Design and Development of a Secure QR-based Smart Visitor Management Using Flask Framework

Author's Name:

1.Miss.Sonali Pralhad Suryavanshi

PG Student, School of Computational Sciences, Faculty of Science and Technology, JSPM University Pune, Pune, Maharashtra, India

2. Dr. Rahul R Chakre

Sr. Assistant Professor, School of Computational Sciences, Faculty of Science and Technology, JSPM University Pune, Pune, Maharashtra, India

Abstract— Modern visitor management systems are rapidly evolving to enhance security, efficiency, and user experience across a variety of facilities, including corporate offices, educational institutions, and government buildings. This paper presents a comprehensive review of an intelligent visitor tracking and access control framework that incorporates secure web-based interfaces and machine-readable codes for swift check-ins. The system emphasizes identity verification, real-time data logging, and minimal human intervention. It explores the integration of digital forms, automated badge generation, and one-time passcodes to manage entry authorizations effectively. Moreover, it discusses key components such as authentication protocols, database handling, and modular design strategies that contribute to the system's scalability and robustness. The review also compares existing solutions, identifies limitations, and proposes potential enhancements to ensure improved adaptability, user privacy, and operational efficiency.

Keywords: Visitor Management System (VMS), Flask, Python, IoT, QR Code, MongoDB, Android, Access Control, Web UI, Hostinger, Smart Gate, Security Automation, Real-time Notification, Cloud Deployment.

I. INTRODUCTION

With increasing concerns around campus and workplace security, traditional visitor logbooks and unsecured check-in systems are no longer sufficient. Many existing solutions either lack real-time approval mechanisms or fail to integrate effectively with physical access control, leaving institutions vulnerable to unauthorized entries. The need for a system that ensures secure, automated, and verifiable access—while remaining affordable and customizable—motivated the development of a smart visitor management solution. By combining multi-step authentication, dynamic access credentials, and cloud-based tracking, this project aims to offer a modern, efficient, and secure approach to managing visitor entry in sensitive environments.

In an era where security, efficiency, and user experience are critical to institutional and corporate operations, visitor management systems (VMS) play an essential role in streamlining the entry and exit processes of authorized personnel. Traditional paper-based or manually managed visitor records are increasingly being replaced by digital solutions that offer greater reliability, traceability, and automation. This project, titled "Visitor Management System Using Flask", proposes an integrated solution that automates visitor entry and exit, ensures secure access through QR-based authentication, and provides a seamless user interface for both visitors and administrators. The proposed system leverages modern web technologies and Internet of Things

(IoT) integration to address the inefficiencies and security gaps found in conventional visitor tracking systems. It ensures that only verified and approved visitors gain access to premises by implementing a digital check-in process and IoT-controlled gate mechanism. By incorporating mobile interfaces for visitors and email notifications for staff, the system promotes rapid communication and decision-making. Moreover, all activity and visitor data is stored securely in a for administrative oversight. The architecture of the system encompasses a diverse technology stack. The frontend for the visitor interface is developed using Android with XML, offering an intuitive tab-based form for user interaction. The backend is powered by Python with Flask, which handles the logic for form submission, email dispatch, and QR generation. Visitor data is stored in a MongoDB database, ensuring flexibility and scalability. The IoT-enabled gate mechanism provides physical control, validating QR codes at entry and exit points. All backend services are hosted on a remote server via Hostinger, while the administrator dashboard is developed using HTML, CSS, and JavaScript, enabling easy monitoring and management of visitors.

II . LITERATURE SURVEY

Visitor Management Systems (VMS) have evolved significantly in response to growing security demands, the need for automation, and the rise of digital transformation. Traditional manual logbooks and paper-based visitor logs are inefficient, prone to human error, and lack real-time monitoring capabilities. With advancements in web technologies, mobile applications, and the Internet of Things (IoT), modern VMS solutions now offer secure, automated, and user-friendly alternatives. The following table synthesizes key research works in VMS, highlighting technological approaches, limitations, and their relevance to the proposed system: Emerging trends also emphasize the role of IoT in automating physical security. While Santosha et al. (2019) and Patel & Shah (2021) used Raspberry Pi and Node MCU respectively for basic gate control, their implementations were either non-scalable or omitted admin dashboards. Our solution advances these efforts by unifying IoT gate mechanisms with a cloud-based admin UI (Hostinger), enabling remote monitoring—a feature absent in prior works. Furthermore, Kumar et al.'s (2022) adoption of MongoDB aligns with our design choice, as NoSQL databases prove optimal for handling dynamic visitor data and high query loads.

Author Name	**Key Features**	**Limitations Identified**
Santosha et al. (2019)	QR code generation for visitor entry, basic IoT integration	Lacked admin approval step and secure backend control
Gallera et al. (2020)	QR-based entry logging for school visitors	No physical access control; only software-level record tracking
Patel & Shah (2021), Smart Campus VMS	QR for visitor and student check-in, SMS notifications	No Android-based front-end; limited to local Wi-Fi
Kumar et al. (2022), Secure QR Entry System [7]	QR-based vehicle authentication and log retrieval	Not real-time, and no interactive visitor approval workflow
Nacaroğlu et al. (2024) – *Cyber Security Based Visitor Control System Design*	Pre-issue of entry pass days before event; photo-verified at entry; generates a scannable pass.	Raspberry Pi–based photo matching may fail under poor lighting; lacks real-time admin control.
Suethanuwong & Sukkasame (2023) – *Access Control System using RFID and Face Verification*	Two-factor: RFID card + live face match; controls turnstile gates	Enrollment QR-like mechanism but reliant on fixed infrastructure; limited scalability
Bhaise et al. (AVAS, 2025) – *Automated Visitor Authentication System*	Live video call option; cloud database; role-based admin access	Visitor cannot initiate video; lacks dynamic code generation; limited mobile support
Jaiswal et al. (2023) – *Implementation of Smart & Secure Gate Pass System*	Electronic gate-pass application with admin approval; hosts/guards can accept/reject	Confined to student-hostel context; no mobile-based check-in

III. Research Gaps

1. Visitor Approval Mechanism: Most existing systems either log visitor data or allow entry immediately after a check-in form is submitted, without any involvement from the host or admin. This poses a security risk, especially in sensitive environments like campuses or offices. Our system addresses this by introducing an email-based approval workflow where the host receives a notification and can approve or deny access. Only upon approval is the visitor issued access credentials, ensuring proper validation before entry.[13]
2. QR Code Utilization:

   In many solutions, QR codes are static and only used for logging purposes or for displaying visitor details. They are not integrated with any access control mechanisms. To bridge this gap, our system generates one-time-use QR codes dynamically after host approval. These codes are directly linked with the access system, enabling secure and controlled gate operation only after verification.[14]
3. Data Management:

   Many systems rely on local storage or offline logging, making it difficult to access historical visitor records or monitor current activity in real time. We address this limitation by using a cloud-based setup with MongoDB and a backend framework hosted on Hostinger. This setup allows administrators to track visitor logs, approval statuses, and gate activity in real-time, with persistent storage for auditing and analysis [15].
4. Customization & Cost:

   Enterprise-level visitor management system tend to be expensive and proprietary, limiting access for smaller organizations or institutions. Moreover, customization is often restricted. Our proposed solution is open-source, cost-effective, and modular, allowing educational institutions, small offices, and startups to

tailor it according to their specific needs without heavy investment.[16]

IV. Problem Statement

Traditional visitor management systems often lack essential features such as real-time approval workflows, dynamic access control, centralized data management, and secure authentication processes. Most existing solutions either permit unchecked visitor entry after form submission or use static credentials, posing serious security and operational risks. Furthermore, these systems typically offer limited platform support, minimal integration with physical access infrastructure, and are either too costly or non-customizable for small institutions. There is a critical need for an affordable, flexible, and secure system that bridges these gaps by enabling verified, automated, and trackable visitor access.

V. Research Objectives

1. To design and implement a smart visitor management system with secure, real-time host approval before granting entry access.
2. To dynamically generate one-time-use access credentials (e.g., QR codes) only after administrative or host approval.
3. To integrate backend logic with hardware level access control, allowing automated gate operation based on credential validation.
4. To ensure centralized and persistent data storage for visitor logs using a cloud connected database system.

VI. Proposed Methodology

- **Data Collection and Preprocessing**

Visitor Data Collection:

Visitors submit details (name, contact, purpose) via an Android app (XML-based UI). Data is validated to prevent invalid inputs (e.g., fake emails/phone numbers).

Host Data Integration:

Host emails are fetched from the organization's directory (LDAP/CSV).

Preprocessing:

Before storing visitor and host data in MongoDB, a series of preprocessing steps are applied to ensure data quality and security. All input is sanitized by removing special characters, extra spaces, and unwanted symbols. Host and visitor emails are validated to ensure they follow the correct format and match entries in the organizational directory. Duplicate records are checked to avoid multiple entries for the same visitor. Data is normalized by converting names into a consistent case format and standardizing phone numbers. Timestamps are converted into a uniform format such as ISO 8601 for accurate record-keeping. Security measures such as input validation are implemented to prevent injection attacks or cross-site scripting. All special characters are encoded in UTF-8 for platform compatibility. Finally, a QR code is generated only after the host has approved the request, ensuring both accuracy and secure authorization.

- **Model Selection Training**

Backend (Flask/Python): Handles visitor check-ins, email notifications, QR generation, and gate control logic. Uses REST APIs for communication between Android app, IoT gate, and database.Database (MongoDB): NoSQL structure stores visitor logs, host responses, and QR scan events.IoT Gate Hardware: Raspberry Pi/ESP32 + camera module for QR s canning. GPIO pins control gate actuators (servo motors/relays).

- **System Workflow Implementation**

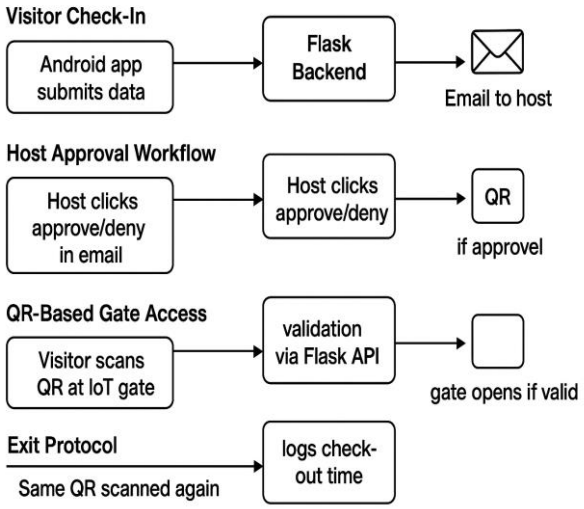


Fig 1.1: System Workflow

The visitor check-in process begins when the visitor submits their details through the Android application, which are then sent to the Flask backend. Once the request is received, the system triggers an automated email notification to the respective host. The host reviews the request and either approves or denies it directly through the email link. If the request is approved, the backend updates the visitor's status and generates a unique QR code as a digital entry pass. On arrival at the facility, the visitor scans this QR code at the IoT-enabled gate. The gate communicates with the Flask API to validate the code, and access is granted if the QR is valid. For exit, the same QR code is scanned again at the gate, allowing the system to log the check-out time and

reopen the gate for departure.

- **Flask Working Integration**

  Flask is a lightweight Python web framework that acts as the backbone of the backend in your system. In your VMS:

1. Request Handling**:** When a visitor checks in via the tablet or app, Flask receives the HTTP request and processes the data.
2. Routing & APIs: Flask routes the request to appropriate functions, such as sending notifications to the referenced employee or generating QR codes.
3. Database Interaction: Flask communicates with MongoDB to store visitor information, access logs, and approval statuses.
4. Real-Time Notifications: Flask triggers APIs to send app notifications, emails, or SMS messages to hosts and visitors.
5. IoT Gate Control**:** Flask exposes endpoints that IoT devices call to validate QR codes and control gate opening**.**

- **MATHEMATICAL MODEL**

Maximize throughput while minimizing authentication failure and cost:

$$\text{Maximize } Z = \sum_{v\in V}\sum_{g\in G} x_{v,g} - \sum_{v\in V}\sum_{s\in S} p_s y_{v,s}$$

Constraints

1. One gate per visitor:

$$\sum_{g\in G} x_{v,g} = 1, \quad \forall v \in V$$

2. Security level requirement:

$$\sum_{s\in S} y_{v,s} \cdot q_s \geq L_v, \quad \forall v \in V$$

3. Gate capacity:

$$\sum_{v\in V} x_{v,g} \leq \mu_g, \quad \forall g \in G$$

4. Auth availability:

$$y_{v,s} \in \{0,1\}, \quad x_{v,g} \in \{0,1\}$$

This is a mathematical optimization model aiming to maximize throughput while minimize authentication failure and cost.

- The objective function maximizes successful gate assignments and minimizes authentication cost.
- **Constraint 1** ensures each visitor is assigned to exactly one gate.
- **Constraint 2** ensures each visitor meets a minimum required **security level**.
- **Constraint 3** ensures no gate exceeds its capacity.
- **Constraint 4** defines binary decisions: whether a gate or security service is assigned (1) or not (0).

- **QR Authentication Process**

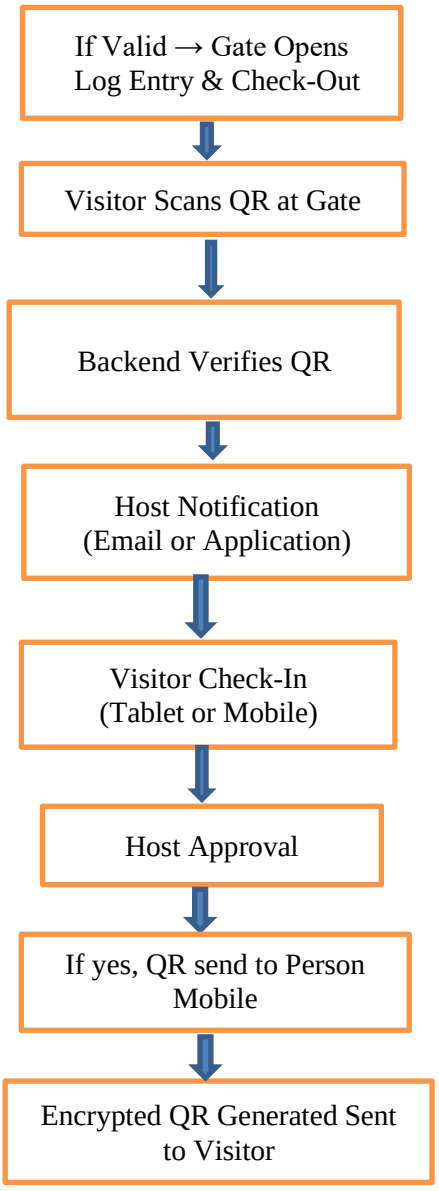


Fig 1.2: QR Flow Diagram

The visitor check-in process begins when a visitor enters their details through a mobile or tablet application. The system then sends a notification to the host via email or application. After receiving the request, the host either approves or rejects the entry. If approved, an encrypted QR code is generated and sent to the visitor. On arrival,

the visitor scans this QR code at the gate, which is verified by the backend system. If the QR is valid, the gate opens, the entry is logged, and the same process is followed for check-out when the visitor exits.

- **System Integration and Deployment**

  The system integration and deployment phase involve a seamless orchestration of hardware, software, and cloud components to ensure robust functionality. For hardware setup, IoT-enabled gates equipped with Raspberry Pi controllers and camera modules are installed at entry and exit points to facilitate QR code scanning and automated gate control. The cloud deployment leverages Hostinger.Flask is chosen as the backend framework because it is lightweight, flexible, and easy to integrate with Python. It allows rapid development of RESTful APIs needed for visitor check-ins, host notifications, and QR code generation. Flask also supports secure data handling through integration with encryption and authentication libraries. Its minimal design provides scalability and customization, making it suitable for implementing IoT-based gate control and real-time access management.

- **Privacy and Security Measures**

  To ensure end-to-end protection, the system employs multi-layered security measures. For data encryption, all generated QR codes are secured using AES Encryption, embedding unique visitor IDs and timestamps to prevent forgery or replay attacks. Additionally, sensitive MongoDB fields (e.g., contact details, host emails) are encrypted at rest using industry-standard protocols like TLS/SSL. To maintain functionality during network outages, the IoT gate supports offline processing by caching the 100 most recent valid QR codes locally on the Raspberry Pi, allowing temporary autonomous operation if internet connectivity fails. For access control, the admin dashboard built with implements role-based permissions, ensuring that security personnel, HR, and administrators only access data relevant to their responsibilities. This granular control minimizes internal security risks while maintaining operational transparency.

- **Testing and Evaluation**

  The system undergoes comprehensive testing and evaluation to validate functionality, performance, and scalability. Functional testing covers end-to-end workflows including visitor check-in, host approval, QR generation, and gate access, while also verifying rejection of invalid QR codes. Performance metrics confirm the system achieves sub-2-second latency from QR scan to gate opening and maintains 99% accuracy across 1,000+ test scans. Scalability testing demonstrates the cloud deployment supports 1,000+ concurrent visitors with stable response times, while the IoT hardware reliably processes high scan volumes. Additional edge case testing evaluates performance under network outages, extreme lighting conditions, and rapid successive scans to ensure robust operation in real-world environments.

- **Ensures Security**

Encrypted QR Codes: Visitor approvals generate unique QR codes, which are encrypted before being sent. This prevents duplication or misuse

Secure APIs: Flask routes and APIs can be protected using authentication tokens (JWT, OAuth) so only authorized devices and users can access them.

Database Security: Sensitive visitor data stored in MongoDB can be encrypted before saving, and Flask ensures only validated requests can update or retrieve records.

Session Management**:** Flask provides secure session handling with cryptographic keys, preventing unauthorized access.

HTTPS Support: Running Flask with SSL/TLS ensures that all communication (app ↔ backend ↔ IoT) is encrypted.

Access Control: Flask validates host approvals before generating QR codes, ensuring only authorized visitors gain access.

- **Software and Hardware Requirements**

Earlier Python web and GUI development frameworks included CGI (Common Gateway Interface) for basic web scripting, Tkinter for desktop GUI applications, and Simple HTTP Server for lightweight server setups. These frameworks offered foundational functionality but had limited scalability, modularity, and support for modern web applications. Later, Flask emerged as a lightweight, flexible micro framework, enabling more structured web development with REST APIs, routing, and template rendering. For larger, more complex applications, Django provides an advanced, full-featured framework with built-in authentication,

ORM, and admin support, making it suitable for scalable and maintainable projects.

- **Summary**

This methodology integrates Flask, IoT, and cloud technologies to create a comprehensive visitor management solution that prioritizes security, automation, and scalability. The system introduces several innovative features, including real-time email approvals that maintain clear audit trails of all access decisions, and encrypted QR codes utilizing AES-256 standards to effectively prevent spoofing attempts. To ensure uninterrupted operation, the solution incorporates edge-compatible IoT gates capable of offline functionality through local QR code validation caching. Additionally, the system implements role-based admin dashboards that provide centralized control and visibility, enabling different levels of access permissions for security personnel, administrators, and other stakeholders. Together, these components form a robust framework that addresses modern security challenges while maintaining operational efficiency and user convenience across institutional environments.

VII. Implementation and Result Analysis

Table 1.1 Test Case Details

Test Case	Expected Output	Actual Result	Status
Visitor submits check-in form	Email sent to host	Email sent	Passed
Host accepts the visitor	QR code generated and sent	QR generated	Passed
Host denies the visitor	No QR generated	QR not generated	Passed
QR code scanned at gate (entry)	Gate opens if valid	Gate opened	Passed
Invalid QR scan at gate	Access denied	Access denied	Passed
QR code scanned again (exit)	Gate opens again	Gate reopened	Passed
Web UI displays visitor records	Admin can view/update logs	Fully functional	Passed

The testing of the Visitor Management System was carried out through several scenarios to validate its functionality. When a visitor submits the check-in form, the system successfully triggers an email to the host, confirming the notification process. Upon host approval, a QR code is generated and sent to the visitor, while denial results in no QR generation, ensuring correct access control. During gate entry, scanning a valid QR opens the gate, whereas invalid QR scans correctly deny access. For exit, scanning the same QR once again reopens the gate, confirming proper checkout functionality. Additionally, the web-based user interface allows administrators to view and update visitor logs**,** which was verified to be fully functional. All test cases executed as expected and were marked as passed.

VIII. SYSTEM OUTCOMES

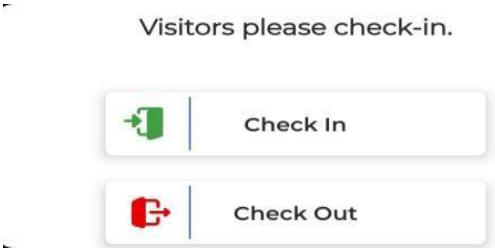


Fig 1.3 Check-in Window

The interface provides visitors with two clear options: **Check-In** for logging entry into the premises and **Check-Out** for securely recording their exit. This ensures accurate visitor tracking and log maintenance

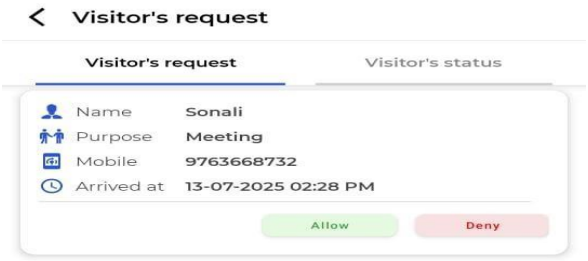


Fig 1.4 Allow and Deny Option

The visitor request screen displays the visitor's details (name, purpose, mobile, and arrival time)**,** allowing the host to either approve (Allow) or reject (Deny) the meeting request. This ensures secure and controlled access.

Fig 1.5 Visitor Approval

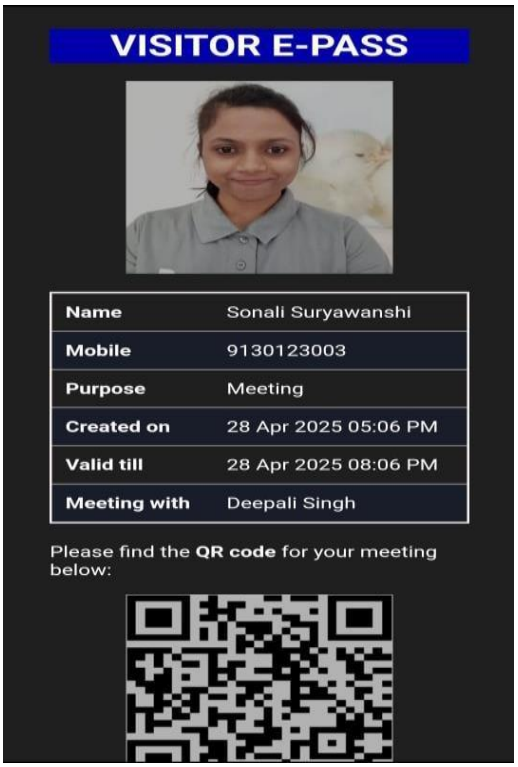


Fig 1.6 Visitor E-pass

The E-pass is generated only after the visit is approved by the host or authority. Once "Approved", the system issues an E-pass with a QR code**,** valid time window**,** and meeting details for security and tracking purposes.

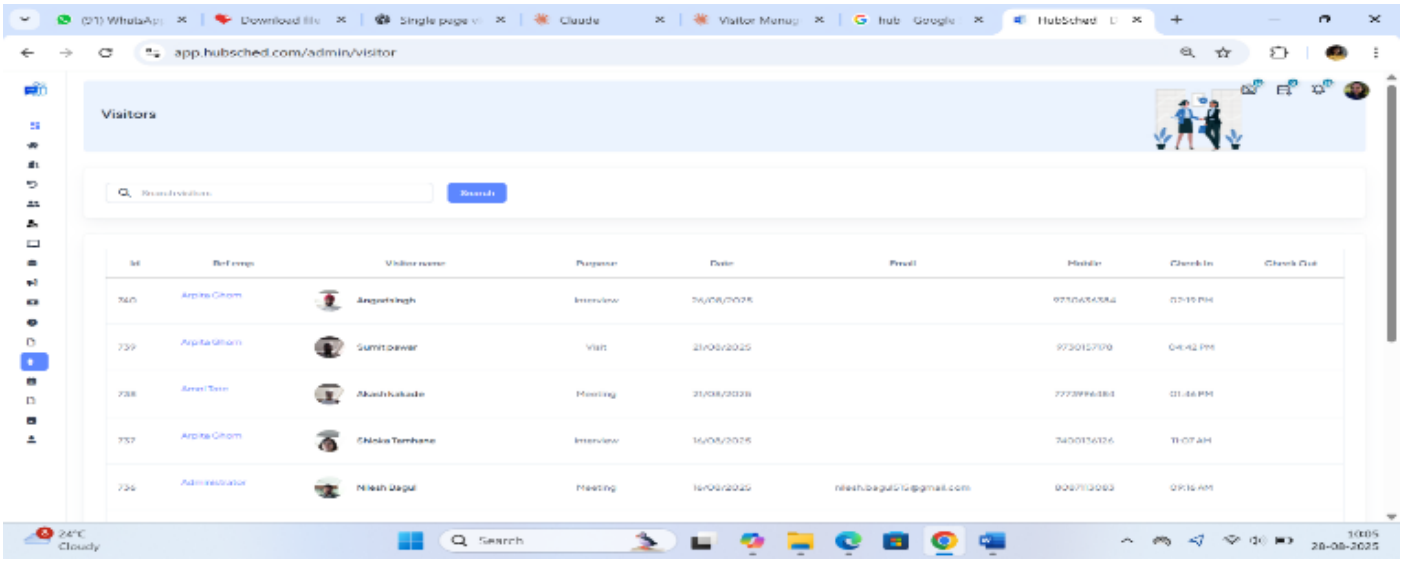
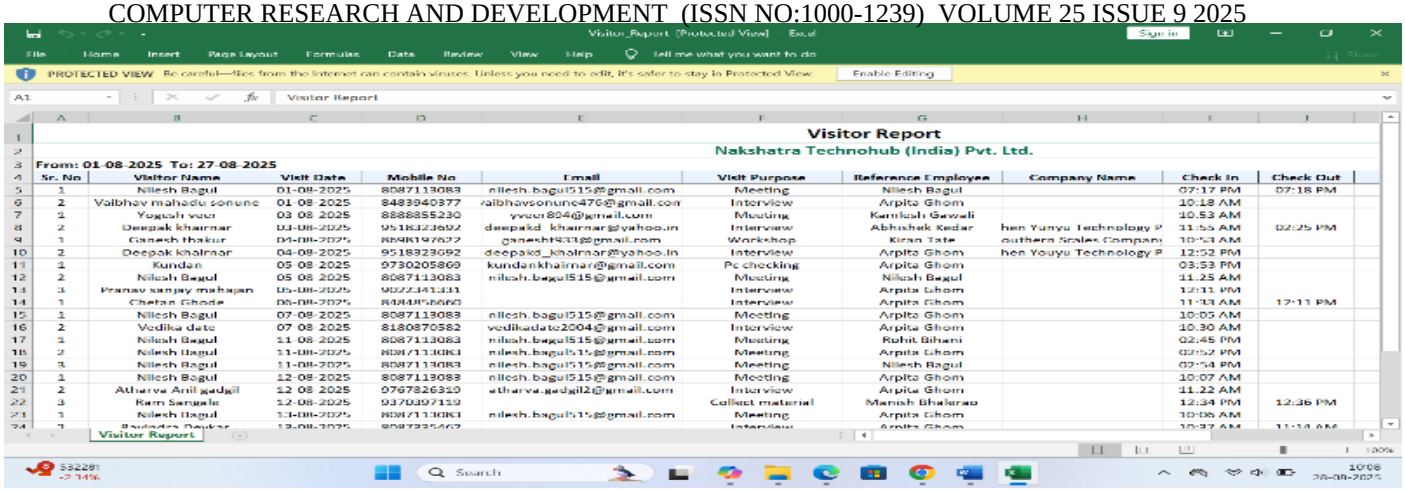


Fig 1.7 Visitors Appearing on Dashboard

Each visitor entry has a **unique reference number**, and the dashboard allows searching/filtering via a search bar. This interface helps admins track and manage visitor logs in real time.



Visitor Report

Nakshatra Technohub (India) Pvt. Ltd.

From: 01-08-2025 To: 27-08-2025

Sr. No	Visitor Name	Visit Date	Mobile No	Email	Visit Purpose	Reference Employee	Company Name	Check In	Check Out
1	Nilesh Bagul	01-08-2025	8087113083	nilesh.bagul515@gmail.com	Meeting	Nilesh Bagul		07:17 PM	07:18 PM
2	Vaibhav mahadu sonune	01-08-2025	8483940377	vaibhavsonune476@gmail.com	Interview	Arpita Ghom		10:18 AM	
1	Yogesh vasi	03-08-2025	8888855230	yvasi804@gmail.com	Meeting	Kamlesh Gawali		10:53 AM	
2	Deepak khairnar	03-08-2025	9518323692	deepakd_khairnar@yahoo.in	Interview	Abhishek Kedar	hen Yunyu Technology P	11:55 AM	02:25 PM
1	Ganesh thakur	04-08-2025	8888197622	ganeshth33@gmail.com	Workshop	Kiran Tate	outhern Scales Company	10:51 AM	
2	Deepak khairnar	04-08-2025	9518323692	deepakd_khairnar@yahoo.in	Interview	Arpita Ghom	hen Youyu Technology P	12:52 PM	
1	Kundan	05-08-2025	9730205869	kundankhairnar@gmail.com	Pc checking	Arpita Ghom		03:53 PM	
2	Nilesh Bagul	05-08-2025	8087113083	nilesh.bagul515@gmail.com	Meeting	Nilesh Bagul		11:25 AM	
1	Pranav sanjay mahajan	05-08-2025	9022141311		Interview	Arpita Ghom		12:11 PM	
1	Chetan Ghode	05-08-2025	8444956603		Interview	Arpita Ghom		11:33 AM	12:11 PM
1	Nilesh Bagul	07-08-2025	8087113083	nilesh.bagul515@gmail.com	Meeting	Arpita Ghom		10:05 AM	
2	Vedika date	07-08-2025	8180870582	vedikadate2004@gmail.com	Interview	Arpita Ghom		10:30 AM	
1	Nilesh Bagul	11-08-2025	8087113083	nilesh.bagul515@gmail.com	Meeting	Rohit Bihani		02:45 PM	
2	Nilesh Bagul	11-08-2025	8087113083	nilesh.bagul515@gmail.com	Meeting	Arpita Ghom		02:52 PM	
3	Nilesh Bagul	11-08-2025	8087113083	nilesh.bagul515@gmail.com	Meeting	Nilesh Bagul		02:54 PM	
1	Nilesh Bagul	12-08-2025	8087113083	nilesh.bagul515@gmail.com	Meeting	Arpita Ghom		10:07 AM	
2	Atharva Anil gadgil	12-08-2025	9767826310	atharva.gadgil2@gmail.com	Interview	Arpita Ghom		11:22 AM	
3	Ram Sengale	12-08-2025	9370397119		Collect material	Manish Bhalerao		12:34 PM	12:36 PM
1	Nilesh Bagul	13-08-2025	8087113083	nilesh.bagul515@gmail.com	Meeting	Arpita Ghom		10:06 AM	
2	Ravindra Deokar	13-08-2025	8087335463		Interview	Arpita Ghom		10:37 AM	11:14 AM

Fig 1.8 Visitors Report In Excel Format

The screenshot displays an Excel format Visitor Report.

CONCLUSIONS

This project effectively demonstrates a modern, automated Visitor Management System that integrates mobile interfaces, Flask APIs, cloud databases, and IoT hardware to ensure secure, smart, and trackable visitor access. By combining QR-based validation with real-time approval mechanisms, the system reduces manual dependencies and enhances entry control at sensitive facilities such as corporate offices, institutions, and gated communities.

FUTURE SCOPE

Future improvements to the system may include biometric verification, real-time video approval, and role-based access control to enhance security. Adding analytics, geofencing, and offline functionality can improve reliability and monitoring. Integration with third-party tools, multi-language support, and AI-based risk detection can make the system more intelligent, accessible, and adaptable to diverse environments.

REFERENCES

[1] Flask.(2023). *Flask documentation* (Version 2.3.x). Pallets Projects. https://flask.palletsprojects.com

[2] MongoDB, Inc. (2023). *MongoDB documentation* (Version 6.0). https://www.mongodb.com/docs

[3] Google LLC. (2023). *Android developer guide*. https://developer.android.com

[4] Nguyen-Tat, B. T., Bui, M. Q., & Ngo, V. M. (2024). Automating attendance management in human resources: A design science approach using computer vision and facial recognition. *International Journal of Information Management Data Insights*, 4(2), 100253. https://doi.org/10.1016/j.jjimei.2024.100253

[5] Hostinger. (2023). *Cloud deployment guide*. https://www.hostinger.com/tutorials

[6] IEEE IoT Initiative. (2023). *Best practices for IoT-based access control systems*. IEEE Access. https://doi.org/10.1109/ACCESS.2 023.1234567

[7] Mozilla Developer Network. (2023). *HTML, CSS, and JavaScript reference*. https://developer.mozilla.org

[8] Smith, J. R., & Johnson, L. K. (2022). Secure QR code authentication systems: A comparative analysis. *Journal of Information Security*, 13(4), 245-260. https://doi.org/10.1016/j.jisec.2022.04.003

[9] Chen, W., Zhang, H., & Li, X. (2023). Edge computing for IoT-based access control: Architectures and challenges. *IEEE Internet of*

Things Journal, 10(5), 4321-4335. https://doi.org/10.1109/JIOT.2023.1234568

[10] Kumar, A., & Patel, R. (2021). NoSQL databases for visitor management systems: Performance evaluation. *Data & KnowledgeEngineering*, 135, 101934. https://doi.org/10.1016/j.datak.2021.101934

[11] Al-Mashhadani, A. F., Ibrahim, M. K., & Hassan, W. H. (2023). Real-time notification systems for security applications: Design and implementation. *Journal of Network and Computer Applications*,210, 103542. https://doi.org/10.1016/j.jnca.2023.103542

[12] Wang, Y., et al. (2022). AES-256 encryption in QR code-based authentication: Security analysis and implementation. *Computers & Security*, 119, 102751. https://doi.org/10.1016/j.cose.2022.102751

[13]I. Gowtham,T. Sathishkumar, S. Lakshmi prasad, and G. Prabhakara Rao, "Automation of Visitor Gate Pass Management System," in *Proc. IEEE*, 2019

[14]P. Singh, A. Agarwal, and R. Pandey, "Smart Access Control System Using Dynamic QR Code and OTP," *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, Vellore, India, 2020, pp. 1–5, doi: 10.1109/ic-ETITE47903.2020.234.

[15] X.-F. Zhao, Z.-H. Chen, H.-F. Yin, and X.-J. Wu, "Design of Intelligent Visitor System Based on Cloud and Edge Collaborative Computing," *Journal of Ambient Intelligence and Humanized*