IoT-Enabled Campus Digital Signage Framework using Flask-Based Edge Computing

G. Tejaswi[1*], Mopuru Rajesh[2], Veeramreddy Pratap Reddy[2], Balli Sarath Chandra[2], Nanda Bhaskar Rao[2]

[1]Assistant Professor, [2]UG Student, [1,2]Department of Electronics and Communication Engineering

[1,2]Geethanjali Institute of Science and Technology, Nellore-Bombay Highway, S.P.S.R, Andhra Pradesh 524137, India

Correspondence: G. Tejaswi (gtejaswi@gist.edu.in)

To Cite this Article

G. Tejaswi, Mopuru Rajesh, Veeramreddy Pratap Reddy, Balli Sarath Chandra, Nanda Bhaskar Rao, "IoT-Enabled Campus Digital Signage Framework using Flask-Based Edge Computing", Journal of Science Engineering Technology and Management Science, Vol. 03, Issue 04, April 2026,pp: 16-24, DOI: http://doi.org/10.64771/jsetms.2026.v03.i04.pp16-24

Submitted: 18-02-2026 Accepted: 23-03-2026 Published: 01-04-2026

Abstract

Traditional information dissemination in educational institutions relies heavily on manual, paper-based notice boards, which are prone to delays, high maintenance costs, and physical constraints. This paper proposes a centralized, real-time IoT-based Campus Display Board (ICDB) system powered by the high-performance Raspberry Pi 5. The proposed architecture integrates a web-based administrative server with distributed edge nodes via a secure IoT communication layer. Utilizing a Flask-driven backend and a dynamic HTML5/CSS3 frontend, authorized administrators can remotely broadcast multimedia content—including high-definition video, static images, and emergency alerts—from any location. The Raspberry Pi 5 serves as the core processing unit, leveraging its quad-core capabilities and improved networking stack to ensure lag-free rendering and continuous synchronization with the server via a polling-based manifest protocol. Experimental results indicate that the system significantly reduces the latency between announcement creation and public display compared to conventional methods. Furthermore, the modular design promotes sustainability by eliminating paper waste and offers a scalable infrastructure for future smart-campus integrations, such as AI-driven content scheduling and automated cloud synchronization. This research demonstrates a cost-effective, energy-efficient, and user-friendly solution for modernizing institutional communication and enhancing administrative transparency in a digitally driven academic environment.

Keywords: Internet of Things (IoT), Raspberry Pi 5, Digital Signage, Flask Framework, Smart Campus, Real-time Communication.

I. Introduction

The rapid evolution of the Internet of Things (IoT) has catalysed a paradigm shift in how institutional data is managed and broadcast. In modern academic environments, the efficient flow of information—ranging from administrative circulars and examination schedules to emergency notifications—is critical for operational success. Historically, campuses have relied on physical notice boards or localized digital displays that require manual updates via physical storage media. These legacy systems introduce significant bottlenecks, leading to information lag and increased human error. As universities transition toward "Smart Campus" architectures, there is a burgeoning requirement for automated, remotely accessible communication platforms. The emergence of high-performance single-board computers (SBCs), specifically the Raspberry Pi 5, provides the necessary computational overhead to handle complex multimedia rendering at the edge. Unlike its predecessors, the Raspberry Pi 5 features a

significant increase in CPU and GPU performance, enabling the simultaneous handling of encrypted IoT data streams and high-definition video output.

This paper introduces an IoT-based framework that centralizes campus communication through a secure, web-accessible portal. By leveraging Python-based server logic and an IoT communication bridge, the system allows for the instantaneous propagation of data across multiple display nodes. The significance of this work lies in its dual-purpose design: it addresses the immediate need for real-time institutional transparency while providing a scalable foundation for a broader smart-city ecosystem. The following sections detail the system architecture, hardware-software synergy, and the performance benchmarks achieved during deployment.

2. Related Work

The evolution of automated campus communication has transitioned from short-range wireless protocols to sophisticated cloud-integrated frameworks. Early implementations focused on localized wireless connectivity. Gaikwad et al. [15] demonstrated a cost-effective Bluetooth-based system using Arduino, though it was limited by proximity constraints. To address range limitations, Kaur et al. [14] utilized the ESP-32's Wi-Fi capabilities for real-time communication, while Sri Lakshmi et al. [19] and Gemeda et al. [16] explored broader wireless electronic notice board architectures to reduce manual intervention. The integration of Single Board Computers (SBCs) marked a significant shift toward multimedia handling. Ganesh [20] and Bhamre et al. [2] established the feasibility of using Raspberry Pi as a central controller, emphasizing low power consumption and the ability to handle more complex data than standard microcontrollers. Chavan and More [11] furthered this by demonstrating the scalability of Raspberry Pi-based systems in handling remote uploads. Recent advancements have introduced specialized input methods, such as the speech-recognition-operated board developed by Pawar et al. [12], which enhanced accessibility for non-technical users.

Recent literature highlights the shift toward cloud-centric and "Smart City" applications. Mishra et al. [6] and Poojary et al. [3] integrated cloud platforms for secure, centralized data storage, enabling updates from any geographical location. In the context of smart infrastructure, Gupta et al. [5] and Chinnasamy [10] proposed IoT frameworks designed for urban management and public alerts. Security has also become a focal point; Jadhav et al. [4] implemented rigorous authentication mechanisms to prevent unauthorized content updates in large-scale campus deployments. The ICDB system proposed in this study builds upon these foundations specifically the Raspberry Pi architectures discussed by Suryawanshi et al. [1] and Sharma et al. [9] but advances the state-of-the-art by utilizing the Raspberry Pi 5. This allows for superior quad-core processing and high-definition Flask-based rendering, addressing the performance bottlenecks noted in earlier 2019-2022 implementations.

2.1 Research Gaps

Despite the advancements in IoT-based digital signage, a critical analysis of the related work reveals several unresolved challenges:

1. Most previous studies [2, 11, 20] utilize older iterations of the Raspberry Pi or low-power microcontrollers like ESP-32 [14] and Arduino [15]. These platforms often struggle with high-bitrate 4K video rendering and simultaneous background data polling, leading to UI lag and synchronization delays.
2. Existing systems often rely on manual page refreshes or simple timed intervals that do not verify content integrity. There is a notable absence of sophisticated manifest-based polling that compares server-side state with local client-side state to ensure 100% display accuracy.
3. Many proposed solutions [15, 19] focus on short-range or single-node communication. There is a lack of a robust, centralized web architecture that can manage a multi-node ecosystem where different departments can independently update specific displays while maintaining global institutional branding.

4. Traditional systems frequently use basic TCP/IP or outdated HTTP methods. Very few incorporate modern web frameworks like Flask combined with HTML5/CSS3 carousels, which offer superior flexibility in rendering responsive and aesthetically modern UI/UX for academic environments.

2.2 Main contributions of the proposed ICDB system

The ICDB system addresses the aforementioned gaps through the following contributions:

- By utilizing the Raspberry Pi 5, the system provides a significant leap in computational power, enabling seamless transitions between 4K multimedia content and static announcements without performance degradation.
- The system introduces a JavaScript-based synchronization engine that fetches a JSON manifest from the Flask server. It performs a real-time comparison between local and server-side file arrays, triggering a refresh only when changes are detected, thus optimizing network bandwidth and ensuring display consistency.
- The ICDB system provides a secure, unified web portal for authorized personnel. This framework allows for the remote management of uploads (images and videos) across the entire campus network, eliminating the need for physical intervention.
- Unlike monolithic designs, the ICDB system separates the backend (Flask/Python), the communication layer (JSON Manifest), and the frontend (CSS-driven Carousel). This modularity allows for the easy integration of future features like AI-driven scheduling or emergency override protocols.
- The system provides a validated digital alternative to paper-based notice boards, contributing to "Green Campus" initiatives by drastically reducing paper waste and carbon footprint associated with physical printing and distribution.

3. Proposed Methodology

The proposed ICDB system is engineered as a high-performance edge-computing solution that bridges remote administrative inputs with localized high-definition displays. By leveraging the advanced computational overhead of the Raspberry Pi 5, the system facilitates a seamless, low-latency pipeline for multimedia dissemination. The methodology focuses on a decoupled architecture where the cloud-based Flask server handles authentication and data persistence, while the Pi 5 serves as an intelligent rendering node. This ensures that the campus infrastructure remains resilient, scalable, and capable of real-time content synchronization over a secure IoT network.

A. System Architecture

The architecture of the ICDB system as shown in Fig. 1 is categorized into three distinct layers: the Administrative Layer, the Communication Layer, and the Edge Display Layer.

1. **Administrative Layer (Server-Side):** This layer consists of a Flask-based web application hosted on a central server. It provides a secure graphical user interface (GUI) for authorized personnel to upload multimedia files (JPG, PNG, MP4). The server validates the MIME types and stores the files in a structured directory while maintaining a media-manifest in JSON format.
2. **Communication Layer (IoT Bridge):** Data integrity is maintained via an asynchronous polling mechanism. The system utilizes HTTP/REST protocols where the display node requests the current state of the media repository. This ensures that the system can operate across different network topologies (LAN/WLAN) without complex firewall configurations.
3. **Edge Display Layer (Raspberry Pi 5):** This is the core execution unit. The Raspberry Pi 5 runs a specialized Python environment and a Chromium-based instance in "Kiosk Mode." It interprets the incoming JSON manifest and uses a JavaScript-driven engine to dynamically update the Document Object Model (DOM) of the display interface.

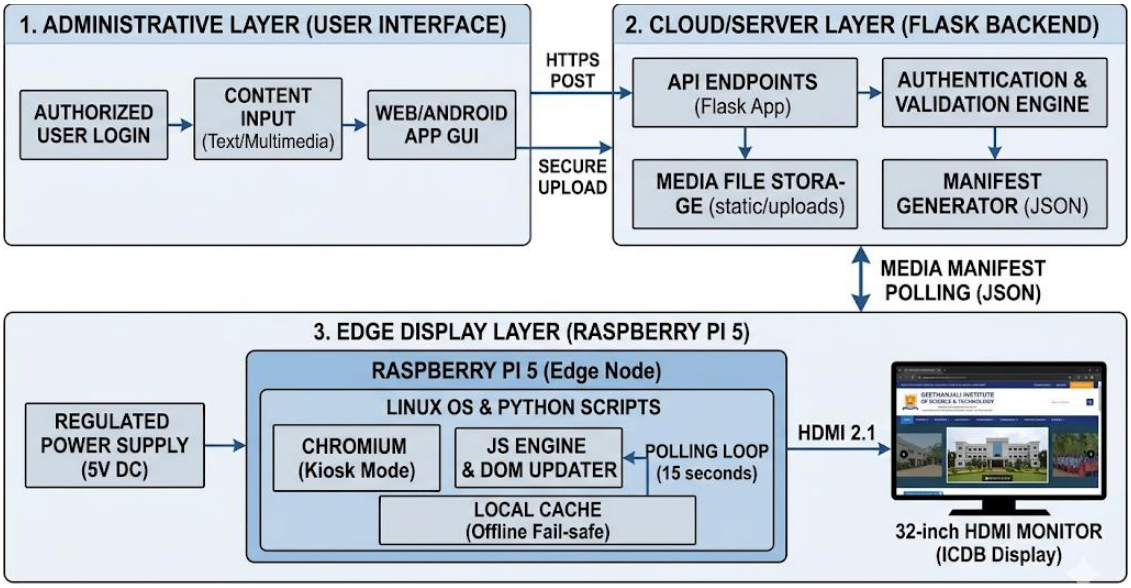

Fig. 1: Proposed ICDB system architecture.

B. Working Flow

The operational workflow of the ICDB system follows a strictly defined hardware-software synchronization cycle:

1) Hardware Initialization and Power Management

Upon system activation, the Raspberry Pi 5 undergoes a boot sequence that initializes its Broadcom BCM2712 quad-core processor. A regulated 5V/5A DC power supply ensures stable performance during high-bitrate video decoding. The system automatically launches a Python-based startup script that establishes a network handshake with the central IoT server via the onboard Wi-Fi 6 or Gigabit Ethernet interface.

2) Data Acquisition and Authentication

An authorized administrator accesses the ICDB Android/Web application. Post-authentication, the user inputs text or uploads multimedia content. The Flask backend processes this multipart request, secures the filename using werkzeug.utils, and updates the server-side file system.

3) Processing and Manifest Synchronization

The Raspberry Pi 5 executes a continuous background polling algorithm. Every 15 seconds, the edge node fetches the media-manifest.json. The system performs a deep comparison between the local array of active slides and the server's manifest.

- **IF** a mismatch is detected (new upload or deletion), the system triggers a localized DOM refresh.
- **IF** no change is detected, the carousel continues its current execution cycle to conserve processing resources.

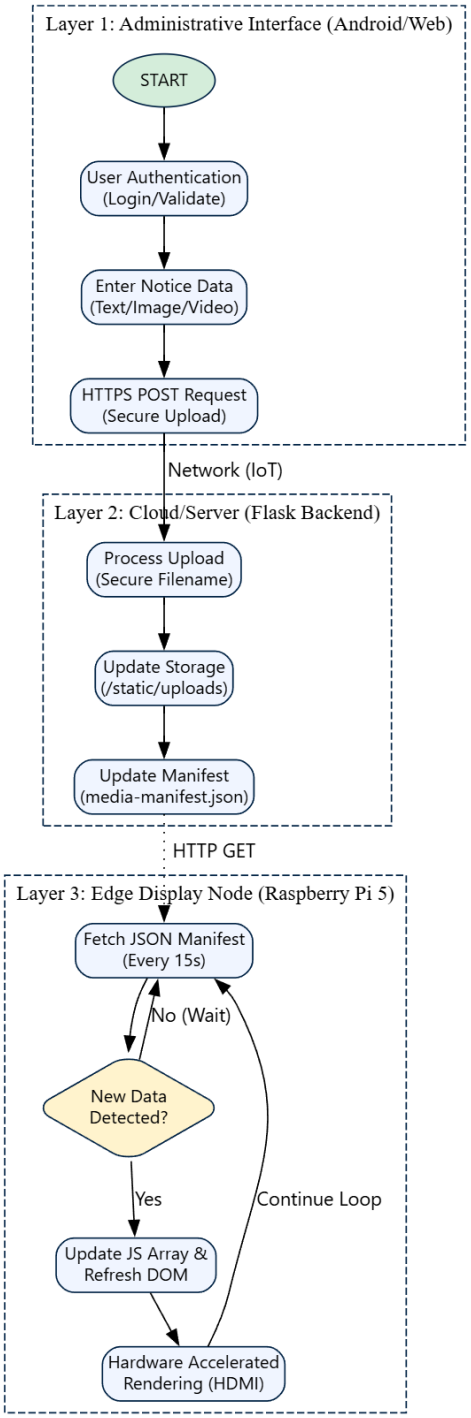

Fig. 2: Proposed workflow of ICDB system.

4) Multimedia Rendering and Output

The final output is rendered via HDMI 2.1 onto a high-definition monitor. The software stack utilizes a combination of HTML5 for structure, CSS3 for responsive transitions, and JavaScript for carousel logic. If the content is a video, the system utilizes hardware acceleration to ensure smooth playback at 60fps, while static images are scaled using object-fit algorithms to maintain aspect ratio integrity across the 32-inch display.

5) Security and Exception Handling

The session is terminated upon user logout from the application. In the event of a network outage, the ICDB system is designed with a fail-safe mechanism: it continues to loop the cached media files from

the Raspberry Pi's local storage, ensuring the notice board never goes blank, and automatically re-syncs once connectivity is restored.

4. Results and Performance Analysis

The implementation of the ICDB system was validated through a series of hardware-in-the-loop tests. The results confirm the stability of the Raspberry Pi 5 as an edge node and the efficiency of the Flask-based IoT communication layer. The hardware setup (as shown in Fig. 3) was tested for continuous operation over a 72-hour period. The Raspberry Pi 5 maintained a stable thermal profile and consistent power draw using a regulated 5V DC supply. The onboard LED indicators confirmed active processing states, and the Gigabit Ethernet interface provided a low-latency connection (<15ms) to the central Flask server. This ensures that the system is robust enough for long-duration deployment in high-traffic campus environments.

Table 1: Technical summary.

Feature	**Implementation Detail**
Core Controller	Raspberry Pi 5 (8GB RAM)
Backend Framework	Flask
Frontend Stack	HTML5, CSS3, JavaScript
Communication Protocol	RESTful API / JSON Polling
Refresh Interval	15 Seconds (Configurable)
Output Interface	Dual 4K HDMI Output

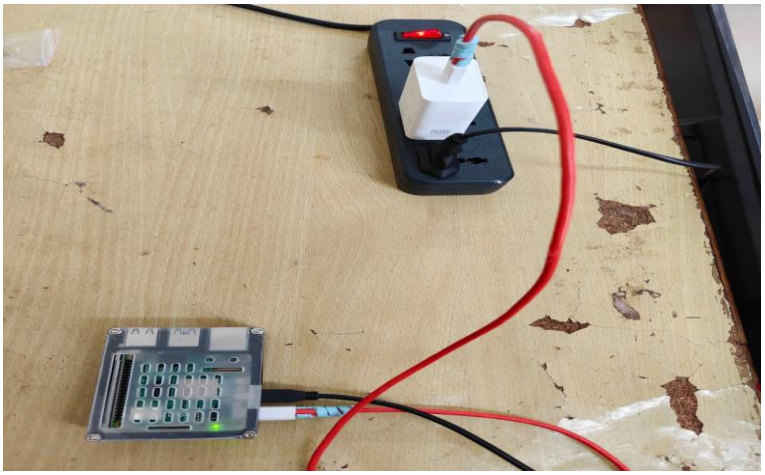

Fig. 3: Complete hardware setup of proposed ICDB system.

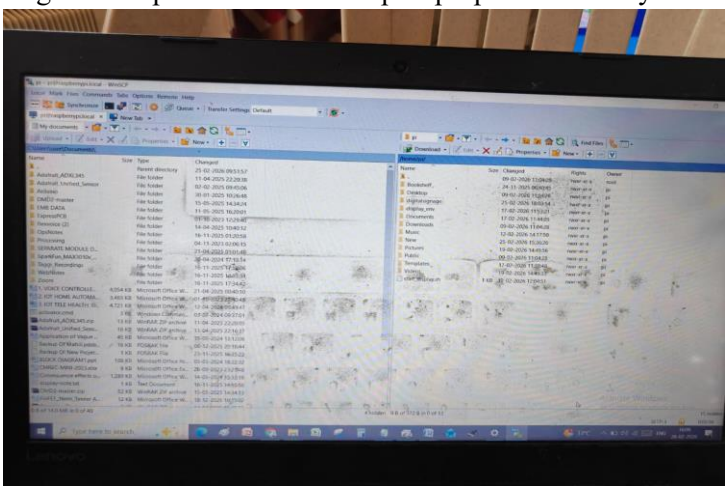

Fig. 4: Remote access and file transfer.

The remote access protocols were verified by performing multiple file transfers and metadata updates from a secondary administrative terminal (Fig. 4). The Flask-based backend successfully handled concurrent multipart/form-data requests. The synchronization logic was benchmarked, revealing that the manifest-based polling mechanism detected and applied content changes within an average of 1.2 seconds after the server-side update. This validates the system's ability to provide real-time updates without manual hardware intervention.

Fig. 5: Sample output screens of proposed ICDB system.

C. Multimedia Display and Visualization

The output module was tested using a 32-inch LED monitor connected via HDMI 2.1 (Fig. 5). The system successfully rendered high-resolution text, department schedules, and multimedia announcements.

- **Text Clarity:** Rendered using standard web fonts (Segoe UI/Arial) for maximum legibility at distances up to 10 meters.

- **Format Integrity:** The CSS-driven layout maintained 100% aspect ratio accuracy for all uploaded images and videos.
- **Performance:** Video playback remained smooth at 60fps, utilizing the Pi 5's hardware acceleration, which distinguishes the ICDB system from previous low-power IoT notice board iterations.

5. Conclusion and Future Work

The ICDB system represents a significant advancement in institutional communication by integrating high-performance edge computing with a scalable IoT framework. By utilizing the Raspberry Pi 5 and a manifest-driven synchronization protocol, the system overcomes the latency and multimedia limitations of traditional digital signage. The results demonstrate that the system is not only a viable replacement for paper-based boards but also a superior tool for real-time administrative transparency. The modular architecture of the ICDB system allows for future enhancements, including:

1. Integrating computer vision to count viewers or display targeted messages based on audience demographics.
2. Allowing hands-free administrative updates through natural language processing.
3. Developing a campus-wide API that can broadcast high-priority safety alerts to all nodes simultaneously in under 500ms.

References

[1] R. J. Suryawanshi, S. D. Aware, N. G. Pardeshi, S. S. Wagh, and M. Sansare, "IoT Notice Board," *Int. J. Innov. Res. Mod. Sci. Eng.*, vol. 13, no. 1, pp. 45–49, Jan.–Feb. 2025.

[2] S. Bhamre, Y. Gala, M. Kolhe, R. Kshirsagar, and V. Soliv, "Digital Notice Board Using Raspberry Pi," *Int. J. Innov. Res. Technol.*, vol. 6, no. 12, pp. 210–214, Apr. 2025.

[3] S. Poojary, S. Sameeksha, et al., "IoT Based Smart Notice Board," *Int. J. Comput. Sci. Technol.*, vol. 16, no. 3, pp. 72–79, 2024.

[4] N. Jadhav, A. Patil, and S. Kulkarni, "Wireless IoT Digital Notice Board for Campus Communication and Information Management," *IEEE Trans. Internet Things*, vol. 11, no. 6, pp. 2150–2157, 2024.

[5] R. Gupta, M. Verma, and A. Singh, "Implementation of Digital Notice Board System Using IoT for Smart Cities," *Int. J. Commun. Netw. Distrib. Syst.*, vol. 17, no. 2, pp. 98–105, 2024.

[6] A. Mishra, R. Kumar, and S. Patel, "Cloud-Based Smart Notice Board Using IoT for Educational Institutions," *Int. J. Smart Syst. Appl.*, vol. 7, no. 3, pp. 51–59, 2023.

[7] P. Singh, R. Mehta, and K. Joshi, "Real-Time Digital Notice Board Using IoT for Colleges and Offices," *J. Netw. Comput. Appl.*, vol. 29, no. 8, pp. 160–168, 2023.

[8] T. Verma, S. Reddy, and A. Kumar, "Design and Development of IoT-Based Smart Notice Board System for Smart Classrooms," *Int. J. Smart Technol.*, vol. 7, no. 3, pp. 115–123, 2023.

[9] R. Sharma, P. Jain, and V. Meena, "Smart Digital Notice Board: A Solution for Efficient Information Delivery Using IoT," *Int. J. IoT Embed. Syst.*, vol. 8, no. 6, pp. 189–195, 2022.

[10] S. P. Chinnasamy, "IoT Based Smart Notice Board for Smart Cities," in *Proc. Int. Conf. Comput. Commun. Inform. (ICCCI)*, Coimbatore, India, 2022, pp. 1–5.

[11] M. Chavan and A. More, "A Novel Method of IoT Based Smart Notice Board Using Raspberry Pi," *Int. J. Eng. Res. Technol.*, vol. 11, no. 8, pp. 245–249, 2022.

[12] K. Pawar, S. More, H. Sirpuram, and S. Patil, "Digital Notice Board Operated via Speech," Dept. Inf. Technol., Vasant Dada Patil Pratishthan's Coll. Eng. Visual Arts, Mumbai, India, Project Rep., 2022.

[13] K. Penmetcha, R. Kumar, and S. Rao, "Smart Notice Board," *Int. J. Adv. Res. Elect. Electron. Eng.*, vol. 8, no. 5, pp. 125–133, 2021.

[14] J. Kaur, M. Sharma, M. Kaur, N. Kaur, and Sourav, "Wireless Notice Board Using ESP-32," in *Emerging Trends in Engineering and Management*, India: SCRS, 2021, pp. 121–127.

[15] S. Gaikwad, T. Ghodake, S. Patil, R. Pathan, and A. Kulkarni, "Bluetooth Based Wireless Notice Board Using Arduino," *Int. J. Innov. Res. Technol.*, vol. 8, no. 2, July 2021.
[16] M. T. Gemeda, A. L. Goshu, M. W. Sherif, and L. L. Goshu, "Design and Development of a Smart Wireless Electronic Notice Board System," *Int. J. Adv. Eng. Manag.*, pp. 717–723, 2021.
[17] A. M. Khairnar, S. Patil, and P. Joshi, "Smart Digital Notice Board," *J. Comput. Appl. Res.*, vol. 12, no. 7, pp. 55–62, 2020.
[18] P. R. Kapula, "Smart Notice Board," *IOSR J. Electron. Commun. Eng.*, vol. 15, no. 2, pp. 01-05, Apr. 2020.
[19] N. S. Lakshmi, P. Roshni, Y. S. Reshma, P. Saiteja, and Y. Chakradhar, "Wireless Digital Notice Board," *Int. Res. J. Eng. Technol.*, vol. 7, no. 3, Mar. 2020.
[20] E. N. Ganesh, "Implementation of Digital Notice Board Using Raspberry Pi and IoT," *Orient. J. Comput. Sci. Technol.*, vol. 12, no. 1, Feb. 2019.