**Research**

1.1 Project X Client Stated Requirements The University wants an automated attendance system. It would be nice if an instructor could use his phone, tablet or computer to identify the students coming in the door. Let's call this phone a registered device, of which there can be several. The phone should be registered to a known lecturer. There can be several lecturers, and each lecturer can have more than 1 phone. Only registered phones may be used by the system. It would be really great if this system would persist this attendance information on a server somewhere. There should be some formal documentation as to requirements, the design and a set of user acceptance tests. The system should be big. We want to be able to generate reports on the attendance, list of students, lecturers, courses and students enrolled on a course. We want to be able add amend delete and view all items in the system Furthermore, the system might be able to locate the phone – should the lecturer ‘get lost’. We also want to be able to take pictures using the phone using the system so we can store a passport style 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 shall be a MySQL relational database established in the cloud. The access to the database shall be via a rest API. 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: Automated Attendance System R1. User Roles and Access Control R1.1. The system shall support three primary roles: Lecturer, Student, and 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 operations on students, lecturers, devices, and courses). R2. Attendance Recording and Data Persistence R2.1. The system shall allow lecturers to record student attendance using a registered mobile device, tablet, or computer. R2.2. Attendance data shall be stored on a cloud-based MySQL database. R2.3. Attendance shall be linked to a course, identifying the student, lecturer, date, and time. R2.4. Attendance shall be retrievable in real-time. R3. Device Registration and Tracking R3.1. A lecturer shall be able to register multiple devices. R3.2. The system shall track the location of registered devices in case a lecturer needs assistance. R3.3. Only registered devices shall be able to access the attendance system. R4. Student Enrollment and Management R4.1. Students shall be able to enroll in courses through the system. R4.2. The system shall store and manage student details, including: R4.2.1. Name R4.2.2. University ID R4.2.3. Profile Picture (captured via the system) R4.3. The system shall allow adding, updating, and deleting student records. R5. Reporting and Data Access R5.1. The system shall provide the following reports: R5.1.1. Attendance reports (by student, by course, by date range). R5.1.2. List of students per course. R5.1.3. List of lecturers and their registered devices. R5.2. Reports shall be accessible to lecturers and administrators. R6. Photo Capture and Storage R6.1. The system shall allow lecturers to capture passport-style photos of students. R6.2. Photos shall be stored as files, and student IDs and names shall be stored in the database. R6.3. The system shall ensure secure access to stored images. R7. System Architecture and API Access R7.1. The system shall use a REST API to interact with the database. R7.2. All user interactions (attendance, enrollment, reporting) shall be handled through the API. R7.3.API requests shall be authenticated and authorized. 8. Testing Criteria R8.1. Unit Testing: Each function (attendance logging, student enrollment, etc.) shall be tested in isolation. R8.2. System Testing: The system shall be tested end-to-end to validate workflows. R8.3. User Acceptance Testing (UAT): The system shall be tested against the above assertions to confirm it meets user needs.

**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.