International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 6, Nov-Dec-2026, ISSN (Online): 2395-566X

Visitor Pass Management System

Pardeep[1],Annpurna[2], Palak[3]
[1]Assistant Professor
[2,3],Research Scholar , Cse Department, Hmritm Ggsipu, Delhi-110036

ABSTRACT- Visitor management is an important part of organizational security because every person entering a workplace, educational institution, hospital, or other controlled premises must be recorded and authorized appropriately. Paper registers and manually prepared passes can lead to long queues, incomplete records, duplicated entries, and difficulty retrieving information during an incident. This paper presents the design of a Visitor Pass Management System (VPMS), a desktop-oriented application intended to digitize visitor registration, appointment handling, pass generation, and entry–exit recording. The proposed application uses React.js for the user interface, Electron.js to provide a desktop environment, Node.js and Express.js for application services, and MongoDB for persistent data storage. Role-based interfaces are planned for administrators, receptionists, security personnel, and hosts so that each user can access functions relevant to their responsibilities. The design also considers QR-code verification as a proposed extension for validating digital passes. Facial recognition and emergency evacuation assistance are treated as future enhancements that require implementation, privacy review, and testing before they can be considered operational. The paper describes the problem, related work, system architecture, development methodology, security considerations, and a testing plan. Since measured implementation results have not been supplied, the paper does not claim that performance improvements or recognition accuracy have already been demonstrated. Instead, it establishes a structured approach for developing and evaluating a more organized, traceable, and maintainable visitor management process.

Keywords: Visitor management, digital visitor pass, appointment management, access control, entry–exit tracking, QR-code verification, desktop application.

I. INTRODUCTION

Organizations receive visitors for meetings, interviews, deliveries, academic activities, maintenance, and other purposes. A visitor may need to provide identity details, identify a host, state the reason for the visit, and receive permission before entering a restricted area. When these activities are managed through paper registers or disconnected spreadsheets, staff may need to repeat data entry and manually search records. The resulting process can be slow and makes it harder to maintain a consistent history of who entered the premises, whom they visited, and when they departed.

A digital visitor management system can centralize these activities. Registration details can be stored in a structured database, appointments can be associated with hosts, and a pass can be generated after the required approval. Entry and exit events can then be recorded against the same visitor record. Research on QR-code-based visitor management has described digital pre-registration, pass validation, and real-time logging as ways to streamline check-in and improve traceability [1]. The present project adopts this general direction while considering a desktop application suitable for a reception or security workstation. The proposed Visitor Pass Management System is designed around a practical workflow rather than a single registration form. A receptionist should be able to create or locate a visitor record, check the appointment details, and issue a pass when authorization requirements are satisfied. Security personnel should be able to confirm that a pass is valid and record entry or exit. Hosts and administrators need access to relevant visit information, subject to their permissions. This separation of responsibilities is important because a system that stores visitor data but does not define who may view or change it can create new security risks.

The application is planned using React.js for interactive screens and Electron.js for desktop delivery. A Node.js and Express.js service layer will handle application requests, while MongoDB will store visitor profiles, appointments, pass identifiers, user roles, and visit events. This separation makes the interface, business rules, and data storage easier to maintain independently. Environment variables will be used for configuration values that should not be embedded directly in source code.

1. Problem Statement

Manual visitor handling creates several operational weaknesses. Paper registers can be difficult to read, misplaced, or incomplete, and they do not automatically enforce required fields. Appointment details may be kept separately from entry records, making it difficult to confirm whether a visitor was expected. Paper passes can also be copied or reused unless staff verify them carefully. When management requests a report, staff may need to review many pages or spreadsheets to calculate visitor counts and find particular visits. These limitations do not mean that

© 2026 IJSRET

every manual process fails; rather, they show why a consistent digital workflow can be useful in environments with regular visitor traffic.

The central problem addressed by this project is the absence of one organized application that connects visitor registration, appointment information, authorization, pass creation, and entry–exit logging. The proposed VPMS aims to reduce repeated data entry and make visit records easier to retrieve while preserving human review for decisions that require judgment. The system is not intended to replace physical security procedures or guarantee that unauthorized access will never occur.

2. Objectives Of The Study

The primary objective is to design a desktop-based system that maintains visitor information and supports a consistent pass-issuance workflow. The system is intended to capture visitor identity and visit details, associate visits with a host and appointment where applicable, generate a unique digital pass identifier, and maintain entry and exit timestamps. It will provide separate interfaces for administrators, receptionists, security staff, and hosts, with permissions aligned to their roles. A further objective is to support searching and reviewing historical visit records without relying on paper registers.

The project also aims to establish a secure and testable technical structure. Input validation, authentication, role-based authorization, controlled database access, and safe configuration management are considered part of the design. QR-code validation is proposed as an additional method for checking pass identifiers. Facial recognition and automated evacuation assistance are possible future extensions, not confirmed capabilities of the current system. Their value and risks must be assessed through separate implementation and evaluation.

II. RELATED WORK AND LITERATURE REVIEW

Digital visitor management has developed from basic electronic sign-in forms toward workflows that connect registration, identity checks, access permissions, and reporting. Rao and colleagues described a QR-code-based visitor management approach for smart offices in which visitors could schedule appointments in advance and receive a QR code to present at arrival [2]. Their work is relevant to the present project because appointment coordination and pass validation are also central requirements of VPMS. It suggests that a digital pass is most useful when connected to an appointment and a defined authorization process rather than treated as an isolated image.

Mohaideen and colleagues presented a visitor management system with QR-code-based check-in [1]. The paper discusses pre-registration, QR-code passes, authentication, logging, host notifications, and analytics. These functions illustrate how a visitor workflow can be organized around a digital record and how the check-in process can be made more consistent. The present project shares the broad goals of reducing manual registration and improving traceability, but it places particular emphasis on a desktop application architecture using React.js, Electron.js, Node.js, Express.js, and MongoDB. The design should be evaluated against its own requirements rather than assuming that results reported by another implementation will automatically apply.

Research on contactless check-in systems also demonstrates that QR-based processes can support structured access workflows, while highlighting the importance of keeping reliable records of visits [3]. Such systems may use check-in information for operational monitoring, but the design of any particular deployment must account for its setting, the sensitivity of the information collected, and the consequences of an incorrect access decision. For a general-purpose VPMS, a QR code should identify a record or a securely generated token; it should not expose unnecessary personal information in a format that anyone can read by scanning it.

The literature points to recurring design themes: reducing manual work, making visit records searchable, linking visits to responsible hosts, validating passes at entry, and

generating useful summaries. At the same time, a QR code alone does not establish a visitor's identity. A copied code may be presented by another person, and a valid pass may be used outside its intended time or location unless the application checks those conditions. A digital system therefore needs clear authorization rules, expiration.

Handling, duplicate-use controls where appropriate, and a manual exception process for cases in which a visitor cannot use the standard workflow.

III. RESEARCH GAP AND PROJECT MOTIVATION

The reviewed work establishes a foundation for QR-based visitor processing, but a student project still needs to define its intended users, deployment environment, data model, and evaluation method. The VPMS described here brings registration, appointment management, pass generation, and entry–exit records into one desktop-oriented workflow. It also identifies role-based access and privacy controls as core design requirements. Facial recognition and evacuation support are not treated as established solutions in this paper because their reliability, legal suitability, and operational usefulness depend on additional implementation and testing.

© 2026 IJSRET

The project contribution is therefore a clearly scoped system design and an evaluation plan that can be validated during development.

1. Proposed System

Use at a reception or security desk. A user signs in and is shown functions permitted by the user's role. A receptionist can register a visitor, search for an existing record, connect the visit to a host or appointment, and prepare a pass after the required checks. Security personnel can verify the pass identifier and record arrival or departure. Administrators can manage user accounts and review system records, while hosts can view visit information relevant to their appointments. The final workflow and permission boundaries should be confirmed with the institution or organization where the system will be used.

- **Visitor Registration And Appointment Management**

- The registration module captures only the information needed to manage a visit, such as the visitor's name, contact details where necessary, organization, purpose of visit, intended host, and visit date. Required fields should be validated before the record is saved. The appointment module associates a visitor with a planned time and host, allowing reception staff to distinguish expected visitors from walk-ins. Duplicate detection can be based on suitable fields and should prompt staff to review possible matches rather than automatically merging records that may belong to different people.

- **Digital Pass Generation**

  After the registration and authorization checks are complete, the system can create a digital visitor pass containing a unique pass identifier, visitor reference, host or destination, validity period, and current status. A pass should not be considered proof of identity by itself; staff must follow the organization's verification procedure. If QR-code support is implemented, the code should contain a random or signed identifier that can be checked by the application, rather than a plain-text collection of sensitive visitor details. Expired, cancelled, or already-used passes should be handled according to configured policy.

- **Entry–Exit Tracking And Record Management**

The entry–exit module records the time of check-in and check-out and associates both events with the same visit. If a visitor leaves without a recorded exit, the record should remain visibly incomplete instead of being silently treated as closed. Search and filtering functions can help authorized staff retrieve records by date, visitor, host, or status. Reporting functions may summarize visits over a selected period, but reports should avoid exposing personal data to users who do not need it.

- **Role-Based Interfaces**

Role-based access is intended to limit each account to the operations required for its work. Administrators manage configuration and user access; receptionists manage registration and appointments; security personnel perform pass checks and entry–exit recording; and hosts review relevant visit requests or arrival information. These are proposed role boundaries and should be adjusted to match the actual organization's procedures. Authorization must be enforced in the server-side API as well as in the interface, because hiding a button in the user interface is not a sufficient security control.

2. System Architecture And Design

The architecture separates presentation, desktop delivery, application logic, and data storage. React.js provides the interface components and form interactions. Electron.js packages the interface as a desktop application and provides the desktop runtime. Node.js runs the server-side

JavaScript environment, while Express.js defines API endpoints for registration, appointments, pass status, user permissions, and visit events. MongoDB stores the records as documents. Communication between the interface and API should use authenticated requests, and the API should validate every operation before changing the database.

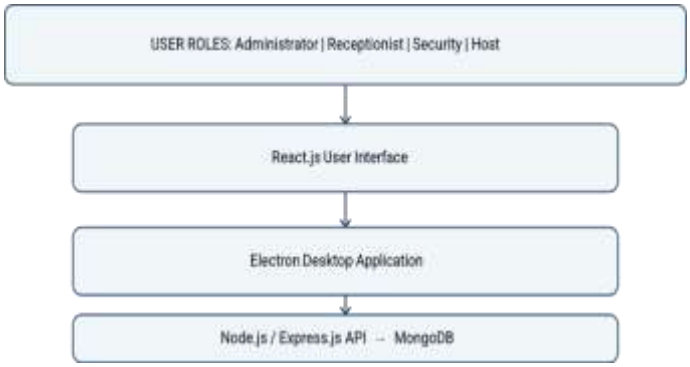

Figure 1. High-level architecture of the proposed Visitor Pass Management System.

The architecture diagram shows the main logical flow rather than a complete deployment topology. User roles interact with the React.js interface through the Electron desktop application. Requests are sent to the Node.js and Express.js service layer, which applies validation and authorization rules before reading or updating MongoDB. QR-code validation, camera-based identity checks, and emergency support are optional supporting functions and are not shown

© 2026 IJSRET

as fully implemented services. If those functions are added, they should be integrated through defined interfaces and tested independently.

3. Data Model

A practical database design may include collections for users, visitors, appointments, passes, and visit events. A user record stores account credentials in a secure form, role, and account status. A visitor record contains the minimum approved personal details needed for identification and contact. An appointment record links a visitor to a host, time window, purpose, and approval status. A pass record stores a unique identifier, validity interval, status, and a reference to the visit. A visit-event record stores check-in and check-out actions, timestamps, and the account that performed each action. Separating events from profile data helps preserve a traceable history when a visit is updated.

Relationships between these records should be maintained through stable identifiers. For example, a pass should reference a visit or appointment instead of relying only on the visitor's name, since names are not guaranteed to be unique. Timestamps should be stored consistently, and the application should define how local time is displayed to users. Database indexes may be added for common searches such as visit date, visitor reference, host, and pass identifier. The actual schema should be revised after requirements analysis and implementation testing.

4. Application Workflow

A typical planned workflow begins when the receptionist creates or locates a visitor record and records the purpose of the visit. If an appointment exists, the system checks its date, host, and status. The authorized staff member then issues a pass. At arrival, security checks the pass and follows the required identity-verification procedure before recording entry. At departure, security records the exit event. If a pass is invalid or a record is incomplete, the application should display a clear status and direct staff to a defined manual review process rather than automatically granting access.

IV. RESEARCH AND DEVELOPMENT METHODOLOGY

The project follows an iterative system-development approach. The first stage is requirement analysis, during which the intended users, visitor workflow, data fields, approval rules, and reporting needs are documented. The second stage is system design, which defines the architecture, database entities, API endpoints, role permissions, and interface navigation. The third stage is implementation, in which the user interface, API, and data layer are developed in small, testable units. The fourth stage is verification, where the system is checked against functional and security requirements. Finally, feedback from representative users can be used to improve usability and correct workflow problems.

1. Requirement Analysis

Functional requirements describe what the application should do: authenticate users, register visitors, manage appointments, create and validate passes, record entry and exit, search records, and restrict operations according to role. Non-functional requirements describe qualities such as usability, reliability, maintainability, response time, privacy, and recoverability. The application should present clear validation messages, preserve records after normal restarts, and handle invalid or incomplete requests without corrupting data. Exact performance targets should be set only after the expected number of users and the deployment hardware are known.

2. Technology Selection

React.js is selected for its component-based approach to interactive user interfaces. Electron.js allows a JavaScript-based interface to run as a desktop application, which suits a staffed reception workstation. Node.js and Express.js provide a JavaScript-based service layer, while MongoDB supports document-oriented storage for visitor

and visit records. HTML, CSS, and JavaScript form the interface foundation. Environment variables separate configuration from source code, but secrets must still be managed securely and must not be exposed to the desktop renderer or committed to a public repository.

3. Implementation Strategy

Development should begin with authentication and role definitions, followed by visitor registration and appointment handling. Pass generation can then be connected to the stored visit record. Entry–exit tracking and search functions should be added after the record lifecycle is stable. The interface should display visit status consistently so that a pending appointment, approved visit, active visit, completed visit, and cancelled pass cannot be confused. Error handling should cover network failures, invalid input, duplicate submissions, expired passes, and database unavailability.

The project should use version control and maintain a small set of documented API contracts. Test data should be synthetic wherever possible, especially during development, so that real visitor information is not copied into demonstration databases. Before a demonstration, the team should verify that the application can be started reliably, that test accounts have the intended roles, and that data can be reset without affecting any real records.

© 2026 IJSRET

4. Evaluation Method

The system should be evaluated using functional tests, role-permission tests, data-integrity checks, and basic usability observations. Functional tests confirm that valid registration, pass generation, check-in, and check-out work as intended. Negative tests verify that missing required fields, invalid pass identifiers, expired passes, and unauthorized API requests are rejected. Data-integrity tests check that each event is attached to the correct visit and that duplicate actions are handled consistently. Usability evaluation can ask representative users to complete common tasks and record whether instructions, labels, and error messages are understandable.

V. SECURITY, PRIVACY, AND RELIABILITY

Visitor records can contain personally identifiable information, so the system should collect only the fields required for a legitimate operational purpose. Access to records should be restricted by role, and the organization should define how long visitor information is retained and how it is deleted when no longer needed. Authentication credentials must never be stored as plain text. Passwords should be processed using an appropriate password-hashing method, and session or token handling should be designed to reduce the risk of account misuse. Sensitive configuration values should be kept outside the source code and protected in the deployment environment.

Authorization checks must be applied at the API layer for every sensitive action. For example, a receptionist should not be able to change administrator permissions merely by altering a request, and a host should not automatically receive access to every visitor record. The application should validate data on both the client and server, while treating server-side validation as authoritative. Database queries should be constructed safely, errors should not reveal secrets or internal configuration, and logs should avoid storing unnecessary personal information.

A QR code is a convenient way to identify a pass record, but it is not inherently secure. The system should use non-guessable identifiers or signed tokens, check validity periods and status, and define whether a pass can be used more than once. The QR content should not include a visitor's full identity details unless there is a specific, justified need. A lost or shared pass should be revocable, and the application should provide a manual verification path for failed scans or accessibility needs.

1. Proposed Facial Recognition

Facial recognition is considered a possible future extension, not a verified feature of the current project. Before it is implemented, the team would need to define the use case, select an appropriate method, evaluate accuracy under realistic lighting and camera conditions, and test how the system behaves for different users. False acceptance and false rejection can both have serious consequences. Any deployment would also require a clear privacy assessment, appropriate consent or other lawful basis, limited retention of biometric data, and a non-biometric alternative for visitors who cannot or do not wish to use the feature.

2. Proposed Emergency Evacuation Assistance

The system could later provide an emergency dashboard based on the latest recorded entry and exit events. Such a dashboard might show visits whose exit has not been recorded and help staff reconcile the list during an evacuation. However, an absent exit timestamp does not prove that a person remains inside: a person may have left without checking out, or a record may be incorrect. The dashboard must therefore be treated as an aid to staff coordination rather than a definitive headcount or a replacement for established emergency procedures. Its

usefulness should be tested through controlled drills and comparison with actual occupancy checks.

3. Reliability And Data Recovery

A visitor management application should define how it behaves when the network or database is unavailable. Staff should receive a clear message when an action has not been saved, and the interface should avoid showing a successful check-in until the server confirms that the event was recorded.

Regular backups, restricted administrative access, and tested restoration procedures are important for preserving visit history. Audit records can help identify who changed a pass status or edited a visit, but their content and retention period should be proportionate to the organization's needs.

4. Testing Plan And Results Discussion

A structured test plan is necessary to determine whether the implemented system meets its requirements. Table 1 lists representative test scenarios. These are proposed tests and not reported outcomes. The team should execute them against the completed application, record the observed result, and document any defect before claiming that a function has passed.

Table 1. Proposed functional and access-control test scenarios.

Test scenario	Expected behavior
rReeqguiisrteedr vfiieslidtosr with valid	rReefecroerndcies issavretdurannedd.a unique visitor

© 2026 IJSRET

Test scenario	Expected behavior
Smuibssminigt rreegqiustiraetdiodna twaith	vSaulbidmaitsiosinonmiessrseajg eectiesdshaondwna.useful
lCinrekaatehaonstapp ointment and	cAoprpreocitntvmiseitnotri,shsot sotr,eadnwd ittihmteh.e
aGuetnheoraizt eda vpiassist for an	aPraes stiodrendtiafniedr dainsdplvaayleidi.ty status
cCahneccekllaend epxapssired or	Pmaasrskiesdraesjeacptepdroavn edde. ntry is not
eRxeictord visitor entry and	cToimrrecstavmispits raerceoardtt.ached to the
aAnttuenmaputhaonr oizpeedrartoiolen with	nTohepr AoPteIcdtednideastathce hoapnegreast.ion and
Search historical visits	tOhneluysmeraatcrehirnegturren ceodr.ds permitted to

No measured execution results are included in this draft because verified test outputs have not been provided. Once the application is implemented, the authors should replace this statement with the actual number of tests executed, the number passed and failed, the environment used, and a short analysis of any defects. Response-time claims should be supported by measurements, and any comparison with a manual process should use a consistent task and a documented measurement procedure.

5. Limitations

The effectiveness of the proposed system depends on accurate data entry, staff adoption, and reliable use of the check-in and check-out workflow. A desktop application may be less convenient for organizations that require access from many locations unless deployment and synchronization are planned carefully. The system cannot independently confirm a person's identity merely because a record or QR code exists. It also cannot guarantee a complete evacuation count if exits are not recorded consistently. These limitations should be made clear to users and administrators.

Additional limitations relate to deployment security and data governance. The application needs appropriate account management, backups, operating-system security, and a defined retention policy. Performance and usability will depend on the actual hardware, visitor volume, and staff workflow. The system should therefore be evaluated in the intended environment before being used for operational security decisions.

6. Future Scope

Future development may include QR scanning at the security desk, automated host notifications, visitor analytics, integration with appointment calendars, and a configurable approval workflow. These enhancements should be prioritized according to user needs and tested for reliability. Facial recognition may be considered only after a separate feasibility, accuracy, and privacy evaluation. Emergency evacuation assistance could be developed as a reconciliation dashboard that clearly displays the age and source of each record and warns staff that recorded occupancy may be incomplete.

VI. CONCLUSION

This paper has presented the design of a Visitor Pass Management System intended to improve the organization of visitor registration, appointment handling, digital pass generation, and entry–exit recording. By connecting these activities through a desktop application, the proposed system aims to make visit information easier to retrieve and reduce dependence on paper-based registers. The architecture uses React.js and Electron.js for the desktop interface, Node.js and Express.js for application services, and MongoDB for data storage. Role-based interfaces, input validation, and server-side authorization are included as important design requirements rather than optional additions.

Related research on QR-code-based visitor management supports the relevance of digital registration and pass validation, but the proposed VPMS must be evaluated on its own implementation. QR-code verification remains a feature to be implemented and tested if it is not already available in the project. Facial recognition and emergency evacuation assistance are future possibilities and must not be presented as completed capabilities without evidence. The next stage is to implement the defined modules, execute the test plan, document actual results, and revise the design based on observed performance and user feedback. This evidence-based process will determine whether the system provides practical improvements in the intended environment.

REFERENCES

1. M. S. Mohaideen, V. Vinu, M. Visagar, and S. V. Arjun, "Visitor Management System with QR Code-Based Check-In," International Research Journal of Education and Technology, vol. 7, no. 3, Mar. 2025. DOI: 10.70127/irjedt.vol.7.issue03.865. Available: https://www.irjweb.com/viewarticle.php?aid=VISITOR-MANAGEMENT-SYSTEM-WITH-QR-CODE-BASED-CHECK-IN
2. S. Rao and C. Jonathan, "QR Code Based Visitor Management System for Smart Offices," International Journal of Recent Technology and Engineering, vol. 8, no. 4, 2019, pp. 8787–8791. DOI: 10.35940/ijrte.D9404.118419.
3. "HealthPass: A Contactless Check-In and Adaptive Access Control System for Lowering Cluster Infection

© 2026 IJSRET

Risk in Public Health Crisis," Frontiers in Public Health, 2024. DOI: 10.3389/fpubh.2024.1448901.
4. React Team, "React Documentation." Available: https://react.dev/
5. Electron, "Electron Documentation." Available: https://www.electronjs.org/docs/latest/
6. Express.js, "Express Documentation." Available: https://expressjs.com/MongoDB, "MongoDB Manual." Available: https://www.mongodb.com/docs/

© 2026 IJSRET