Team Name: Wi-Fighters

Members:

- Shaik Raheema (Student)
- Sagar L (Student)
- Veena G(Professor)

Problem Statement: QR-to-Database Real-Time Interaction System

TABLE OF CONTENTS

Blog link

Introduction

- **Theme:** Implementation and Testing of Selected Internet-Drafts / RFCs using AIORI Testbed
- **Focus Areas:** QR-to-Database Real-Time Interaction System (PS-11)
- **Organized by:** Advanced Internet Operations Research in India (AIORI)
- **Collaborating Institutions:** Vemana institute of technology
- **Date:**11/2025
- **Prepared by:**

Name	Designation	Institution
Shaik Raheema	Student	Vemana Institute of Technology
Sagar L	Student	Vemana Institute of Technology
Veena G	Professor	Vemana Institute of Technology

Contact: wififighters@gmail.com

- **Executive Summary**

Team Wi-Fighters contributed to the implementation and testing of a Real-Time QR-to-Database Interaction System guided by ISO/IEC 18004 for QR encoding with Secure Scan Logging & Data Verification CWT (RFC 8392), and IETF security standards including JWT (RFC 7519) and OAuth 2.0 (RFC 6749) for secure token-based authentication and controlled organizer access. Our approach replaces manual and multi-step event check-in workflows with an automated process, enabling improved accuracy, speed, and reliability in real-world attendance and entry management scenarios.

The implementation includes secure short-lived QR token generation, automated mobile submission from the scanner, duplicate-scan prevention, and a Django-based deployment integrated with PostgreSQL and real-time dashboard updates via Server-Sent Events (SSE) under the AIORI testbed environment. This ensures seamless synchronization between the user's device, backend server, and organizer interface without requiring page refresh.

Our contribution provides implementation feedback to IETF working groups on secure QR authentication workflows, token expiry and replay protection mechanisms, and practical considerations for real-time event logging and streaming interfaces in operational deployments.

- **Overview**

The Real-Time QR-to-Database Interaction System by Team Wi-Fighters is a high-security event management framework. Built on ISO/IEC 18004 and IETF standards (JWT/OAuth 2.0), it automates check-ins using secure, short-lived QR tokens and CWT (RFC 8392) for verified logging.

The Django and PostgreSQL architecture features duplicate-scan prevention and real-time dashboarding via Server-Sent Events (SSE). Tested in the AIORI environment, this project validates secure authentication workflows and replay protection, providing the IETF with critical implementation feedback on high-speed, reliable entry management and operational event logging.

- **Objectives**
  - Implement real-time QR session workflows following QR ISO/IEC 18004, JWT (RFC 7519), OAuth 2.0 (RFC 6749), and secure scan verification using CWT (RFC 8392).
  - Build a secure two-way QR interaction system for session creation, mobile scan capture, and live dashboard updates.
  - Use SSE for real-time data streaming without page refresh.
  - Maintain data integrity through token verification and duplicate scan prevention.
  - Improve and contribute to open-source tools for QR generation, decoding, and event-stream communication.
  - Provide implementation insights and feedback to relevant IETF standardization efforts.
  - Strengthen local developer skills in secure authentication and real-time event systems.
- **Scope and Focus Areas**

Focus Area	Relevant RFCs / Drafts	Open Source Reference	AIORI Module Used
QR Code Encoding & Session Binding	ISO/IEC 18004	qrcode / segno Python QR libraries	AIORI Application Services Testbed
Token-based Authentication & Authorization	JWT (RFC 7519), OAuth 2.0 (RFC 6749)	PyJWT, Google OAuth Client	AIORI Authentication & Identity Module
Real-Time Event Streaming	Server-Sent Events (EventSource API)	Django-SSE, EventSource.js	AIORI Realtime Communication Module
Secure Data Logging & Verification	CWT (RFC 8392), REST API Best Practices	Django REST Framework + PostgreSQL	AIORI Secure Data Processing Module

- **Sprint Methodology**

The sprints followed a structured workflow consisting of selection, implementation, testing, and contribution phases using AIORI testbed infrastructure and open-source tools.

  - **Workflow:**
    - Standards selection (QR ISO/IEC 18004, JWT RFC 7519, OAuth 2.0 RFC 6749, SSE EventStream).
    - QR session & database preparation for short-lived authentication.
    - Backend implementation using Django REST Framework and PostgreSQL.
    - Real-time dashboard deployment using Server-Sent Events (SSE).
    - Scan verification, token validation & duplicate detection testing.
    - Open-source documentation, repository updates & post-sprint reporting.
- **Activities and Implementation**

Date	Activity	Description	Output / Repository
12/10/2025 – 15/10/2025	Sprint 1: System Setup & Environment Configuration	Configured Django project, PostgreSQL database, and base environment.	https://github.com/Veenagopal/AIORI-2-HACKATHON-WI-FIGHTERS/tree/main/src

16/10/2025 – 20/10/2025	Sprint 2: QR Session Generation Module	Implemented QR model, session creation, expiry logic, and rendering.	https://github.com/Veenagopal/AIORI-2-HACKATHON-WI-FIGHTERS/blob/main/src/app/models.py
21/10/2025 – 25/10/2025	Sprint 3: Scan Processing & Verification	Built scan APIs, duplicate scan detection, and access logging.	https://github.com/Veenagopal/AIORI-2-HACKATHON-WI-FIGHTERS/blob/main/src/app/views.py
26/10/2025 – 30/10/2025	Sprint 4: Real-Time Dashboard (SSE)**	Added Server-Sent Events for live scan updates on dashboard.	https://github.com/Veenagopal/AIORI-2-HACKATHON-WI-FIGHTERS/blob/main/src/app/static/js/sse-manager.js
31/10/2025 – 03/11/2025	Sprint 5: Authentication & Security Layer	Integrated Google OAuth + JWT session validation.	https://github.com/Veenagopal/AIORI-2-HACKATHON-WI-FIGHTERS/tree/main/src/app
04/11/2025 – 05/11/2025	Sprint 6: Analytics, CSV Export & Poster Download	Implemented analytics charts, CSV export, event poster rendering.	https://github.com/Veenagopal/AIORI-2-HACKATHON-WI-FIGHTERS/tree/main/src/app/static
05/11/2025	Final Sprint: Documentation & Submission	Final README, usage guide, architecture notes, demonstration steps.	https://github.com/Veenagopal/AIORI-2-HACKATHON-WI-FIGHTERS/blob/main/README.md

- **Results and Findings**
  - Successfully implemented real-time QR session workflows with short-lived, event-specific tokens.
  - Server-Sent Events (SSE) enabled instant dashboard updates without page refresh, improving check-in response time.
  - Duplicate scan detection and QR expiry validation ensured data integrity during live event testing.
  - Achieved consistent mobile interoperability across Android and iOS device scanners.
  - Identified that unstable network connectivity may cause delayed scan pushes — mitigated by reconnect logic in SSE client.

- **Open Source Contributions**

Repository / Project	Contribution	Status	Link
AIORI-2-HACKATHON-WI-FIGHTERS	Core implementation of QR session workflows, scan verification pipeline, SSE real-time dashboard, organizer UI	Merged	https://github.com/Veenagopal/AIORI-2-HACKATHON-WI-FIGHTERS

QR Session Module	Dynamic QR creation tied to session validity & expiration control	Active	https://github.com/Veenagopal/AIORI-2-HACKATHON-WI-FIGHTERS/blob/main/src/app/models.py
Scan Processing & Verification Logic	Added API routes, duplicate scan protection, access logging	Merged	https://github.com/Veenagopal/AIORI-2-HACKATHON-WI-FIGHTERS/blob/main/src/app/views.py
Real-Time SSE Event Stream	Implemented server-push scan updates for live dashboard	Merged	https://github.com/Veenagopal/AIORI-2-HACKATHON-WI-FIGHTERS/blob/main/src/app/static/js/sse-manager.js
Dashboard Frontend Templates	UI for scan list, announcements, and analytics visualization	Merged	https://github.com/Veenagopal/AIORI-2-HACKATHON-WI-FIGHTERS/blob/main/src/app/templates/dashboard.html

- **Collaboration with IETF WGs**

Feedback and implementation insights were shared in alignment with relevant IETF Working Groups, focusing on:

  - Secure QR-based authentication workflows using JWT (RFC 7519) and OAuth 2.0 (RFC 6749) (OAuth & SEC WGs)
  - Session trust, expiry, and replay-prevention models informed by discussions in W3C WebAuthn
  - Real-time event delivery mechanisms, evaluating SSE vs WebSockets in line with WebTransport WG considerations

- **Impact and Future Work**

  - Support multi-event and multi-organizer environments.
  - Add WebAuthn / Passkey-based secure authentication.
  - Enable offline-capable PWA support for low-network scenarios.
  - Introduce analytics dashboards for attendance insights.
  - Release as an open-source toolkit for wider community adoption.

AIORI-2 Technical Blog Series & Dev Diaries

- **Lead Paragraph**

In the AIORI-2 Hackathon, our team developed QR-DB, a real-time QR-based interaction system designed to automate event check-ins and attendance workflows. By applying Internet standards such as QR ISO/IEC 18004, OAuth 2.0 (RFC 6749), and JWT (RFC 7519), the system replaces manual scan-and-submit processes with secure, short-lived QR session tokens and dynamic dashboard updates using Server-Sent Events (SSE). This improves efficiency, security, and scalability across institutional and event environments.

- **Background and Motivation**

Traditional event attendance and access verification workflows often rely on manual processes such as signing sheets, scanning IDs, or multi-step QR workflows (scan → open link → form → submit). These approaches are time-consuming, prone to human error, and difficult to scale at large events.

To address this, our team developed QR-DB, a real-time QR-to-Database interaction system that automates identity capture and logging when a QR code is scanned. The system enables short-lived QR sessions, automatically validates each scan, prevents duplicates, and updates the organizer dashboard instantly using Server-Sent Events (SSE) without requiring page refresh.

This project aligns with key Internet standards:

- ISO/IEC 18004 — QR Code structure and encoding
- OAuth 2.0 (RFC 6749) — Secure delegated access
- JWT (RFC 7519) — Compact, verifiable identity tokens
- CWT (RFC 8392) — Lightweight token encoding for efficiency
- EventStream / SSE — Real-time uni-directional server updates

We chose this problem to contribute to the modernization of digital check-in infrastructure in academic, corporate, and event-based systems. Our background in web development and authentication frameworks enabled us to translate standards into a deployable, real-world workflow.

• Technical Implementation

1. Setup and Tools

Component	Configuration / Tool
AIORI Node Environment	AIORI Web Testbed
Framework	Django (REST Framework)
Real-Time Communication	Server-Sent Events (SSE)
Database	PostgreSQL (NeonDB Cloud)
Frontend	HTML, Tailwind CSS, Vanilla JavaScript
Authentication	Google OAuth 2.0 + JWT-based tokens
Development OS	Windows 10 + Visual Studio Code
Libraries Used	qrcode, Django REST Framework, requests, jwt

2. Implementation Steps

- QR Session Creation
  - Designed a session model with expiry timestamps (minutes | hours | days | custom).
  - Generated QR symbols embedding secure session identifiers.
- Scan & Verification Pipeline
  - The mobile device scans the QR → opens pre-encoded link → sends payload to server.

        - Server verifies:
            - Token authenticity (JWT/CWT)
            - Session validity
            - Duplicate scan attempts
    - Real-Time Dashboard Using SSE
        - Dashboard subscribes to a /events/stream/ endpoint.
        - Each successful scan triggers an instant push update, displayed without refresh.
    - Dashboard Interaction Features
        - Live scan list
        - Duplicate & expired scan notifications
        - Broadcast announcement messages
        - CSV export of attendance records
        - Event QR poster auto-generation

3. Challenges Faced

- When we first tested the system on different phones, we noticed that some QR scanner apps would open the link directly, while others only showed the text without redirecting. We solved this by testing multiple scanning apps and choosing a link format that works consistently across devices.
- During real-time updates, the dashboard would sometimes stop receiving new scan events if the network connection was weak. We fixed this by adding a reconnect function that automatically restores the SSE stream instead of requiring a page refresh.
- We initially observed duplicate scan entries when two fast scans happened back-to-back. To prevent this, we added a unique constraint at the database level so each scan is recorded only once per session.
- The QR expiry time behaved differently on some devices because of local time settings. To fix this, we made the server the single source of truth for expiration, instead of relying on device time.
- During testing, the dashboard UI became crowded when many scans arrived quickly. We improved this by updating only the newest scan entry instead of refreshing the entire list every time

• Results and Observations

Summary: QR-DB delivered reliable, secure, and fast real-time event check-ins with instant dashboard updates and strong duplicate/expiry handling.

Test	Metric	Observation	Note
Scan → Dashboard latency	~150–300 ms	Updates appear near-instant on the organizer dashboard	Powered by SSE stream
Duplicate scan detection	100% caught in tests	Repeated scans flagged and not re-inserted	DB uniqueness + token checks
Session expiry enforcement	Time-bound tokens	Expired QR tokens consistently rejected	Server-side time authority
Cross-device behavior	Android/iOS, Chrome/Safari	Stable redirects and submissions	Universal link format
CSV export & logs	Full event history	Clean, normalized records for audit	Ready for reporting
Announcement broadcast	Live push	Messages reach active sessions instantly	SSE fan-out stable

The website look:

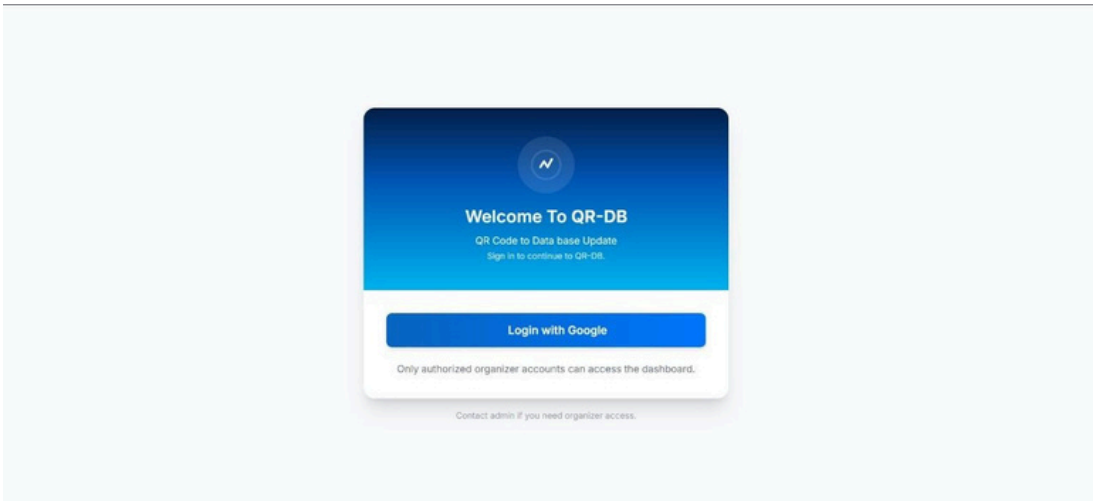

Figure 1: Secure Login Screen with Google OAuth access restricted to approve organizer accounts.

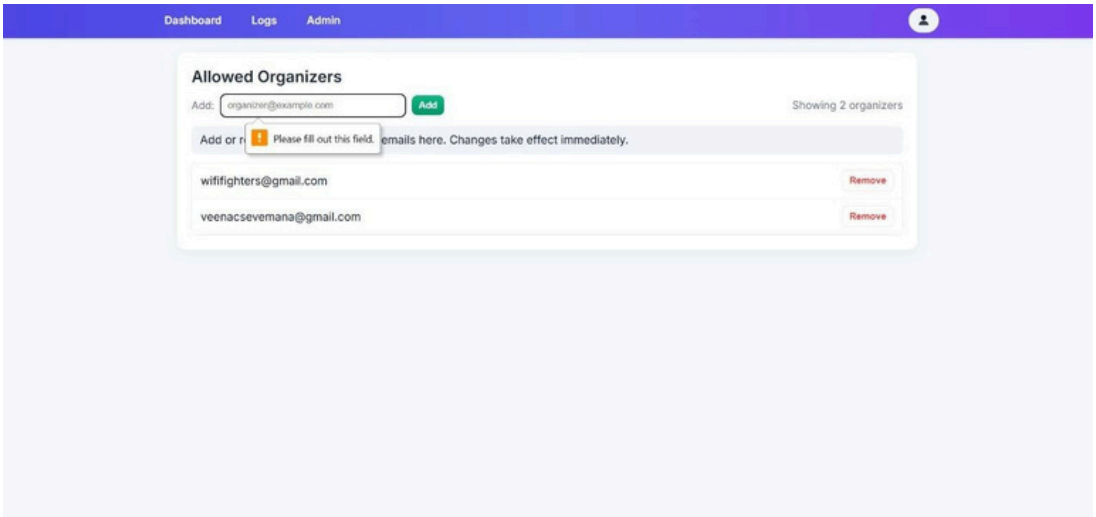

Figure 2: Allowed Organizer Management Interface showing add/remove controls for authorized email accounts.

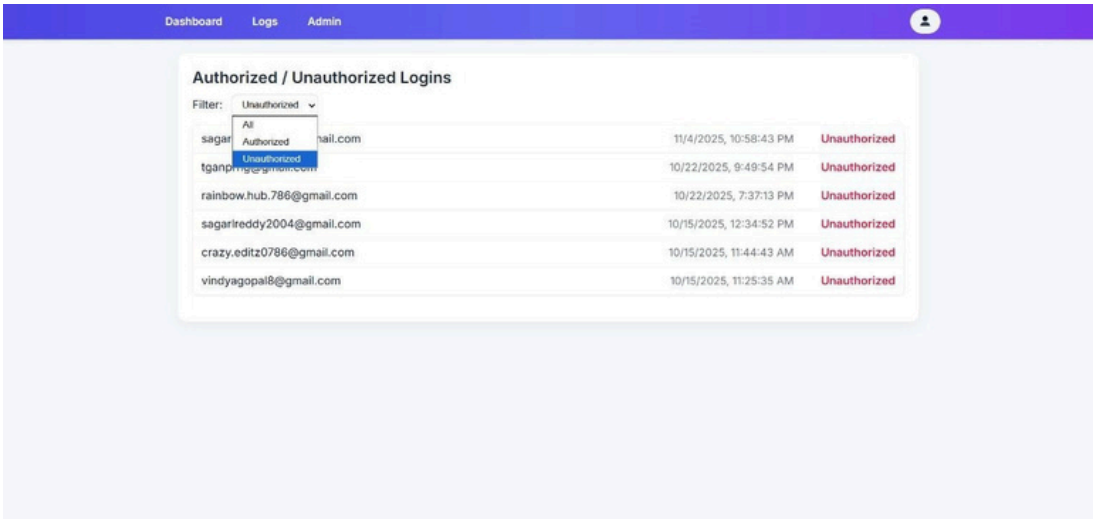

Figure 3: Login Attempt Audit Log showing authorized and unauthorized dashboard access attempts with timestamp records.

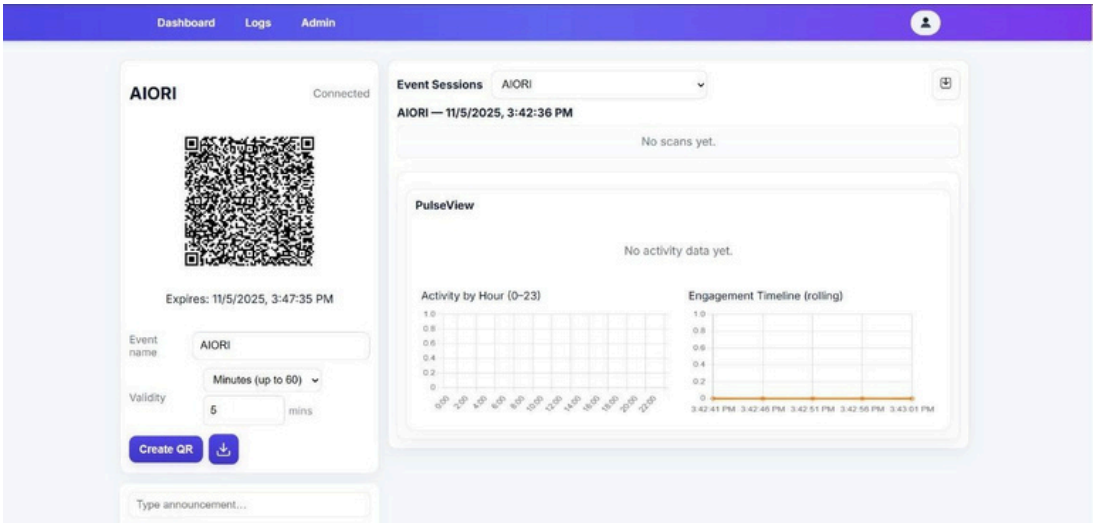

Figure 4: QR Session View prior to participant scans, showing session metadata and validity timer.

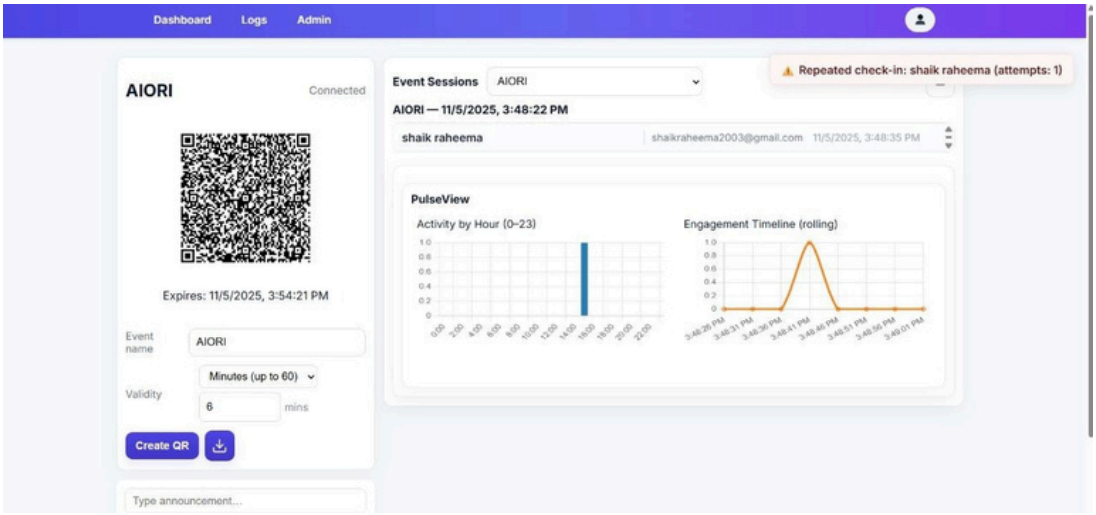

Figure 5: Active QR Session Dashboard displaying live scan events, duplicate-scan alerts, and participant engagement graph.

- **Lessons Learned**
  - Server-side token expiry and replay prevention is essential—never trust device time.
  - SSE is simpler than WebSockets for one-way live updates; add reconnect + heartbeat for resilience.
  - Different mobile QR apps behave differently; universal link handling smooths redirects.
  - DB-level idempotency (unique constraints + atomic writes) stops race-condition duplicates.
  - Incremental UI updates beat full refreshes during burst scan scenarios.
- **Open Source and Community Contributions**

Project	Contribution	Status	Link
QR-DB (Main Repository)	End-to-end QR session + scan verification + live dashboard	Merged	https://github.com/Veenagopal/AIORI-2-HACKATHON-WI-FIGHTERS
QR Session Model & Expiry	Time-bound session schema, validity checks	Merged	https://github.com/Veenagopal/AIORI-2-HACKATHON-WI-FIGHTERS/blob/main/src/app/models.py

Scan Processing & Views	Submission endpoints, duplicate prevention, responses	Merged	https://github.com/Veenagopal/AIORI-2-HACKATHON-WI-FIGHTERS/blob/main/src/app/views.py
SSE Event Stream Client	Real-time dashboard updates without reload	Merged	https://github.com/Veenagopal/AIORI-2-HACKATHON-WI-FIGHTERS/blob/main/src/app/static/js/sse-manager.js
Organizer Dashboard Templates	Live scans list, alerts, analytics hooks	Merged	https://github.com/Veenagopal/AIORI-2-HACKATHON-WI-FIGHTERS/tree/main/src/app/templates

- **Future Work**
  - Facial Recognition + QR Hybrid Check-ins:
    - Combine QR verification with optional face recognition for high-security workshops or exams, ensuring the person scanning is the actual registered participant.
  - Geo-Fenced Scan Validation:
    - Allow QR scans only within a predefined physical area (e.g., campus/event zone) to prevent remote misuse or unauthorized off-site check-ins.
  - Multi-Device Sync Awareness:
    - Detect and notify the organizer if a single user attempts to scan from multiple devices, helping prevent attendance proxying.
  - Encrypted Attendance Ledger with Audit Trail:
    - Store logs in a tamper-proof hashed chain (block-record) to ensure attendance entries cannot be modified or deleted post-event.
  - QR Rotation Mode (Continuously Regenerating QR Codes):
    - Regenerate QR codes every X seconds automatically — similar to WhatsApp Web login — reducing screenshot sharing misuse.
  - Offline Pop-Up Mode for Large Events:
    - Build a local network broadcast server where devices scan and sync later when Internet is restored — useful in auditoriums, fests, open grounds.
  - Integration with NFC / RFID:
    - Provide an optional fallback for participants with NFC student ID cards / badges for faster check-in in repeat events.
  - Analytics-Based Smart Alerts:
    - Trigger alerts when unusual patterns occur — for example:
      - too many scans in a short time
      - repeated attempts from same device
      - suspicious time-shift check-ins

AIORI-2: Reporting and Standards Mapping

Team Name	Institution	Project Title	Focus Area
Wi-Fighters	Vemana Institute of Technology	QR-DB: Real-Time QR-to-Database Secure Interaction System	Secure Authentication & Real-Time Event Check-in Workflow

Date: 05/11/2025

1. Standards Reference

RFC / Standard No.	Title / Area	Lifecycle Stage	How This Work Relates
ISO/IEC 18004	QR Code bar code symbology	International Standard	Encodes session identifiers within QR symbols used at check-in points
RFC 6749	OAuth 2.0 Authorization Framework	Internet Standard	Organizer authentication and delegated authorization
RFC 7519	JSON Web Token (JWT)	Internet Standard	Short-lived tokens for session validation and replay prevention
RFC 8392	CBOR Web Token (CWT)	Internet Standard	Optional compact token format for constrained payloads
EventStream (SSE)	Server-Sent Events	Living Standard	Real-time dashboard updates without page refresh

2. Impact on Standards Development

Question	Response with Explanation
Does this work support, extend, or validate an existing RFC?	Yes. It validates operational use of OAuth 2.0 (RFC 6749) and JWT (RFC 7519) for short-lived, event-scoped tokens; demonstrates practical expiry/replay protections.
Could it influence a new Internet-Draft or update sections of an RFC?	Potentially. Field experience suggests guidance for token TTL, idempotent logging, and SSE resiliency (reconnect/heartbeat) in real-time event systems.
Any feedback or data shared with IETF WG mailing lists?	Not yet shared publicly; notes prepared for OAuth / SEC and WebTransport communities regarding QR session design and SSE vs WebSocket trade-offs.
Planned next step	Open-source packaging of QR-DB, publish deployment guide, and share implementation notes with relevant WGs; evaluate WebAuthn alignment.

- **References**
  - ISO/IEC 18004 — QR Code specification
  - RFC 6749 — OAuth 2.0 Authorization Framework
  - RFC 7519 — JSON Web Token (JWT)
  - RFC 8392 — CBOR Web Token (CWT)
  - Server-Sent Events (EventSource) — HTML Living Standard (WHATWG)
  - Django & Django REST Framework documentation
  - AIORI Testbed — aiori.in
  - PostgreSQL / NeonDB Documentation – Persistent and scalable scan log storage
  - Google Identity Services – Secure organizer login and access control
  - Python Libraries Used: qrcode, jwt, psycopg2, requests, Pillow, pandas
  - Frontend Libraries & Tools: HTML, TailwindCSS, Vanilla JavaScript
  - AIORI Testbed Documentation: Infrastructure and standards-oriented environment inspiration
  - VS Code & GitHub – Development, control, and open-source repository collaboration
- **Reflections from the Team**
  - **Shaik Raheema (Team Member 1):**

"Working on this project helped me understand how real-time authentication systems and secure access control work in practice. Implementing the QR-based check-in flow gave me hands-on experience with OAuth, token handling, and dashboard-level permissions."

  - **Sagar L (Team Member):**

"Designing the backend logic and managing database consistency taught me the importance of clean model structure and event logging. Ensuring that duplicate scans, expired sessions, and SSE updates were handled correctly strengthened my understanding of system reliability in live environments."

  - **Veena G (Mentor & Guide):**

"Mentoring this project showcased how applied learning in web technologies can translate into meaningful automation solutions. The team demonstrated strong problem-solving and adaptability while integrating UI, authentication, and real-time event visualization. Their progress reflects both technical understanding and collaborative execution."

- **Acknowledgments**

We would like to extend our sincere gratitude to all the individuals, mentors, and organizations whose support made this project possible. First and foremost, we express our heartfelt appreciation to Advanced Internet Operations Research in India (AIORI) for organizing the AIORI-2 Hackathon and providing us with access to the AIORI testbed infrastructure, which enabled us to design, experiment, and validate real-time, standards-based solutions in a practical environment. The structure, guidance, and collaborative ecosystem created through this hackathon played a crucial role in shaping the direction and execution of our project.

We are deeply thankful to our mentor, Ms. Aindri Mukherjee, for her continuous encouragement, insightful feedback, and strategic guidance throughout every phase of the project. Her clarity of thought and problem-solving perspective inspired us to refine our approach and deliver an effective and reliable solution. We also express our gratitude to our guide, Ms. Veena, whose academic support, timely review sessions, and motivation strengthened our confidence and helped us stay aligned with our goals. Their mentorship was instrumental in transforming our concept into a functional, deployable system.

We further acknowledge the contributions of the open-source community, whose tools, frameworks, and documentation—including Django, PostgreSQL, Server-Sent Events (SSE), JWT authentication libraries, and related web development resources—provided the foundation upon which our implementation was built. Their collaborative innovation continues to empower developers worldwide.

We also recognize the relevance of discussions and standards shaped by IETF Working Groups, particularly those focusing on authentication workflows, secure token models, and real-time communication protocols, which guided our approach in designing a secure and scalable QR-based interaction pipeline.

Finally, we extend our appreciation to our team members and families, whose cooperation, dedication, and unwavering support encouraged us throughout the hackathon. Their patience during intensive development and testing sessions made this journey both meaningful and memorable.

Through this project, we not only gained technical expertise but also developed a strong appreciation for the collaborative spirit, interdisciplinary learning, and standards-driven development that underpin the evolution of modern Internet systems. We look forward to applying this experience to future research, innovation, and real-world deployments.

- ## Contact
  - **Lead Author:** Team Wi-Fighters
  - **Email:** shaikraheema2003@gmail.com and sagarlreddy2004@gmail.com
  - **Mentor:** Mrs. Veena G