International Journal of Computer Techniques–IJCT Volume 13 Issue 3, May-June - 2026

Open Access and Peer Review Journal ISSN 2394-2231 https://ijctjournal.org/

Design And Implementation of a Real Time Campus Service Management System

Udosen, Alfred Akpan[1], Ademeso, Ademola Michael[2*], Ogunde, Stephen Oluwatofunmi[3], Udofa, Ekomobong Anthony[4], Daramola, Christiana Jumoke[5]

[1,2,3,4]*Department of Computer Science, School of Computing, Babcock University, Ilishan-Remo, Ogun State, Nigeria*

[5]*Department of Software Engineering, School of Computing, Babcock University, Ilishan-Remo, Ogun State, Nigeria*

[*]***Corresponding Author**: ademesoa9735@student.babcock.edu.ng*

Abstract

Communication and responsibility that concerns technical support services on today's modern university campuses can be fragmented. This study proposes CamPulse, a live campus service management system that can centralize and streamline the reporting and management of technical problems with a main objective to create a single Web-based platform accessible to students, technicians and administrators through diverse interfaces to replace the previous mundane manual methods such as telephone calls and office visits. Current architecture that includes a socket service for live communication, a Node backend, and a React frontend, connected to a PostgreSQL database were used to develop the solution, and the results of this research provided a complete set of tools: smart ticket submission, instant status update with an effective network backup plan and detailed performance analysis for management, making the platform highly responsive and able to process multiple simultaneous requests and deliver a professional user experience. Finally, the system demonstrates how an affordable, lightweight app solution can strongly support a proactive culture for providing support at an educational institution without the cost of enterprise-level software. Future versions of the system are recommended with hardware asset tracking, dedicated mobile applications for Android and iOS and predictive analytics that can forecast equipment failures before they happen.

Keywords: *Campus, Information and Communication Technology, Management System, Real Time, Support System*

1. Introduction

The role of Information and Communication Technology (ICT) in the academic and administrative activities of universities has been growing. Online learning, online examination, data management and communication on campus rely on well-developed infrastructure. But there are still many institutions that use informal or fragmented approaches to handle IT services, especially incident reporting, service request processing and complaints resolving. These approaches often bring about delays, redundant work, and low accountability among IT support teams (Hamidu, 2025). In Nigerian universities, technical problems are generally reported through email, telephone or face-to-face basis, often leaving the report poorly documented or lost. If there is no central service tracking, universities face challenges in maintaining resolution timelines, identifying recurring issues, or effectively staffing services (Adesina & Ademola Popoola, 2023).

2. Problem Statement

If a network disconnects or a portal stop functioning on a university site, it's often a messy process to report it. Pupils and staff are trapped in a situation where they are making random emails or physically walking to the ICT unit to gripe face to face. There is no real structure to this communication, it can lead to reports going astray, response delays, and there is virtually no accountability. The root issue is a huge lack of visibility. Users are unable to monitor request status and management does not have the information about staff performance. There are large enterprise IT tools available to address these very issues, but they are too costly and too over-complicated for a typical university to manage.

3. Aim and Objectives

The major goal of this project is to develop and deploy a real-life campus service management system. Specific Objectives: To create a web application for the campus community to report faults centrally, to develop a user-friendly web-based dashboard to monitor the status of the faults and to test the final system to ensure it is efficient and reliable for normal operations of the campus.

4. Literature Review

The management of information technologies is a necessity in today's universities. Many universities and colleges are beginning to use proven industry frameworks, such as ITIL to restructure the processes of support that are not organized. A recent study actually tested ITIL principles with the particular group of campus IT management and discovered real measurable improvements in the dependability of systems and the speed of workflows (Machaladze, 2025).

Alternatively, the commercial aspect is dominated by giant cloud platforms having a wide range of automation and control over the monitoring of services, such as ServiceNow. In most cases, however, these enterprise tools are not technically feasible or economically viable for a smaller educational institution (Okechukwu & Yusuf, 2024). This budget constraint makes the selection of open-source solutions such as osTicket and GLPI (Ahmed & Patel, 2023) a more popular choice on campus. These platforms provide helpful ticket tools, but often don't include user feedback tools and the performance dashboards needed to successfully oversee an instructional environment.

For the universities in the developing areas, the practical situation is faced with a high restriction on budgets, unpredictable power networks and lack of technical staffs for dedicated tasks. It is quite challenging to pay for and sustain large enterprise software, and a lot of campuses resort to manual reporting (Ogunyemi et al., 2023). The answer to this is to create lightweight, custom-made web-based applications tailored for the campus to stay within budget and highly functional.

4.1 Review of Related Works

There are some studies that have examined the modernisation of technical support in educational settings. Machaladze (2025) studied the IT infrastructure management within a framework provided for standard practices and showed that with the implementation of the structured practices, the reliability of the IT systems can be improved in a measurable way. In this study, however, the emphasis was much more on infrastructure, as opposed to tracking of user complaints. The implementation part, Gwanmak and Thomas (2024), created a web-based support desk for a Nigerian University. Their platform was able to effectively capture incidents and assign technicians, but did not have the modern analytics and complaint tracking system.

Likewise, Kristianto et al. (2024) used rapid application development to create a quick service request system, however, the solution did not include important components such as feedback mechanisms for students and escalation handling procedures. Furthermore, there are a wide range of prevalent opensource platforms like the osTicket Project and the GLPI Project, which offer powerful ticketing systems. and asset management capabilities (osTicket Project, 2024; GLPI Project, 2024). The enterprise platforms, however, can be technically complex, and may not come with a universe-particular logic that a small campus might need.

4.2 Gap Analysis

Critical study of the relevant literature evidently shows a definite lack in the management services provided for developing the skills in tertiary establishments as the existing systems did not have user feedback mechanisms or have large open-source systems that are complex and resource intensive for a small university to sustain. Furthermore, none of the analyzed local implementations added support for live communications over socket in addition to comprehensive administration analytics. This project solved the problem through a lightweight, custom built web application with incident logging, live status, and performance information in a single application tailored to the operational challenges of a modern campus.

5. Methodology

The Software Development Life Cycle adopted for this project was the incremental approach. In this particular model, the engineering process is divided into smaller steps and a functional version of the software is produced early in the process for testing and modification.

5.1 Functional Requirements

During the analysis phase, the following functional requirements were identified for the system to meet the gaps:

1. Students/Staff should be able to create incident requests, include image evidence and monitor the status of the request in real time.
2. Technicians need to be able to see their designated ticket queues, update the resolution and communicate with each other.
3. Administrators should have the ability to configure users' roles, control departmental categories, and create visual performance reports.
4. Live alerts should be pushed to active users, without forcing them to refresh their browsers.

5.2 System Architecture

The developed system, CamPulse's architectural structure is based on a basic three-tier structure:

1. The Presentation Layer: It is developed with React and Tailwind CSS, and is responsible for everything the user interacts with and captures all user inputs.
2. The Application Layer: It is the core intelligence power that manages user authentication, notifications, and ticket routing. Executed on the server, it leverages the capabilities of Node and Express.
3. Data Layer: Primarily, PostgreSQL database was used for its data storage as it guarantees the security of all user accounts, service, and incident reports.

5.3 Dual Protocol Communication

We employed a twin protocol communication solution due to the amount of traffic expected in a campus-based IT environment. In addition, the system uses the standard Hypertext Transfer Protocol (HTTP) requests to load basic data and WebSockets to get updates in real-time. When a ticket is modified via an API call, the server will process the change and send a real-time notification via the socket channel to every

client that needs to be notified, without the need to refresh the browser. The architecture of the platform, which includes how the front-end, back-end, and database are connected, is shown in Figure 1 below.

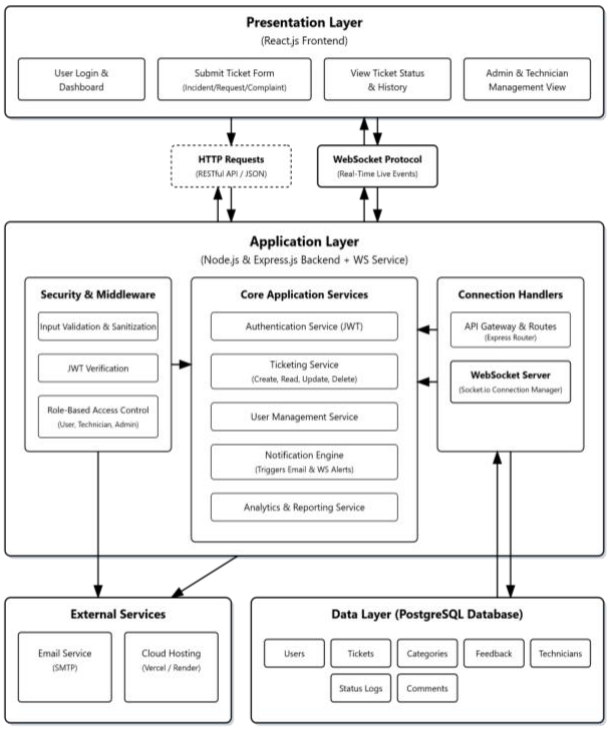


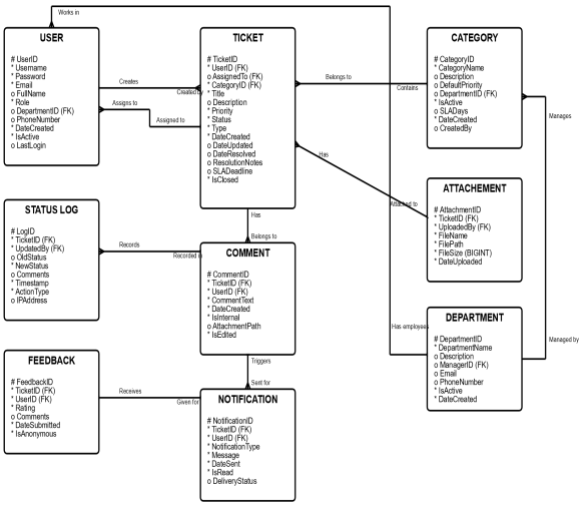
Figure 1: System Architecture Diagram

5.4 Database Design and Entity Relationship

The database has been designed with high relational integrity for all platform operations. The elements of the core entities and their relationships are:

1. Users Table: Credentials, access Level (Student, Technician, Admin), and Department Foreign Keys.
2. Tickets Table: The table that holds the issue description, priority, current status, and timestamps. It has many-to-one relationships with Users table (as creator) and Technicians table (as assignee).
3. Feedback and Logs Tables: These are additional tables that are associated to a particular ticket and record the changes in its state over time and also user satisfaction after the ticket has been resolved.

The Entity Relationship Diagram shown in Figure 2 below describes the structure of the database and how the various entities are logically related within the system.



.Figure 2: Entity Relationship Diagram

5.5 System Flow and Interaction Diagrams

To fully visualize the inner workings of the CamPulse platform, behavioural and structural models were created to show the flow of data through the platform as well as the interactions which the various users have with the tools available to them.

5.5.1 Use case Diagram

Use case diagram is used to represent different interactions in between three main actors and platform functionalities. Students can only create new tickets, and monitor their own requests and resolution feedback. Technicians can now access and claim assignments and update job statuses, and Administrators have full access to all of the control functions: user management, report generation, etc. We defined the interactions between the different system actors and the platform functionalities by depicting the use case diagram as shown below in the figure 3.

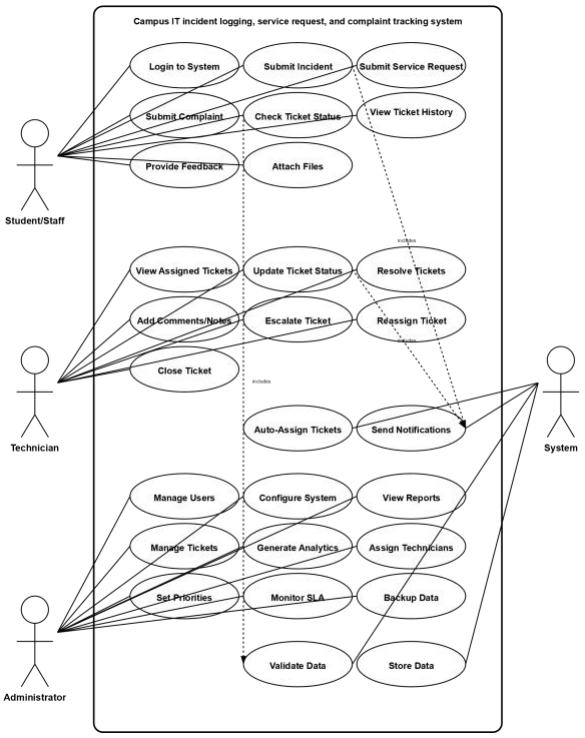


Figure 3: Use Case Diagram

5.5.2 Sequence Diagram

The sequence diagram shows the time-order of the messages exchanged among the different components of the system in a typical operation. A student submits a ticket, and the React frontend will make a secure request to the Node backend. The server decides on the priority logic, stores the record in the PostgreSQL database and immediately notifies the assigned technician with a live update over the WebSocket connection. The sequence diagram is shown in figure 4 below which shows the chronological order of messages and events in the system.

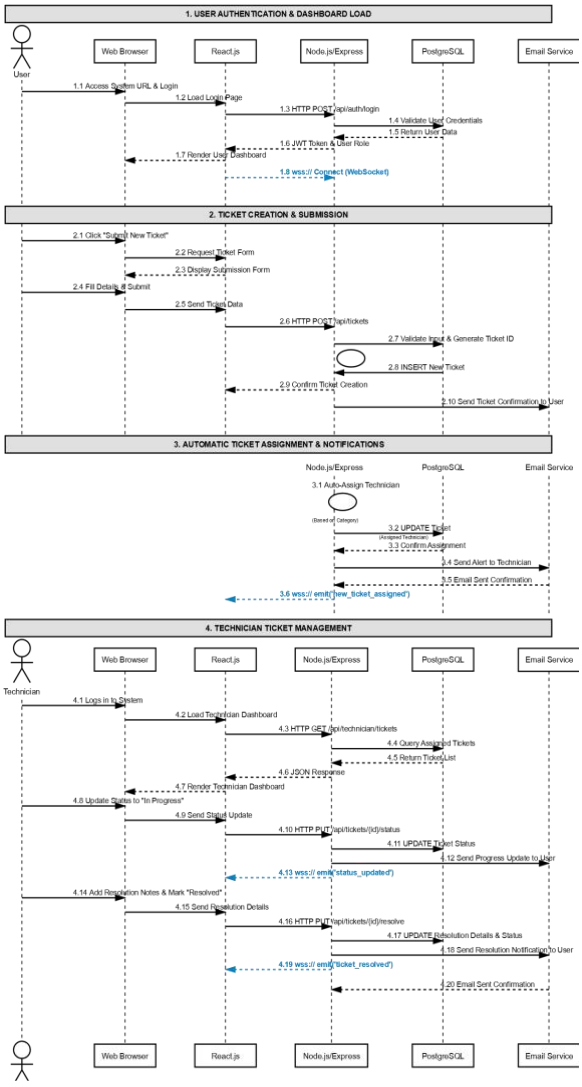


Figure 4: Sequence Diagram

5.5.3 Flowchart Diagram

A system flowchart is used to show the logical flow of a user from reporting the issue to the end of the process where the issue is resolved. After logging on and completing a new incident/service request, the system automatically validates the information and creates a ticket ID. This request is then securely stored in the database and forwarded to the relevant staff member of the IT sector according to its priority level and type of the request. The system sends a dual notification protocol as the ticket moves through various stages, including in progress and eventually resolved, as the technician does. A real-time WebSocket event updates the user's dashboard,

which the user will see without refreshing the browser. Meanwhile, an automated email is sent, to ensure a permanent record of the update. Once the technician has resolved the issue, the student is asked for a comment prior to the issue closing. The entire process enables administrators to extract accurate performance reports and ensure that every service interaction is completely traced with transparency. The flowchart diagram in Figure 5 below shows the logical step by step processes of the application.

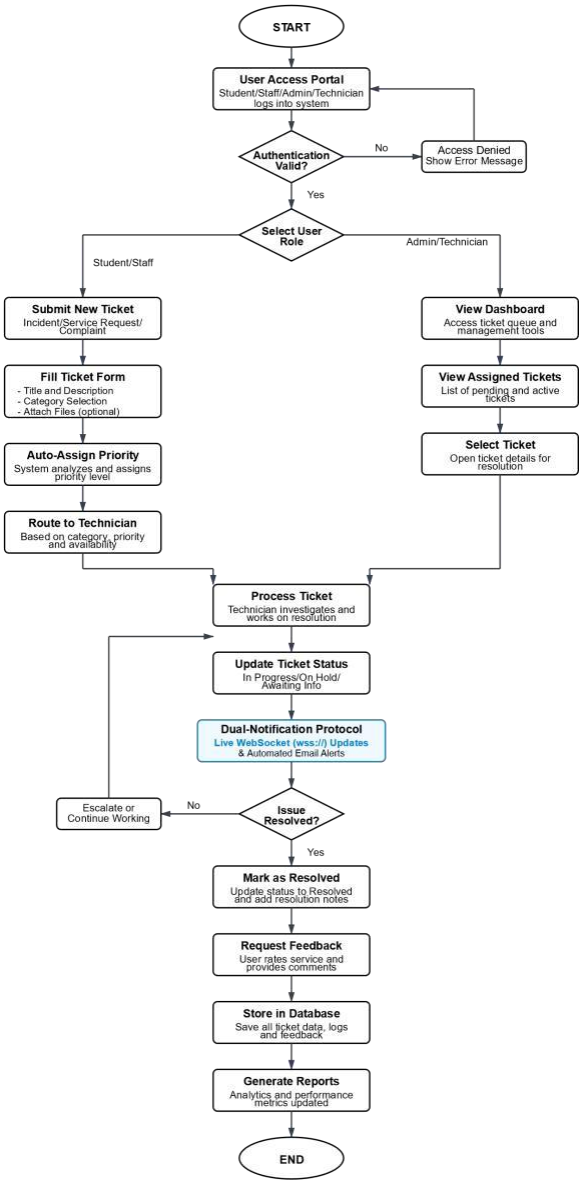


Figure 5: Flowchart Diagram

6. Implementation and Results

The system was powered using the latest web tools, static site hosted by Vercel and the server hosted by Render. Lasting fifteen different visual interfaces, the system design was translated into a working application.

6.1 User Interfaces

The platform is completely role-aware, which means that the layout of the platform automatically adjusts to the user's permissions.

1. Service Desk: A smart submission form is used by students and staff for submitting documents, with automatic priority detection according to the text entered along with drag and drop submitted files. A real-time queue of their open tickets is shown in the user dashboard, each with a colour coded status.
2. Specific Tickets: The dedicated space for the Information Technology (IT) support team displays only the tickets that are assigned to them, along with real-time monitoring of workloads and activity visualized on charts.
3. Administration Panel: Administration team gets a high-level view of the whole campus using detailed graphs and statistics created by the library Recharts. They can check on ticket allocation and the effectiveness of the IT team in solving tickets.

The user dashboard is presented in figure 6 below, which will give students an overview of their active requests. However, Figure 7 below emphasizes the technician console, which provides IT personnel with an allotted area to handle their queue.

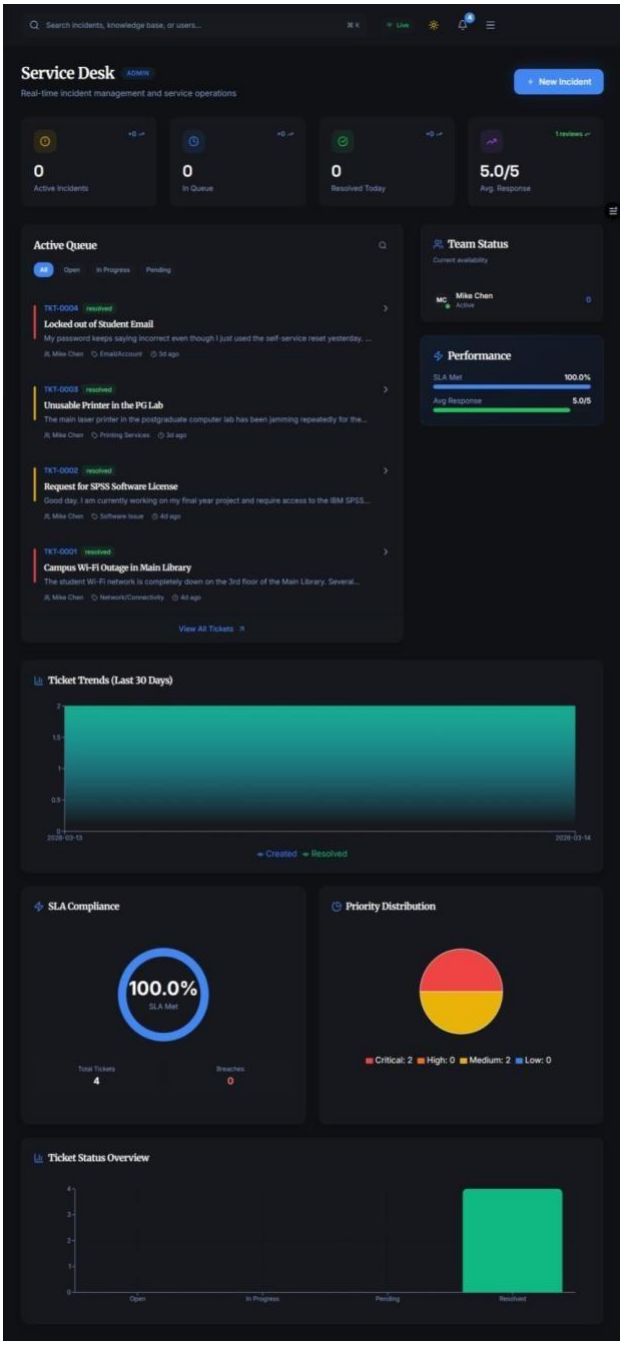


Figure 6: User Dashboard

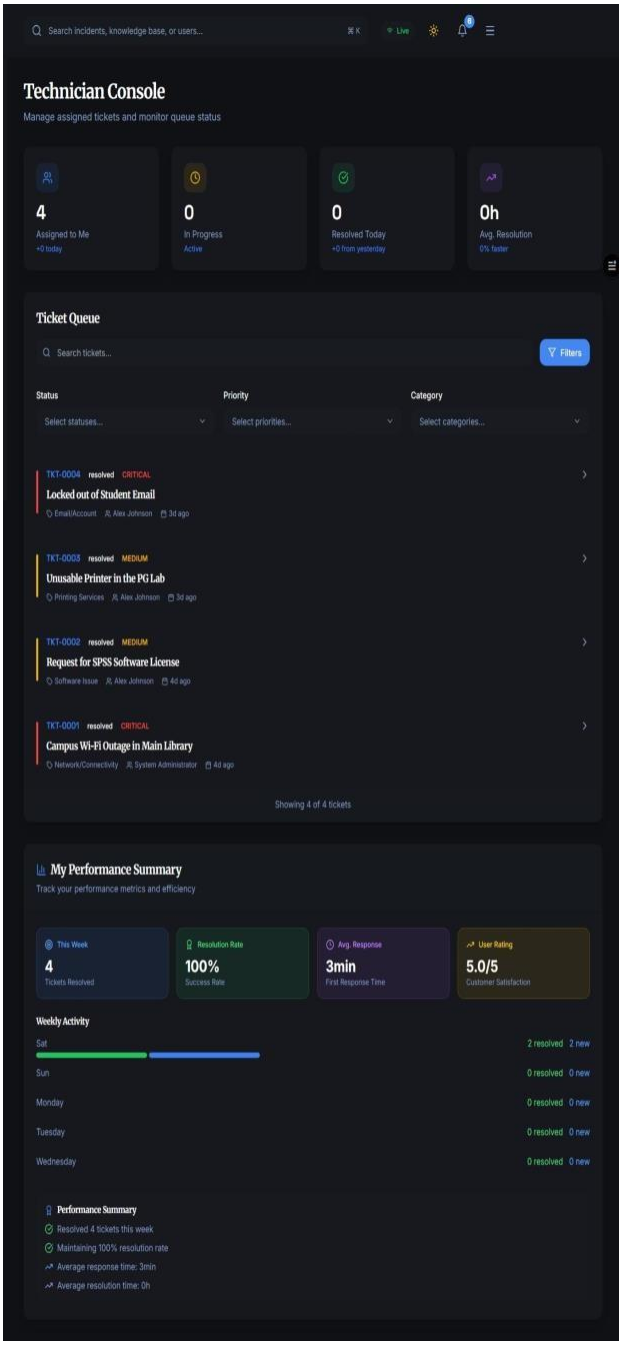


Figure 7: Technician Dashboard

6.2 System Evaluation and Testing

The completed program was tested extensively to ensure that it met all technical and operational requirements. Test case scenarios were then run to verify the functional and performance criteria of the system with simulated campus loads. The results of the functional test and performance evaluation are shown in table 1 below.

Table 1. Functionality and Evaluation of Systems

Test Scenario	Expected Outcome	Actual Observation	Status
Ticket Submission	System automatically detects priority and saves the record.	Priority assigned correctly based on keywords; data stored securely in PostgreSQL.	Pass
Role Based Routing	Ticket is routed exclusively to the relevant department technician.	Technician dashboard updated dynamically with only relevant category tickets.	Pass
Live Notifications	Alert delivered via WebSocket without manual page refresh.	In-app notification delivered with an average latency of under 100 milliseconds.	Pass
Network Fallback	System switches to HTTP polling if WebSocket drops.	System silently transitioned to 30-second polling during simulated network drops.	Pass
Usability Check	Users can navigate the lifecycle without technical assistance.	Sample group successfully created, tracked, and closed tickets independently.	Pass

Speed tests were conducted showing that each page was loaded in less than 2 seconds and at the back end several users could submit tickets concurrently without any slowdown. In the network fallback test, the seamless switching of the system was observed, which showed its fault tolerance.

7. Conclusion and Recommendation

7.1 Summary and Conclusion

The primary objective of developing the CamPulse platform was to address the current slow and unorganized manner in which IT support is usually provided in a university campus. With only one central web application working for all involved, we were able to replace the scattered methods of communication.

The development of this system has demonstrated the impact of well-designed IT service system on day-to-day campus operation by ensuring the students and staff can report issues and monitoring them in real time. Lastly, CamPulse demonstrates how a university could not be required to invest millions of dollars in some huge enterprise application to deliver quality IT support. This light-weight, custom application was designed for university campuses foe easy promotion od a proactive support culture and maintain a well-functioning campus technology.

7.2 Recommendations for Future Research

Leveraging the success of the current system, its capabilities can be improved through:

1. Predictive Maintenance: Leveraging the richness of ticket data, machine learning model should be used to predict hardware failure in advance.
2. Native Mobile Applications: Native applications for iOS and Android would allow for the submission of problems, including photos taken with their phone camera by the students in real time.
3. Smart Language Processing: Advanced natural language processing can ensure that
4. the software can be made to identify duplicates of inbound requests and to

automatically categorize them accurately.

5. Extension: If the users' internet connection fails, they can still be alerted to important events through an SMS gateway, thus extending the reach of the notification engine.

References

Adebayo, S., & Falola, T. (2021). Evaluating the effectiveness of ICT-based service management practices in Nigerian tertiary institutions. *African Journal of Information Systems and Technology, 8*(1), 112–121.

Adesina, D. S., & Ademola Popoola, D. (2023). Exploring the status and challenges of ICT network services and broadband utilization in Nigerian universities. *Nigerian Journal of Technological Development, 22*(1), 8–14.

Ahmed, N., & Patel, A. (2023). An evaluation of open-source helpdesk platforms for higher education: Features, deployment, and analytics. *International Journal of Educational Technology in Higher Education*.

Aljarallah, M., & Alharbi, H. (2023). Usability testing and user experience evaluation in web applications: A practical framework for academic systems. *International Journal of Human–Computer Studies*.

GLPI Project. (2024). *GLPI: Smart IT service management, helpdesk & asset tracking*. Retrieved from https://www.glpi-project.org/

Gwanmak, B. P., & Thomas, G. (2020). Development of a help desk support system (A case study of the University of Jos ICT Directorate). *International Journal of Informatics, Technology & Computers, 6*(1), 1–15.

Hamidu, M. (2025). Evaluating the role of ICT in enhancing service delivery in tertiary institutions in Nigeria. *International Journal of African Innovation and Multidisciplinary Research, 7*(2).

Machaladze, O. (2025). IT infrastructure management in educational institutions using the ITIL framework. *International Scientific Journal of Engineering and Applications*.

Nassif, A. B., & Capretz, L. F. (2021). Evaluating software reliability through functional and regression testing techniques. *Journal of Systems and Software Engineering*.

Ogunlade, A. T., Musa, A. O., & Nwankwo, C. (2023). Implementation and evaluation of lightweight IT service platforms for Nigerian universities. *African Journal of Computing and ICT, 18*(2), 44–57.

Ogunyemi, T., Ajayi, O., & Lawal, F. O. (2023). Assessment of ICT infrastructure and support services in Nigerian universities: Challenges and prospects. *International Journal of Innovative Research in Computer Science and Technology, 11*(3), 45–53.

Okechukwu, M., & Yusuf, S. (2024). Evaluation of open source ITSM tools for small scale educational institutions. *International Journal of Computer Applications and Education*.

osTicket Project. (2024). *Open-source ticketing system platform*. Retrieved from https://osticket.com/

Srinivasan, V., Banerjee, R., & Qureshi, M. (2022). Web application performance testing and benchmarking under concurrent user load. *IEEE Access, 10*, 67532–67545.