DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

Final Project

Student Name: Manjot Singh
UID: 23BCS12549
Branch: CSE
Section/Group: KRG-2B
Semester: 5th
Subject Code: 23CSH-304
Subject Name: PBJL

1. Aim:

To develop a scalable, secure, and real-time attendance tracking web application that allows administrators to generate time-sensitive QR codes for specific class sessions and enables authenticated students to mark their presence instantly by scanning the code

2. Objective:

- Implement a robust, atomic database strategy in MongoDB to prevent data corruption and concurrency issues during simultaneous student check-ins.
- Establish a clean, decoupled MVC architecture using Spring Boot (Controller/Service/Repository) and React (View).
- Utilize token-based authentication principles (simulated `X-User-Id` header) and robust network handling to ensure secure data transfer.
- Provide a dynamic, real-time administration dashboard for session creation, monitoring, and attendance record retrieval.

3. Technology Stack

Component	Technology	Role
Backend Framework	Java (Spring Boot)	Provides the RESTful API endpoints, handles business logic, and manages security integration.
Database	MongoDB	Stores user profiles and attendance sessions, used for high-performance atomic array updates.
Frontend Framework	React.js	Presents the user interface (Admin Dashboard, Student Profile) and manages client-side state.
Styling	Tailwind CSS	Provides rapid, utility-first styling for a responsive and modern user interface.
QR Generation	qrcodegen (JS Library)	Client-side utility for reliable, offline generation of session QR codes.
QR Scanning	html5-qrcode (JS Library)	Client-side utility for activating the student's camera to decode the attendance token.

4. Methodology

The project follows a standard layered architecture with critical focus paid to data integrity and concurrency handling:

1. Data Modeling and Persistence (MongoDB)

- **User Model:** Contains nested data, specifically the **attendanceRecords** array, where each element stores the session ID, presence status, and join time. This complex structure necessitated advanced database handling.
- **Session Model:** Stores the unique token (UUID), section, name, and, critically, the **expiresAt** timestamp to enforce time limits.

2. Atomic Update Strategy (Concurrency Solution)

The most significant technical challenge was preventing duplicate records or lost updates when multiple students check in simultaneously or when the session is initialized for many students.

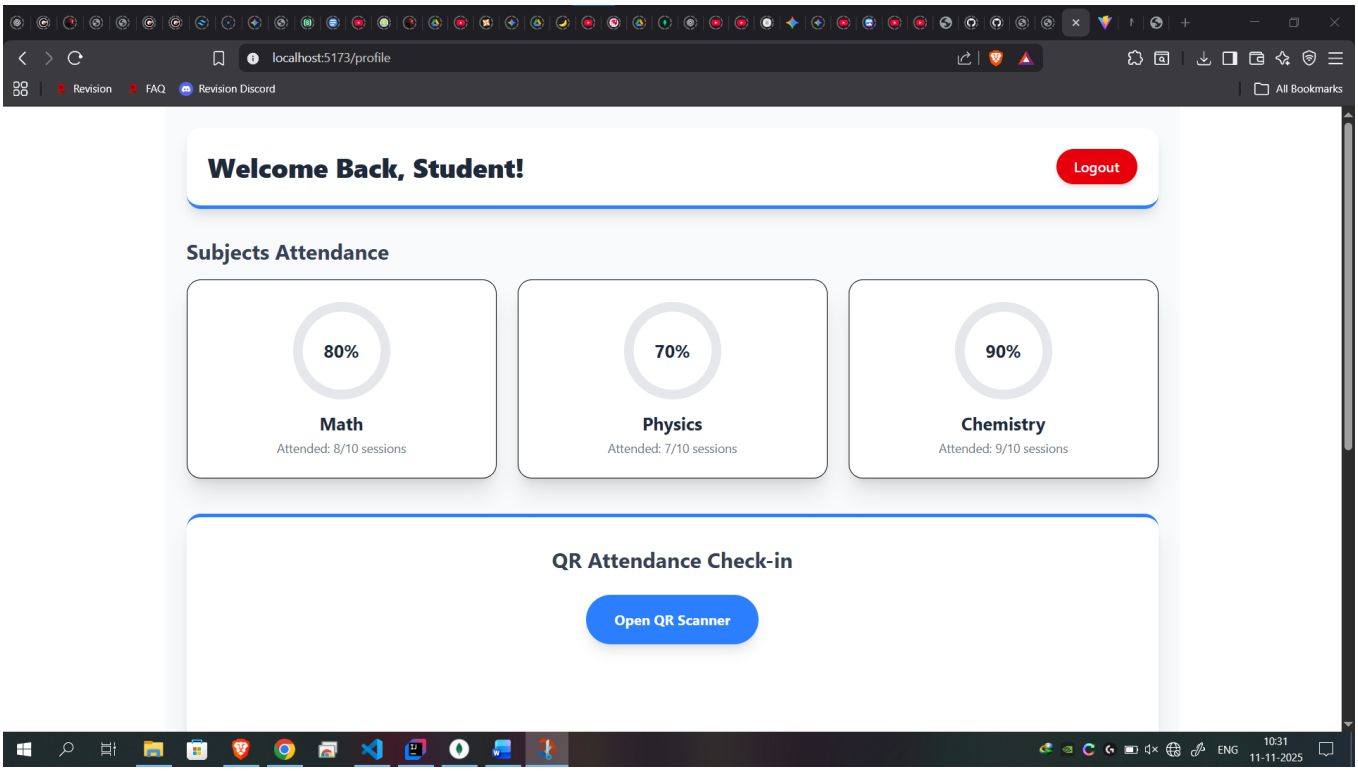
- **Token Generation Fix:** The **AttendanceService.generateToken()** method uses a single, atomic MongoDB operation (mongoTemplate.updateMulti) with a negative query condition ($ne: token). This ensures that the absent record is only added to a student's array *if it does not already exist*, successfully solving the duplicate insertion problem.
- **Check-in Update:** The **AttendanceService.checkIn()** method uses the MongoDB positional operator ($) via mongoTemplate.updateFirst to update only the specific array element matching the token. This avoids loading and saving the entire large user document, guaranteeing speed and atomicity for the check-in transaction.

3. Frontend Reliability and Experience

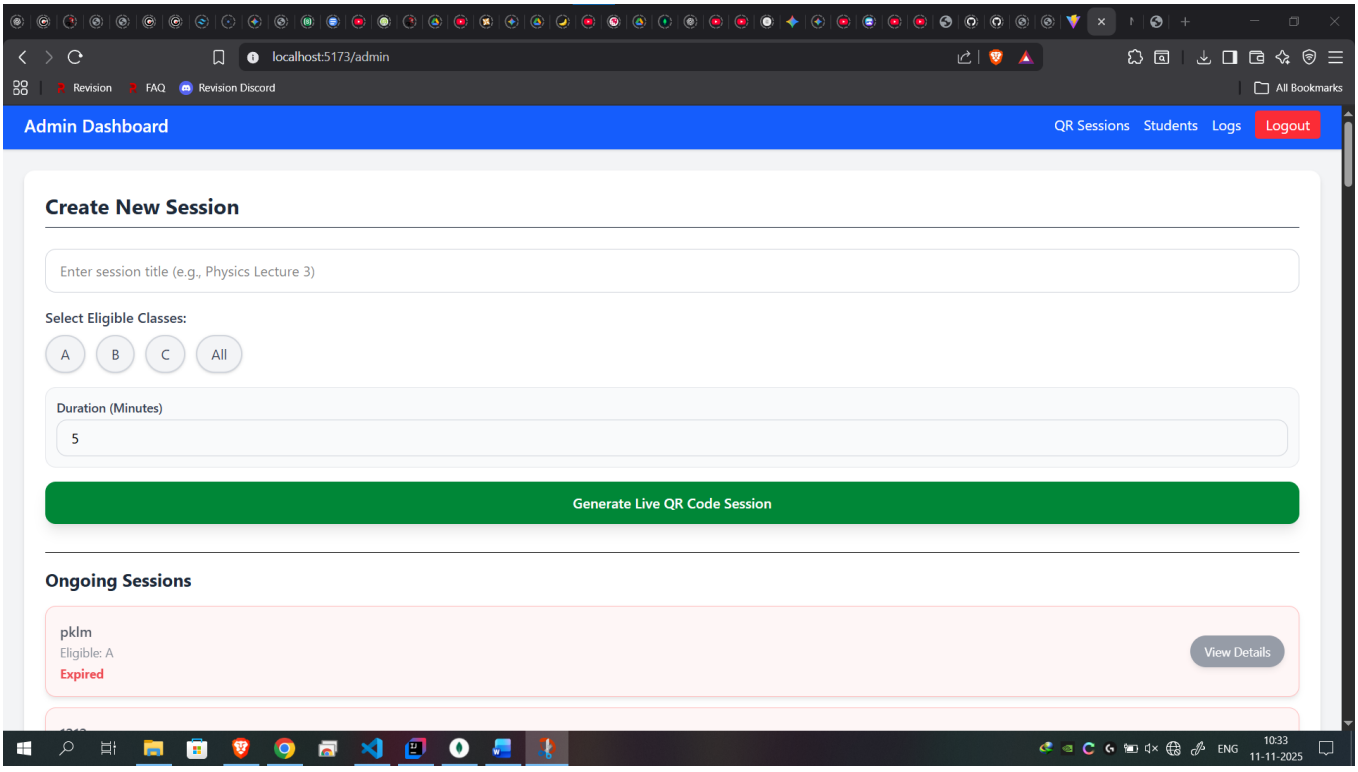
- **Anti-Concurrency Measures:** The **fetchWithRetry** utility was modified to perform **only one attempt** for POST requests (like token generation and check-in). This prevents the browser from automatically retrying a write operation on network failure, which was identified as a source of duplicate records.
- **Real-time Monitoring:** The SessionTab.jsx component implements a client-side useEffect hook to run a **countdown timer** every second, providing the administrator with a real-time view of remaining session time.
- **Dashboard Sorting:** Sessions are sorted dynamically by status, ensuring **Active Sessions** always appear above Expired Sessions for immediate oversight.

6. Output:

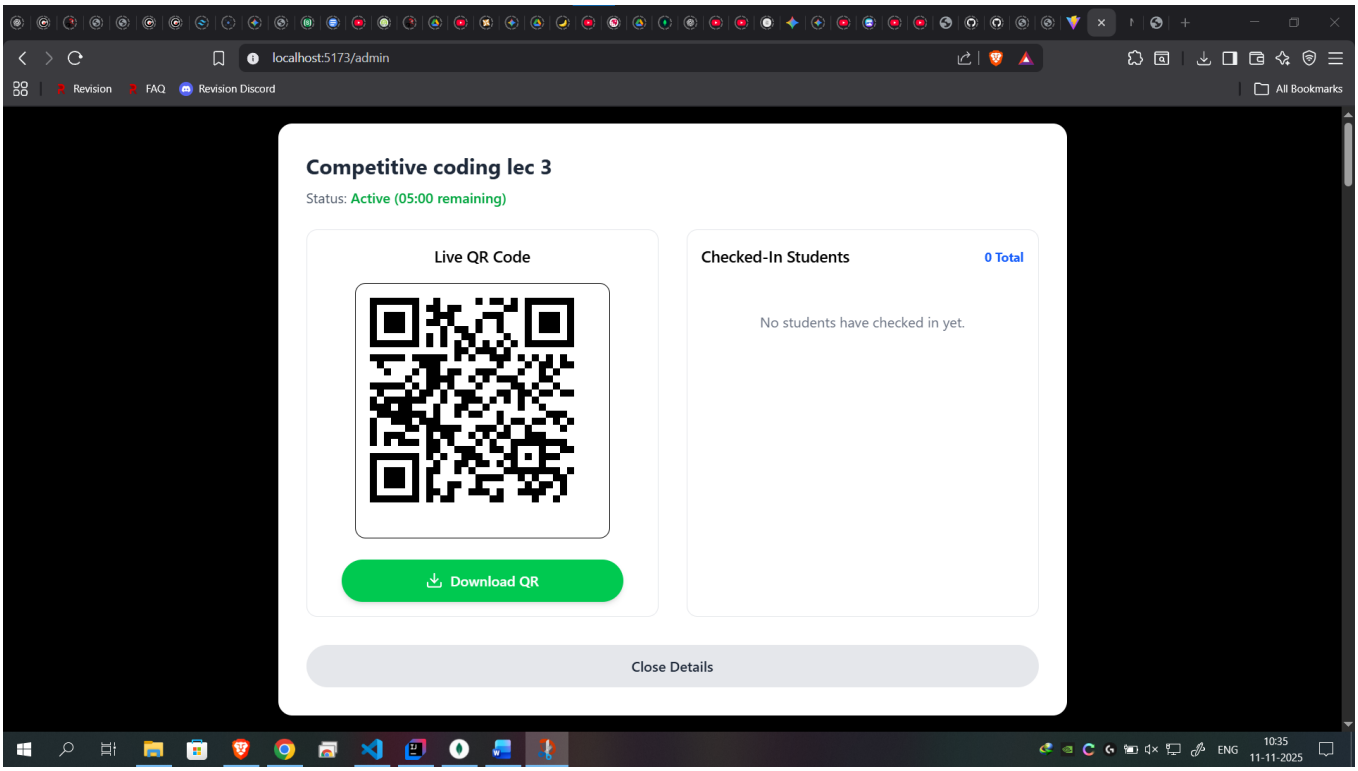
Student Interface



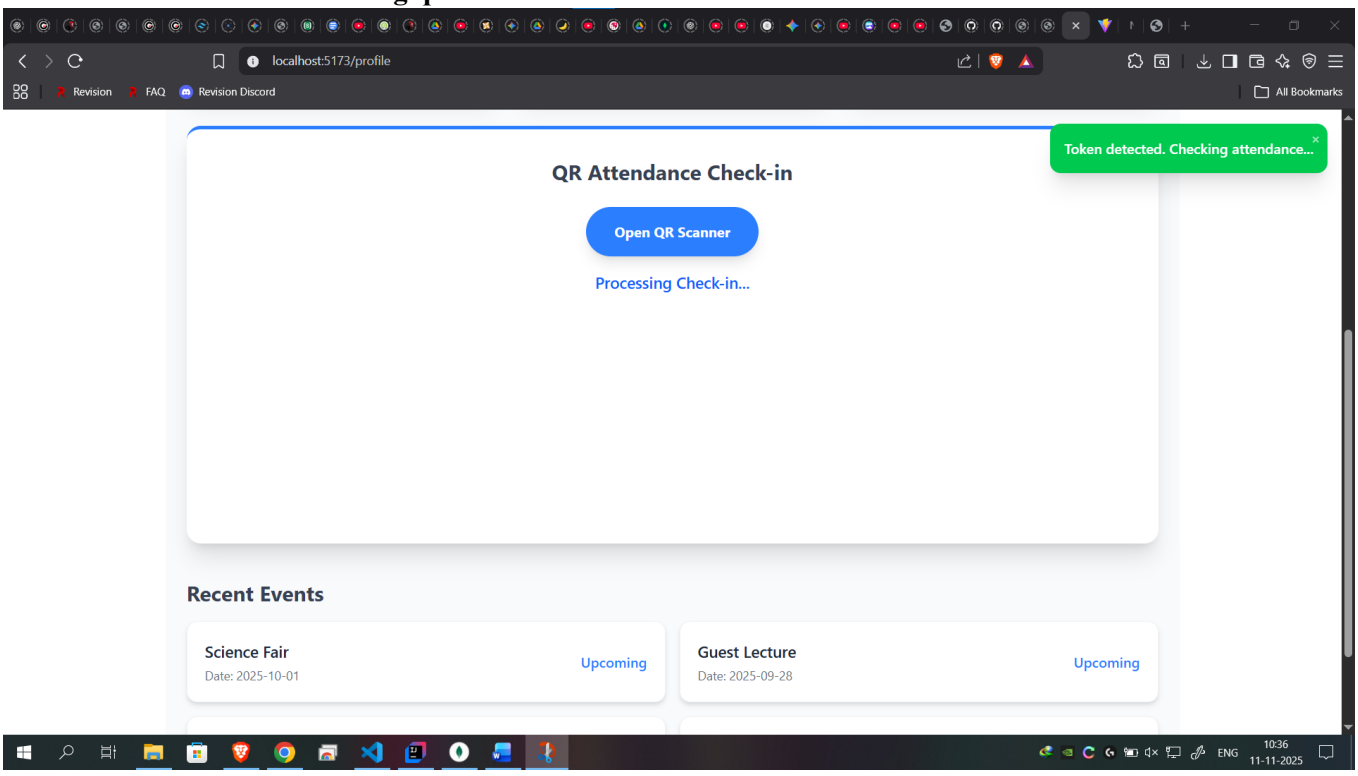
Admin Interface



Admin Creates New Session



Student Interface after scanning qr code



7. Conclusion and Future Scope

The project successfully delivered a fully functional, high-integrity attendance system. The core objective of implementing atomic updates for nested MongoDB arrays was achieved, creating a reliable foundation for scaling concurrent attendance operations.

Future Scope:

1. **Full JWT Authentication:** Replace the simulated X-User-Id header with secure, industry-standard **JSON Web Tokens (JWTs)**, integrating user validation and token management directly into the Spring Security filter chain.
2. **Role-Based Access Control (RBAC):** Implement Spring Security roles (ROLE_ADMIN, ROLE_STUDENT) to strictly enforce which users can access the AdminController versus the AttendanceController.
3. **Data Persistence for Student Profile:** Implement API endpoints to fetch a student's *actual* attendance history from the database instead of using frontend mock data, populating the Profile.jsx dashboard dynamically.