

Republic of the Philippines

DAVAO ORIENTAL STATE UNIVERSITY

Guang-guang, Dahican, City of Mati, Davao Oriental

Faculty of Computing, Data Sciences, Engineering and Technology

Information Technology Program

Project X

Automated Attendance System

BSIT 3B

Presented by:

Motlog, Christian Dave B.

Talara, Jelaie G.

Alvar, Rikki Mae

I. INTRODUCTION

1.1 Purpose

This study presents the development of a digital attendance management system designed to overcome the inefficiencies inherent in traditional methods such as paper logs and spreadsheets. These manual approaches are prone to inaccuracies, tampering, and unnecessary time consumption, which compromises data reliability and detracts from instructional time (Yadav & Singh, 2019). To mitigate these challenges, the proposed system introduces a tech-driven solution with the following capabilities:

* Utilization of authorized devices for check-in procedures
* Student identity verification via captured photos
* Secure, cloud-based data storage
* Location tracking to deter proxy attendance (Kumar & Goudar, 2014)

The aim is to streamline attendance tracking through automation, thereby improving accuracy and safeguarding student records. The expected deliverables include a functional prototype, performance assessments, and deployment recommendations to support modernized academic management.

1.2 Objectives of the Study

General Objective:  
To create Project X, a secure, QR code-enabled automated attendance solution for Davao Oriental State University (DOrSU), aimed at improving the precision, efficiency, and integrity of student attendance data.

Specific Objectives:

1. Replace manual attendance tracking with an efficient digital alternative
2. Implement QR scanning for verified student identity confirmation
3. Utilize cloud-based infrastructure for live data access and storage
4. Assess and document system performance, scalability, and deployment readiness for broader institutional adoption

1.3 Scope and Limitations

Scope:  
The system is tailored for academic institutions and features:

* Device-based faculty check-in capabilities
* QR code-based verification for student identity
* GPS validation to confirm physical presence in the classroom
* Secure MongoDB for storing all records

Limitations:

* Requires devices with camera and internet connectivity
* QR scanning performance may be affected by environmental lighting or camera quality
* No integration with existing LMS/SIS platforms
* Advanced security features such as multi-factor authentication are not implemented
* Testing is limited to DOrSU only; performance at larger institutions remains unverified
* User engagement from faculty and students is critical for full functionality

V. REQUIREMENTS

2.1 FUNCTIONAL REQUIREMENTS

2.1.1 User Roles & Permissions

The system recognizes and supports three distinct user categories:

1. Administrator – Oversees user management, device authorization, course allocation, and report generation
2. Lecturer/Instructor – Handles student attendance, manages assigned devices, and captures student photos
3. Student – Recognized by the system and has their attendance recorded automatically

2.1.2 Device Registration & Validation

* Lecturers must register their devices (smartphone, tablet, or PC) for attendance logging
* Multiple devices can be registered per lecturer, but each must have a unique identifier
* Only verified devices are permitted system access

2.1.3 Attendance Monitoring

* Attendance may be logged manually or automatically when a student enters the classroom
* Each record includes student ID, full name, timestamp, and course information
* Data is saved instantly to the cloud database
* Manual correction options are available in case of errors

2.1.4 Student Identification via Photos

* Lecturers can capture student images using registered devices
* Student photos must follow passport-style guidelines and are stored as image files
* Student names and IDs are saved in the backend database

2.1.5 Geolocation Services

* Lost devices can be tracked in real-time by authorized users
* Location information is stored securely and only accessible to permitted personnel

2.1.6 Reporting & Data Handling

* The system can produce reports including:
  + Session-wise attendance summaries
  + Overall attendance patterns
  + Lecturer and course assignments
  + Student enrollments by course
* Full CRUD (Create, Read, Update, Delete) operations are available for users, courses, and attendance logs

2.1.7 API & System Access

* All backend operations are handled via RESTful APIs
* Secure login and access control are enforced
* MySQL is used for cloud-hosted database operations

2.1.8 System Testing

Testing will be conducted at three levels:

1. User Acceptance Testing (UAT) – Verifying that system outputs align with institutional goals
2. System Testing – Validating API, UI, and database connectivity across the full application
3. Unit Testing – Testing individual functions/modules to ensure component-level reliability

2.2 NON-FUNCTIONAL REQUIREMENTS

2.2.1 System Performance

* The platform should respond to inputs within 2 seconds under average load conditions
* Attendance check-ins and QR code scans must be processed in real-time

2.2.2 Scalability

* The system is designed for DOrSU campuses, with support for multiple departments, users, and devices
* The database infrastructure will auto-scale with increases in data volume

2.2.3 Availability

* Maintains 75% uptime during academic hours
* Backup procedures (e.g., manual logs, Google Forms) are available for emergency use

2.2.4 Security

* Strong password rules and optional two-factor authentication (2FA)
* All communications will be protected by HTTPS encryption
* Role-based access ensures restricted features are only available to authorized users
* Personally, identifiable data (e.g., IDs, photos) must comply with privacy laws such as GDPR

2.2.5 Usability

* The user interface will be intuitive and tailored to each role (Admin, Lecturer, Student)
* User manuals and onboarding guides will support ease of use
* Design will be responsive and mobile-compatible

2.2.6 Maintainability

* MVVM architecture enables modular updates and efficient maintenance
* Built-in diagnostics and error logs assist administrators with troubleshooting

2.2.7 Compatibility

* The application is compatible with popular platforms including Windows, macOS, Android, and iOS

2.2.8 Backup & Recovery

* Daily cloud backups ensure data integrity
* In case of failure, full recovery of attendance records is expected within 24 hours

SYSTEM ARCHITECTURE

High-Level Design (HLD)

This section provides an overview of the Automated Attendance System’s key components and their interactions. The system incorporates QR-based scanning, role-specific access rights, and real-time monitoring. It defines how administrators, lecturers, and students interface with the system, laying the foundation for development, integration, and future scalability.

System Architecture Components

|  |  |  |
| --- | --- | --- |
| Layer | Component | Technology / Description |
| Presentation Layer | Mobile Frontend | React.js – User Interfaces for Students, Lecturers, Admins |
| Application Layer | RESTful API Server | Node.js – Handles all business logic, data validation, and routing |
| Data Layer | Relational Database | MongoDB – Stores all users, courses, attendance, and system logs |
| QR Code Module | QR Code Generator & Scanner | Generates time-bound QR codes; integrated with mobile scanning |
| Authentication | Auth Module | JWT-based Login System with hashed passwords and role-based access control |
| Notification Module | Optional | Email or SMS-based alerts (absences, announcements, etc.) – future expansion |
| Admin Dashboard | Admin Interface | Manages courses, users, schedules, and logs |
| Logs | Logging/Audit System | Tracks user activity, login/logout records |

Module Overview

|  |  |
| --- | --- |
| Module | Functions |
| Student Module | - View attendance history - Scan QR code to check-in |
| Lecturer Module | - Generate QR code - Monitor attendance - View enrolled students |
| Admin Module | - Manage users, roles, and courses - View system logs |
| Attendance Module | - Record attendance from QR scans - Validate student and session details |
| Course Management | - Add/update courses - Assign lecturers to classes |
| QR Management | - Create time-limited QR codes - Mark QR codes as expired automatically |
| Login & Security | - Hash passwords - Generate/verify JWT tokens - Role-based access |

Technology Stack

|  |  |
| --- | --- |
| **Component** | **Technology Suggestion** |
| Frontend | React.js |
| Backend API | Node.js |
| Database | MongoDB |
| QR Code Handling | QRCode libraries |

Security Considerations

* Passwords stored using strong hashing.
* Role-based access control to restrict data exposure.
* Expiring QR codes to prevent re-use.
* Input validation on all form entries and API endpoints.
* Secure API tokens using HTTPS.

Future Enhancements

* SMS/Email Notifications
* Analytics Dashboard for Admin
* Machine Learning for detecting anomalies in attendance
* Biometric integration

LOW LEVEL DESIGN

Table: User (Base Class)

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Description |
| id\_user | INT | PK, AI | Primary key |
| email | VARCHAR(100) | UNIQUE | Email used for login |
| username | VARCHAR(45) | UNIQUE | Username |
| password | VARCHAR(255) |  | Hashed password |
| role | ENUM('student', 'instructor', 'admin') |  | User role for access control |

Table: Student (Inherits from User)

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Description |
| id\_student | INT | PK, AI | Primary key |
| id\_user | INT | FK | References User(id\_user) |
| fname | VARCHAR(100) |  | First name |
| lname | VARCHAR(100) |  | Last name |
| studentID | VARCHAR(45) | UNIQUE | Official student number |

Table: Instructor (Inherits from User)

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Description |
| id\_instructor | INT | PK, AI | Primary key |
| id\_user | INT | FK | References User(id\_user) |
| fname | VARCHAR(100) |  | First name |
| lname | VARCHAR(100) |  | Last name |
| employeeID | VARCHAR(45) | UNIQUE | Instructor ID |

Table: Course

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Description |
| id\_course | INT | PK, AI | Primary key |
| course\_code | VARCHAR(20) | UNIQUE | Course code (e.g., ITC130) |
| course\_name | VARCHAR(100) |  | Descriptive course name |
| id\_instructor | INT | FK | References Instructor(id\_instructor) |

Table: Enrollment

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Description |
| id\_enrollment | INT | PK, AI | Primary key |
| id\_student | INT | FK | References Student(id\_student) |
| id\_course | INT | FK | References Course(id\_course) |
| enrolled\_at | DATETIME |  | Date/time student enrolled |

Table: Attendance

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Description |
| id\_attendance | INT | PK, AI | Primary key |
| id\_student | INT | FK | References Student(id\_student) |
| id\_course | INT | FK | References Course(id\_course) |
| date | DATE |  | Date of attendance |
| status | ENUM('Present', 'Absent', 'Late') |  | Attendance status |

Table: QRCode

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Description |
| id\_qr | INT | PK, AI | Primary key |
| id\_course | INT | FK | References Course(id\_course) |
| generated\_at | DATETIME |  | Date/time the QR code was created |
| qr\_data | TEXT |  | Encoded data for attendance |