**Research**

1.1 Requirements stated by the Project X client The university wants an automated method for tracking attendance. If a teacher could identify the kids entering the room using his computer, tablet, or phone, that would be great. This phone, of which there may be multiple, will be referred to as a registered device. A reputable lecturer should be listed on the phone. Each lecturer may have more than one phone, and there may be multiple lecturers. The system can only use phones that are registered. If this system could store the attendance data on a server somewhere, that would be fantastic. A set of user acceptability tests, the design, and the requirements should all be formally documented. The system ought to be large. The ability to create reports on attendance, students, lecturers, courses, and students enrolled in a course is something we desire. In addition to being able to add, edit, remove, and see every item in the system, the system may also be able to find the phone in the event that the lecturer "gets lost." We also want to be able to snap images using the phone using the system so we can store a passport type photo and the student’ name and university ID and store the picture as a file and the student’s id and name in the database. The database will be a cloud-based relational MySQL database. A rest API will be used to access the database. Testing should be at the following levels

1.User Acceptance (UAT)

2.Systems

3.Unit testing – where possible

System Overview

The university requires an automated attendance system that allows instructors to use registered devices (phones, tablets, or computers) to track student attendance. The system should persist attendance records on a cloud-based MySQL database, accessible via a REST API. Additional functionalities include report generation, CRUD operations, location tracking, and student photo capturing.

Functional Requirements

User Management

* Lecturer Registration: Each lecturer should have an account.
* Device Registration: Each lecturer can register multiple devices (only registered devices can access the system).
* Student Registration: Each student has a unique University ID, name, and profile photo.

Attendance Tracking

* Student Identification: The system should allow lecturers to identify and mark attendance for students entering the classroom using a registered device.
* Data Persistence: Attendance records should be stored on a cloud-based MySQL database.

Reports & Data Management

* Generate reports for:
  + Attendance records
  + List of students
  + List of lecturers
  + Courses and enrolled students
* Perform CRUD (Create, Read, Update, Delete) operations on:
  + Lecturers
  + Students
  + Courses
  + Attendance records

Non-Functional Requirements

Database & API

* Database: MySQL (hosted in the cloud).
* API Access: The database should be accessed via a REST API.

Testing Levels

1. System Testing – Verifies end-to-end functionality of the entire system.
2. Unit Testing – Tests individual components, where feasib

1.2 Project X: Attendance Tracking System: Initial Requirements Document

1. User Management Requirements

R01. The system must allow authorized user to register students and instructors.

* R01.01 Authorized user must be able to upload student ID photos along with personal details (full name, student ID, year level, program, and faculty) during registration.
* R01.02 The system must allow authorized user to remove students or instructors from the system.
* R01.03 Authorized user must have the authority to unregister an instructor’s phone if necessary.
* R01.04 Only authorized users can access specific features.
* R01.05 Only authorized users can locate lost registered devices of the instructor.

2. Attendance Tracking Requirements

R02. The system must provide attendance tracking functionality for instructors.

* R02.01 Instructors must be able to take attendance using ID Barcode scanning via registered devices (phones or laptops).
* R02.02 Instructors must be able to take attendance using ID Barcode scanning via registered devices (phones or laptops).
* R02.03 The system must allow instructors to drop students who have accumulated three consecutive absences.
* R02.05 The system must generate attendance report for authorized users.
* R02.06 Students must be able to view their attendance records per course.

3. Security Requirements

R04. The system must implement strong security measures to protect data integrity.

* R04.01 Role-Based Access Control (RBAC) must be enforced to restrict access to sensitive data.
* R04.02 Admin-controlled data encryption must be used to ensure data confidentiality.
* R04.03 Multi-Factor Authentication (MFA) must be required for instructor logins.
* R04.04 The system must operate over HTTPS using SSL security protocols.

5. Infrastructure & Deployment Requirements

R05. The system must be cloud-based and ensure scalability.

* R05.01 The system must use MongoDB Atlas or MySQL for cloud-based data storage.
* R05.02 The system must be accessible and scalable.
* R05.03 The system must not provide an API for integration with other university systems.

6. Deployment Timeline

R06. On May 15, 2025 is the initial checking for the system.

R07. The system must be fully operational and deployed on May 22, 2025.

By following these structured requirements, the system will streamline attendance processes, improve security, and enhance data visualization for better decision-making.

Requirements Traceability Matrix (RTM) with the column format you requested:

| Requirement ID | Requirement Description | UC1 (Authentication & Access Control) | UC2 (Attendance Recording) | UC3 (Device Registration & Tracking) | UC4 (Student Enrollment & Management) |
| --- | --- | --- | --- | --- | --- |
| R1.1 | The system shall support three primary roles: Lecturer, Student, Administrator. | ✔ |  |  |  |
| R1.2 | Only registered lecturers shall be able to record attendance. | ✔ | ✔ |  |  |
| R1.3 | Only registered devices shall be authorized for attendance recording. |  | ✔ | ✔ |  |
| R1.4 | Administrators shall have full control over system data (CRUD on students, lecturers, devices, and courses). | ✔ | ✔ | ✔ | ✔ |
| R2.1 | Lecturers can record attendance using a registered device. |  | ✔ | ✔ |  |
| R2.2 | Attendance data shall be stored in a cloud-based MySQL database. |  | ✔ |  |  |
| R2.3 | Attendance shall be linked to a course, student, lecturer, date, and time. |  | ✔ |  | ✔ |
| R2.4 | Attendance shall be retrievable in real-time. |  | ✔ |  |  |
| R3.1 | A lecturer shall be able to register multiple devices. |  |  | ✔ |  |
| R3.2 | The system shall track the location of registered devices. |  |  | ✔ |  |
| R3.3 | Only registered devices shall access the attendance system. |  | ✔ | ✔ |  |

1.3 Explanation of the updated High-Level Architecture diagram for Project X: Automated Attendance System, tailored for a system using MongoDB and secure file storage.

High-Level Architecture Explanation (MongoDB Version)

Actors / Users

There are three main user types:

1. Administrator
   * Manages system data (students, lecturers, devices, courses).
   * Has full CRUD (Create, Read, Update, Delete) privileges.
2. Lecturer
   * Uses mobile or web apps to:
     + Record attendance
     + Capture student photos
     + View attendance reports
3. Student
   * Enrolls in courses
   * Views their own attendance records

Client Interfaces

.

1. Mobile App
   * Primarily used by lecturers to:
     + Take attendance
     + Capture and upload student photo.

Backend Services

1. Authentication & Authorization Service
   * Handles login, token validation, and role-based access.
   * Ensures only registered devices and users access the system.
2. REST API Layer
   * Exposes endpoints for:
     + Student enrollment
     + Attendance recording
     + Device registration
     + Reporting

MongoDB Collections (Database)

This is where all structured data is stored:

| Collection | Description |
| --- | --- |
| students | Stores student details (name, ID, profile picture filename, etc.) |
| lecturers | Lecturer profiles and linked device IDs |
| courses | Course info and enrolled students |
| attendance\_logs | Logs of attendance by course, student, date, and time |
| devices | Registered lecturer devices |
| photos | Metadata (e.g., student ID, filename) for captured photos |

MongoDB stores these as **JSON-like documents**

Interaction Flow

1. A lecturer logs in on a registered mobile device.
2. The mobile app sends a request to the REST API, which verifies the device and user role.
3. The lecturer records attendance or takes a student photo.
4. The backend:
   * Saves the attendance record in attendance\_logs.
   * Uploads the photo to Secure File Storage.
   * Saves metadata (like filename, student ID) in the photos collection.
5. Reports can be generated in real time for both lecturers and administrators.

Summary

* MongoDB handles all structured data in collections.
* A REST API connects clients (web and mobile) to backend logic.
* Secure file storage is used for image files.
* Role-based access ensures secure and appropriate use of system features.

1.4 If QR Code Scanning **is** part of your system (even implicitly), here's how it would fit:

Where QR Code Scanning Fits in the Architecture:

1. Lecturer’s Role:
   * Generates a QR code for a specific course session.
   * Displays it via the Mobile App or Web Interface.
2. Student’s Role:
   * Uses their mobile app to scan the QR code to mark attendance.
3. System Workflow:
   * After scanning:
     + The app sends the scanned QR data to the REST API.
     + The Business Logic Layer validates:
       - The QR code’s authenticity and expiry.
       - The student's enrollment in that course.
     + An entry is added to the attendance\_logs collection in MongoDB.

Architecture Impact:

If you do include QR scanning, the diagram would need:

* A “QR Code Generator” component (on lecturer side).
* A “QR Code Scanner” feature in the student mobile app.
* The API must handle QR code validation and attendance logging.

1.5 The high-level architecture diagram with Mobile App only and QR code scanning.

High-Level Architecture Explanation

The Automated Attendance System (Project X) is designed to provide secure and efficient attendance tracking using a mobile app for both lecturers and students, with backend services supported by a REST API and MongoDB database.

Key Components:

* Users:
  + Administrator: Manages system data such as student records, lecturers, devices, and courses via the mobile app.
  + Lecturer: Uses the mobile app to register attendance, capture student photos, generate attendance reports, and register devices.
  + Student: Uses the mobile app to scan QR codes for marking attendance and to enroll in courses.
* Client Interface:
  + Mobile App: The single interface used by all users to interact with the system. Lecturers generate QR codes for attendance sessions and capture photos, while students scan these QR codes to record their attendance.
* Backend System:
  + Authentication & Role-Based Access Service: Ensures secure login and authorizes users based on their roles (Administrator, Lecturer, Student).
  + REST API Layer: Acts as the communication bridge between the mobile app and backend services. All requests such as attendance recording, student enrollment, and report retrieval are processed here.
  + Business Logic Layer: Implements core functionalities, including validating QR codes, managing attendance records, handling student and lecturer data, and enforcing access control.
* QR Code Components:
  + QR Code Generator: Allows lecturers to create unique QR codes for each attendance session. These codes encode session information such as course, date, and time.
  + QR Code Scanner: Embedded in the mobile app for students to scan the lecturer-generated QR codes to mark their attendance securely.
* Database and Storage:
  + MongoDB: Stores collections for students, lecturers, courses, attendance logs, registered devices, and photo metadata.
  + Secure File Storage: Stores student photos captured by lecturers securely and links them to student records.

System Workflow Summary:

1. A lecturer logs in through the mobile app and generates a QR code for the current attendance session.
2. Students use the mobile app to scan the QR code, which sends scanned data to the backend via the REST API.
3. The backend validates the QR code, verifies student enrollment, and records the attendance entry in MongoDB.
4. Administrators and lecturers can access reports and manage data through the mobile app.
5. The system enforces strict role-based access and device registration to ensure only authorized users and devices can record or modify attendance data.