Smart Gate: Automating Security & Logistics via OCR & Mobile Technology

[Mr. Dhruv Maurya], [Ms. Riddhi Mehta]
Department of Computer Science and Engineering
Parul Institute of Technology, Parul University, Gujarat, India

Abstract—Modern gate management in residential societies, educational campuses, industrial complexes, and smart cities demands a unified system capable of simultaneously ensuring security, recording visitor data, and streamlining logistics operations. Conventional manual entry systems are plagued by delays, inaccurate record-keeping, unauthorized access risks, and an inability to generate meaningful analytics. This paper presents Smart Gate, an intelligent, IoT-integrated system that automates gate security and logistics management through Optical Character Recognition (OCR), mobile technology, and cloud-connected infrastructure. The system captures and recognizes vehicle license plates in real time, authenticates visitors via QR code-based mobile passes, delivers instant push notifications to residents or security personnel, and maintains a tamper-proof digital logbook accessible through a responsive web and mobile dashboard. Smart Gate further incorporates role-based access control (RBAC), delivery management, an emergency override mechanism, and analytical reporting modules. Experimental evaluation demonstrates that Smart Gate reduces gate processing time by up to 78%, eliminates manual data-entry errors, and achieves a license plate recognition accuracy of 94.3% under varied lighting conditions. The proposed system provides a scalable, costeffective, and interoperable architecture that is deployable across diverse environments without requiring specialized hardware.

Index Terms—Smart Gate, Optical Character Recognition, OCR, Mobile Technology, License Plate Recognition, Visitor Management, IoT Security, QR Code Authentication, Role-Based Access Control, Logistics Automation, Edge Computing

I. INTRODUCTION

The concept of smart infrastructure has gained remarkable momentum in the wake of rapid urbanization and the proliferation of Internet of Things (IoT) devices. Gate systems—the primary interface between controlled premises and the external environment—remain one of the most underdeveloped segments of smart infrastructure. Despite significant advancements in networking and mobile computing, many residential complexes, educational institutions, industrial facilities, and commercial establishments continue to rely on paper-based visitor registers, telephone-based verifications, and manual inspection of identity documents. These legacy processes introduce substantial security vulnerabilities and operational inefficiencies.

Residential societies with thousands of flat owners face a particularly acute challenge: managing the daily ingress and egress of residents, domestic workers, delivery personnel, and guests while simultaneously keeping meticulous security records. Industrial gates must handle vehicular traffic carrying goods, raw materials, and equipment—a process that requires weighment records, driver authentication, and cargo manifests. Educational campuses need to distinguish between students, faculty, contractors, and visitors while preventing unauthorized entry. In each of these contexts, the human element introduces delays, inconsistencies, and potential corruption of records.

Smart Gate addresses these pain points through a cohesive architecture that integrates three complementary technologies: Optical Character Recognition (OCR) for automated license plate number extraction, mobile application infrastructure for realtime notifications and digital visitor passes, and cloud-hosted data management for persistent, accessible, and analytically rich record-keeping. Unlike existing commercial solutions that are either prohibitively expensive, proprietary-hardware-dependent, or limited in scope, Smart Gate is engineered to operate on commodity cameras and smartphones, making it financially viable for a wide range of deployment environments.

The key contributions of this paper are as follows: (1) A novel end-to-end gate automation framework integrating OCR, mobile push notifications, and QR code authentication; (2) A robust multi-role access control model designed for heterogeneous stakeholder groups; (3) A real-time vehicle recognition pipeline optimized for low-light and partial-occlusion conditions; (4) A delivery and logistics management sub-system with chain-of-custody tracking; and (5) Comprehensive experimental evaluation including accuracy benchmarking, latency measurement, and user experience assessment.

The remainder of this paper is organized as follows: Section II reviews related work; Section III describes the system architecture; Section IV details the methodology and algorithmic design; Section V covers implementation specifics; Section VI presents results and analysis; Section VII examines security considerations; Section VIII discusses advantages and limitations; Section IX outlines future directions; and Section X concludes the paper.

II. LITERATURE REVIEW

The automation of gate access and visitor management has been approached from multiple research angles, each contributing distinct insights yet leaving gaps that Smart Gate seeks to address.

A. Traditional Access Control Systems

Early access control systems relied on RFID tags and proximity cards [1]. These systems offered deterministic authentication but required each authorized entity to carry a physical credential, limiting scalability and increasing administrative overhead for large populations. Chen et al. [2] extended RFID-based systems with centralized databases, improving audit trail quality but failing to address the challenge of unregistered visitors or delivery personnel who lack pre-issued credentials.

B. License Plate Recognition (LPR)

Automatic License Plate Recognition (ALPR) has been studied extensively since the 1990s. Classical approaches relied on edge detection, morphological operations, and template matching [3]. Deep learning transformed the field: convolutional neural networks (CNNs) now achieve recognition accuracies exceeding 97% on curated datasets [4]. However, real-world deployment introduces challenges such as variable illumination, motion blur, partial occlusion by mud or stickers, and the diversity of international plate formats. Kumar et al. [5] demonstrated that YOLO-based detection combined with CRNN-based sequence recognition yields strong performance under adverse conditions, a finding that influenced the design of Smart Gate's recognition pipeline.

C. Mobile-Based Visitor Management

Smartphone-centric visitor management has gained commercial traction through platforms such as Envoy and Sine. Academic contributions include work by Patel et al. [6], who proposed a QR code invitation system for corporate campuses, and Zhao et al. [7], who incorporated facial recognition alongside QR codes for dual-factor authentication. These systems demonstrated reduced reception workload and improved visitor experience but were evaluated only in single-building office environments and did not address vehicular logistics.

D. IoT-Integrated Gate Systems

IoT-based gate systems have been explored for smart homes [8] and smart cities [9]. Typically, these systems involve a microcontroller-operated barrier connected to cloud MQTT brokers. Mishra et al. [10] integrated such a system with voice command interfaces, while Gupta et al. [11] added solar-powered edge computing nodes to enable offline operation. However, none of these systems combined full logistics management, multi-role access control, and analytical dashboards in a single coherent platform.

E. Research Gap

A synthesis of existing literature reveals three principal gaps: (i) the absence of a unified system that jointly handles pedestrian visitors and vehicular logistics; (ii) the lack of role-aware notification workflows that differentiate between residents, security guards, and administrators; and (iii) insufficient attention to deployment in resource-constrained environments without proprietary hardware. Smart Gate is designed specifically to close these gaps.

III. SYSTEM ARCHITECTURE

Smart Gate employs a layered, microservices-oriented architecture comprising five principal layers: Perception, Communication, Processing, Application, and Presentation. Each layer encapsulates a cohesive set of responsibilities, promoting modularity, independent scalability, and testability.

A. Perception Layer

The Perception Layer constitutes the physical sensing frontier of Smart Gate. It consists of one or more IP cameras mounted at entry and exit points, capable of capturing video streams at a minimum resolution of 1280×720 pixels at 25 frames per second. The cameras feed video to an on-premise edge node—typically a Raspberry Pi 4 or an equivalent single-board computer—that performs initial frame selection using a motion-triggered algorithm. This approach reduces unnecessary processing load by activating the OCR pipeline only when a vehicle or pedestrian is detected within a designated region of interest (ROI). Additionally, the perception layer incorporates an infrared proximity sensor to detect vehicle presence at the barrier, ensuring that OCR processing is triggered precisely when a vehicle halts at the gate.

B. Communication Layer

The Communication Layer mediates data exchange between the edge node and the cloud backend. It employs a hybrid protocol stack: MQTT for lightweight telemetry data (sensor triggers, barrier commands) and HTTPS REST for structured payloads (visitor records, image uploads, notification requests). WebSockets are used to maintain a persistent connection between the

mobile application and the backend server, enabling real-time, bidirectional notification delivery without polling overhead. All communications are secured using TLS 1.3, and API endpoints require JWT-based authentication.

C. Processing Layer

The Processing Layer hosts the computational core of Smart Gate. It consists of a Node.js-based API server deployed on a cloud virtual machine (or alternatively on a local server for air-gapped environments). This layer performs OCR result validation, visitor record creation, image storage, access decision arbitration, and notification dispatch. A Python-based OCR microservice, running on the edge node, performs license plate segmentation and character recognition before transmitting structured results to the API server. A rule engine within the Processing Layer evaluates access policies based on vehicle registration status, visitor pre-authorization, time-of-day restrictions, and blacklist entries.

D. Application Layer

The Application Layer encompasses the domain-specific logic of Smart Gate. It manages the lifecycle of gate events, from initial detection through final disposition (admitted, denied, flagged). Sub-modules within this layer include: the Visitor Management Module (handling pre-registration, QR code generation, and walk-in logging), the Delivery Management Module (tracking consignment arrival, handover, and recipient acknowledgment), the Resident Notification Module (dispatching mobile push notifications with visitor images and approve/deny actions), and the Analytics Module (computing traffic statistics, peak hours, frequent visitors, and anomaly flags).

E. Presentation Layer

The Presentation Layer provides user-facing interfaces for all stakeholder groups. A React Native mobile application serves residents and administrators with a unified notification center, gate event history, pre-authorization management, and delivery tracking. A React.js web dashboard serves security guards and facility managers with a real-time gate monitoring console, manual override controls, report generation, and system configuration interfaces. Both interfaces consume the same REST API, ensuring data consistency across platforms.

IV. METHODOLOGY

A. License Plate Detection and OCR Pipeline

The license plate recognition pipeline operates in three sequential stages: detection, segmentation, and recognition. In the detection stage, a YOLOv8-nano model—chosen for its balance of accuracy and edge-deployable weight size—processes each motion-triggered video frame to produce bounding boxes around candidate license plate regions. The model was fine-tuned on a dataset of 18,000 annotated images encompassing Indian license plates (both old and high-security registration plate formats), collected from diverse lighting conditions, angles, and weather scenarios.

In the segmentation stage, the detected plate region is extracted and subjected to a pre-processing pipeline comprising adaptive thresholding, morphological dilation for character boundary enhancement, and perspective correction using homographic transformation to normalize skewed plates. This pipeline significantly improves recognition accuracy for plates captured at angles deviating by up to 25 degrees from the camera axis.

The recognition stage employs a CRNN architecture—combining a CNN feature extractor with a bidirectional LSTM sequence modeler and a CTC decoder—to convert the pre-processed plate image into a character sequence. Post-recognition, a validation module checks the output against Indian Regional Transport Office (RTO) number plate format regular expressions to reject spurious readings. Valid plate numbers are then queried against the vehicle database to determine registration status.

B. QR Code-Based Visitor Authentication

For pre-registered visitors, Smart Gate generates time-bound, cryptographically signed QR codes containing an encrypted payload that encodes the visitor's name, host flat number, permitted entry window, and a unique nonce. When the visitor presents the QR code at the gate terminal (either a smartphone screen or a printed copy), the edge node's QR scanner decodes and transmits the payload to the backend. The backend validates the digital signature using HMAC-SHA256, verifies the time window, and checks the nonce against a replay-protection database. Valid QR codes trigger automatic gate opening with a log entry and simultaneous notification to the host.

C. Role-Based Access Control (RBAC) Model

Smart Gate implements a hierarchical RBAC model with five principal roles: Super Administrator, Facility Manager, Security Guard, Resident, and Delivery Agent. Each role is associated with a defined set of permissions covering gate event visibility, manual override capability, visitor pre-authorization, analytics access, and system configuration. The permission matrix is stored in the cloud database and is evaluated server-side with every API request, ensuring that no privilege escalation can occur through client-side manipulation.

D. Delivery and Logistics Management

The Delivery Management Module addresses the specific workflow of commercial deliveries, which constitute a significant fraction of gate traffic in residential and commercial settings. When a delivery agent arrives, the security guard logs the consignment details—courier company, tracking number, recipient flat, package count—using the guard mobile interface. The system dispatches a notification to the recipient, who can acknowledge receipt digitally. For missed deliveries, the module records the delivery attempt and allows the resident to schedule a re-delivery or authorize release to a neighbor or concierge. A chain-of-custody ledger maintains an immutable record of each handover event.

E. Analytics and Reporting

The Analytics Module aggregates gate event data to produce operational intelligence. Key metrics include: hourly and daily traffic volume, average gate processing time, peak hour identification, visitor frequency distribution, unauthorized access attempt frequency, and delivery success rate. Reports are generated on-demand or on a scheduled basis and are exportable in PDF and CSV formats. Anomaly detection algorithms flag statistical outliers—such as unusually high visitor frequency for a particular flat or repeated access denials for a specific vehicle—for security review.

V. IMPLEMENTATION

A. Technology Stack

The Smart Gate system is implemented using a carefully selected, open-source-oriented technology stack. The mobile application is developed with React Native (v0.73), enabling a single codebase to target both Android (API 26+) and iOS (14+) platforms. The web dashboard is built with React.js (v18), utilizing Tailwind CSS for responsive styling and Recharts for data visualization components. The backend API server runs on Node.js (v20 LTS) with the Express.js framework, connected to a PostgreSQL database for structured relational data and Firebase Cloud Storage for image archival.

The OCR microservice is implemented in Python 3.11 using PyTorch for the YOLOv8 and CRNN models, OpenCV for image pre-processing, and the pyzbar library for QR code decoding. The edge node software is containerized using Docker, enabling reproducible deployment on any Linux-based single-board computer. Push notifications are delivered via Firebase Cloud Messaging (FCM) for Android and Apple Push Notification Service (APNs) for iOS, abstracted behind a unified notification service layer in the backend.

B. Database Schema Design

The PostgreSQL database schema is organized around five core entities: Users (residents, guards, administrators), Vehicles (registered vehicles linked to residents), GateEvents (timestamped entry/exit records linking vehicles or visitors to gate decisions), DeliveryRecords (consignment tracking entries), and NotificationLogs (records of dispatched and acknowledged notifications). Foreign key relationships enforce referential integrity, and composite indexes on GateEvents (gate_id, timestamp) optimize time-range queries for the analytics module.

C. Edge Node Deployment

The edge node software is packaged as a Docker Compose application consisting of three containers: the OCR microservice, a local MQTT broker (Mosquitto), and an edge agent that interfaces with the physical camera and sensors via the V4L2 API. The edge node synchronizes processed events with the cloud backend using an asynchronous message queue, ensuring that gate events are preserved locally in a SQLite buffer during internet outages and uploaded upon connectivity restoration. This design provides graceful degradation, maintaining gate functionality even without continuous cloud connectivity.

D. Mobile Application Features

The resident mobile application provides the following principal screens: Home Dashboard (showing current gate status, recent events, and active visitors), Visitor Invite (allowing residents to pre-register visitors by name, purpose, and date/time window, generating a shareable QR code), Delivery Tracking (listing pending and completed deliveries), Notification Center (displaying gate events with visitor photos and approve/deny action buttons), and Settings (managing profile, vehicle registrations, and notification preferences). The application implements offline caching of recent events using AsyncStorage, ensuring usability in low-connectivity scenarios.

E. Security Implementation Details

All API communications enforce HTTPS with certificate pinning in the mobile application to prevent man-in-the-middle attacks. Passwords are stored as Argon2id hashes with per-user salts. QR code payloads are encrypted with AES-256-GCM before being embedded, adding a layer of confidentiality beyond the HMAC signature. Access tokens are short-lived JWTs (15-minute expiry) refreshed via rotating refresh tokens stored in HttpOnly secure cookies. The backend implements rate limiting (100 requests per minute per IP) and input validation using the Joi schema library to prevent injection attacks.

VI. RESULTS AND ANALYSIS

A. License Plate Recognition Accuracy

The OCR pipeline was evaluated on a test dataset of 2,400 license plate images, spanning four lighting conditions (bright sunlight, overcast, twilight, and artificial night illumination) and three camera angle categories (frontal ±5°, angled 6°-15°, and steep 16°-25°). The overall recognition accuracy—defined as exact-match of all characters in the plate number—was 94.3%. Accuracy was highest under bright sunlight at frontal angles (98.1%) and lowest under artificial night illumination at steep angles (87.6%). Post-recognition format validation rejected 96.2% of erroneous readings, reducing the rate of incorrect vehicle identifications presented to the backend.

B. Gate Processing Latency

End-to-end gate processing latency—measured from vehicle halt detection by the proximity sensor to barrier opening command issuance—was evaluated across 500 gate events in a controlled deployment. For registered vehicles, the mean latency was 2.3 seconds (standard deviation: 0.4s), representing a 78.2% reduction compared to the baseline manual check process, which averaged 10.6 seconds. For QR code-authenticated visitors, the mean latency was 1.8 seconds. For walk-in visitors requiring guard logging and resident notification, the mean latency was 22.4 seconds, primarily attributable to resident response time for approve/deny notification actions.

C. Notification Delivery Performance

Push notification delivery was evaluated over 1,200 notification events. The mean end-to-end notification delivery latency (from gate event detection to notification receipt on the resident device) was 1.2 seconds for devices with active FCM/APNs connections. Delivery success rate was 99.1% within a 30-second window. The 0.9% failure rate was attributable to resident devices in airplane mode or with app notification permissions disabled, in which cases the system fell back to SMS notifications as a secondary channel.

D. System Reliability

A 72-hour continuous operation stress test was conducted with simulated gate events at 300% of expected peak load. The system maintained a request success rate of 99.7%, with no data loss observed due to the edge-node buffering mechanism. Memory consumption on the edge node remained stable at approximately 420 MB RSS, and CPU utilization peaked at 73% during simultaneous OCR processing and notification dispatch. The cloud backend scaled horizontally during the stress test, demonstrating the effectiveness of the stateless API design.

E. User Experience Evaluation

A user study was conducted with 48 participants comprising 30 residents, 12 security guards, and 6 administrators across a pilot deployment at a 200-unit residential complex. Participants completed a System Usability Scale (SUS) questionnaire after two weeks of use. The overall mean SUS score was 82.4 (out of 100), placing Smart Gate in the 'Excellent' category. Residents specifically appreciated the real-time notification with visitor photo (rated 4.7/5), while security guards valued the reduced manual paperwork (rated 4.6/5). The primary pain point reported was the occasional 3-4 second delay in notification approval response on older smartphones.

VII. SECURITY ANALYSIS

A systematic threat modeling exercise using the STRIDE framework (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) was conducted to identify and mitigate security risks in Smart Gate.

Spoofing threats—such as presenting a photograph of a valid license plate or a screenshot of a legitimate QR code—are mitigated by requiring QR codes to be presented within a five-minute validity window (reducing replay window) and by equipping the entry terminal with liveness detection capability using infrared depth sensing. Tampering with gate event records is prevented by maintaining an append-only event ledger in the database with cryptographic hash chaining, making retroactive modification detectable. Repudiation is addressed by timestamping all events with NTP-synchronized clocks and retaining photographic evidence of each gate interaction for 90 days.

Information disclosure risks are minimized through end-to-end encryption of all data in transit and at rest, role-based data access restrictions ensuring that guards cannot view resident communication details, and annual third-party security audits. Denial of service protection is implemented at the API gateway level using adaptive rate limiting and geographic IP filtering. Privilege escalation attempts are defeated by the server-side RBAC enforcement, which cannot be bypassed by client-side token manipulation.

VIII. ADVANTAGES AND LIMITATIONS

A. Advantages

Smart Gate offers numerous compelling advantages over conventional and prior art solutions. First, the system operates on commodity hardware—standard IP cameras and Android/iOS smartphones—eliminating the need for proprietary readers or specialized terminals. Second, the unified platform approach means that a single deployment satisfies security, visitor management, and logistics needs simultaneously, reducing integration complexity. Third, the offline-capable edge node design ensures continuity of gate operations during internet outages, a critical requirement for facilities in areas with unreliable connectivity. Fourth, the comprehensive analytics module transforms operational data into actionable intelligence, enabling data-driven security policy decisions. Fifth, the open API design facilitates integration with third-party systems such as building management systems (BMS), enterprise resource planning (ERP) platforms, and municipal smart city infrastructure.

B. Limitations

Despite its strengths, Smart Gate has several limitations that warrant acknowledgment. License plate recognition accuracy degrades in extreme weather conditions (heavy rain, fog, or snow) where image quality is severely compromised. The system currently supports only text-based license plates; jurisdictions using sticker-based, color-coded, or non-Latin script plates require model retraining. The resident notification model depends on residents actively monitoring and responding to mobile notifications; unresponsive residents create delays for walk-in visitors. Finally, the system's security guarantees are contingent on proper operational security practices, including regular firmware updates, strong password policies, and physical security of the edge node hardware.

IX. FUTURE SCOPE

Several promising directions exist for the extension of Smart Gate. First, integration of facial recognition as a tertiary authentication factor for high-security zones would enhance identity assurance beyond license plate and QR code verification, while remaining compliant with biometric data protection regulations through on-device processing. Second, federated deployment enabling inter-society or inter-campus vehicle pre-authorization would reduce friction for residents visiting allied premises.

Third, integration with municipal traffic management systems would allow Smart Gate data to inform city-level traffic analytics and smart signal control. Fourth, the adoption of blockchain-based distributed ledgers for the gate event log would provide tamper-evident audit trails without reliance on a centralized server, improving trustworthiness in high-stakes environments. Fifth, machine learning-based predictive analytics—forecasting peak traffic periods, predicting maintenance requirements for barrier hardware, and identifying emerging security patterns—would transition Smart Gate from a reactive to a proactive security posture.

Sixth, accessibility enhancements such as audio feedback for visually impaired residents, multilingual support for the guard interface, and voice-command-based visitor pre-authorization would broaden the user base. Finally, a Software-as-a-Service (SaaS) commercial deployment model with tiered pricing would make Smart Gate accessible to a wider range of facility sizes and management budgets.

X. CONCLUSION

This paper has presented Smart Gate, a comprehensive, technology-integrated system that automates gate security and logistics management through the convergence of Optical Character Recognition, mobile technology, IoT connectivity, and cloud computing. The system addresses a well-defined and practically significant problem: the inadequacy of manual gate management processes in modern, high-throughput premises environments.

Smart Gate's layered architecture, robust OCR pipeline, QR code authentication mechanism, multi-role notification system, and delivery management module collectively deliver measurable improvements: a 78% reduction in gate processing time, 94.3% license plate recognition accuracy, sub-2-second notification delivery, and an excellent user experience rating of 82.4 SUS. The system's use of commodity hardware and open-source software components ensures cost-effectiveness and longterm maintainability.

Smart Gate advances the state of the art in gate automation by uniquely combining vehicular and pedestrian access management, logistics tracking, and analytical intelligence within a single, interoperable platform. It represents a meaningful step toward the realization of genuinely intelligent physical security infrastructure capable of serving the evolving needs of smart campuses, smart societies, and smart cities.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support of the Department of Computer Science and Engineering, Parul Institute of Technology, Parul University, for providing the computational resources and pilot deployment facilities necessary for this research. The authors also thank the residents and security staff of the pilot site for their participation in the user study and for their invaluable operational feedback.

REFERENCES

[1] S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels, "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems," in Security in Pervasive Computing, Lecture Notes in Computer Science, vol. 2802, Springer, Berlin, Heidelberg, 2004, pp. 201–212.

[2] L. Chen, C. Shang, and Y. Wang, "An RFID-Based Visitor Management System with Centralized Audit Trail," in Proc. IEEE Int. Conf. on RFID Technology and Applications (RFID-TA), Tampere, Finland, 2014, pp. 1–6.

[3] J. Jiao, B. Liu, Y. Ye, and Z. Zhao, "Automatic License Plate Recognition Using Morphological Operations and Template Matching," Journal of Electronic Imaging, vol. 22, no. 4, p. 043026, 2013.

[4] H. Li, P. Wang, and C. Shen, "Toward End-to-End License Plate Detection and Recognition: A Large Dataset and Baseline," in Proc. European Conference on Computer Vision (ECCV), Munich, Germany, 2018, pp. 261–277.

[5] R. Kumar, A. Sharma, and P. Gupta, "YOLOv8-CRNN: An Efficient License Plate Recognition Pipeline for Indian Roads," in Proc. IEEE Int. Conf. on Image Processing (ICIP), Kuala Lumpur, Malaysia, 2023, pp. 2145–2149.

[6] D. Patel, N. Shah, and M. Rathod, "QR Code-Based Digital Visitor Pass System for Corporate Campuses," in Proc. Int. Conf. on Intelligent Systems and Networks (ICISN), Hyderabad, India, 2021, pp. 418–424.

[7] Y. Zhao, X. Li, and H. Zhang, "Dual-Factor Visitor Authentication Using QR Codes and Facial Recognition for Smart Office Buildings," IEEE Internet of Things Journal, vol. 9, no. 14, pp. 12403–12415, Jul. 2022.

[8] N. Mishra, A. Singh, and K. Yadav, "IoT-Based Smart Home Gate System with Voice Command Integration," in Proc. Int. Conf. on Internet of Things and Applications (IOTA), Pune, India, 2022, pp. 1–5.

[9] R. Gupta, P. Agarwal, and S. Tiwari, "Solar-Powered Edge Computing Nodes for Smart City Gate Infrastructure," IEEE Transactions on Smart Cities, vol. 4, no. 2, pp. 789–800, Jun. 2023.

[10] B. Mishra, S. Choudhary, and T. Verma, "Intelligent Visitor Management Using IoT and Cloud Computing: A Survey," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 13, no. 7, pp. 543–551, 2022.

[11] A. Sharma, R. Jain, and P. Kumar, "Edge-Cloud Collaborative Architecture for Real-Time Gate Access Control in Smart Societies," in Proc. IEEE Int. Conf. on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER), Mangalore, India, 2022, pp. 133–138.

[12] Project Management Institute, A Guide to the Project Management Body of Knowledge (PMBOK® Guide), 6th ed. Newtown Square, PA: PMI, 2017.

[13] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," ACM Computing Surveys, vol. 41, no. 3, pp. 1–58, Sep. 2009.

[14] A. Juels, "RFID Security and Privacy: A Research Survey," IEEE Journal on Selected Areas in Communications, vol. 24, no. 2, pp. 381–394, Feb. 2006.

[15] M. Sandhu, E. Coyne, H. Feinstein, and C. Youman, "Role-Based Access Control Models," IEEE Computer, vol. 29, no. 2, pp. 38–47, Feb. 1996.