A centralized QR-based campus application for access to institutional services

Aarushi Singh, Archie Singh, Riddhi Jain, Riya Dhyawna
Department of Computer Science Engineering
Indian Institute of Technology, Jodhpur

I. Problem Statement

A. Objective

- To automate student entry and exit tracking using QR technology
- To provide real-time lecture attendance monitoring
- To reduce manual effort and errors in attendance recording
- To maintain a centralized and secure database of student records
- To provide role-based access for guards, students, faculty, and administrators

B. Motivation

- Manual attendance systems are inefficient and prone to mistakes
- Institutions require better monitoring of student movement and presence
- Digital solutions improve efficiency, transparency, and security
- Integration with academic schedules ensures accurate attendance mapping
- Automation reduces workload on guards and faculty

II. Alpha Team

A. Alpha Team Structure

The project is developed by Alpha Team, consisting of four members:

- Aarushi – Project workflow, opportunities, and coordination
- Archie – Team processes and collaboration strategy
- Riddhi – Software system design and stakeholder analysis
- Riya – Requirements engineering and documentation

The Alpha Team is responsible for requirement analysis, system design, sprint planning, implementation strategy, UML modelling, and documentation.

B. Roles and Responsibilities

- Project Coordination and Sprint Planning: Managed collaboratively by the team
- SRS Development and Documentation: Prepared and reviewed by Alpha Team
- UML Design and Architecture Decisions: Joint responsibility with internal reviews
- Traceability and Version Control: Maintained across SRS versions
- Quality Assurance: Peer review of diagrams, modules, and sprint deliverables

C. Approval Authority

Version 1.0 and Version 2.0 of the SRS were reviewed and approved internally by Alpha Team members after analysis and feedback incorporation.

III. Review of Existing Technologies for Campus Access & Attendance Management

University Campuses and even company organisations use technologies like RFID (Radio-Frequency Identification), biometric systems, bluetooth systems, mobile based NFC (Near Field Communication) systems and QR based systems. The strengths and shortcomings are analysed below to motivate why a QR-based solution will help replace our university's (IIT Jodhpur) manual entry/exit and attendance system (currently the scope of the project).

A. Manual Attendance and Entry Logging

Current classroom and gate systems typically rely on paper registers or manual data entry performed by the student themselves in the presence of a supervising entity.

- Advantages:
  - Low cost; requires no investment in technology.
  - Simple to understand and immediately implement.
  - No dependency on electronic devices, power, or infrastructure.
- Disadvantages:
  - Highly prone to human errors (e.g., illegible handwriting, missed entries).
  - Time-consuming, especially for large groups or peak hours, leading to long queues and cluttering.
  - Lacks real-time tracking, analytics, or digital audit logs.
  - Easy to manipulate via the security guards to hide alibi or put attendance by proxy.

B. RFID Card-Based Systems

Radio-Frequency Identification (RFID) cards are issued to users and the proximity is detected to grant access or log entry/exit etc [2].

- Advantages:

  - Faster than manual entry
  - Cards are reusable and durable.
  - Effective for high-traffic locations.
  - Easily integrated with physical barriers like turnstiles.
- Disadvantages:
  - Cards can be lost, stolen, or shared, leading to proxy attendance and fake entry/exit logs to hide alibi.
  - Requires installation of physical readers at every access point (high infrastructure cost).
  - Data are often limited to simple presence logs without context.
  - Tedious task of distributing and replacing physical cards.

C. Biometric Systems (Fingerprint/Iris/Face)

These systems capture physical traits (fingerprint, face, iris) and match them against stored templates to verify identity [3].

- Advantages:
  - Very high identification accuracy.
  - Extremely difficult to fake or share (eliminates proxy attendance).
  - Removes the need for users to carry physical ID cards.
- Disadvantages:
  - Very expensive hardware and setup.
  - Collection of sensitive biometric data raises significant privacy concerns.
  - Can be slow to process large queues.
  - Susceptible to false negatives due to environmental factors (e.g., wet fingers, poor lighting, if there is a damage to that body part like burning of fingerprints, losing eyes).

D. Mobile NFC/Bluetooth/BLE Solutions

Utilize mobile devices to trigger attendance or access via Near Field Communication (NFC) or Bluetooth Low Energy (BLE) beacons [4].

- Advantages:
  - Contactless and rapid processing.
  - Uses a familiar user interface (personal smartphones).
  - Can integrate location data and additional context.
- Disadvantages:
  - Requires users to have phones with specific NFC/BLE hardware.
  - Dependent on battery life and device connectivity.
  - Compatibility issues across different operating systems (current issue for the library based application used in IIT Jodhpur).
  - Security risks if the transmission is not properly encrypted, as the communication is open.

E. QR-Based Systems (Camera / Scanner)

Users present a dynamic or static QR code. Standard cameras or dedicated QR scanners read the code to log the event [5].

- Advantages:
  - Low cost; QR codes can be displayed on phones or etched on physical id cards (for static only).
  - Compatible with standard smartphone cameras and inexpensive scanners.
  - Offers a good balance of speed and security.
  - Can encode detailed, encrypted information (ID, timestamp, session data).
- Disadvantages:
  - Static QR codes can be photographed and shared (spoofing risk, proxies, fake alibi).
  - Requires adequate lighting and camera resolution.
  - Real-time validation typically requires network connectivity.

1) Security Trade-Off Analysis and Design Clarification: While static QR codes present spoofing risks due to possible screenshot sharing or photographic duplication, the proposed system does not rely solely on static QR validation. Although each student is assigned a single QR identifier for their academic tenure, the security model incorporates multiple mitigation layers.

First, all QR scans undergo server-side validation, ensuring that merely presenting a copied QR image is insufficient without backend verification. Second, the system displays the registered student photograph on the guard interface during each scan, enabling manual identity confirmation. This introduces a human verification layer that reduces impersonation risk.

Additionally, anti-passback logic prevents repeated "IN" or "OUT" entries without corresponding state transitions, thereby limiting replay attacks using duplicated QR images. The QR identifier functions as a reference token rather than a standalone authentication artifact.

Thus, while the identifier remains tenure-bound for lifecycle simplicity and database consistency, real-time validation, state management, and role-based verification collectively mitigate risks typically associated with static QR implementations. This represents a balanced trade-off between infrastructure cost, usability, and institutional security requirements.

F. Comparative Summary

A comparison of the key attributes of these technologies is presented in Table I.

G. Justification for QR-Based Approach

Analysing and comparing these advantages and disadvantages of each system, QR-based systems are identified to be the most optimal solution for an academic institution (our target). The factors include but are not limited to:

- It utilizes existing devices and used in the form of an application (smartphones) and requires minimal hardware investment.

TABLE I
COMPARATIVE ANALYSIS OF ACCESS TECHNOLOGIES

Technology	Cost	Accuracy	Spoof Risk	Infrastructure
Manual	Very Low	Low	High	Minimal
RFID	Medium	Medium	Medium	Readers Needed
Biometric	High	Very High	Low	Biometric Sensors
NFC/BLE	Medium	Medium	Medium	NFC Devices
QR-Based	Low	High	Medium*	Cameras/Scanners

*QR spoof risk can be mitigated by server-side validation.

- It eliminates the need for specialized biometric sensors or expensive RFID readers.
- It works universally across different user devices.
- It easily supports multiple functions including attendance tracking, gate entry/exit, and digital identity verification within a single application. This is consistent with integrated QR code-based approaches used in modern university e-class systems. [1].
- The system is purely software-driven with the reference to a database explained in detail in the solution section of this document, making updates and maintenance straightforward.

To avoid security risks, we have decided the QR based technology to refer to the photo for verification by the security and other info of the student getting the QR scanned from their phone/ ID card (if etched on the ID card, a possibility currently in review). QR based systems improve efficiency and are cost effective, and currently a version of this is used in library systems, and some classrooms (based on the instructor) and also the food mess (using hashing to avoid other forms of spoofing) in IIT Jodhpur.

IV. PROPOSED SOLUTION AND DEVELOPMENT TRAJECTORY

This section explains how the proposed QR-based campus system was designed and refined. The development was carried out in stages, starting with an initial understanding of the problem and gradually improving the design based on feasibility checks, design reviews, and academic use cases. Instead of fixing all details at once, the system was improved multiple versions of the Software Requirements Specifications (SRS).

This approach helped ensure that the system remained clear, practical, and suitable for real institutional use.

A. Evolution of Software Requirements

SRS version 1 focused on defining main goals and basic functionality of the system. It included features such as QR based identity verification, student entry and exit logging, and attendance marking. At this stage, the requirements were kept general to define the scope of the system.

As the design developed, some functional and non-functional requirements were unclear and not detailed enough. Some important aspects such as performance limits, security, data retention, and QR code validity were not defined.

To address these issues, SRS version 2.0 was prepared. This version refined the functional requirements and clearly defined non-functional requirements. It introduced rules such as assigning a single QR code per student for the entire academic tenure, maintaining user status values (Active, Graduated), invalidating QR codes after graduation, and archiving student records. Version 2.0 also improved clarity by linking requirements with UML diagrams and traceability matrices.

B. System Architecture Overview

The system is designed using a layered architecture so that different parts of the system can operate independently. This makes the system easier to understand, maintain, and extend in the future.

The architecture consists of three main layers:

- **Presentation Layer**: Provides the User Interface for Students, Guards, and administrators. It handles QR display, scanning, and showing verification results.
- **Application Layer**: Contains the main system logic, including QR Validation, access control, attendance mapping, and checks to prevent incorrect repeated entries.
- **Data Layer**: Stores user information, attendance records, access logs, and QR related data while following defined data retention rules.

This layered structure helps keep the system organized and reliable.

C. Technology Stack selection and justification

The technology stack was chosen based on simplicity, reliability, and ease of use in academic environment like IIT Jodhpur.

QR code generation is handled using the open-source Python library `python-qrcode`. This library was selected because it is easy to integrate with a Python backend and follows standard QR encoding methods. Server-side validation is used to reduce the risk of QR misuse.

A relational database is used to store structured data such as user details, attendance logs, and entry records. This ensures data consistency and supports clear traceability. The system relies on camera-based scanning, allowing it to work with commonly available devices without the need for specialized hardware.

D. Sprint Planning and Agile Development Process

The project followed an agile development approach, where work was divided into multiple sprints.Each sprint focused on refining a specific part of the system design.

- **Sprint 1**: Requirement analysis and preparation of SRS version 1
- **Sprint 2**: Requirement refinement, UML modeling, and development of SRS version 2.0.
- **Sprint 3**: Logical validation, traceability mapping, and documentation updates.

At the end of each sprint, the design was reviewed and improved based on identified issues. This allowed gradual and controlled improvement of the system.

E. Agile Implementation Evidence

To ensure that the agile methodology described was not limited to documentation evolution, practical sprint-level planning and monitoring artifacts were also prepared during development.

1) Sprint Backlog Sample: Each sprint included a defined backlog of tasks mapped to system modules. A sample backlog from Sprint 2 is shown in Table II.

TABLE II
SAMPLE SPRINT BACKLOG (SPRINT 2)

Task ID	Task Description	Owner	Status
SB-1	Refine functional requirements	Riddhi	Completed
SB-2	Design QR validation logic	Riya	Completed
SB-3	Update UML diagrams	Aarushi	Completed
SB-4	Traceability mapping	Archie	Completed
SB-5	Draft NFR specifications	Team	Completed

2) Sprint Velocity Observation: Sprint velocity was measured based on completion of planned backlog items within each sprint cycle.

TABLE III
SPRINT VELOCITY SUMMARY

Sprint	Planned Tasks	Completed Tasks
Sprint 1	6	6
Sprint 2	8	8
Sprint 3	7	7

All planned tasks were completed within sprint timelines, indicating stable sprint velocity and realistic workload distribution.

3) Burndown Trend Representation: Although formal project management tools were not used, sprint progress was tracked manually. Work completion followed a steady reduction pattern across sprint duration, similar to a standard burndown trend.

TABLE IV
ILLUSTRATIVE SPRINT BURNDOWN DATA (SPRINT 2)

Day	Remaining Tasks
Day 1	8
Day 2	6
Day 3	4
Day 4	2
Day 5	0

The decreasing trend indicates consistent progress and balanced task allocation across team members.

4) Agile Implementation Interpretation: The agile framework in this project focused on iterative requirement refinement, UML validation, and prototype planning rather than large-scale coding sprints. Each sprint resulted in a measurable improvement in system clarity, requirement precision, and architectural stability.

Thus, the agile process was applied to:

- Requirement engineering and validation
- System architecture refinement
- Traceability and documentation evolution
- Prototype planning and feasibility checks

This ensured continuous improvement and structured progress throughout the development lifecycle.

F. Scrum Team Roles and Responsibilities

A simplified Scrum Framework was used to manage Development activities.

The Product Owner handled requirement definition and feature prioritization. The Scrum Master coordinated sprint activities and ensured smooth progress. The development Team was responsible for requirement refinement, design validation, UML preparation and documentation.

This structure helped maintain clear responsibility and effective coordination.

G. Development and coding Protocols

Standard development practices were followed throughout the project. Version control was used to track changes across SRS versions and design updates. Clear documentation standards were maintained to ensure consistency between requirements, UML diagrams, and traceability matrices.

Logical checks were applied during requirement refinement to prevent issues such as duplicate entry logs, invalid QR reuse, and incorrect status transitions.

H. Ethical and Privacy Considerations

Since the proposed system tracks student identity, attendance records, and entry/exit movement, ethical handling of data is a critical design requirement.

1) Data Protection and Compliance: The system follows data minimization principles by storing only essential information required for authentication, attendance logging, and institutional reporting. Sensitive data such as biometric identifiers are not collected. Student records are archived after graduation and may be purged according to defined data retention policies, reducing long-term storage risks.

2) Student Consent and Transparency: The system is designed to operate under institutional authorization and informed student consent. Students are informed that their QR identifier is used solely for identity verification, attendance logging, and campus access control. Access logs are visible only to authorized administrative roles, ensuring transparency in how data is used.

3) Secure Data Transmission and Encryption: All QR validation requests are designed to be processed via secure server-side APIs. Communication between client applications and backend services is assumed to be protected using HTTPS (TLS encryption) to prevent interception or tampering during transmission. Stored data within the database is accessible only through authenticated service layers, reducing unauthorized access risk.

4) Role-Based Access Control: Access to sensitive student information is restricted through role-based authorization mechanisms. Guards can verify identity and entry status but cannot modify academic records. Administrative overrides are logged with timestamps and guard identification to ensure accountability and auditability.

By integrating privacy-aware architecture decisions alongside functional requirements, the proposed system aims to balance institutional security needs with ethical responsibility and student data protection.

V. Results, Evaluation and Discussion

A. Design-Level Results: Refinement from Version 1.0 to Version 2.0

At the current stage, the project has completed the development of SRS Version 2.0. The results discussed in this section are based on requirement refinement, logical validation, and structural improvements made between Version 1.0 and Version 2.0.

The transition to Version 2.0 reflects feedback incorporation, clearer definition of system behavior, and improved alignment between requirements and UML diagrams. The updated SRS demonstrates improved clarity, reduced ambiguity, and better feasibility for future implementation.

B. Logical Validation of Entry/Exit Control

During requirement analysis, a potential issue was identified: repeated QR scans could generate incorrect multiple "IN" or "OUT" logs. The design was refined to include a validation step that checks the user's last recorded status before allowing a new entry or exit.

This led to the inclusion of anti-passback logic in the updated UML class structure and functional requirements. The refinement strengthens system reliability and prevents logical inconsistencies in attendance and gate logging.

C. QR Lifecycle and Data Governance Improvements

In Version 1.0, the lifecycle of QR codes and treatment of graduated student records were not clearly defined. Version 2.0 introduced the following improvements:

- Each user is assigned one QR code for their entire academic tenure.
- A user status attribute (Active, Graduated, Blocked) was added.
- QR codes are automatically invalidated after graduation.
- Records of graduated students are archived and may be purged after a defined retention period.

These changes improve long-term data governance and ensure realistic institutional applicability. It is important to note that although the QR identifier remains constant during a student's tenure, it does not function as a static offline credential. Each scan is validated against live backend records, and the system checks current status flags, entry history, and authorization constraints before confirming access. Therefore, the tenure-based identifier simplifies the tedious structure already present without compromising operational security.

D. Technology Selection Outcome

The project finalized the use of the open-source Python library `python-qrcode` [6] for QR code generation. The selection was based on:

- Ease of integration with Python backend
- Standardized QR encoding support

An established Python library reduces development complexity and ensures uniformity in QR generation and decoding.

E. Theoretical Validation and Architectural Decision

Since the current phase of the project focuses on building a small working prototype, architectural decisions were evaluated with respect to both prototype simplicity and future scalability.

Three lookup strategies were conceptually analyzed:

- **Baseline Method:** Sequential record scan for each QR validation (linear search).
- **Indexed Database Retrieval:** Primary-key indexing using database structures (approximately $\mathcal{O}(\log n)$).
- **Hash-Based Lookup:** In-memory key-value retrieval using hashing (average-case $\mathcal{O}(1)$).

For the prototype stage, a Python dictionary was selected to store user records. Python dictionaries internally use hash tables, enabling near-constant average lookup time. This makes them efficient and simple to implement for small-scale systems.

Synthetic latency modeling was performed to compare the theoretical growth behavior of these strategies.

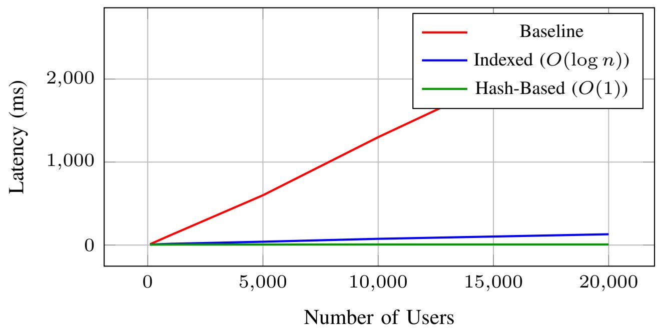

Fig. 1. Theoretical scalability comparison of lookup strategies

The modeling indicates that while indexed database retrieval significantly improves scalability compared to naive sequential scanning, hash-based lookup maintains near-constant latency even as dataset size increases.

Given that the current implementation targets a small prototype environment, the hash-based approach using Python dictionaries was selected for its simplicity and performance benefits. However, for large institutional deployment, database indexing or distributed storage mechanisms would be more appropriate to ensure persistence, reliability, and scalability.

F. Summary of Design Evolution

The summary of improvements across versions is presented in Table IX.

TABLE V
REVISION HISTORY

Version	Date	Name	Reasons for changes
1.0	21/01/2026	Initial SRS Submission	N/A
2.0	28/01/2026	SRS Version 2	• Modified UML diagrams + inserted UML • Added clarification about need for changing QR • Added functionality about deleting data for deregistered students • Highlighted Alpha-team recommendations regarding v1.0 approval • Clarified location indicator in UI to reduce confusion • Used "justify" alignment for document • Documented Traceability matrices for FR and NFR • Clarified manual entry mandate (see use case/traceability) • Added Design and Coding standards (template chosen) • Added techstack • Added realistic sprint plan for modules

TABLE VI
TRACEABILITY MATRIX FUNCTIONAL REQUIREMENTS

FR ID	Functional Requirement	Components	Holder	Source	UML Map
FR-1	Decode QR codes	Unified App (Student/Guard), QR Processor	Riya	QR Proc.	Use Case, Seq
FR-2	Validation service returns DB response within time interval	API Gateway, QR Processor	Aarushi	Valid. & Perf.	Seq, Activity
FR-3	Display student's photo on guard's screen after scan	Unified App (Guard), Data Access	Archie	Identity Verif.	Sequence
FR-4	Assign one QR code per user for academic tenure	Auth Service, Data Access	Riddhi	QR Lifecycle	Class, Use Case
FR-5	Maintain user status (Active, Grad, Blocked) & archiving	Auth Service, Data Access	Archie	User Mgmt	Class, Comp
FR-6	Invalidate QR on Grad; archive records; no alumni storage	Data Sync, Data Access	Riddhi	Data Retention	Class, Seq, Comp
FR-7	Maintain current_status flag for every user	Campus Actors DB, Data Access	Riya	Presence Track	Class, DB
FR-8	Reset status to "Outside" only after successful exit	QR Processor, Data Access	Aarushi	Entry/Exit	Seq, Activity
FR-9	Require guard auth before override functions	Auth Service, Unified App	Riya, Aarushi	Security	Use Case, Comp
FR-10	Log manual override with Guard ID and reason	Auth Service, Data Access	Archie, Riddhi	Audit Log	Seq, Activity

G. *Overall Discussion*

The refinement of SRS Version 2.0 demonstrates systematic design improvement. Logical inconsistencies were identified and addressed, lifecycle policies were clarified, and technology components were selected to support future development.

The traceability matrix confirms alignment between updated functional requirements, UML diagrams, and planned implementation modules. The current results indicate that the proposed QR-based identity and attendance system is conceptually sound and ready for the implementation phase.

The following section presents prototype-level experimental evaluation. Unlike earlier design and theoretical validation, these results are based on actual execution of a working prototype to assess correctness and response time behavior under controlled conditions.

VI. PROTOTYPE IMPLEMENTATION EVIDENCE

To validate the feasibility of the proposed QR-based campus system, a small working prototype was developed. The prototype focuses on QR generation, scanning simulation, and database lookup validation.

A. *Working Prototype Overview*

A basic prototype was implemented using Python to simulate the core workflow:

- Generation of unique QR code for each student using `python-qrcode`
- Storage of student records in a Python dictionary (hash-based lookup)
- Simulated scanning and validation process
- Display of student identity and status after scan

TABLE VII
TRACEABILITY MATRIX NON-FUNCTIONAL REQUIREMENTS

ID	Requirement	Components	Holder	Category	Source	Feature	Artifacts	Ver.
NFR-1	The system shall authenticate users before allowing access to personal data.	API Gateway, Auth Service	Archie	Security	Prevent misuse of shared QR	Digital ID, QR Scan	Use-Case, Seq Diag	V2.0 clr
NFR-2	The system shall ensure secure data transmission between client and server.	API Gateway, Unified App	Riya	Security	Protect sensitive records	All modules	Comp, Deploy Diag	V2.0 unc
NFR-3	The system shall respond to a QR scan request within 2 seconds under normal network conditions.	QR Proc, Data Access	Riddhi	Perf.	Smooth workflow	Gate Entry, Attend.	Timing Diagram	V2.0 add
NFR-4	The system shall be available during institutional working hours with minimal downtime.	API Gateway, Backend (All)	Aarushi	Avail.	Daily usage	All modules	Deploy Diagram	V2.0 unc
NFR-5	The system shall support mobile, tablet, and desktop devices.	Unified App, REST API	Riya	Compat.	Multi-device access	Digital ID, Admin	Deploy Diagram	V2.0 unc
NFR-6	The system should be easy to use by guards with minimal training.	Unified App (Guard View)	Archie	Usability	Guard efficiency	Gate Entry Module	Activity Diagram	V2.0 ref
NFR-7	The system shall store attendance and entry logs in a structured and searchable format.	Data Access, Campus Actors DB	Riddhi	Maint.	Reporting & audits	Attend, Gate Entry	Logical DB Design	V2.0 unc
NFR-8	The system shall automatically deactivate records of graduated students.	Data Sync, Data Access	Aarushi	Data Mgmt	Instructor feedback	Student Mgmt	Class Diagram	V2.0 add
NFR-9	The system shall allow manual guard entry as a fallback when QR scanning fails.	Unified App (Guard), Auth	Riya, Archie, Riddhi, Aarushi	Reliab.	Instructor Feedback	Gate Entry Module	Use-Case Diagram	V2.0 clr

TABLE VIII
THEORETICAL LOOKUP LATENCY COMPARISON

Users	Baseline (ms)	Indexed ($O(\log n)$)	Hash-Based ($O(1)$)
100	10	6	5
1,000	120	15	6
5,000	600	40	6
10,000	1300	75	7
20,000	2600	130	7

TABLE IX
SUMMARY OF DESIGN EVOLUTION

Aspect	Ver 1.0	Ver 2.0	Improvement
Duplicate logging	Not defined	Status validation added	Improved accuracy
QR lifecycle	Undefined	Tenure-based validity	Clear governance
Data retention	Not specified	Archive + purge policy	Better data control
QR handling	Not finalized	`python-qrcode` selected	Practical tech choice

Fig. 2. Prototype

This prototype demonstrates the practical feasibility of the proposed system architecture.

B. Sample Validation Output

When a QR code is scanned, the system retrieves student details from the database and verifies status. A sample console output is shown below:

```
QR Scan Successful
Student Name: Riddhi Jain
Roll Number: B24CS1063
Status: Active
Entry Allowed
```

```
Timestamp: 14:32:18
```

This confirms correct mapping between QR data and stored student records.

C. Test Case Evaluation

A small test dataset was used to validate correctness of the system.

TABLE X
PROTOTYPE TEST CASE RESULTS

Test ID	Scenario	Expected Result	Status
TC1	Valid QR scan	Student verified	Pass
TC2	Invalid QR data	Access denied	Pass
TC3	Repeated entry scan	Duplicate blocked	Pass
TC4	Graduated student QR	Access denied	Pass

D. Timing Measurement (Hash-Based Lookup Prototype)

To evaluate real-time feasibility, QR validation time was measured using a small-scale prototype implemented in Python. Student records were stored using a hash-based dictionary structure to simulate database lookup.

The test was conducted on a standard laptop environment under normal execution conditions.

Each response time value represents an average over multiple executions to minimize measurement noise.

TABLE XI
MEASURED QR VALIDATION TIME (HASH-BASED LOOKUP)

Number of Records	Avg Response Time (ms)
10	2 ms
50	3 ms
100	4 ms

The response time remains nearly constant as the number of records increases. This validates the use of hashing for small-scale deployment, as hash-based lookup provides approximately $\mathcal{O}(1)$ average time complexity.

Even with 100 active records, the response time remains well below the defined non-functional requirement of 2 seconds, confirming suitability for real-time campus use in prototype-scale deployment.

E. Implementation Significance

Although the current prototype is limited in scale, it validates:

- Correct QR-to-user mapping
- Fast hash-based lookup
- Prevention of duplicate entry logs
- Feasibility of real-time validation

These results demonstrate that the proposed system design is practically implementable and scalable for institutional deployment.

VII. CONSTRAINTS AND PRACTICAL LIMITATIONS

During the development and prototype testing phases, several constraints and dependencies emerged that were not initially accounted for in the early planning stages.

A. Infrastructure and Environmental Constraints

The initial design assumed ideal scanning conditions. However, practical testing revealed that low-light environments (e.g., campus gates at night) significantly impede the camera's ability to decode static QR codes quickly. Furthermore, the system relies entirely on the user's smartphone battery availability; if a student's device is discharged, no alternative manual fallback (other than the guard's manual override) exists in the current automated flow.

B. Network Dependencies

The current hash-based validation relies on real-time server connectivity. The design does not yet account for network partitions or latency spikes in the campus WiFi. In a real-world deployment, a lack of connectivity would render the scanning terminals non-functional, necessitating an offline-first architecture which is currently out of scope.

C. Deployment and Scalability Issues

While the Python dictionary lookups ($\mathcal{O}(1)$) work for the prototype, utilizing in-memory storage is volatile. Any server restart results in data loss. Transitioning to a persistent SQL database introduces I/O latency that was not modeled in the theoretical calculations, potentially affecting the non-functional requirement of a 2-second response time under heavy load.

VIII. RISK ANALYSIS AND MITIGATION

Despite careful system design, certain operational and security risks were identified during analysis and prototype development. The major risks and their mitigation strategies are outlined below.

- **QR screenshot sharing and Spoofing Risk:** Static QR codes may be photographed or shared, enabling proxy access. This risk is mitigated through server-side validation, display of the registered student photograph on the guard interface, and manual identity verification during scanning.
- **Replay Attacks:** Repeated scanning of the same QR code could lead to false multiple entry or exit logs. The system mitigates this risk using anti-passback logic that validates the user's previous status before accepting a new scan.
- **Network dependency Risk:** The system relies on real-time connectivity for QR validation. Temporary network failures may disrupt automated scanning. A manual guard override mechanism is provided as a fallback to maintain operational continuity.
- **Data Breach and Privacy Risks** Unauthorized access to student records could compromise privacy. This risk is mitigated through role-based access control, restricted permissions for guards, and secure server-side APIs.

- **Scalability Risk:** The prototype uses in-memory storage which is suitable only for small deployments. For large-scale institutional use, the system is designed to migrate to a persistent indexed database to maintain performance and reliability.

These risks are comparable to those present in existing digital campus systems and are addressed through layered technical controls and institutional oversight.

IX. Conclusion and Future Scope

A. Summary of Achievements

The Alpha Team successfully progressed from an abstract problem statement to a fully defined Software Requirements Specification (Version 2.0). Key achievements include the definition of a tenure-based QR lifecycle, the implementation of logic to prevent anti-passback violations, and the validation of a hash-based lookup prototype that meets performance benchmarks.

B. Assessment of Objectives

The primary objectives of automating entry tracking and ensuring a secure database structure were met through the layered architecture design. The constraints regarding role-based access were successfully mapped to the UML class diagrams and validated through the prototype's logic tests.

C. Future Work

To transform this prototype into a production-ready system for IIT Jodhpur, future development will focus on:

- **Database Migration:** replacing in-memory storage with a persistent PostgreSQL database.
- **Offline Mode:** Implementing local caching to allow scans during network downtime.
- **Mobile Application:** Developing a dedicated frontend (Flutter/React Native) to replace the command-line prototype interface.

X. Generative AI Usage Declaration

Although Artificial Intelligence tools have been used to a limited degree during the development of the project, their assistance has been mostly in the following areas:

- Structuring documentation and improving language clarity
- Reviewing suggestions for format and documentation structure

All technical decisions, architectural planning of the system, use cases, sprint planning, and design logic were performed and verified by the members of Alpha Team. Any content generated using AI tools was critically reviewed, modified, and validated by the team to maintain academic integrity.

The development process was carried out with a primary focus on natural intelligence, team discussions, and collaborative decision-making.

XI. Final Compilation and Formatting

A. Report Compilation and Formatting

The final report was prepared to maintain uniformity across all sections. The document follows IEEE double column formatting guidelines, including standardized headings, tables, and figure alignment.

The formatting practices followed are:

- Justified text alignment for professional presentation
- Consistent font styles and hierarchical headings
- Standardized requirement numbering and traceability references
- Uniform formatting and layout of UML diagrams
- Version tracking using traceability matrices and change logs

B. Final Compilation Activities

- Coordinating contributions from all team members into a single consolidated report
- Verifying continuity and consistency between different document versions
- Cross-checking sprint plans with implementation and design traces
- Ensuring compliance with page limits and submission guidelines

References

[1] abu bakar, Suwaibah & Salleh, Shahril & Rasidi, Azamuddin & Tasmin, Rosmaini & Aziati, A. & Muhammad Nda, Ramatu & Che Rusuli, Ts. Dr. Muhamad Saufi. (2020). Integrating QR Code-Based Approach to University e-Class System for Managing Student Attendance. 10.1007/978-981-15-4409-5_34.

[2] R. Want, "An introduction to RFID technology," IEEE Pervasive Computing, vol. 5, no. 1, 2006.

[3] A. K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," IEEE Transactions on Circuits and Systems for Video Technology, vol. 14, no. 1, 2004.

[4] V. Coskun, B. Ozdenizci, and K. Ok, "A survey on near field communication (NFC) technology," Wireless personal communications, vol. 71, no. 3, 2013.

[5] S. Tiwari, "An Introduction to QR Code Technology," in 2016 International Conference on Information Technology (ICIT), 2016.

[6] Lincoln Loop, "python-qrcode: Python QR Code image generator," *GitHub repository*, 2024. [Online]. Available: https://github.com/lincolnloop/python-qrcode.

XII. Appendix

A. System Activity Diagram

The activity diagram in Fig. 3 represents the workflow of the QR-based campus entry, exit, and attendance system. It shows QR validation, identity verification, operation selection, and database update process.

B. System Sequence Diagram

The Sequence diagram in Fig. 4 illustrates the interaction between the student, QR scanner, server, and database during QR validation, identity verification, and entry/attendance logging.

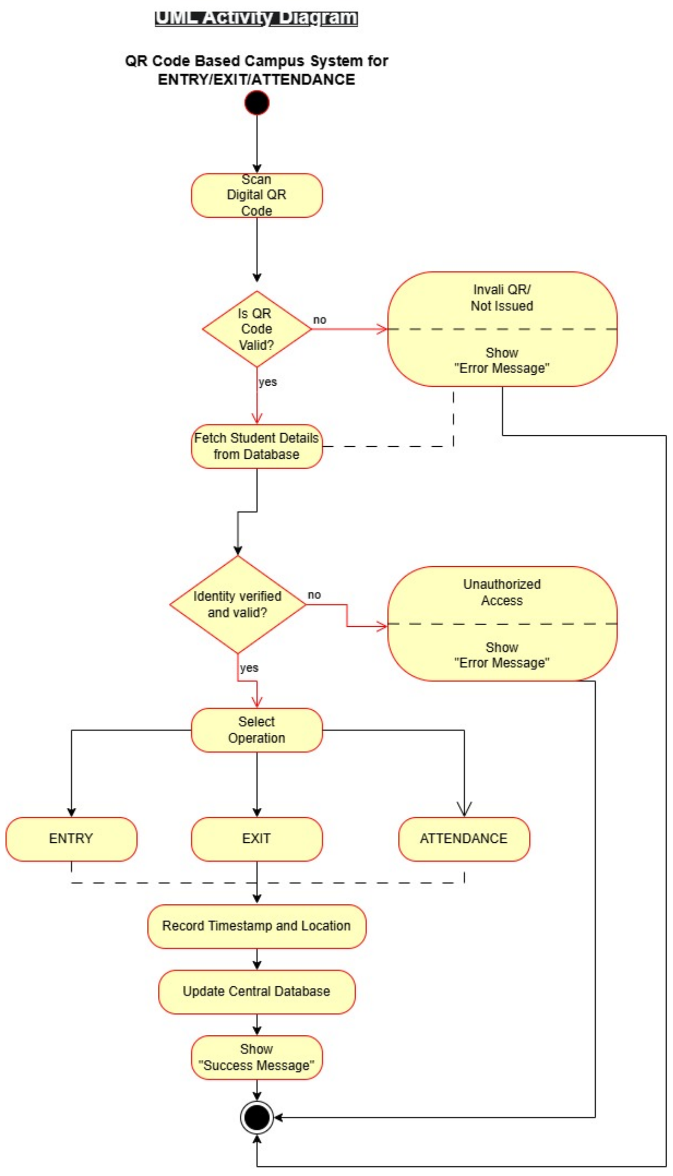

Fig. 3. Activity Diagram for QR-Based Campus Entry/Exit and Attendance System

C. *Use Case Diagram*

The Use Case diagram in Fig. 5 details the interactions between the primary actors (Student, Guard, Faculty) and the system. It highlights core functionalities such as scanning QR codes, verifying identity via the anti-passback engine, and managing attendance records.

D. *System Component Diagram*

The Component diagram in Fig. 6 visualizes the high-level architecture, demonstrating the separation of concerns between the Client Layer (Unified App views), the Backend System (API Gateway, Logic Layer), and the Data Layer.

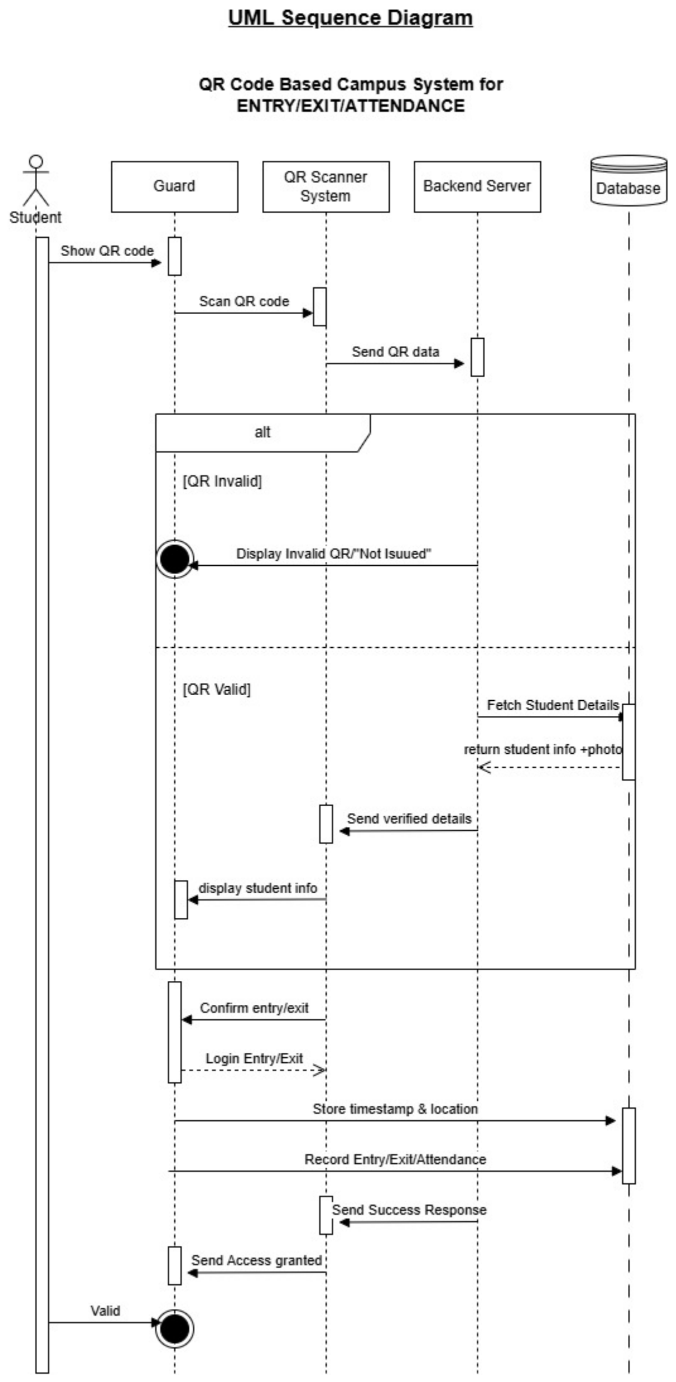

Fig. 4. Sequence Diagram for QR-Based Campus Entry/Exit and Attendance System

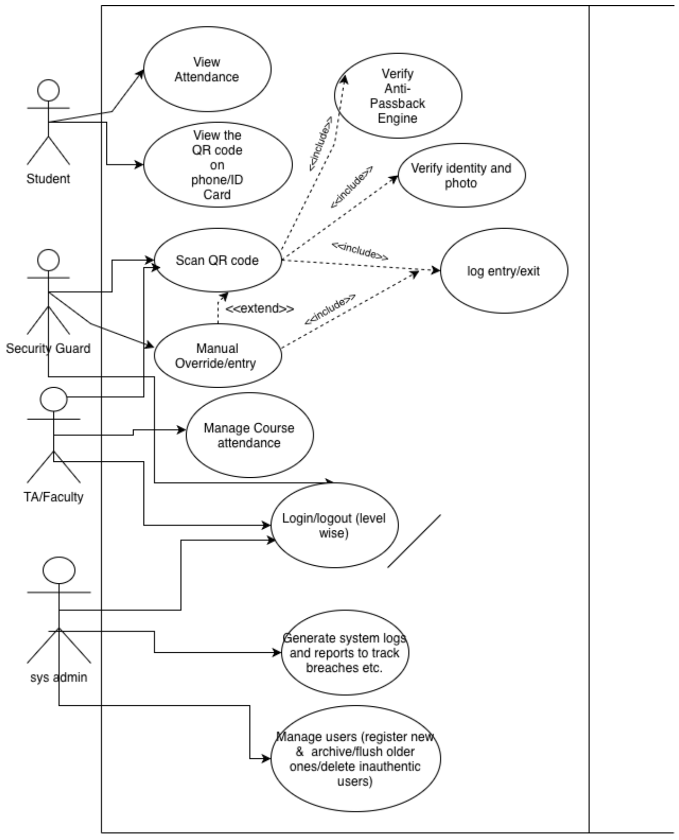
View Attendance
View the QR code on phone/ID Card
Verify Anti-Passback Engine
Verify identity and photo
Scan QR code
log entry/exit
Manual Override/entry
Manage Course attendance
Login/logout (level wise)
Generate system logs and reports to track breaches etc.
Manage users (register new & archive/flush older ones/delete inauthentic users)
<<include>>
<<include>>
<<include>>
<<include>>
<<extend>>
Student
Security Guard
TA/Faculty
sys admin

Fig. 5. Use Case Diagram illustrating actor interactions and system boundaries

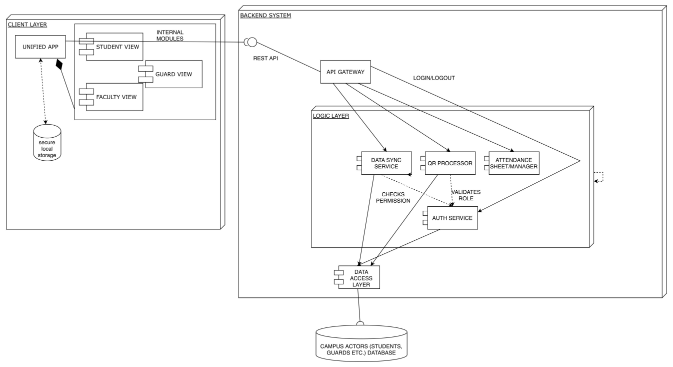

Fig. 6. Component Diagram showing the Client, Backend, and Data Access layers

E. Class UML Diagram

The Class diagram in Fig. 7 illustrates the core classes such as User, Student, Guard, Admin, Course, Attendance, EntryExitLog, and QRCodeService, along with their attributes, methods, and relationships.

F. Composite UML Diagram

The Composite diagram in Fig. 8 illustrates the internal structure of the main system components and the interaction between their internal parts.

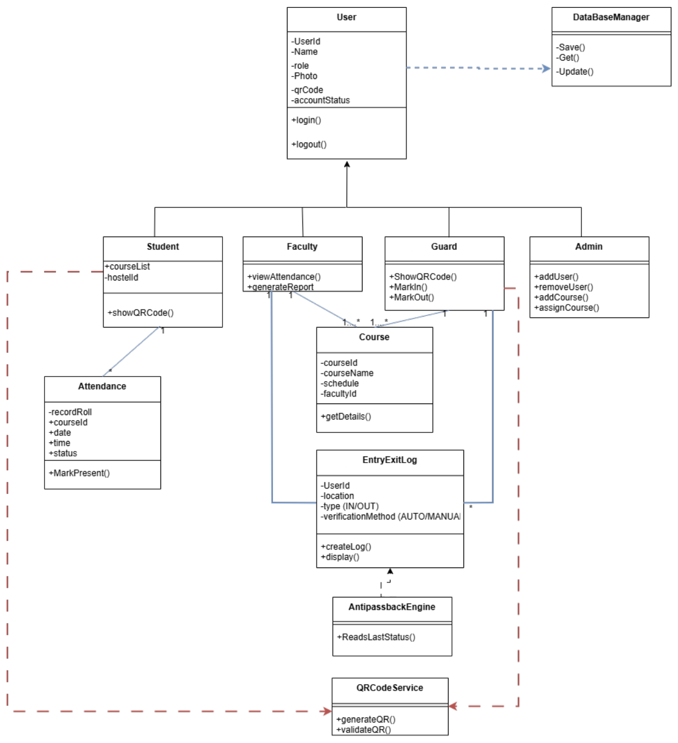

Fig. 7. Class UML Diagram

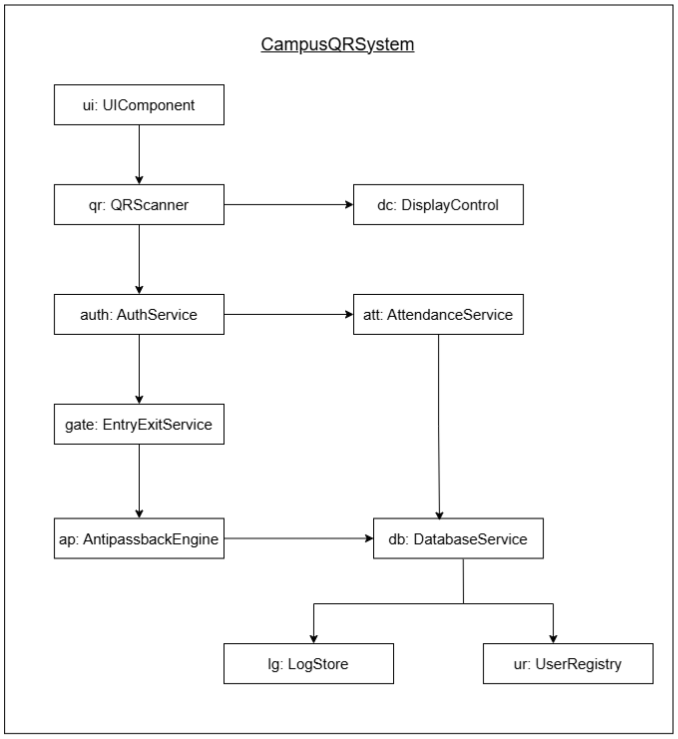

Fig. 8. Composite UML Diagram

G. Timing UML Diagram

The Timing diagram in Fig. 9 represents the temporal behavior of the QR validation process. It illustrates the sequence of events over time, including QR scan initiation, server-side validation, database lookup, identity verification, and response generation.

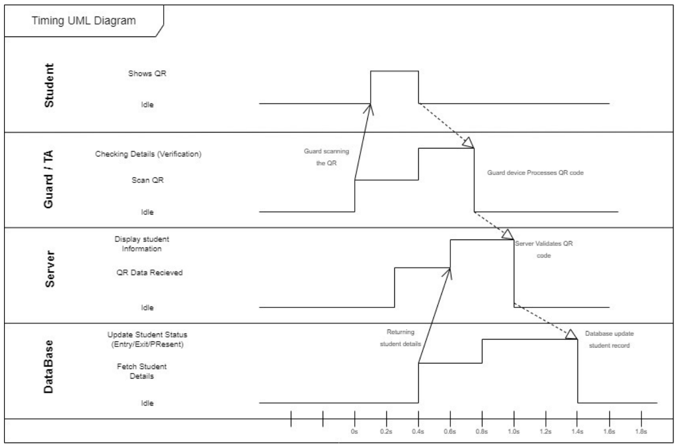

Fig. 9. Timing UML Diagram

H. Deployment UML Diagram

The Deployment diagram in Fig. 10 illustrates the physical deployment architecture of the proposed QR-based campus system. It shows the interaction between client devices (student and guard smartphones), the application server, and the backend database.

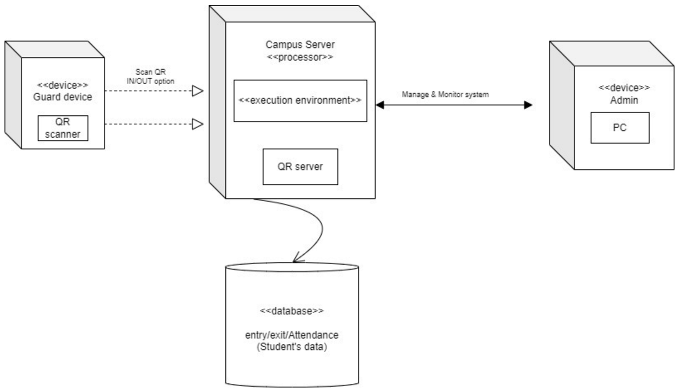

Fig. 10. Timing UML Diagram