway and Database IPs, installed languages, and storage capacity are also presented. A visual area depicts the system structure, showing a ”Wing A Building” element. The interface includes a left navigation bar and an option to edit the layout, offering a centralized view for monitoring and managing the system.

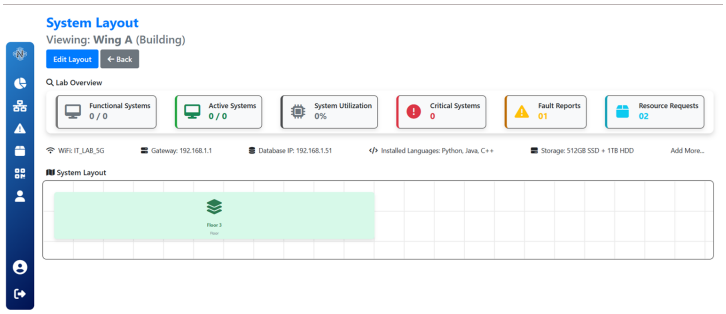


Figure 4.13: System Layout Floor:1

This image displays a system layout interface focused on a specific section, ”Wing A (Building)”, as indicated by the subtitle. Similar to the previous view, it presents key metrics like functional and active systems, utilization, critical systems, fault reports, and resource requests, along with connectivity and system details such as WiFi, IPs, installed languages, and storage. The central visual area now depicts a more granular view within Wing A, showing a box labeled ”Floor 3” with a stacked icon, suggesting a hierarchical representation of the system layout focusing on a particular floor within the building. Navigation options are available on the left sidebar, and ”Edit Layout” and ”Back” buttons allow for modifying the view and returning to a previous level.

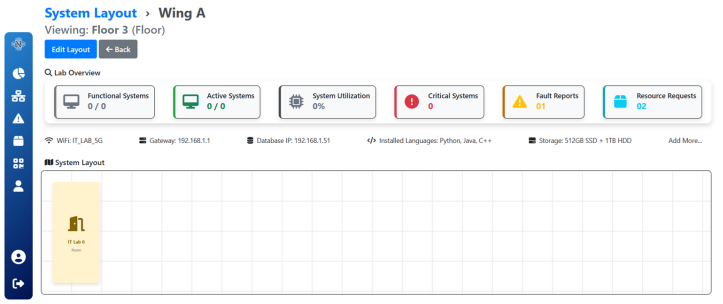


Figure 4.14: IT Lab A Room Layout

Viewing Floor 3 of Wing A, this system layout shows the specific ”IT Lab A Room” within that floor. The dashboard maintains the display of key system metrics and network details relevant to this level.

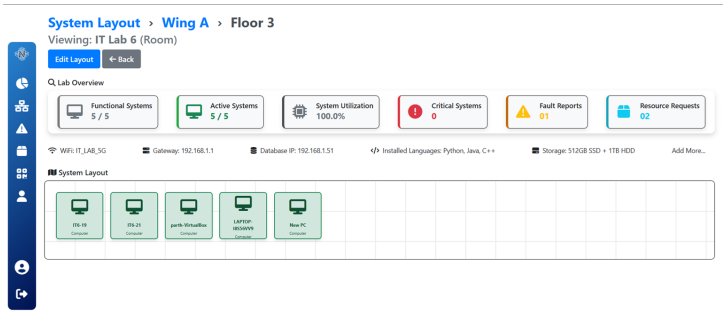


Figure 4.15: IT Lab 6 Room Details

This view of the system layout navigates down to ”IT Lab 6 (Room)” within Wing A, Floor 3. The dashboard reflects the status of this specific room, showing all 5 functional and active systems are in use with 100percent utilization. The visual layout clearly displays icons representing individual computers or systems present in IT Lab 6, each labeled with a unique identifier. This detailed view allows for monitoring the status of individual machines within the room.

4.9 System Information

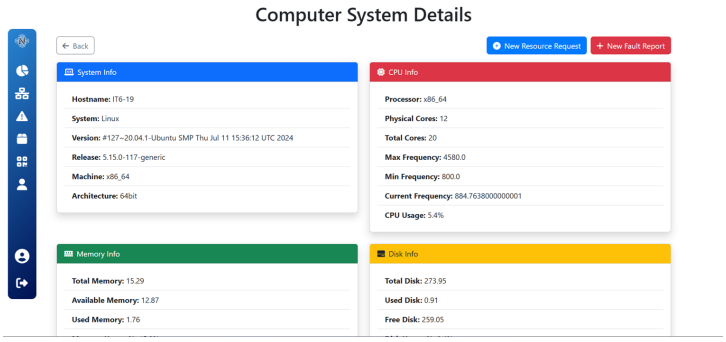


Figure 4.16: System Information

This image displays a detailed view of a single computer system, titled ”Computer System Details”. The information is organized into four distinct sections. The ”System Info” section provides fundamental details including the hostname (IG-19), operating system (Linux), version, release, machine type (x86_64) and architecture (64bit). The ”CPU Info” section offers processor specifications

such as the processor type (x86_64), physical cores (12), total cores (20), maximum, minimum, and current frequencies, and current CPU usage (5.4%). Below, the ”Memory Info” section details the total, available, and used memory in Gigabytes. Finally, the ”Disk Info” section provides a summary of the total, used, and free disk space, also in Gigabytes. Navigation options are available on the left sidebar, and buttons to request a new resource or report a new fault are present at the top right.

4.10 User Privileges

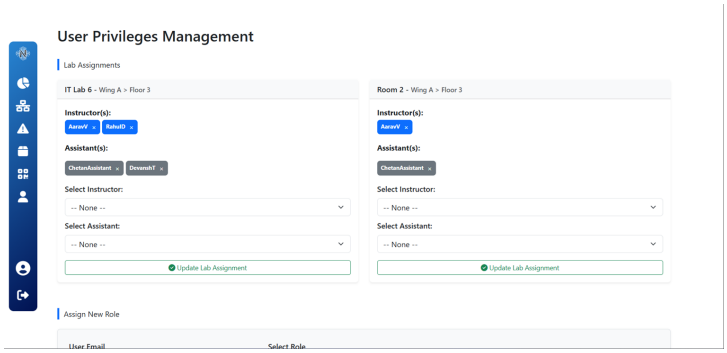


Figure 4.17: User Privileges Management Interface

This image depicts a ”User Privileges Management” interface, designed for controlling access and roles within a lab environment. The primary section, ”Lab Assignments,” is organized by lab location, specifically showing configurations for ”IT Lab 6 - Wing A ¿ Floor 3” and ”Room 2 - Wing A ¿ Floor 3”. For each location, it lists currently assigned instructors and assistants, providing controls to remove existing personnel and dropdown menus to assign new instructors and assistants, with an ”Update Lab Assignment” button to apply changes. A secondary section, partially visible at the bottom, is dedicated to ”Assign New Role,” featuring input fields for user email and selecting a role, indicating the functionality to grant new roles to users across the system. The interface includes a left sidebar for navigation.

This image shows further details of the ”User Privileges Management” interface. It clearly presents the ”Assign New Role” section, where an administrator can enter a user’s email, select a role from a dropdown menu, and click ”Assign Role” to grant system privileges. Below this, the ”Users with Roles” section lists existing users along with their assigned roles and email addresses. This

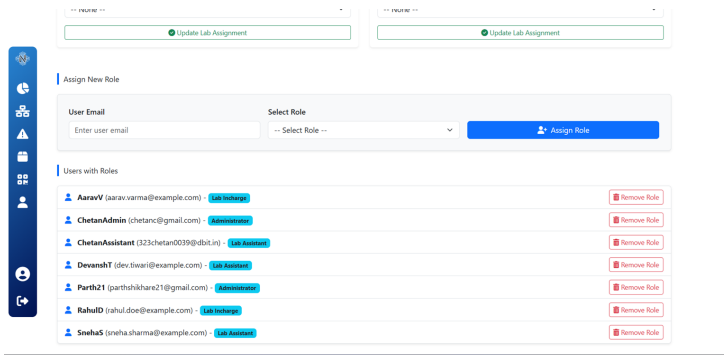


Figure 4.18: Role Assignment and Listing

section provides a clear overview of current user permissions and includes a ”Remove Role” button next to each entry, allowing for the revocation of assigned roles. The interface maintains the left sidebar for navigation.

Chapter 5

Results and Discussion

This chapter would contain the intermediate result and their analysis .Final result and their analysis.In case of application comparison between the your application and existing.

5.1 Intermediate Results and Analysis

Throughout the development of NexusGrid, key modules were tested individually to validate their functionality and integration. The role-based login system and dashboards operated smoothly, ensuring proper access control and navigation. QR code redirection worked efficiently, enabling quick fault identification. Real-time monitoring using psutil accurately captured system stats and displayed them using interactive graphs. Fault reporting, database interactions, and automated email alerts functioned reliably, while frontend validations and OTP-based login added a layer of security. These results confirmed the system's stability and effectiveness during development, paving the way for successful integration and deployment.

5.1.1 Role-Based Login and User Authentication

Intermediate Result: The login and user authentication module was developed using Django's built-in auth system. User roles (Admin, Manager, Employee)

were assigned using Django's Group and Permission models. Passwords were enforced with strength validation, and OTP verification was integrated to add security. Redirect logic was implemented to take users to their appropriate dashboard post-login.

Analysis: Login/logout and session management worked without issues during testing. Each user type could access only their designated dashboard and features. OTP system was tested and verified for accuracy and timely delivery. Secure login practices were successfully implemented and validated.

5.1.2 QR Code System Integration

Intermediate Result: QR codes were generated for each system using the qrcode Python library. Each QR mapped to a unique system ID and was intended to redirect users to a pre-filled fault reporting form. Testing was conducted on both mobile and desktop QR scanning tools.

Analysis: QR scanning redirected users correctly to the respective system fault form. Time taken for reporting reduced significantly—manual system identification was eliminated. Error rates dropped, especially in high-volume environments like computer labs. System responsiveness to QR redirection was smooth and accurate.

5.1.3 Real-Time System Monitoring

Intermediate Result: System health parameters (CPU, memory, disk, temperature) were fetched in real time using psutil and pySMART. Visual graphs were created using Plotly and embedded into dashboards. The backend periodically polled data and updated frontend without refreshing the page (AJAX/Channels concept).

Analysis: Live graphs updated in real time, clearly reflecting system behavior. Performance remained stable even under continuous data fetching. Potential issues like system overheating or high CPU usage were visible at a glance. Created scope for future enhancements like automated alerts based on thresholds.

5.1.4 Fault Reporting and Resource Request Management

Intermediate Result: Fault reports and resource requests were submitted via structured forms from the user dashboard. Data was stored using Django models

and retrieved for tracking/updating by Admin/Manager. Django's email backend was used to send alert mails upon submission.

Analysis: Fault logs were saved and tracked successfully through the system. Admins/Managers were able to update the status (Pending, In Progress, Resolved). Emails were triggered correctly and received on time. Enhanced overall system accountability and transparency.

5.1.5 Frontend Functionality and Validation

Intermediate Result: Forms across modules were validated using JavaScript for completeness and correct input formats. Password fields included live strength meters. The UI was designed using HTML/CSS and Bootstrap to be responsive and clean.

Analysis: All forms were easy to fill and validated input efficiently. Error messages appeared promptly, reducing submission errors. UI worked consistently across devices and resolutions. User feedback was positive on design clarity and interaction flow.

5.1.6 Database and Backend Integrity

Intermediate Result: Database schema was created using Django's ORM, focused on modular and scalable design. CRUD operations were implemented for user data, system metadata, faults, and resource requests. Unit testing was done on Django views, forms, and models.

Analysis: Data stored and retrieved accurately across all modules. Backend functions remained stable even under concurrent usage. Form submissions, dashboard population, and user status updates worked seamlessly. System proved to be scalable and maintainable during code extensions.

5.1.7 Integration Tests and Team Coordination

Intermediate Result: All independently developed modules were gradually merged using GitHub. Team communication was maintained via WhatsApp and regular Google Meet check-ins. Roles were clearly distributed: frontend (Chetan), backend/API (Parth), database (Nischay).

Analysis: Integration was smooth due to modular coding practices. No major merge conflicts or logical clashes occurred. Team stayed on schedule, adapted quickly to issues, and maintained a unified vision. Mentor feedback was incorporated efficiently after each review round.

5.2 Final Results and Analysis

5.2.1 System Performance:

The final implementation of NexusGrid demonstrated excellent responsiveness and smooth functionality across all modules. The user interface was fast-loading and intuitive, offering seamless navigation between login, dashboard, and reporting pages. Role-based access was executed efficiently, with minimal delay during redirection and secure session handling. QR code scanning triggered immediate redirection to fault-reporting forms, enabling users to report issues within seconds. The real-time system monitoring dashboard, powered by psutil and Plotly, continuously updated metrics like CPU load and memory usage without noticeable lag. Backend tasks like database operations, report generation, and email notifications were handled reliably with consistent performance, even during simultaneous user activity. The frontend remained responsive across various screen sizes and devices, maintaining a consistent experience for all user roles.

5.2.2 System Accuracy:

NexusGrid also performed with high precision in terms of data handling, user flow, and monitoring outputs. The system consistently directed users to the correct dashboards based on their assigned roles, enforcing access boundaries accurately. QR code redirection maintained accurate mapping between systems and their digital IDs, with no mismatches recorded during testing. Real-time monitoring values—such as CPU temperature, memory load, and network activity—were captured accurately and reflected immediately on the dashboard, aligning closely with the actual system behavior. Fault reports and resource requests submitted by users were stored without error, and updates to their status were tracked and displayed correctly. Email notifications were reliably triggered and delivered, ensuring timely communication. Throughout testing, no data loss, misrouting, or inconsistencies were observed, confirming the system's overall reliability and high level of accuracy.

5.3 Comparison with Existing Applications

5.3.1 Feature Comparison

Compared to existing system management tools—many of which rely on basic ticketing systems or manual logs—NexusGrid offers a more comprehensive, automated, and interactive solution. Traditional tools often lack features like QR-based fault identification, requiring users to describe the faulty system manually, which increases error rates and delays. NexusGrid simplifies this through unique QR code tagging, ensuring faster and more accurate fault reporting.

While some existing platforms provide monitoring tools, they usually lack role-specific dashboards, making it difficult to segregate responsibilities or streamline views for different users. NexusGrid introduces custom dashboards for Admins, Managers, and Employees, improving workflow clarity and task distribution. Additionally, many current systems don't offer visual interactive workspace layouts, whereas NexusGrid includes a UI-based layout that color-codes system status in real time. Features such as automated email alerts, OTP-secured login, and resource request tracking add further value and aren't commonly available in simpler solutions.

5.3.2 Performance Comparison

In terms of performance, NexusGrid outperforms many conventional platforms that rely on slower backend processing or lack real-time interactivity. While existing tools may take longer to update system status or reflect new fault entries, NexusGrid uses real-time monitoring with libraries like psutil and Django Channels, ensuring live system stats are always current.

Most traditional applications also depend heavily on manual refresh or polling, often leading to data lag or missed updates. NexusGrid maintains high responsiveness, even during continuous data streaming and simultaneous user actions. Email notifications and database updates are near-instantaneous, while frontend responsiveness across mobile and desktop ensures a smooth user experience. Moreover, integration testing proved that NexusGrid scales efficiently with load, which is often a limitation in legacy systems. Overall, NexusGrid not only matches but exceeds the performance benchmarks of many existing infrastructure monitoring solutions in both speed and reliability.

5.4 Summary

NexusGrid is a full-stack, intelligent system designed to address the common challenges of system monitoring, fault reporting, and resource management in institutional and workplace settings. By combining real-time data tracking, QR code-based identification, role-specific dashboards, and automated alerts, it streamlines IT infrastructure management with speed and accuracy. The project leverages Django, Python libraries, and modern web technologies to deliver a scalable, secure, and user-friendly platform. Through continuous testing and refinement, NexusGrid demonstrated strong performance, accurate data handling, and seamless integration of all modules. Compared to existing solutions, it offers superior automation, interactivity, and real-time response, making it a robust and future-ready tool for smart system administration.

Chapter 6

Conclusion & Future Work

This chapter summarizes the key findings and conclusions drawn from the results and discussions presented in the previous chapters. Additionally, it reflects on the lessons learned in terms of project management and outlines potential areas for future improvement and development.

6.1 Conclusion

The NexusGrid project successfully delivers a comprehensive and automated solution for real-time system monitoring, fault reporting, and resource management in institutional and organizational environments. By integrating advanced technologies into a user-friendly platform, it bridges the gap between manual IT maintenance practices and modern digital infrastructure. The project not only fulfills its core objectives but also sets the foundation for scalable and intelligent infrastructure management systems.

Key Outcomes and Highlights:

- **Real-Time Monitoring:** Continuous tracking of system health using Python libraries like psutil and visual dashboards.
- **Fault Reporting via QR Codes:** Easy fault registration by scanning workspace-specific QR codes for quick redirection.

- **Role-Based Dashboards:** Custom dashboards for Admin, Manager, and Employee roles to ensure tailored functionality and access.
- **Automated Resource Allocation:** Smart distribution of tasks and issue assignments with minimal manual intervention.
- **Technology Stack Integration:** Smooth integration of Django, HTML/CSS, Bootstrap, JavaScript, Celery, and Plotly.
- **Secure Login System:** OTP verification, password validation, and multi-platform login support to enhance security.
- **Interactive UI/UX:** Clean, responsive, and intuitive interface that enhances usability and system navigation.

In conclusion, NexusGrid demonstrates the real-world applicability of full-stack web development to solve genuine infrastructure problems in academic and professional settings. Its modular design, focus on user experience, and future-ready architecture make it not just a project, but a prototype for next-generation infrastructure monitoring solutions. The system is scalable, maintainable, and ready for integration with larger IT ecosystems.

6.2 Lessons Learned from Project Management

Working on NexusGrid has been a transformative experience that combined technical learning with real-world problem-solving. The development journey offered deep insights into full-stack web application design, team collaboration, and practical deployment strategies. It also emphasized the importance of user-centric thinking and iterative improvement in creating a reliable and scalable solution.

Key Lessons Gained:

- **Full-Stack Development Exposure:** Gained hands-on experience in Django framework, frontend technologies like HTML, CSS, Bootstrap, and JavaScript, as well as backend logic integration.
- **Modular Design Thinking:** Understood the importance of separating concerns (like login modules, dashboards, QR functionalities) for maintainability and scalability.

- **Security Awareness:** Learned how to implement and validate secure login systems using OTP and password validation techniques.
- **Real-Time Data Handling:** Gained knowledge about using psutil, Django Channels, and other libraries to monitor and display live system stats.
- **Data Visualization and Reporting:** Understood how to present data meaningfully through visual libraries like Plotly to support decision-making.
- **Team Coordination:** Learned how crucial communication and role distribution are in collaborative development.
- **Debugging and Testing:** Understood the iterative nature of testing, debugging, and refining features through real-time feedback.

Through this project, we not only developed a functional tool but also honed the skills required to design and deploy end-to-end solutions. The practical challenges helped us grow technically and adaptively, preparing us for more complex software development tasks in the future.

6.2.1 Requirement Gathering and Understanding

The development of NexusGrid began with a clear focus on identifying real-world issues in system monitoring, fault management, and resource allocation—especially within institutional and organizational environments. To ensure the solution was relevant, user-friendly, and efficient, extensive effort was placed into gathering, analyzing, and refining the requirements at the initial stage.

Key Requirement Gathering activities:

- **Stakeholder Discussions:** Held informal interviews with students, lab assistants, and faculty to understand frequent problems in lab system maintenance.
- **Problem Observation:** Noted the repetitive issues in system health, fault reporting delays, and unstructured communication in physical workspaces.
- **Requirement Documentation:** Drafted initial documents listing functional needs (login system, QR redirection, dashboards) and non-functional needs (scalability, security, UI responsiveness).

- **Use Case Analysis:** Mapped user roles (Admin, Manager, Employee) to corresponding use cases to define specific access levels and workflows.
- **echnical Feasibility Checks:** Assessed the viability of features like real-time monitoring using psutil, background task handling using Celery, and frontend interactivity with JavaScrip

6.2.2 Time Management

Effective time management played a crucial role in the successful execution of the NexusGrid project. The team followed a structured weekly timeline, starting with team formation and domain selection, followed by in-depth research, technology learning, and system design. Clear milestones were set for UI development, backend integration, and feature testing, which helped maintain consistent progress. Regular mentor feedback and internal discussions ensured that tasks were prioritized efficiently and any delays were addressed promptly. By balancing individual responsibilities with collaborative goals, the team was able to stay on track and deliver a fully functional and well-documented system within the academic timeline.

6.2.3 Collaboration and Communication

Strong collaboration and clear communication were at the heart of the NexusGrid project's success. Each team member was assigned distinct responsibilities—frontend, backend, and database—which allowed focused contributions while still encouraging cross-functional support. Regular meetings were held to discuss progress, share challenges, and align on goals, ensuring that everyone stayed updated and coordinated. Tools like WhatsApp,Discord and Google Meet were used for real-time discussions, while shared documents and GitHub helped track code changes and maintain transparency. Constructive feedback from the mentor also played a key role in refining the project. This collaborative environment fostered teamwork, encouraged problem-solving, and ultimately led to a smooth and efficient development process.

6.3 Future Work

Building upon the foundation established by NexusGrid, several avenues can be pursued to enhance its capabilities, scalability, and adaptability to evolving IT environments. By integrating advanced technologies, expanding platform

reach, and deepening analytics, NexusGrid can transform from a robust monitoring tool into an intelligent, predictive, and fully integrated infrastructure management ecosystem.

6.3.1 Predictive Analytics and Machine Learning:

Incorporate anomaly-detection models to predict hardware failures or performance degradation before they occur. Leverage historical system metrics to train algorithms that flag unusual patterns and recommend proactive maintenance.

6.3.2 Microservices and Cloud-Native Deployment:

Refactor core modules into microservices and deploy on Kubernetes or Docker Swarm. This will improve fault isolation, enable independent scaling of services (e.g., monitoring, alerting, reporting), and streamline continuous integration/continuous deployment (CI/CD) pipelines.

6.3.3 Mobile and Voice-Driven Interfaces:

Develop native mobile apps (iOS/Android) to allow administrators and users to monitor system health and receive alerts on the go. Integrate voice assistants (e.g., via Amazon Alexa or Google Assistant) for hands-free fault reporting and status checks.

6.3.4 IoT and SNMP Device Integration:

Extend system health checks to networked hardware—such as routers, switches, and IoT sensors—using SNMP (Simple Network Management Protocol) and MQTT. This will enable comprehensive, end-to-end visibility across all infrastructure components.

6.3.5 Advanced Energy Management:

Integrate with smart power strips and building management systems to not only shut down idle machines but also optimize heating, cooling, and lighting based on real-time occupancy and usage patterns.

6.3.6 Role-Based SLA Tracking:

Introduce Service Level Agreement (SLA) monitoring per department or user group, automatically tracking uptime/downtime metrics and generating compliance reports for managers and stakeholders.

6.3.7 Plugin Architecture and Third-Party Integrations:

Design a plugin framework that allows easy addition of connectors—for example, to Slack, Microsoft Teams, Jira, or PagerDuty—so that alerts and reports can flow directly into existing organizational workflows.

6.3.8 Enhanced Reporting and Dashboards:

Provide customizable dashboard widgets and drill-down analytics, with support for geospatial mapping of distributed assets, heat maps of usage, and multi-dimensional pivot tables for deeper insights.

Next-Generation Vision: By pursuing these enhancements, NexusGrid can evolve into an AI-driven, cloud-native platform that seamlessly bridges the gap between system operations and strategic IT decision-making. Future iterations will not only monitor and report but also intelligently orchestrate resources, optimize energy usage across entire facilities, and embed itself as an indispensable tool in any forward-looking IT department.

6.4 Summary

The presentation introduces *NexusGrid*, a Python-based mini-project designed to automate system monitoring, fault reporting, and resource management in workspaces. It addresses common issues in manual systems, such as delayed fault detection, inefficient resource allocation, and lack of real-time monitoring, which contribute to downtime, reduced productivity, and increased operational costs.

NexusGrid features role-based dashboards tailored for Admins, Supervisors, and Users. Key functionalities include real-time system health monitoring (CPU, network, temperature), simplified fault reporting with tracking, resource request management, interactive workspace layouts with visual system status indicators, bulk software management, QR code integration for system identification, and detailed analytics for performance, faults, and resource usage.

The technology stack includes HTML, CSS, Bootstrap, and JavaScript for the frontend; Django (Python) for the backend; and SQLite for the database. Various Python libraries support tasks such as real-time graphing, background processes, email notifications, system monitoring, automation, data analysis, and QR code generation. The presentation also outlines the project timeline and the division of technical responsibilities across development areas.

References

[1] Django Documentation: https://www.djangoproject.com/

[2] Chart.js: https://www.chartjs.org/

[3] ReportLab: https://www.reportlab.com/

[4] SQLite Documentation:https://www.sqlite.org/docs.html

[5] PostgreSQL:https://www.postgresql.org/docs/

[6] jango Video Tutorial:https://www.youtube.com/watch?v=rHux0gMZ3Eg

Acknowledgements

We, the undersigned, would like to extend our sincere gratitude to everyone who supported us throughout the completion of our college mini project. First and foremost, we are deeply thankful to our project guide Prof. Shiv Negi, for their expert guidance, constructive feedback, and constant encouragement. Their insights into software development and practical knowledge helped us navigate the challenges we encountered during the development of this System Monitoring and Control software.

We would also like to thank our college for providing us with the resources and opportunity to work on this project. The experience has enhanced our understanding of software design, teamwork, and project management. Finally, we are immensely grateful to our families and friends for their unwavering support, understanding, and motivation throughout this project.

(————————————-)
Parth Shikhare - 53

(————————————-)
Nischay Chavan - 14

(————————————-)
Chetan Chaudhari - 10

Date: