

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 3C

Presented by:

Sitti Monera Martin

Kyle Alonzo

Honey Jane P. Pangasate

**I. INTRODUCTION**

**1.1 Purpose**

This study focuses on creating an automated attendance system to resolve the inefficiencies of manual tracking methods in academic settings. Conventional approaches like paper logs or spreadsheet entries are vulnerable to errors, tampering, and time wastage, undermining record accuracy and diverting time from educational activities (Yadav & Singh, 2019). To address these issues, this project proposes a digital solution leveraging modern technology. The system will:

* Use registered devices for check-ins.
* Verify identities via student photos.
* Employ cloud storage for secure data access.
* Incorporate geolocation to prevent proxy attendance (Kumar & Goudar, 2014).

The goal is to enhance attendance management through automation, ensuring reliability and security. Outcomes include a functional prototype, evaluation results, and deployment guidelines, supporting the digital evolution of academic administration.

**1.2 Objectives of the Study**

**General Objective:**To develop *Project X*, a secure automated attendance system for Davao Oriental State University (DOrSU), utilizing QR code technology to enhance accuracy, efficiency, and data integrity.

**Specific Objectives:**

1. Replace paper-based attendance with a digital platform to reduce manual effort.
2. Implement QR code scanning for secure student authentication.
3. Use cloud-based databases for real-time record storage and access.
4. Document system performance and scalability recommendations for wider adoption.

**1.3 Scope and Limitations**

**Scope:**

* **Designed for academic institutions, the system includes:**
  + Faculty-registered device check-ins.
  + QR-based student verification.
  + Geolocation validation for classroom presence.
  + Secure MongoDB database storage.

**Limitations:**

* Requires camera-equipped devices and stable internet.
* QR scanning may be affected by lighting or camera quality.
* Excludes integration with LMS/SIS systems.
* Advanced security (e.g., multi-factor authentication) not included.
* Testing confined to DOrSU; scalability to larger institutions untested.
* Effectiveness hinges on user adoption by faculty and students.

**V. REQUIREMENTS**

**2.1 FUNCTIONAL REQUIREMENTS**

**2.1.1 User Roles & Access**

• The system shall support three user roles:

1. **Administrator** – manages users, devices, courses, and reports.
2. **Lecturer** – records student attendance, manages their own registered devices, and takes student photos.
3. **Student** – is identified by the system and has their attendance recorded.

**2.1.2 Device Registration & Authentication**

* A lecturer must register a device (phone, tablet, or computer) before it can be used for attendance tracking.
* A lecturer may register multiple devices, but each device must be uniquely identifiable.
* Only registered devices shall be allowed to access the attendance system.

**2.1.3 Attendance Tracking**

* The system shall automatically or manually record student attendance when a student enters the classroom.
* Each attendance record shall include the student’s ID, name, timestamp, and course details.
* Attendance data shall be immediately stored in the cloud database.
* The system shall provide an option for manual override in case of errors.

**2.1.4 Student Identification & Photo Capture**

* The system shall support photo capture of students using the lecturer’s registered device.
* The photo shall be stored as a file, while the student’s ID and name are stored in the database.
* The system shall enforce passport-style guidelines for student photos.

**2.1.5 Location Tracking**

* The system may allow lecturers to track the real-time location of their registered device in case it is lost.
* Location data shall be securely stored and accessible only to authorized users.

**2.1.6 Reporting & Data Management**

* The system shall generate attendance reports, listing:
* Students present/absent per session
* Overall attendance trends
* Lecturers and their assigned courses
* Students enrolled in each course
* Users shall be able to add, edit, delete, and view all items in the system, including students, lecturers, courses, and attendance records.

**2.1.7 System Access & API Integration**

* The system shall use a REST API for all database operations.
* Access control shall be implemented using secure authentication mechanisms.
* The database shall be cloud-based using MySQL.

**2.1.8 Testing Requirements**

Testing shall be performed at three levels:

1. User Acceptance Testing (UAT): Ensuring that the system meets business requirements and client expectations.
2. System Testing: Verifying system-wide functionality, including API interactions and database operations.
3. Unit Testing: Testing individual components where applicable to ensure proper functionality.
4. 4. System Development Life Cycle

**2.2 NON – FUNCTIONAL REQUIREMENTS**

**2.2.1 Performance**

* The system shall respond to user input within 2 seconds under normal load conditions.
* Attendance check – ins and qr code scanning shall be processed in real – time.

**2.2.2 Scalability**

* The system shall be designed to support only campuses of DOrSU , each with separate departments, users, and devices.
* The cloud database shall scale automatically to accommodate increasing numbers of students, courses, and attendance records.

**2.2.3 Availability**

* The system is maintained 75% uptime, ensuring high availability for all users during academic hours.
* Backup systems are placed to minimize downtime in the event of failure such as manual attendance, google form and such.

**2.2.4 Security**

* User authentication is to be enforced using strong password policies and optional two-factor authentication (2FA).

* All data transmissions between client devices and the server shall use HTTPS encryption.
* Role – based access control (RBAC) shall prevent unauthorized access to restricted features.
* Sensitive information such as student IDs and photos shall be stored securely and comply with privacy standards such as GDPR or local data protection laws.

**2.2.5 Usability**

* The system shall provide an intuitive and accessible interface for all user roles (Administrator, Lecturer, Student).
* Training documentation and help guides shall be provided to assist first-time users.
* The interface shall be responsive and mobile-friendly.

**2.2.6 Maintainability**

* The system architecture (MVVM) shall ensure modular design, allowing for easy updates and maintenance.
* Error logs and system diagnostics shall be accessible to administrators or IT staff for debugging and monitoring.

**2.2.7 Compatibility**

* The system should be compatible with major mobile app devices.
* The lecturer device module shall support Windows, macOS, Android, and iOS platforms.

**2.2.8 Backup and Recovery**

* Attendance data shall be backed up daily on secure cloud storage.
* In the event of data loss or corruption, the system shall support full recovery within 24 hours.

## SYSTEM ARCHITECTURE

## HIGH LEVEL DESIGN

**Introduction**

The High-Level Design (HLD) describes the major components of the Automated Attendance System. It outlines the system's architecture, core modules, and how the components interact. The system utilizes QR code-based attendance tracking, role-based access, and real-time monitoring for students, lecturers, and administrators.

**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., IT101) |
| 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**

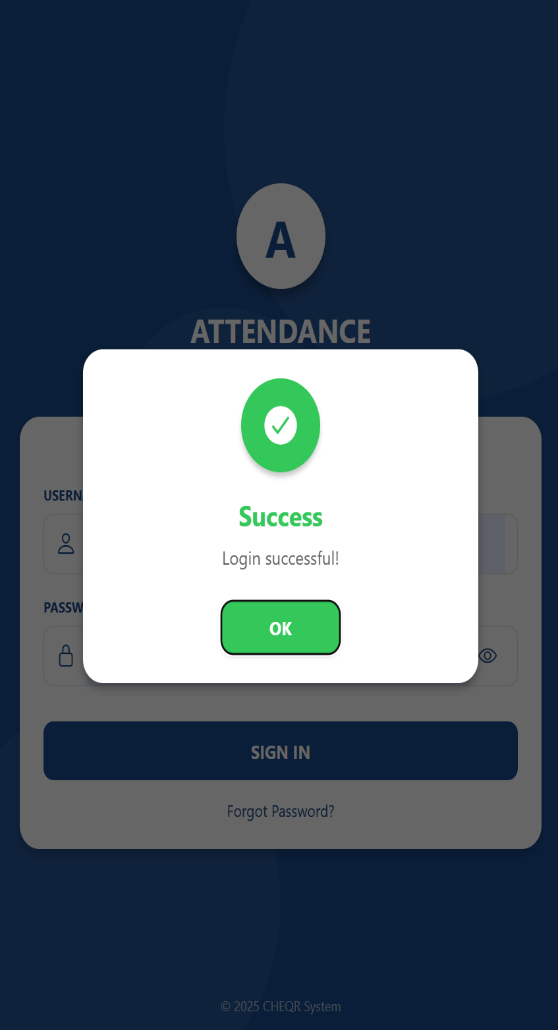
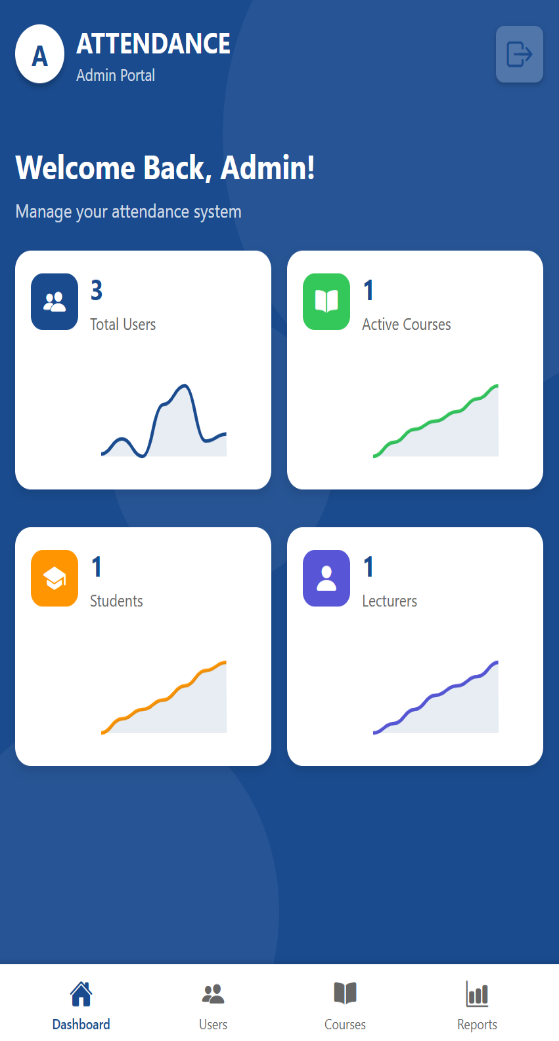
|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Key** | **Description** |
| id\_attendance | INT | PK, AI | Primary key |
| id\_student | INT | FK | References Student(id\_student) |
| id\_course | INT | FK | References Course(id\_course) |
| date | DATE |  | Date of attendance |
|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

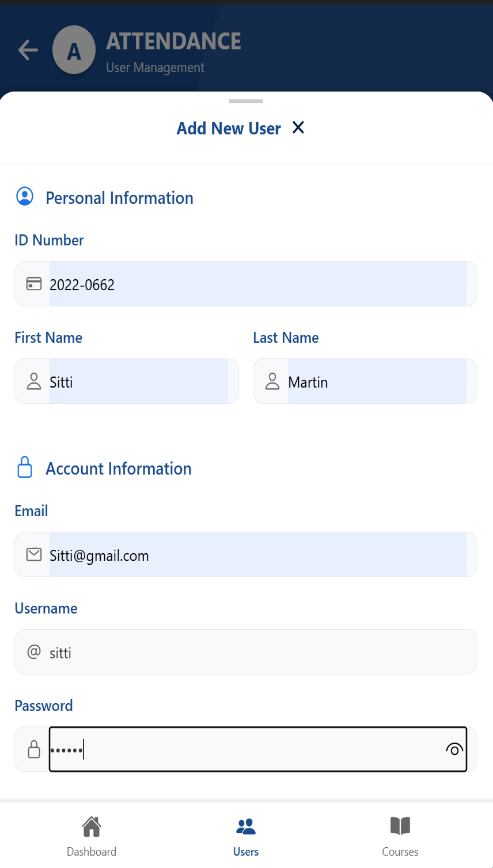
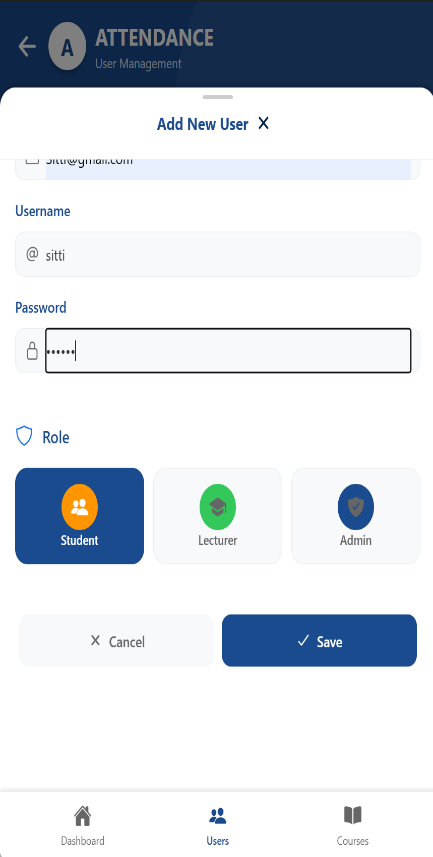
**Table: QRCode**

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |

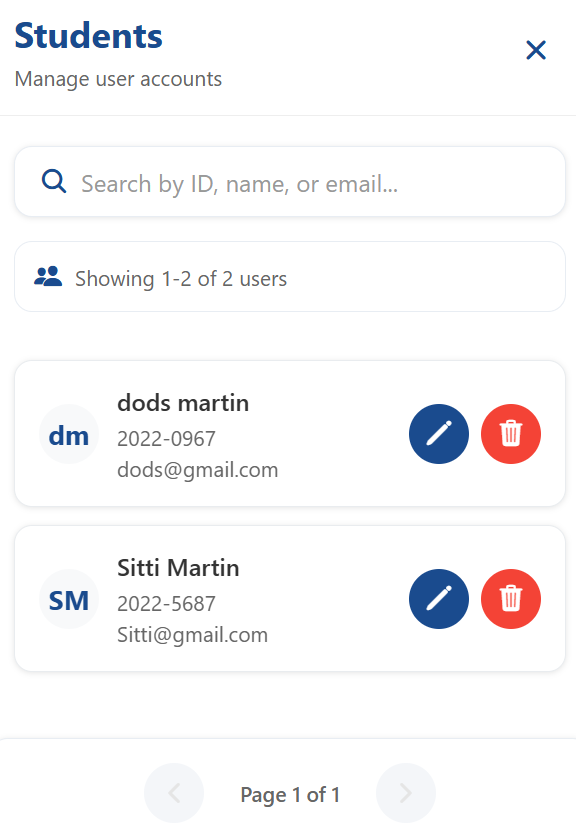
**APP**

ADMIN TESTING   
  
STEP 1: LOGIN

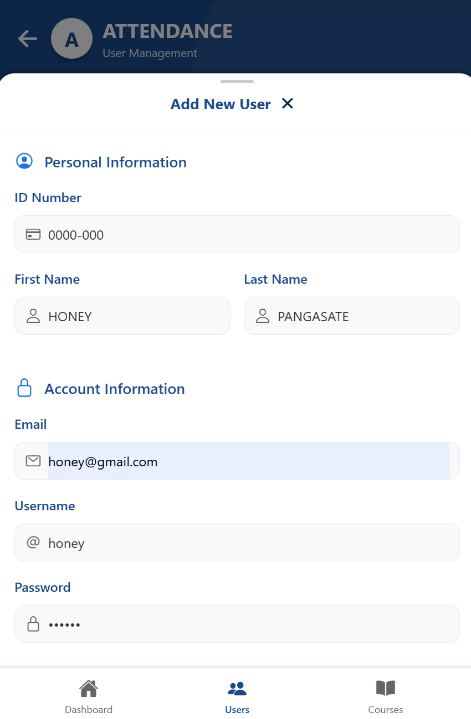
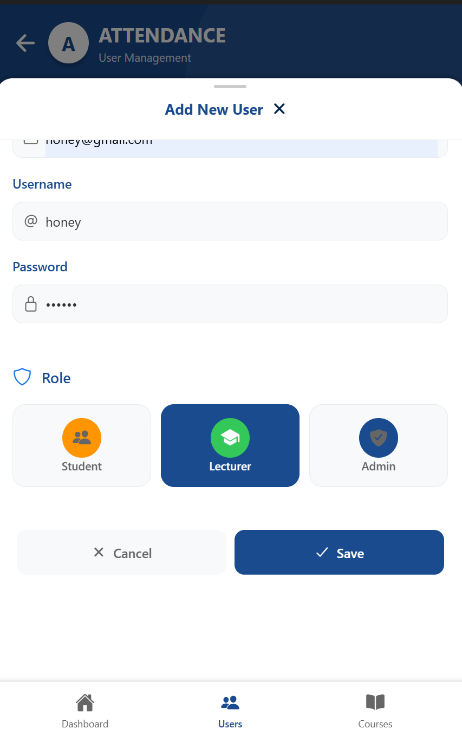


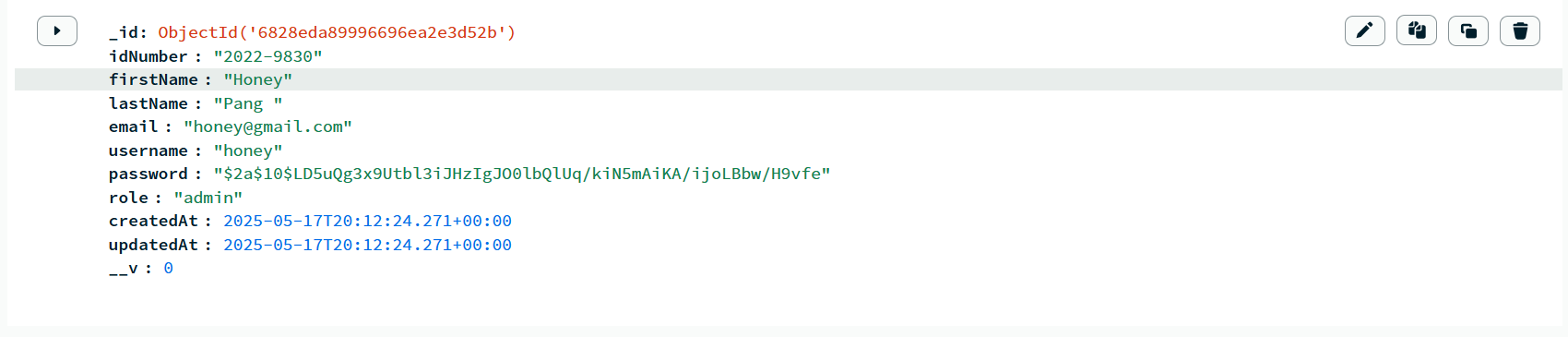
STEP 2: ADD USERS   
2.1 ROLE STUDENT

2.2 Save in the db

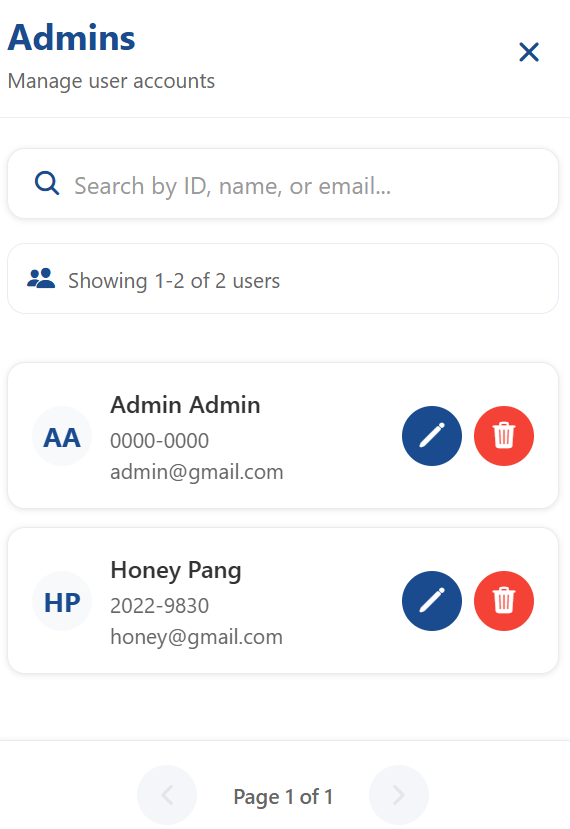
2.3 view in the list

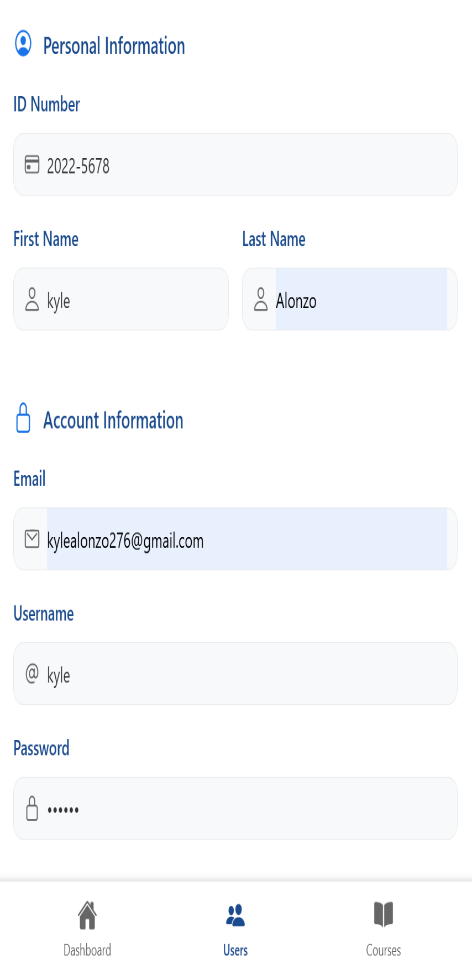
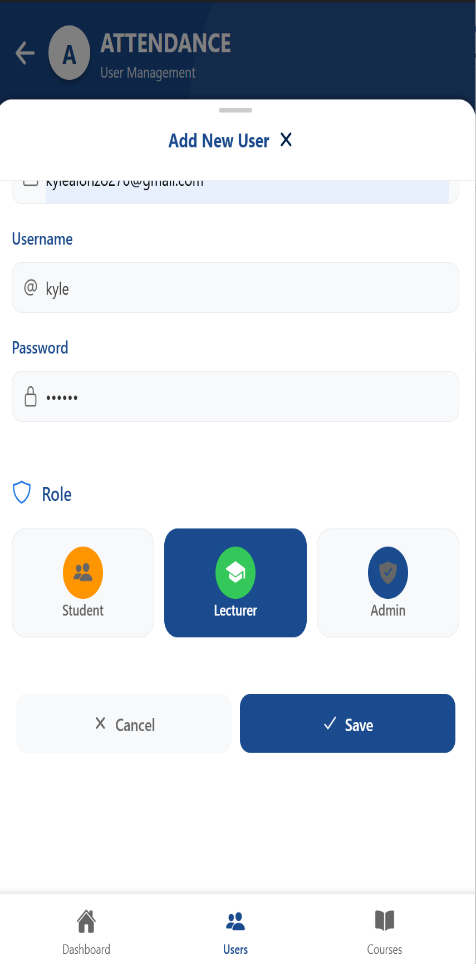
2.4ROLE ADMIN



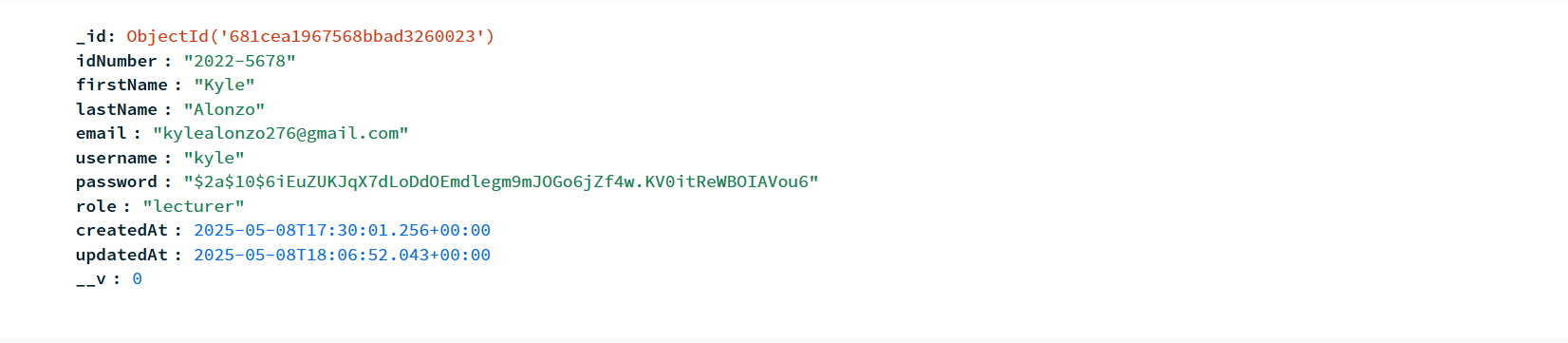
2.5 save in the db

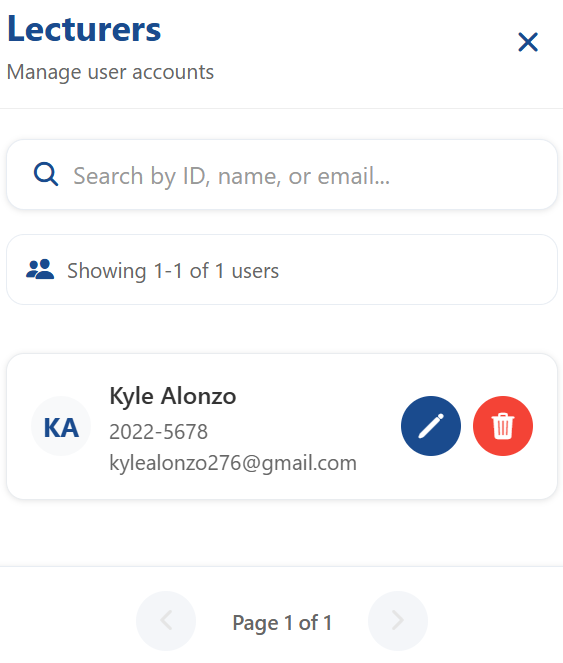
2.6. view in the list

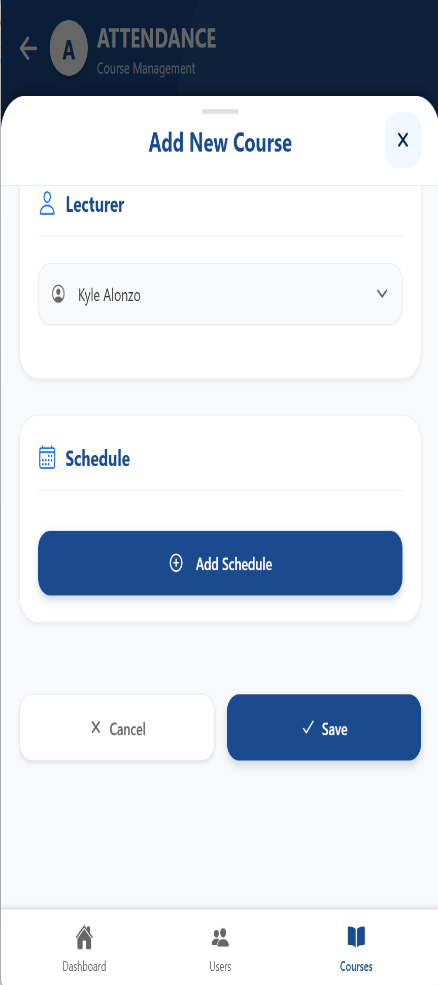
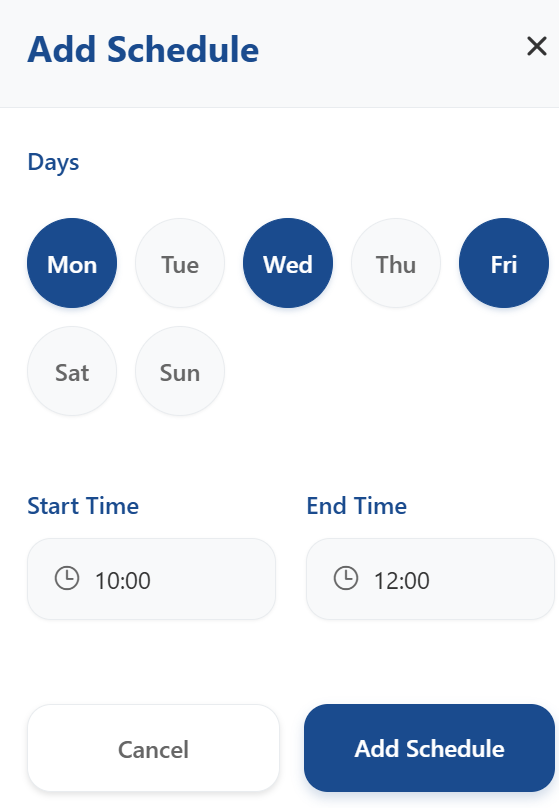


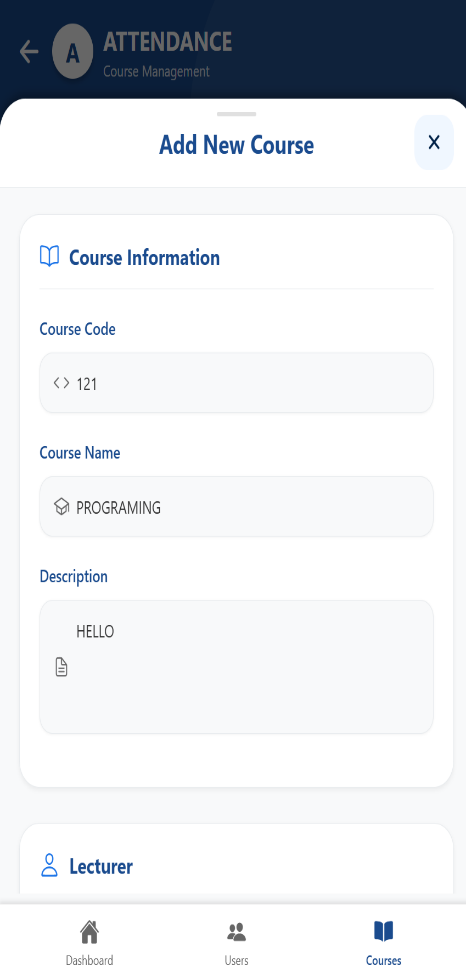
2.7 ROLE LECTURER

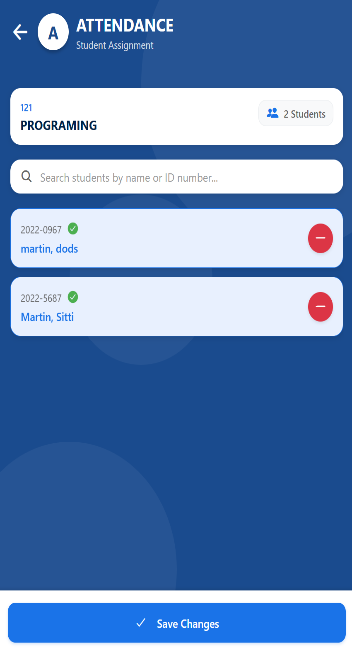
2.8 save in the db

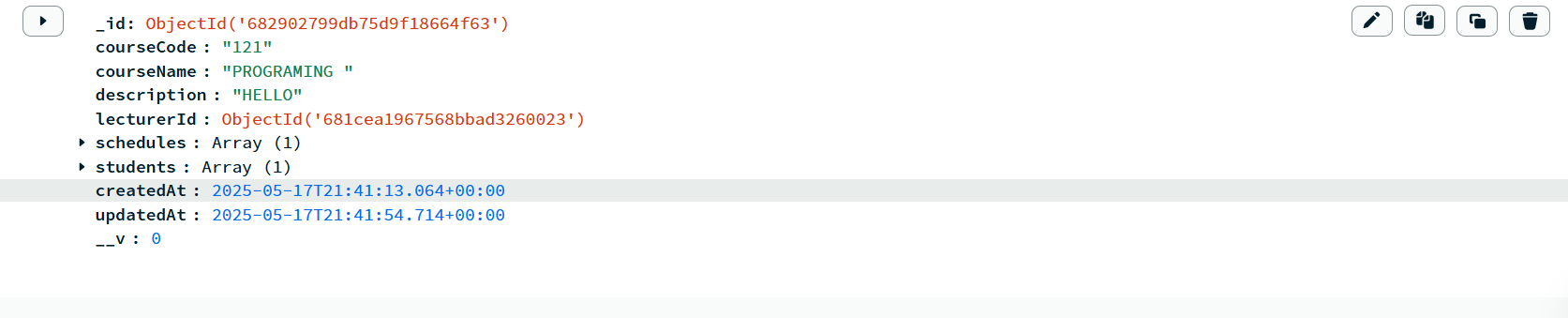


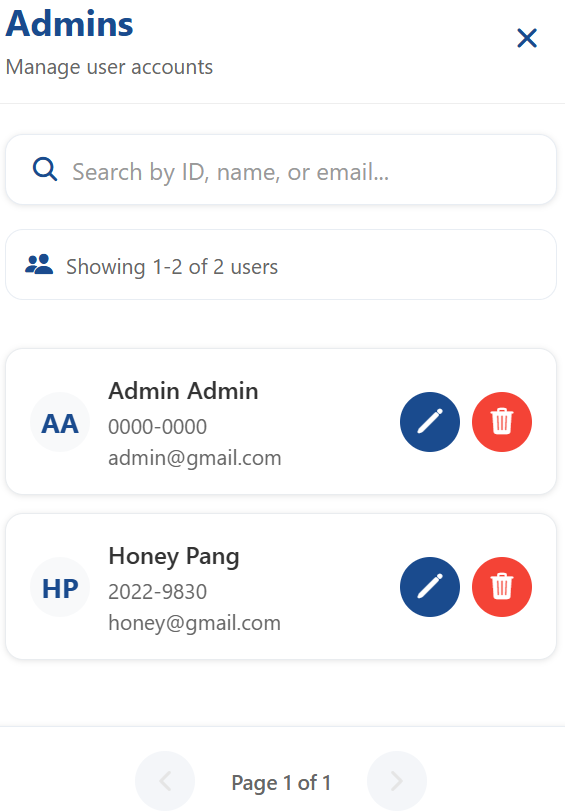
2.9 View the list

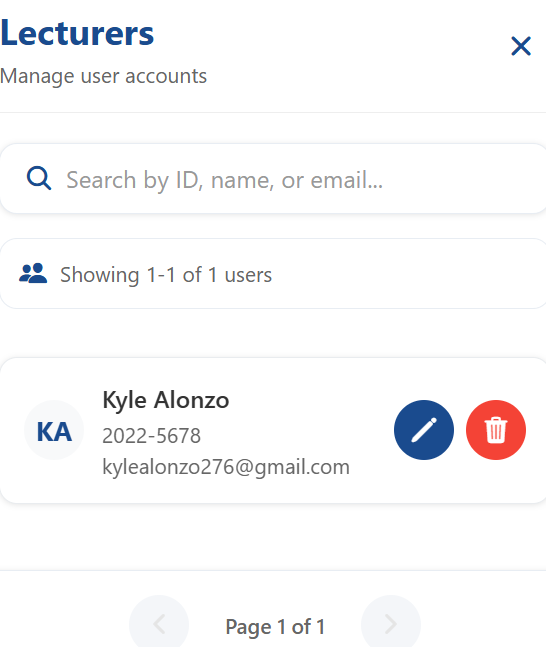
3. ADD COURSE

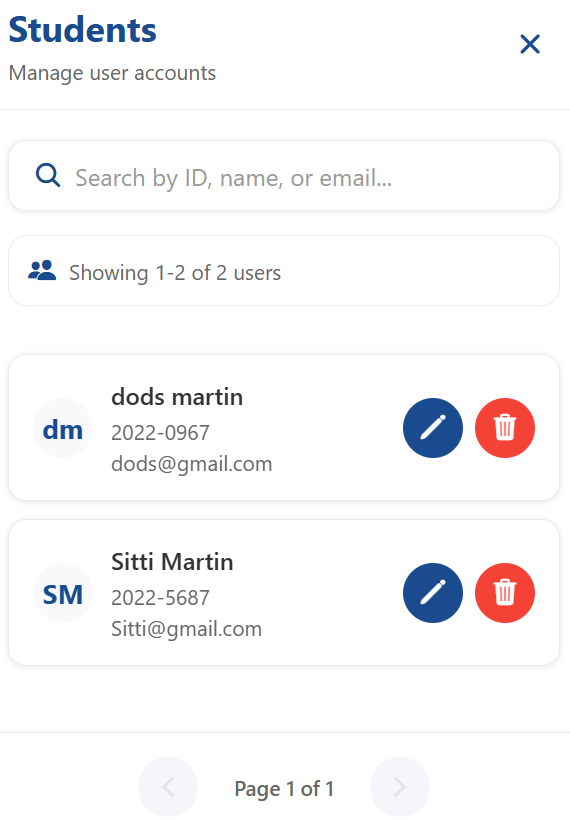


3.2 ADD STUDENT

3.3 save in the db

4. view admin details or list

5. view lecturer details or list

6. view students details or list