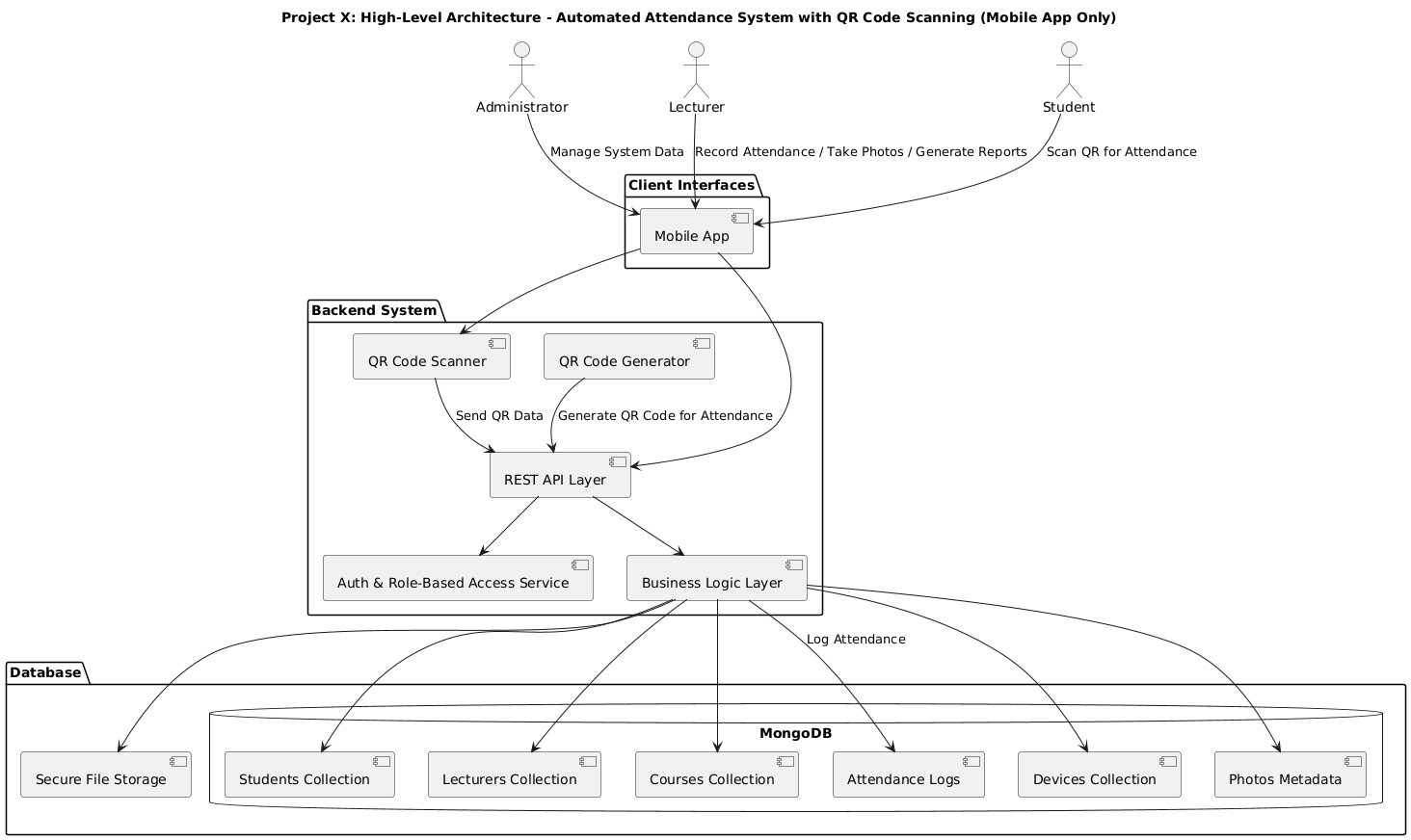
**High-Level Design Document**

**Project Name:** Attendance Tracking System  
**Version:** 1.0  
**Prepared Date:** May 26, 2025

**1. Project Overview**

A web-based Attendance Tracking System that enables authorized users to register students and instructors, manage attendance via barcode scanning, and generate attendance reports. The system emphasizes role-based access control, multi-factor authentication, and secure cloud deployment.

**2. Architecture Diagram**



**3. System Architecture**

* **Frontend:**
  + SPA using Expo
  + Instructor interface for scanning barcodes
  + Student portal to view attendance
  + Admin dashboard for managing users and reports
* **Backend:**
  + Spring Boot REST API
  + Handles RBAC, MFA, and CRUD operations
  + Attendance processing logic
  + Report generation
* **Database:**
  + Cloud DB (MySQL or MongoDB Atlas)
  + Tables/collections for Users, Devices, Attendance Logs, Courses, and Audit Trails

**4. Modules and Components**

| **Module** | **Description** | **Technology** |
| --- | --- | --- |
| User Management | Register students/instructors, assign roles, unregister | Spring Boot + MySQL/MongoDB |
| Attendance Tracker | Scan barcodes, drop inactive students, attendance view | React + API |
| Reports | Generate attendance reports | Spring Boot |
| Security | MFA, RBAC, device tracking, encryption | Spring Security |
| Infrastructure | Cloud deployment, secure access, scalability | AWS/Azure/GCP |

**5. Data Flow Description**

1. **Registration:**  
   Authorized user registers students/instructors → system stores data + ID photo in DB
2. **Attendance:**  
   Instructor scans barcode → system validates → records timestamped attendance
3. **Viewing:**  
   Students login → view their attendance per course
4. **Reporting:**  
   Admin generates attendance summary reports

**6. API Design**

| **Endpoint** | **Method** | **Description** | **Auth Role** |
| --- | --- | --- | --- |
| /api/register/student | POST | Register a student | Authorized User |
| /api/register/instructor | POST | Register an instructor | Authorized User |
| /api/attendance/scan | POST | Submit scanned barcode | Instructor |
| /api/attendance/student/{id} | GET | View student attendance | Student |
| /api/reports/attendance | GET | Generate attendance report | Admin |
| /api/auth/login | POST | Login with MFA | All |
|  |  |  |  |

**7. Database Design**

* users (id, name, role, email, password\_hash, mfa\_enabled)
* students (id, student\_id, name, photo\_url, year\_level, program, faculty)
* instructors (id, name, registered\_device\_id)
* attendance\_logs (id, student\_id, instructor\_id, timestamp, course)
* devices (device\_id, instructor\_id, status)

**8. Security**

* **RBAC:** Enforced for Admin, Instructor, and Student roles
* **MFA:** Required for instructor logins
* **Encryption:** Admin-controlled for sensitive data
* **Transport Layer Security:** HTTPS using SSL
* **Access Restrictions:** Authorized users only

**9. Technology Stack**

| **Layer** | **Technology** |
| --- | --- |
| Frontend | Expo.js |
| Database | MySQL / MongoDB Atlas |
| Auth & Security | Spring Security + MFA |

**10. Deployment & Infrastructure**

* **Cloud Deployment:** Cloud-native with auto-scaling enabled
* **Database Hosting:** MongoDB Atlas or MySQL on Cloud VM
* **Device Integration:** Android phones and laptops for scanning

**11. Assumptions & Constraints**

* QR scanning works via camera-enabled devices
* University users will register manually via web dashboard
* System does not integrate with other university APIs (per R05.03)

**12. Timeline**

* **Initial System Check:** May 15, 2025
* **Full Deployment:** May 22, 2025