UNIVERSITY OF BUEA
Buea, South West Region
Cameroon
P.O. Box 63,
Tel: (237) 3332 21 34/3332 26 90
Fax: (237) 3332 22 72

REPUBLIC OF CAMEROON
PEACE-WORK-FATHERLAND

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

UI and Implementation of a mobile base Attendance system with Facial recognition and Geo-fencing

By: Group 24

Course Master:
Dr. Nkemeni Valery
University of Buea

2024/2025 Academic Year

Group 24 Members

Name	Matricule
BILLA SOPHIA	FE22A176
EKANE METUGE AKAME FAVOUR	FE22199
EYONG GODWILL NGANG	FE22A214
ONYA MARTHA	FE22A292
NEBOTA ISMAEL OWAMBA	FE22A256

Catalog

1. Introduction

1.1 Purpose of the Task

The purpose of this task is to design and implement a visually appealing, user-friendly, and functional user interface (UI) for the **Mobile-Based Attendance Management System** that integrates **geofencing** and **facial recognition**. The UI design aims to ensure an intuitive user experience for both students and instructors while reflecting the app's core identity and purpose of real-time attendance tracking. This task emphasizes the development of a consistent design language, interactive components, and responsive layouts that work seamlessly across mobile devices.

1.2 Scope of the UI Design and Implementation

This app focuses on three core areas:

- **App Identity**: Establishing a recognizable and professional brand identity through logo design, typography, color schemes, and app icon creation.
- **Visual Design**: Creating low- and high-fidelity wireframes, defining UI components such as buttons and forms, and setting up layout structures and navigation flows. Accessibility and responsiveness are also considered to support a wide range of users and devices.
- **Frontend Implementation**: Translating the design into functional interfaces using a mobile development framework(flutter). This includes screen development, component integration, animations, and frontend logic for real-time interaction, including face detection, GPS validation, and user notifications.

1.3 Tools and Technologies Used

The following tools and technologies are used in the UI design and frontend implementation:

- **Design Tools**: Figma (wireframing, mockups), Canva (logo/icon design).
- **Frontend Framework**: Flutter (cross-platform mobile development).
- **Programming Language**: Dart

- **Supporting Libraries**:
  - google_maps_flutter for geolocation and geofencing.
  - `camera`, and tflite_flutter for face detection and recognition.
  - provider for state management.
- **Testing Tools**: Flutter DevTools, integration_test package for UI and UX validation

2. App Identity

2.1 Brand Name & Logo Design

- **Brand Name**: **AuraCheck**
- **Meaning & Concept**:
  - *"Aura"* symbolizes presence, identity, and authenticity core themes of facial recognition.
  - *"Check"* emphasizes verification, attendance, and validation.
  - Together, *AuraCheck* communicates a high-tech, secure, and presence-focused solution.
- **Logo Design Concept**:
  - A clean, modern logo that integrates:
    - ✧ A minimal school base (symbolizing attendance checking).
    - ✧ A **location pin or shield** (representing geofencing and secure
- **File Formats**: SVG and PNG (for app splash screen)

2.2 Color Scheme & Typography

Purpose	**Color**	**Hex-Code**
Primary color	Blue	#0053AD
Secondary color	Black	#212121
Success	Green	#115B43
Warning	Yellow	#E48900
Danger	Red	#C33025
Tertiary color	White	#FFFFFF

Why these colors?:

- Blue conveys **trust, technology**, and **clarity**.
- Yellow adds **energy and attention** to CTAs like "Check In".
- Neutrals keep the UI readable and minimal.
- **Green** conveys Success or Confirmation and is Used to show that an action (like check-in) was completed successfully.

Typography

- **Primary Font (Headings)**: *Poppins Bold*
- **Secondary Font (Body Text)**: Inter semi bold.
- **Button Font**: *Poppins Medium.*
- **Font Sizes**:

  - Title: 20–24px
  - Subtitle: 16–18px
  - Body: 14–16px
  - Buttons: 16px

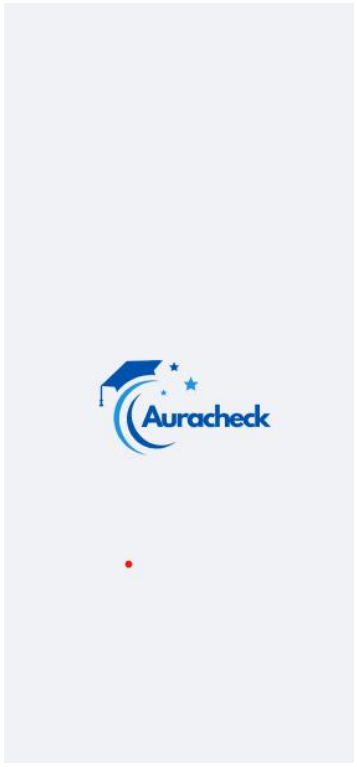
2.3 App Icon Design

- **Icon Style**: Flat, minimalistic, instantly recognizable
- **Icon Elements**:

  - A stylized **face outline** surrounded by a soft A**uracheck.**
  - **Graduation cap** which Symbolizes Education, learning, student identity.
  - A subtle Star **mark** integrated into the design which Symbolize**s** Accuracy, innovation, excellence.
  - **Curved Circles/Swooshes** which Symbolizes Aura, presence, dynamic motion.

- **Color Palette**: Primary blue background with white icon elements.
- **Usability**: Designed to look sharp on both small (app list) and large (splash screen)

Logo



Splash Screen

3. Visual Design

3.1 UI Style Guide (Buttons, Icons, Fonts Consistency)

- **Buttons**
  - **Primary Button:**
    - Color: #0053AD (Blue).
    - Text: White, Bold

- ✧ Usage: "Check In", "Submit", "Confirm"
- ✧ Rounded corners, slight shadow for emphasis

- **Secondary Button**:

  - ✧ Color: Transparent with border `#0053AD`
  - ✧ Text: Blue, Medium weight
  - ✧ Usage: “Cancel”, “Go Back”, “Retry”

- **Danger Button**:

  - ✧ Background: `#C33025` (Red)
  - ✧ Text: WhiteUsage: “Access Denied”, “Retry Scan”

  - ○

➢ **Icons**

- Simple line icons (Flat style, 24px standard size)
- Icons from **Material Icons** or **Feather Icons**

Examples:

- Attendance – calendar-check
- Location – map-pin
- Face ID – user-check
- Alerts – alert-circle

➢ **Fonts**

- **Header Font**: Poppins Bold
- **Body Font**: Inter Regular
- **Button Font**: Poppins Medium

➢ **Font Sizes**:

- Titles: 20–24px
- Labels/Subtitles: 16–18px

- Body Text: 14–16px
- Captions & Icons: 12–14px

3.2 Wireframes & Mockups

3.2.1 Low-Fidelity Wireframes

Low-fidelity wire-frames are used in the **early stages of UI design** to map out:

- The layout of each screen.
- Navigation paths.
- Placement of buttons, forms, and icons.
- **User experience flow** without distractions of colors or graphics.

Screen overview

1. Login/Sign-in Screen

Goal: Help users to securely access the app

Wireframe Elements:

- App logo (top center)
- Welcome text: "Login to AuraCheck"
- Input fields: Email/Username and Password
- “Forgot Password?” link
- **Login button** (Primary CTA)
- Option for role selection (e.g., "Student / Instructor")
- **Flow:** On successful login → Navigate to the relevant dashboard

2. Student Dashboard

Goal: Provide students with quick access to attendance tools and history

Wireframe Elements:

- Greeting header: "Welcome, [Student Name]"
- Quick status: "Next Class: [Course], Time"
- **Check-in Button** (large and central)
- Navigation bar: Home | History | Profile

- Notifications (optional bell icon)

Flow: Tap "Check-In" → Start facial + GPS scan

3. **Instructor Dashboard**

Goal: Help instructors monitor class attendance in real-time

Wireframe Elements:

- Greeting header: "Hello, Instructor [Name]"
- Course list dropdown (to select which class to view)
- **Today's Class Summary**: Total students, Checked-in, Absent
- Quick links/buttons:
    - View Attendance Logs
    - Manage Class Settings
- Sidebar menu: Dashboard | Courses | Logs | Logout

Flow: Select course → View student check-ins

4. Attendance Check-in (Face + GPS)
5. Attendance History
6. Admin Panel (Manage Courses, View Logs)

These wire-frames focus on simplicity, clear CTAs, and screen flow clarity.

3.2.2 High-Fidelity Mock-ups

High-fidelity mock-ups are helps to **polish detailed screen designs** that show exactly how the final app will look and feel. They include:

- **Branding** (logo, colors, typography)
- **UI elements** like real buttons, icons, and text
- **Screen layouts** that closely match what users will interact with
- **Interactive features** like button states, shadows, notifications, and animations

These mock-ups are used for:

- Developer hand-off
- Stakeholder approval
- Prototyping real user experien

3.3 Screen Layouts & Navigation Flow

➢ **Main Screens:**

- **Login Screen**
- **Home Dashboard** (Dynamic — student or instructor)
- **Attendance Check-In**
- Face Scan → GPS Check → Result
- **History View**
- Filter by date/course
- **Admin/Instructor Panel**

✧ Course List → Student Attendance Overview

3.4 Accessibility Considerations (Contrast, Font Size, etc.)

Color Contrast

- ➢ All text contrasts meet **WCAG 2.1** standards:
- ➢ Primary text: Black #212121 on white
- ➢ Buttons: White on blue (#0053AD) and green (#115B43)
- ➢ Warnings/errors: Yellow and red with black or white text

Text Size & Readability

- ➢ Minimum font size: **14px**
- ➢ No dense text blocks
- ➢ **Line spacing** maintained for readability

Iconography

- ➢ All icons have text labels

- High-contrast icons for color-blind users

Touch Targets

- Buttons and icons are at least **48x48dp**
- Form inputs spaced for finger accessibility

Feedback & Animation

- Screen reader friendly (using ARIA labels or equivalents)
- Visual cues: color + icon + text for every status message

4.1 Technology Stack:

Technology Chosen: Flutter.

Definition:

Flutter is an open-source, cross-platform framework developed by Google. It allows developers to build natively compiled applications for mobile, web, and desktop from a single codebase.

Why Flutter?

Cross-platform compatibility:Flutter enables the development of applications that run seamlessly on multiple platforms, including Android, iOS, web, and desktop.

Fast development: Flutter's hot reload feature allows developers to see the changes they make to the codebase in real-time, without requiring a full rebuild.

Native performance: Flutter applications are compiled to native code, providing fast and efficient performance.

4.2 Key Screens & Components

The following key screens and components have been implemented:

For the implementation of our app, the following was been implemented:

- ✓ Login/Registration screens: Custom-designed login and registration screens with form validation and error handling for all users in the system.
- ✓ Dashboard/Homepage: Each user has his/her own reponsive dashboard depending on the user logged into the system. A responsive dashboard with navigation menu, providing easy access to main features
- ✓ Navigation menu: A customizable navigation menu with smooth transitions and animations.

- ✓ Facial Recognition Menu: Guided the students on how biometric data is to be collected.
- ✓ Profile Summary: Displays user characteristics on the system.

Also, some specific components of our users where implemented as follows:

1. Admininistrator:

- ❖ *Management of Users*
- ❖ *Management of Classes/Courses*
- ❖ *Define Locations*

2. Student:

- ❖ *Register for courses.*
- ❖ *View attendant history.*
- ❖ *Facial Enrollment.*
- ❖ *Check-In Location*

3. Instructor:

- ❖ *View Courses/Report*
- ❖ *History*
- ❖ *Edit User Profile*
- ❖ *Filtering*

4.3 Responsive Design (Multi-Device Adaptability)

So far, we have tested our application on just a single platform (Android), but the implementation shall ensure a responsive design, allowing for seamless adaptability across various devices, including:

- ■ ***Desktop*s:** Optimized for large screens with multiple columns and rows.
- ■ ***Laptops***: Adapted for smaller screens with adjustable columns and rows.
- ■ ***Tablets***: Optimized for touch-based interfaces with larger tap targets.
- ■ ***Mobile phones***: Adapted for small screens with single-column layouts.

4.4 User Interaction & Animations.

The implementation incorporates engaging user interactions and animations, including:

- ❖ Smooth navigation transitions: Animated transitions between screens and pages.
- ❖ Button hover effects: Visual effects on buttons when hovered or pressed.
- ❖ Audit Logging Validation: The system takes into consideration Real-time loggings of each action

performed by either of the Actors on the system.

4.5 Testing & Debugging (UI/UX Validation).

We were able to test our UI/UX design using

A. *Unit Testing*: Testing individual components or units of code.

B. *Acceptance Testing*: Testing to ensure the software meets the requirements.

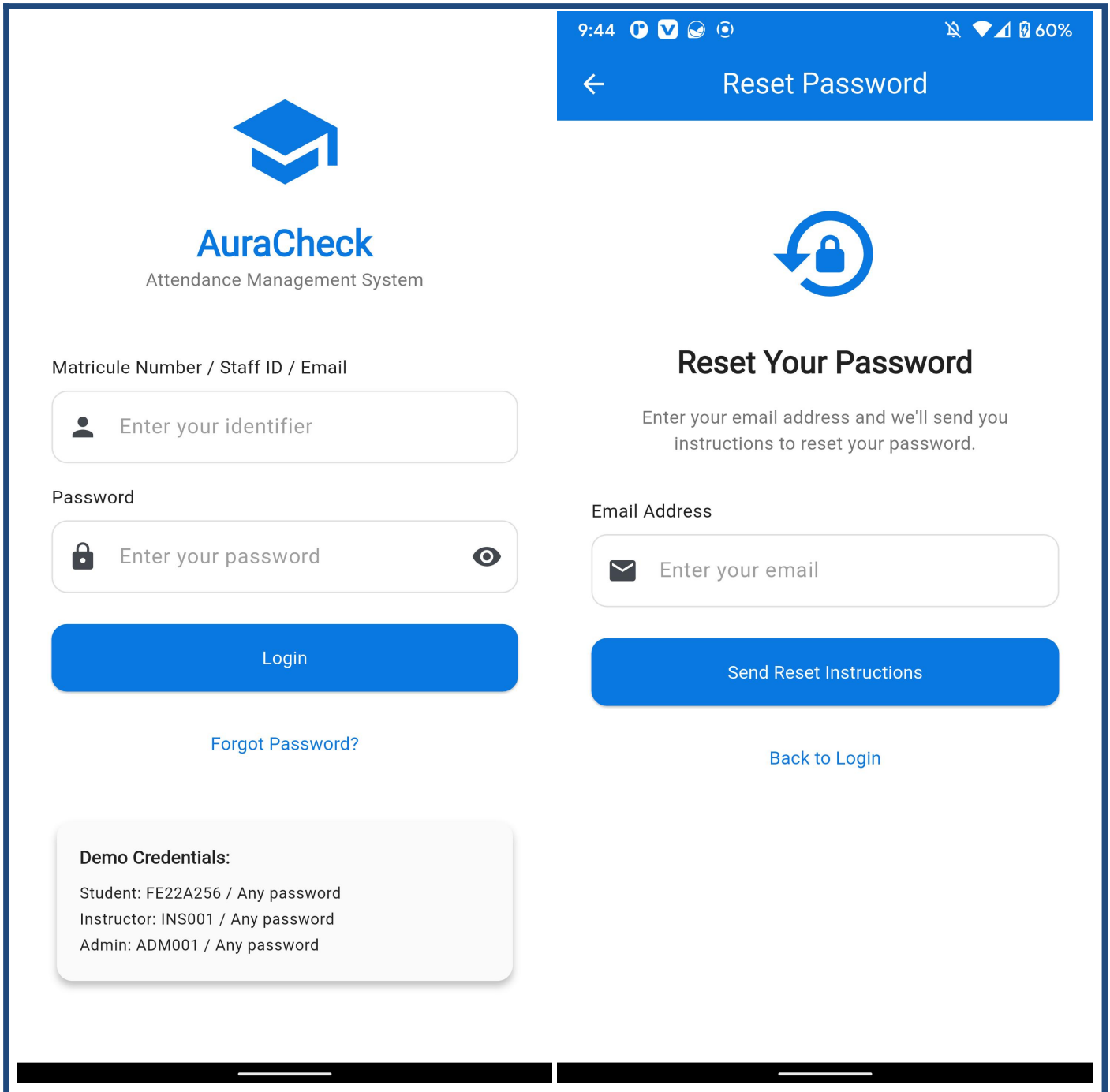
C. *Regression Testing*: Testing to ensure that changes in our system has not introduced new bugs (errors).

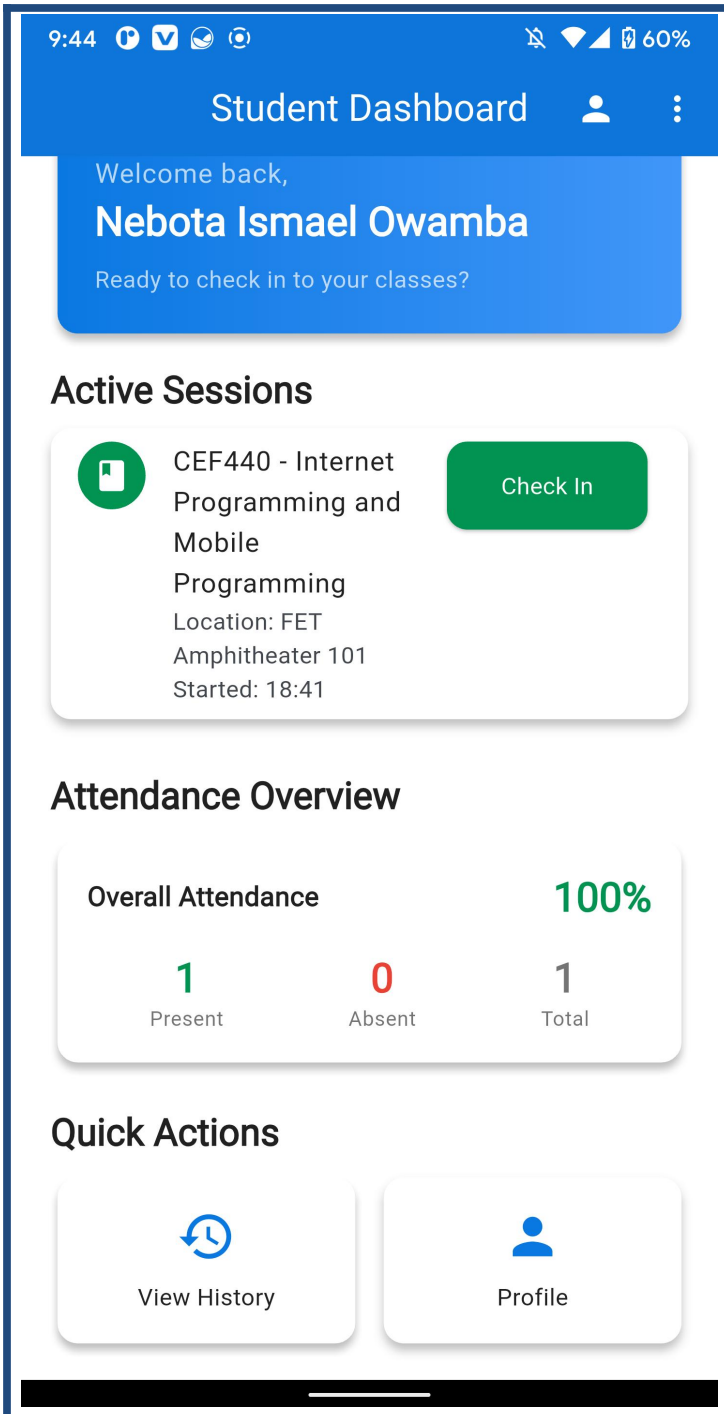
Thorough testing and debugging have been conducted to ensure UI/UX validation, including:

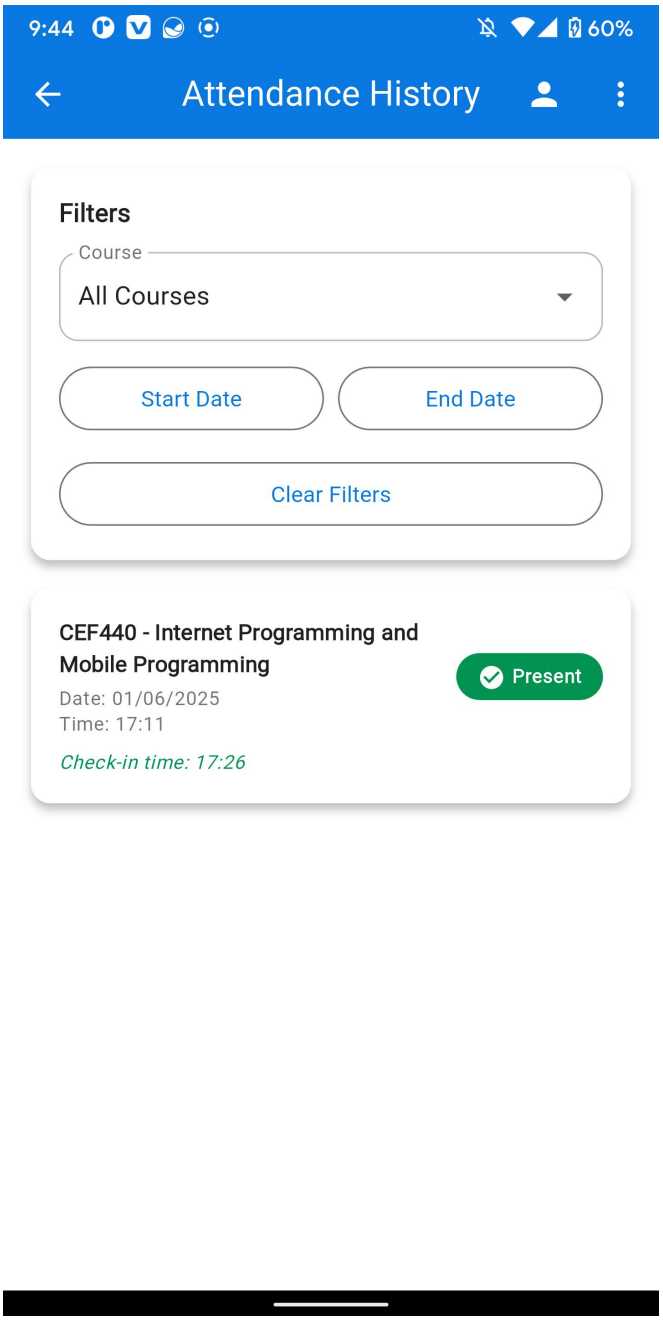
- Device responsiveness testing: Testing on various devices to ensure responsiveness.
- User interaction testing: Testing user interactions, such as button clicks and form submissions.

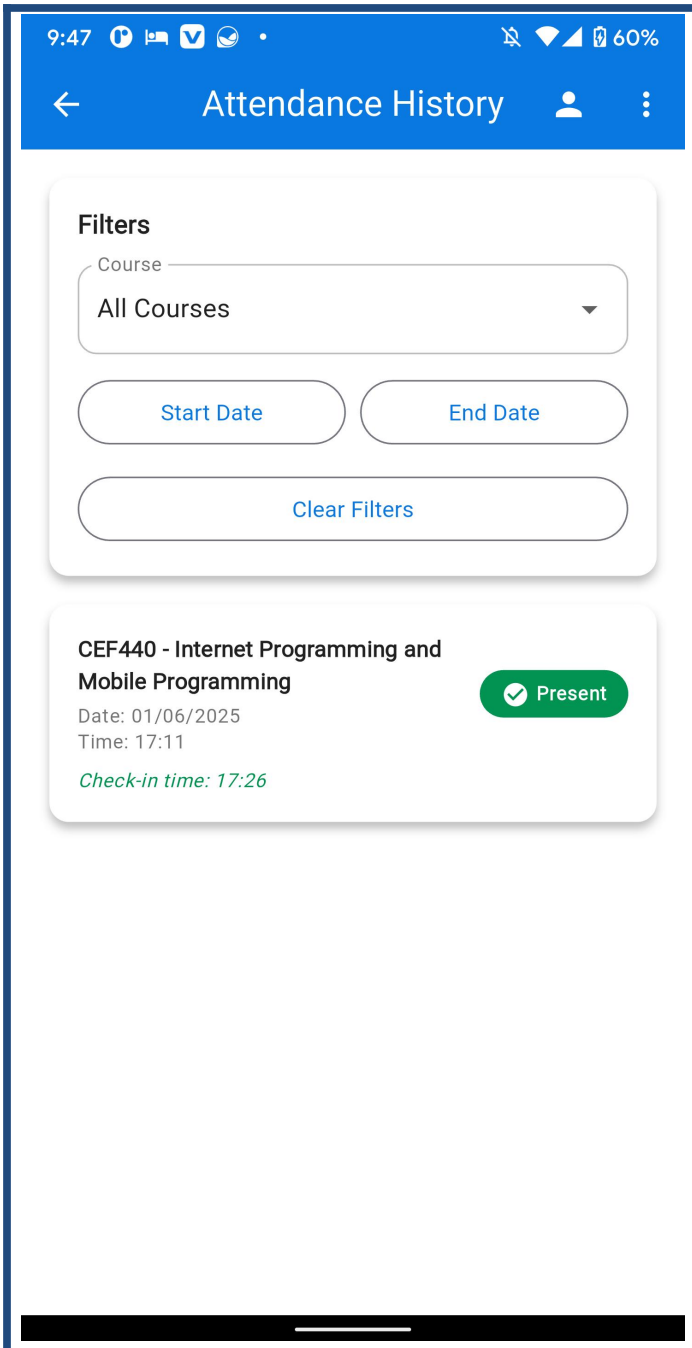
5. Technical Architecture

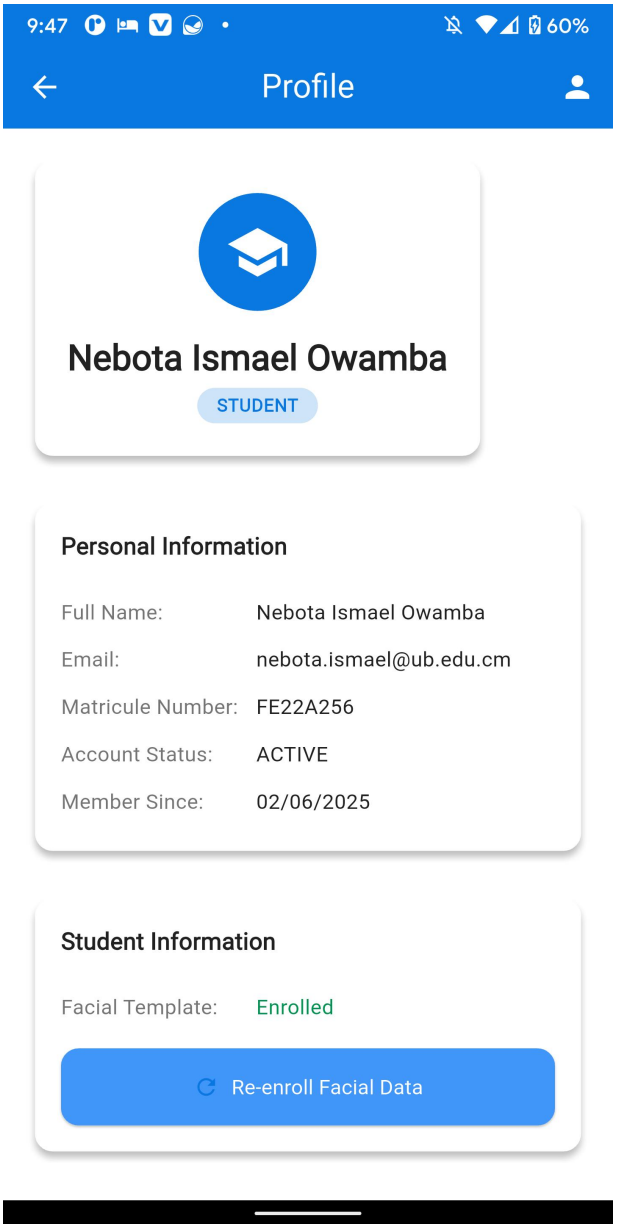
5.1 Technology Stack

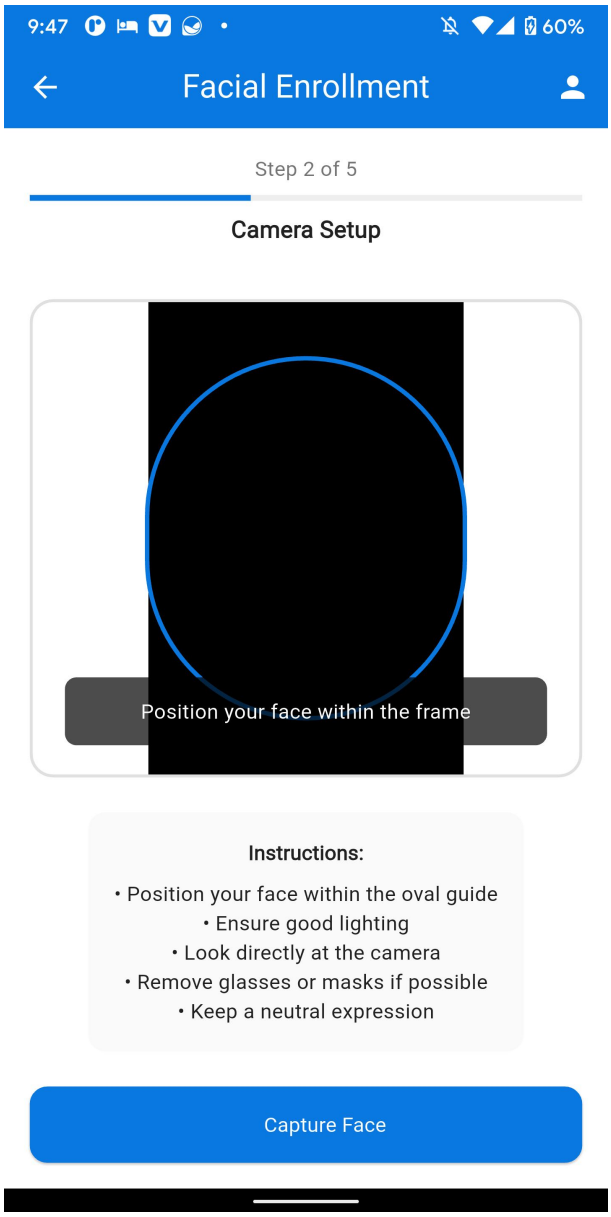

AuraCheck
Attendance Management System
Matricule Number / Staff ID / Email
Enter your identifier
Password
Enter your password
Login
Forgot Password?
Demo Credentials:
Student: FE22A256 / Any password
Instructor: INS001 / Any password
Admin: ADM001 / Any password
9:44
60%
Reset Password
Reset Your Password
Enter your email address and we'll send you instructions to reset your password.
Email Address
Enter your email
Send Reset Instructions
Back to Login

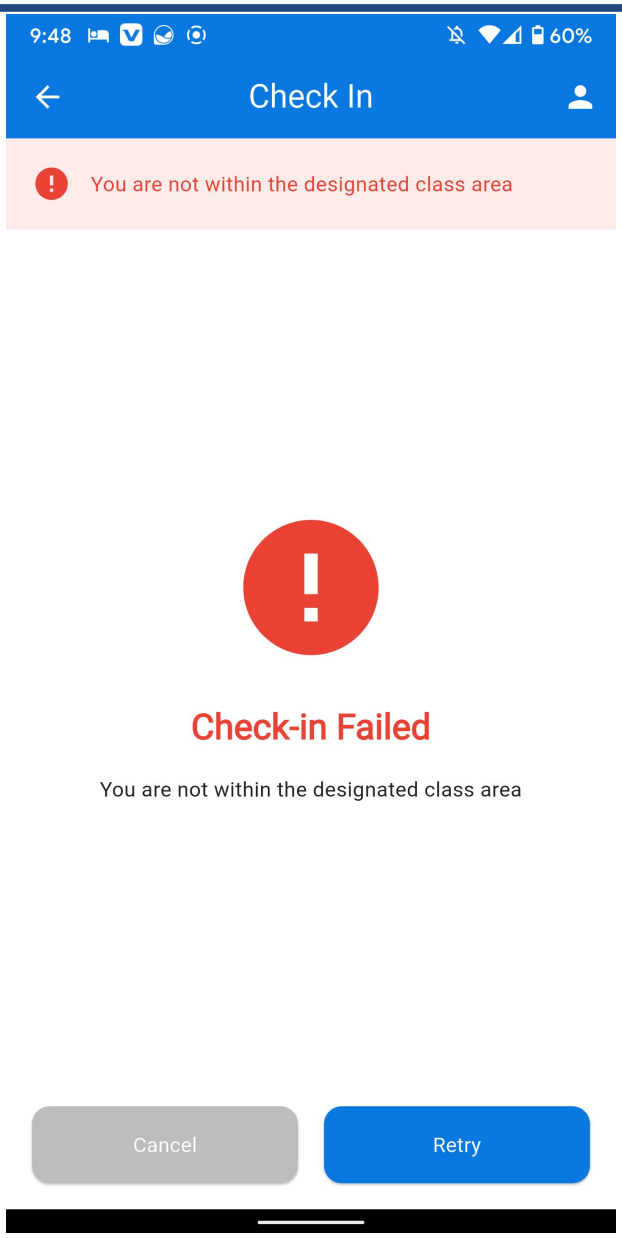

9:44
60%
Student Dashboard
Welcome back,
Nebota Ismael Owamba
Ready to check in to your classes?
Active Sessions
CEF440 - Internet Programming and Mobile Programming
Location: FET Amphitheater 101
Started: 18:41
Check In
Attendance Overview
Overall Attendance
100%
1
Present
0
Absent
1
Total
Quick Actions
View History
Profile

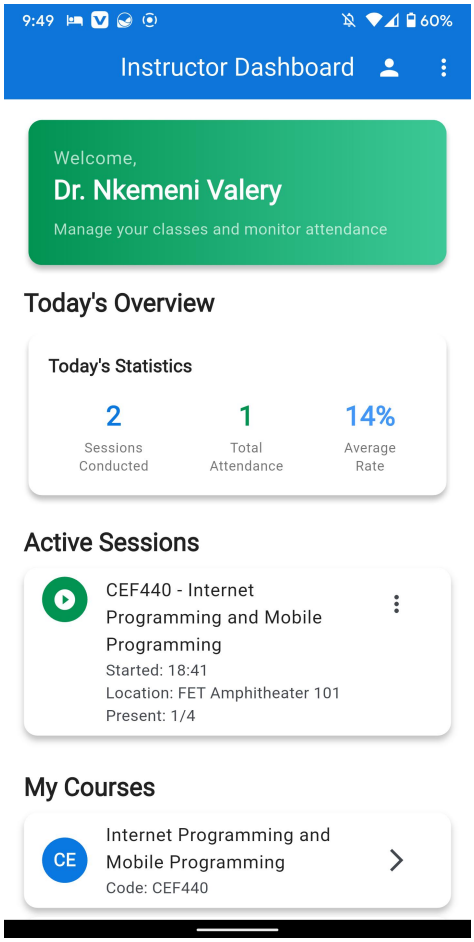

9:44
60%
Attendance History
Filters
Course
All Courses
Start Date
End Date
Clear Filters
CEF440 - Internet Programming and Mobile Programming
Date: 01/06/2025
Time: 17:11
Present
Check-in time: 17:26

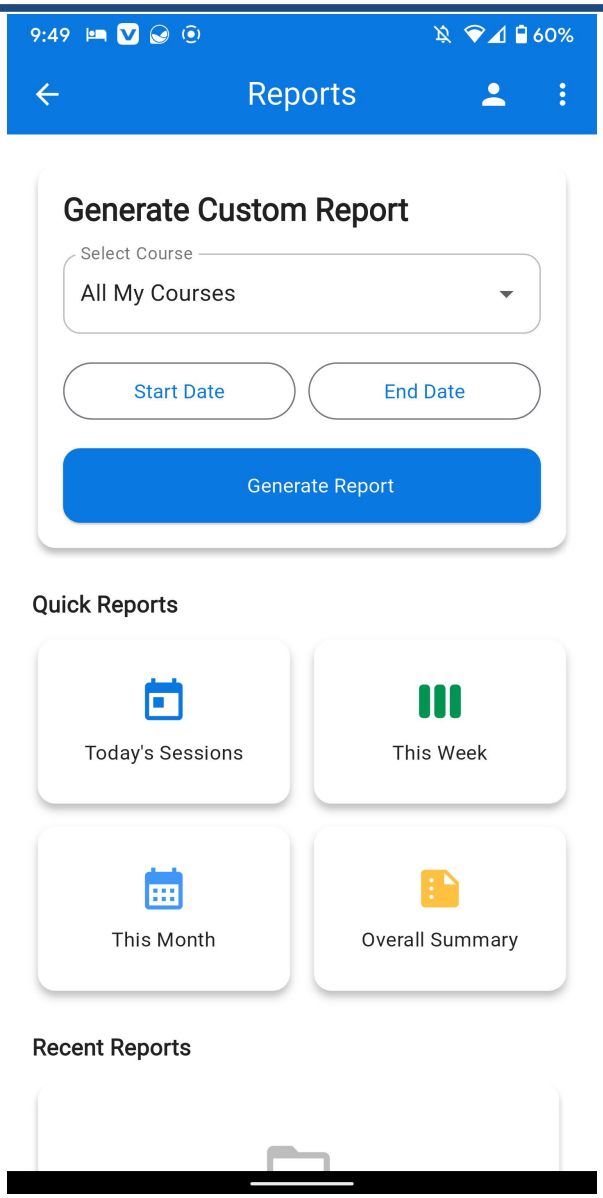

9:47
60%
Attendance History
Filters
Course
All Courses
Start Date
End Date
Clear Filters
CEF440 - Internet Programming and Mobile Programming
Present
Date: 01/06/2025
Time: 17:11
Check-in time: 17:26

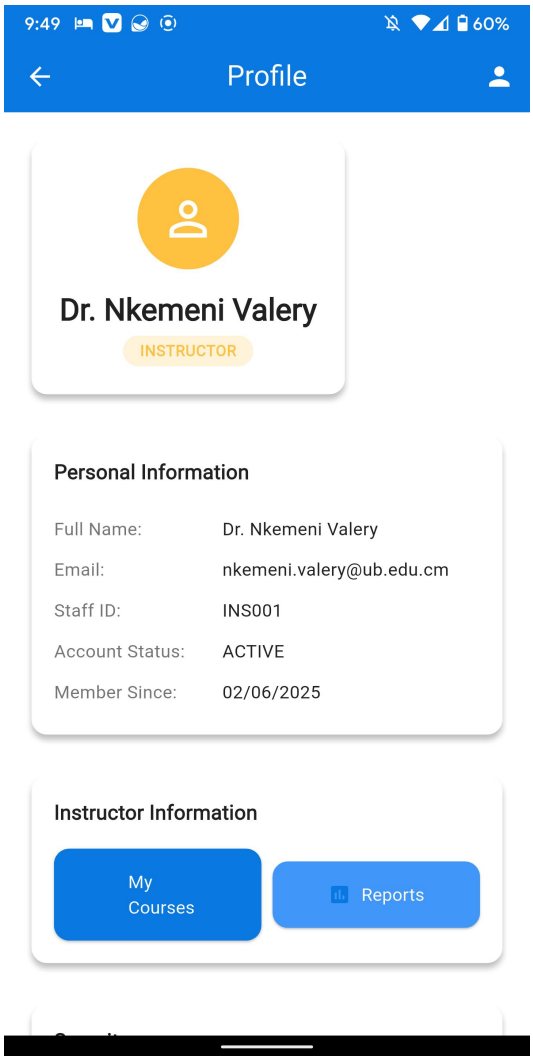

9:47
60%
Profile
Nebota Ismael Owamba
STUDENT
Personal Information
Full Name: Nebota Ismael Owamba
Email: nebota.ismael@ub.edu.cm
Matricule Number: FE22A256
Account Status: ACTIVE
Member Since: 02/06/2025
Student Information
Facial Template: Enrolled
Re-enroll Facial Data

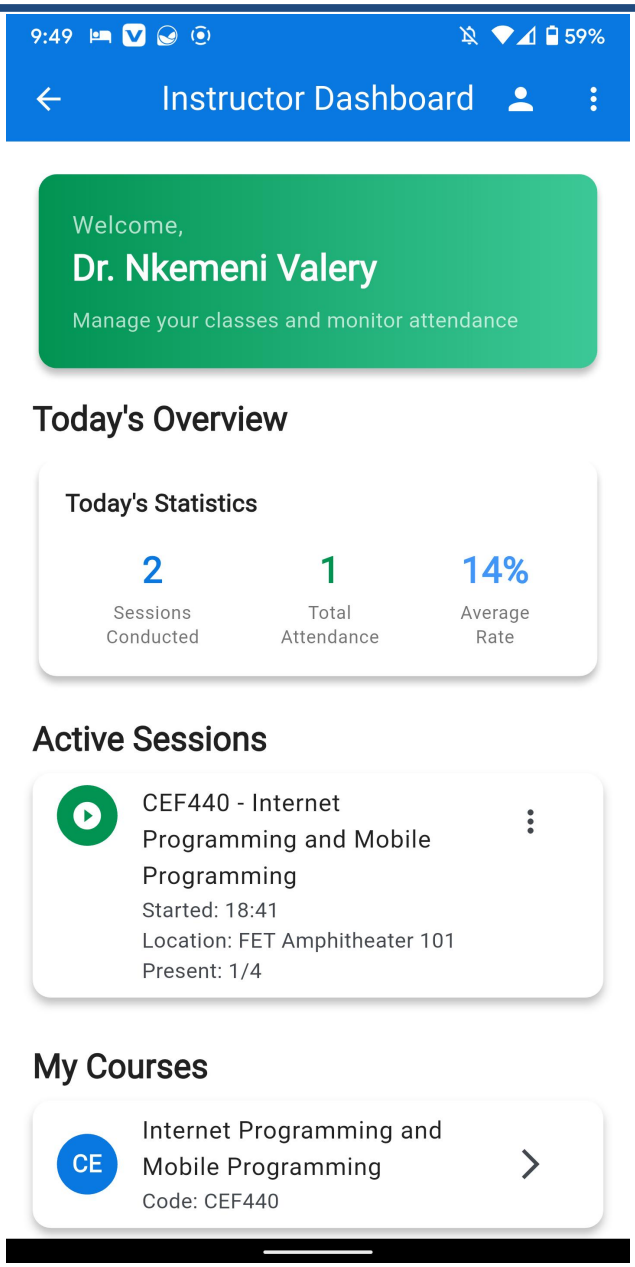

9:47
60%
Facial Enrollment
Step 2 of 5
Camera Setup
Position your face within the frame
Instructions:
• Position your face within the oval guide
• Ensure good lighting
• Look directly at the camera
• Remove glasses or masks if possible
• Keep a neutral expression
Capture Face

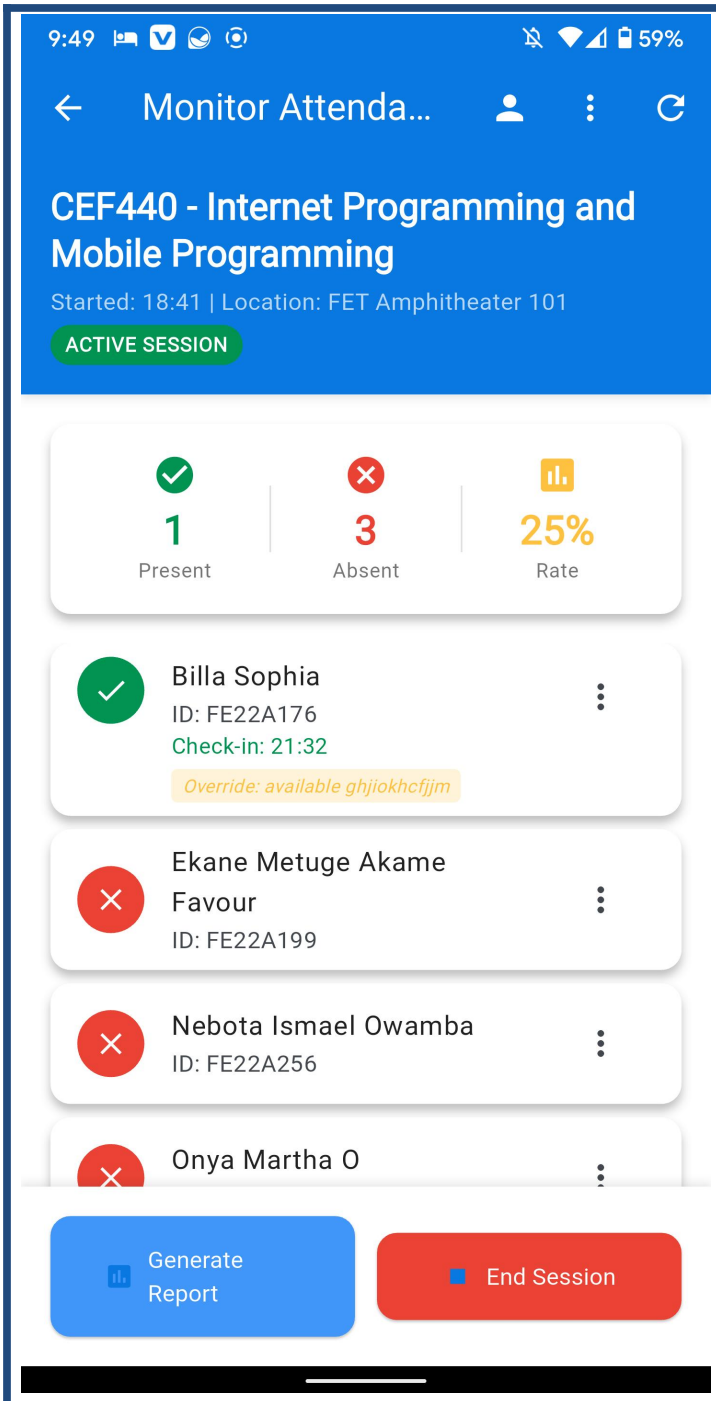

9:48
60%
Check In
You are not within the designated class area
Check-in Failed
You are not within the designated class area
Cancel
Retry

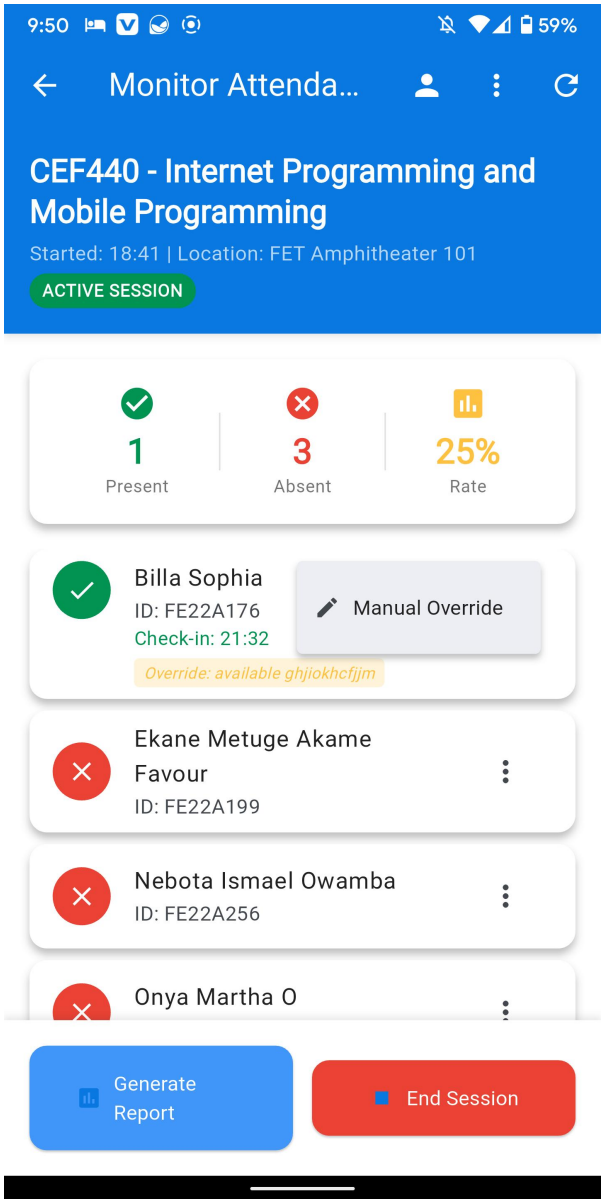

9:49
60%
Instructor Dashboard
Welcome,
Dr. Nkemeni Valery
Manage your classes and monitor attendance
Today's Overview
Today's Statistics
2
Sessions Conducted
1
Total Attendance
14%
Average Rate
Active Sessions
CEF440 - Internet Programming and Mobile Programming
Started: 18:41
Location: FET Amphitheater 101
Present: 1/4
My Courses
CE
Internet Programming and Mobile Programming
Code: CEF440

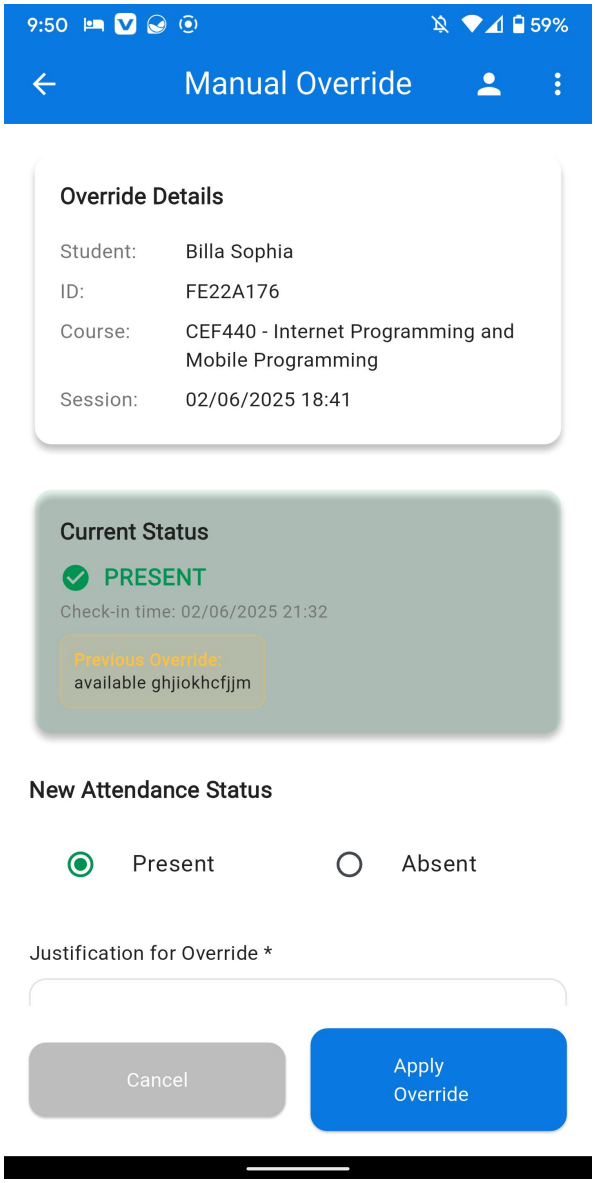

9:49
60%
Reports
Generate Custom Report
Select Course
All My Courses
Start Date
End Date
Generate Report
Quick Reports
Today's Sessions
This Week
This Month
Overall Summary
Recent Reports

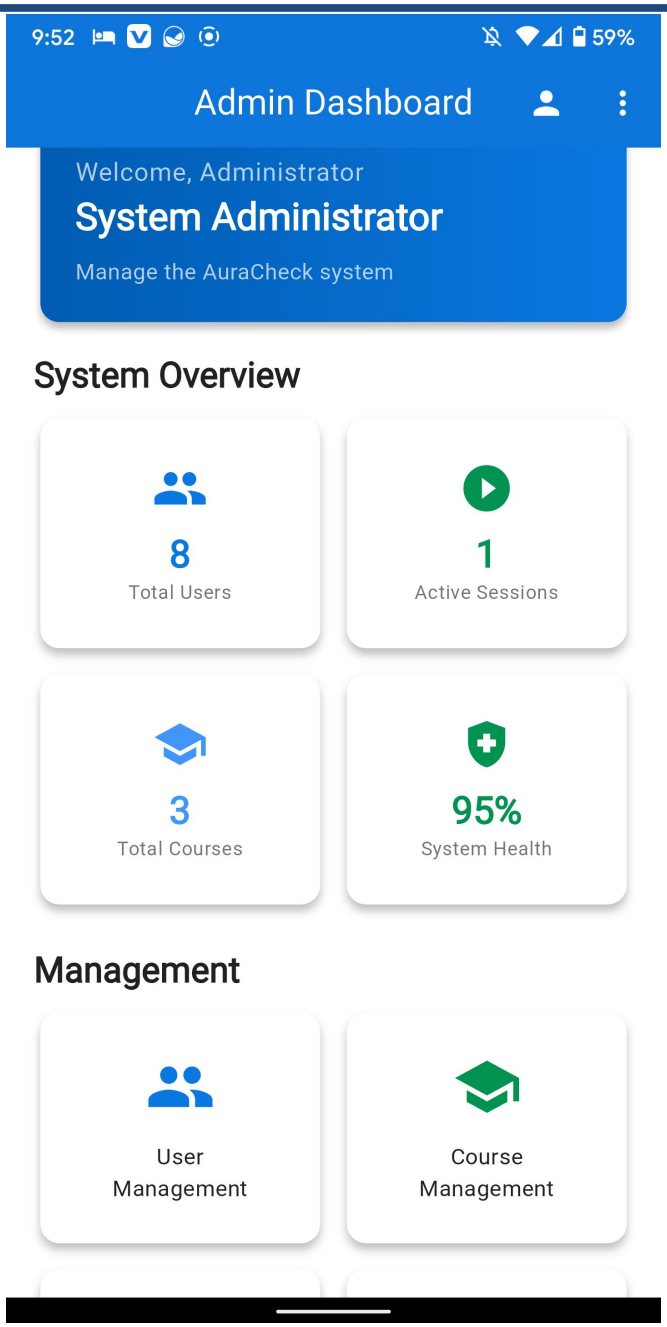

9:49
60%
Profile
Dr. Nkemeni Valery
INSTRUCTOR
Personal Information
Full Name: Dr. Nkemeni Valery
Email: nkemeni.valery@ub.edu.cm
Staff ID: INS001
Account Status: ACTIVE
Member Since: 02/06/2025
Instructor Information
My Courses
Reports

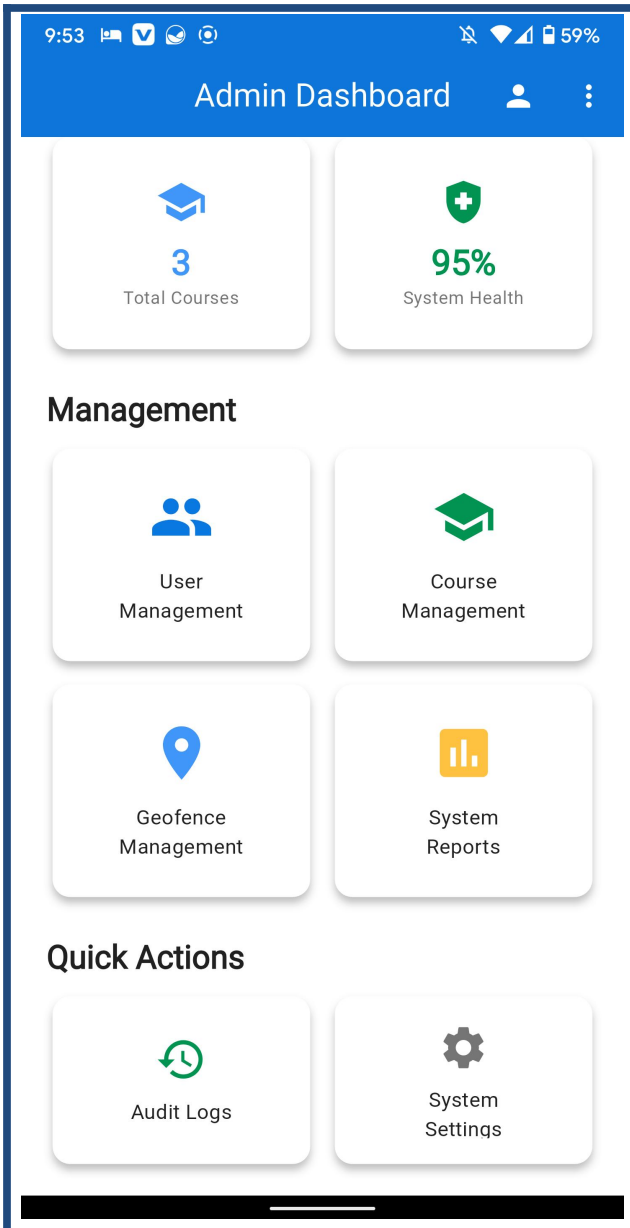

9:49
59%
Instructor Dashboard
Welcome,
Dr. Nkemeni Valery
Manage your classes and monitor attendance
Today's Overview
Today's Statistics
2
Sessions Conducted
1
Total Attendance
14%
Average Rate
Active Sessions
CEF440 - Internet Programming and Mobile Programming
Started: 18:41
Location: FET Amphitheater 101
Present: 1/4
My Courses
CE
Internet Programming and Mobile Programming
Code: CEF440

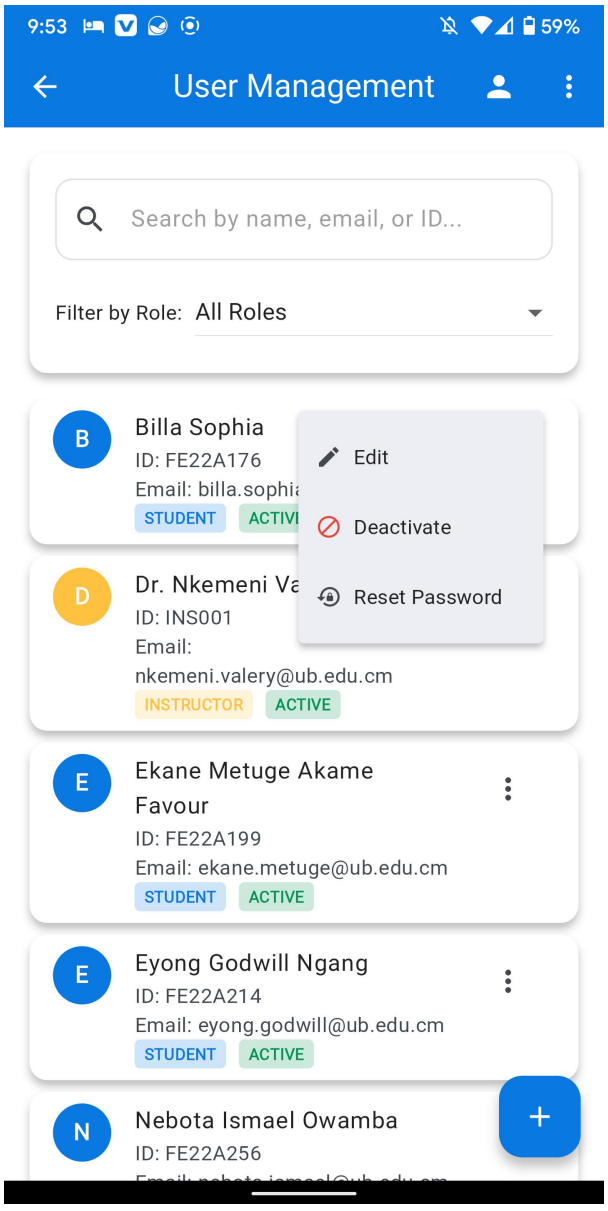

9:49
59%
Monitor Attenda...
CEF440 - Internet Programming and Mobile Programming
Started: 18:41 | Location: FET Amphitheater 101
ACTIVE SESSION
1
Present
3
Absent
25%
Rate
Billa Sophia
ID: FE22A176
Check-in: 21:32
Override: available ghjiokhcfjjm
Ekane Metuge Akame Favour
ID: FE22A199
Nebota Ismael Owamba
ID: FE22A256
Onya Martha O
Generate Report
End Session

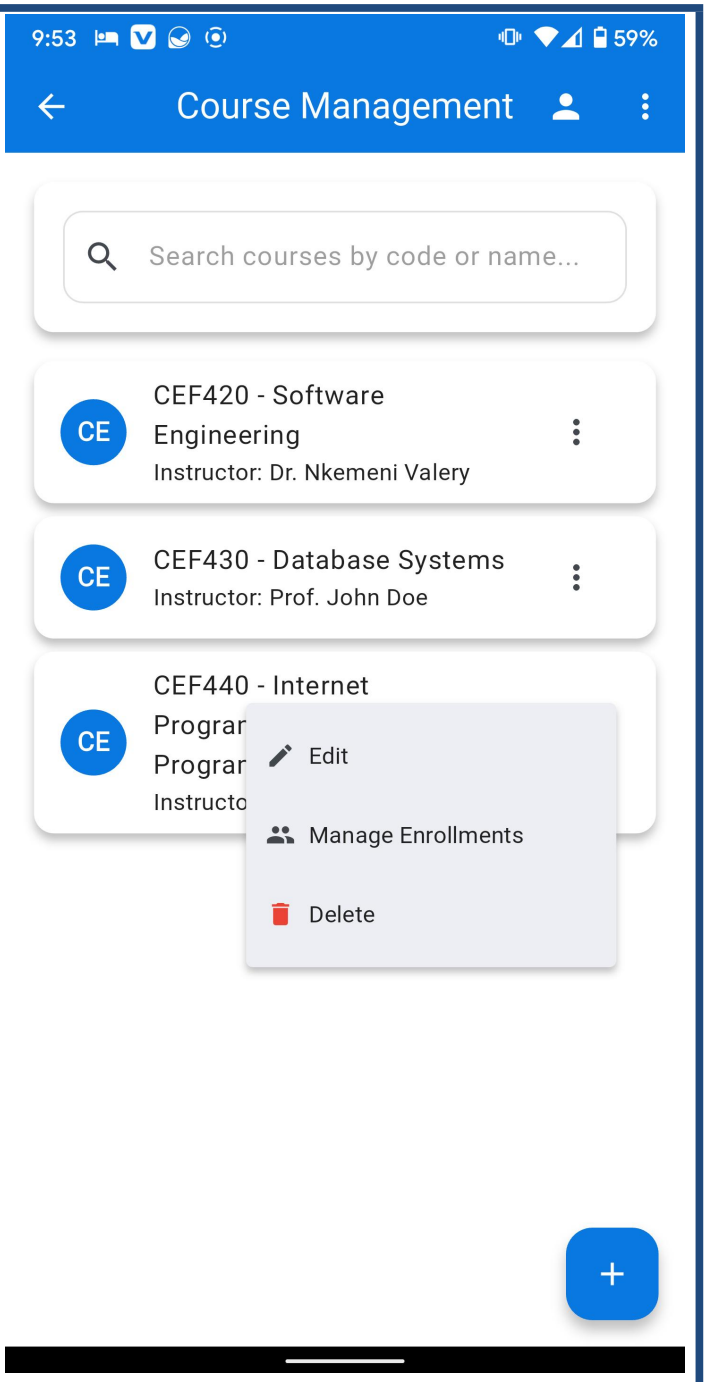

9:50
59%
Monitor Attenda...
CEF440 - Internet Programming and Mobile Programming
Started: 18:41 | Location: FET Amphitheater 101
ACTIVE SESSION
1
Present
3
Absent
25%
Rate
Billa Sophia
ID: FE22A176
Check-in: 21:32
Override: available ghjiokhcfjjm
Manual Override
Ekane Metuge Akame Favour
ID: FE22A199
Nebota Ismael Owamba
ID: FE22A256
Onya Martha O
Generate Report
End Session

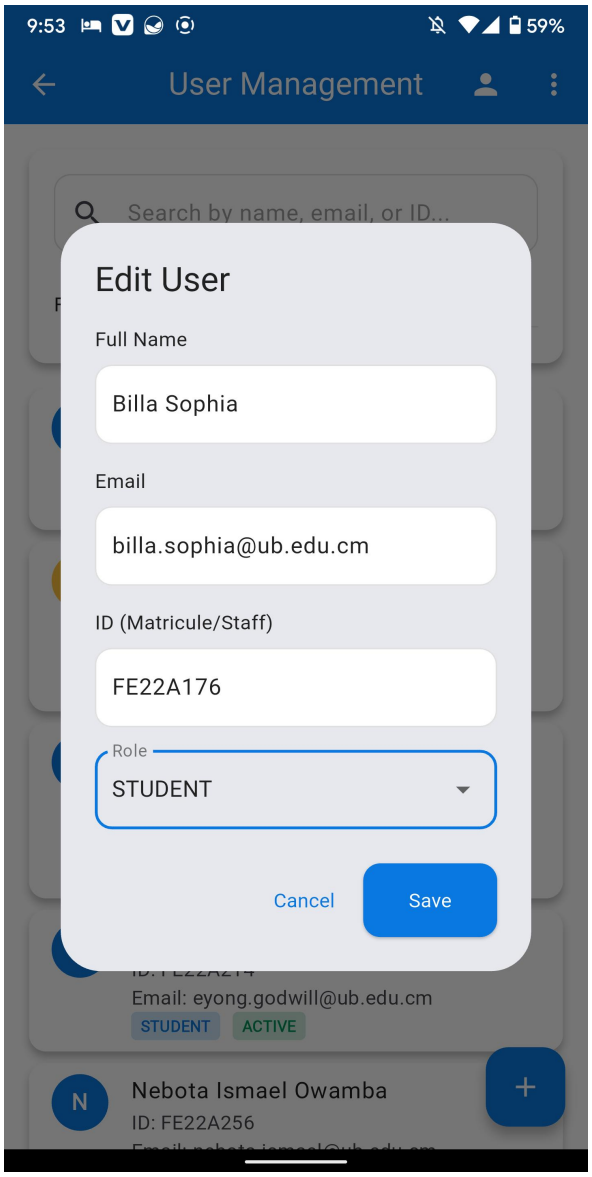

9:50
59%
Manual Override
Override Details
Student: Billa Sophia
ID: FE22A176
Course: CEF440 - Internet Programming and Mobile Programming
Session: 02/06/2025 18:41
Current Status
PRESENT
Check-in time: 02/06/2025 21:32
Previous Override:
available ghjiokhcfjjm
New Attendance Status
Present
Absent
Justification for Override *
Cancel
Apply Override

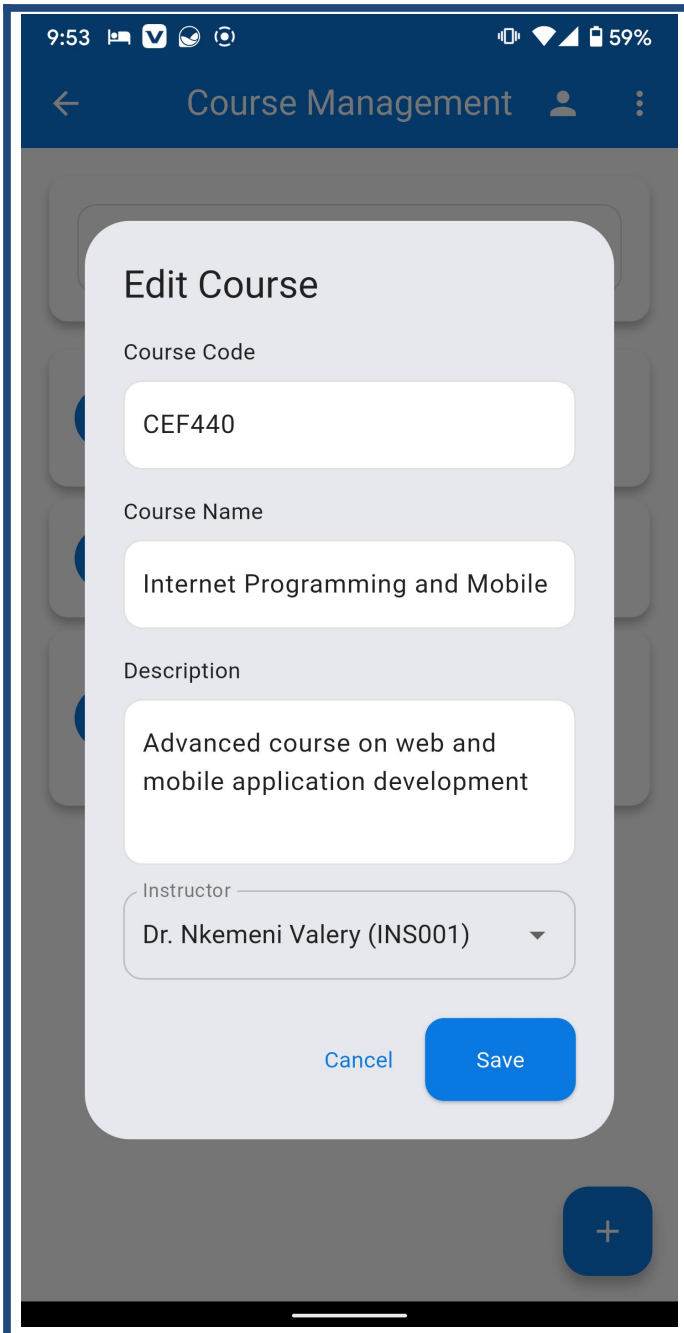

9:52
59%
Admin Dashboard
Welcome, Administrator
System Administrator
Manage the AuraCheck system
System Overview
8
Total Users
1
Active Sessions
3
Total Courses
95%
System Health
Management
User Management
Course Management

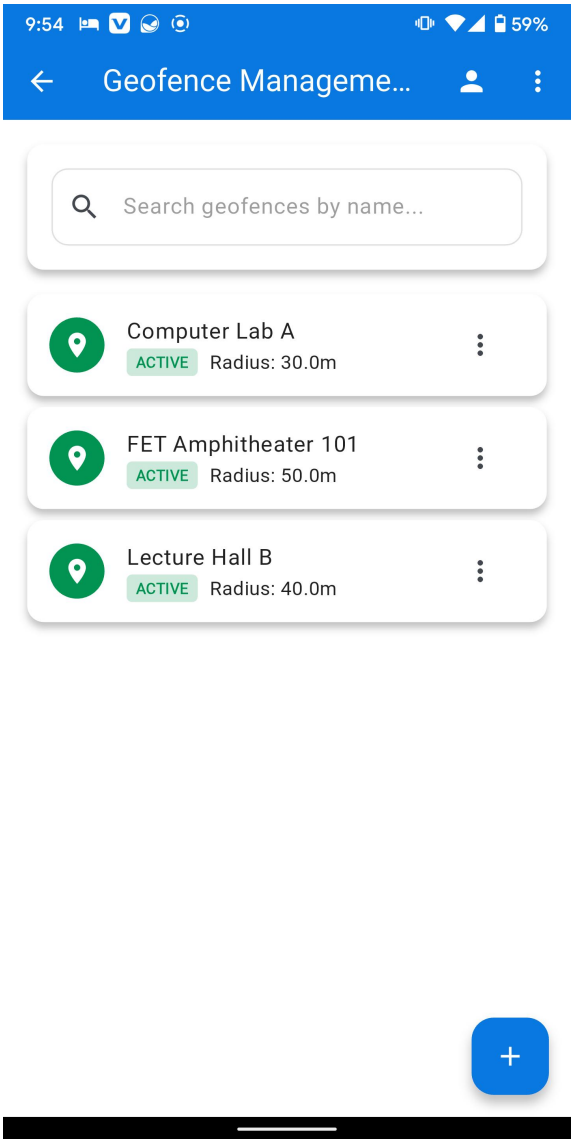

9:53
59%
Admin Dashboard
3
Total Courses
95%
System Health
Management
User Management
Course Management
Geofence Management
System Reports
Quick Actions
Audit Logs
System Settings

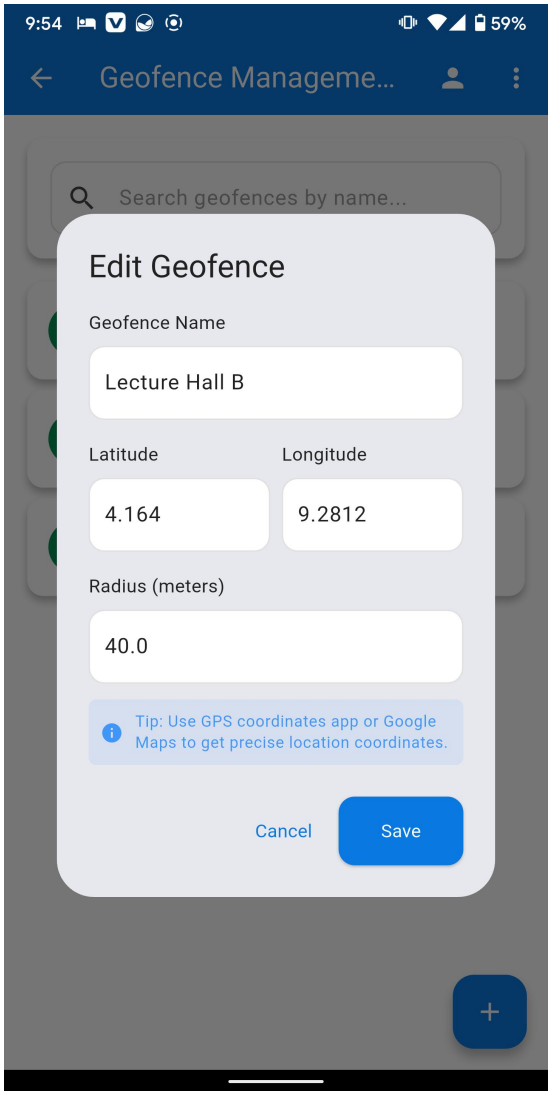

9:53
59%
User Management
Search by name, email, or ID...
Filter by Role: All Roles
B
Billa Sophia
ID: FE22A176
Email: billa.sophia
STUDENT
ACTIVE
Edit
Deactivate
Reset Password
D
Dr. Nkemeni Va
ID: INS001
Email: nkemeni.valery@ub.edu.cm
INSTRUCTOR
ACTIVE
E
Ekane Metuge Akame Favour
ID: FE22A199
Email: ekane.metuge@ub.edu.cm
STUDENT
ACTIVE
E
Eyong Godwill Ngang
ID: FE22A214
Email: eyong.godwill@ub.edu.cm
STUDENT
ACTIVE
N
Nebota Ismael Owamba
ID: FE22A256

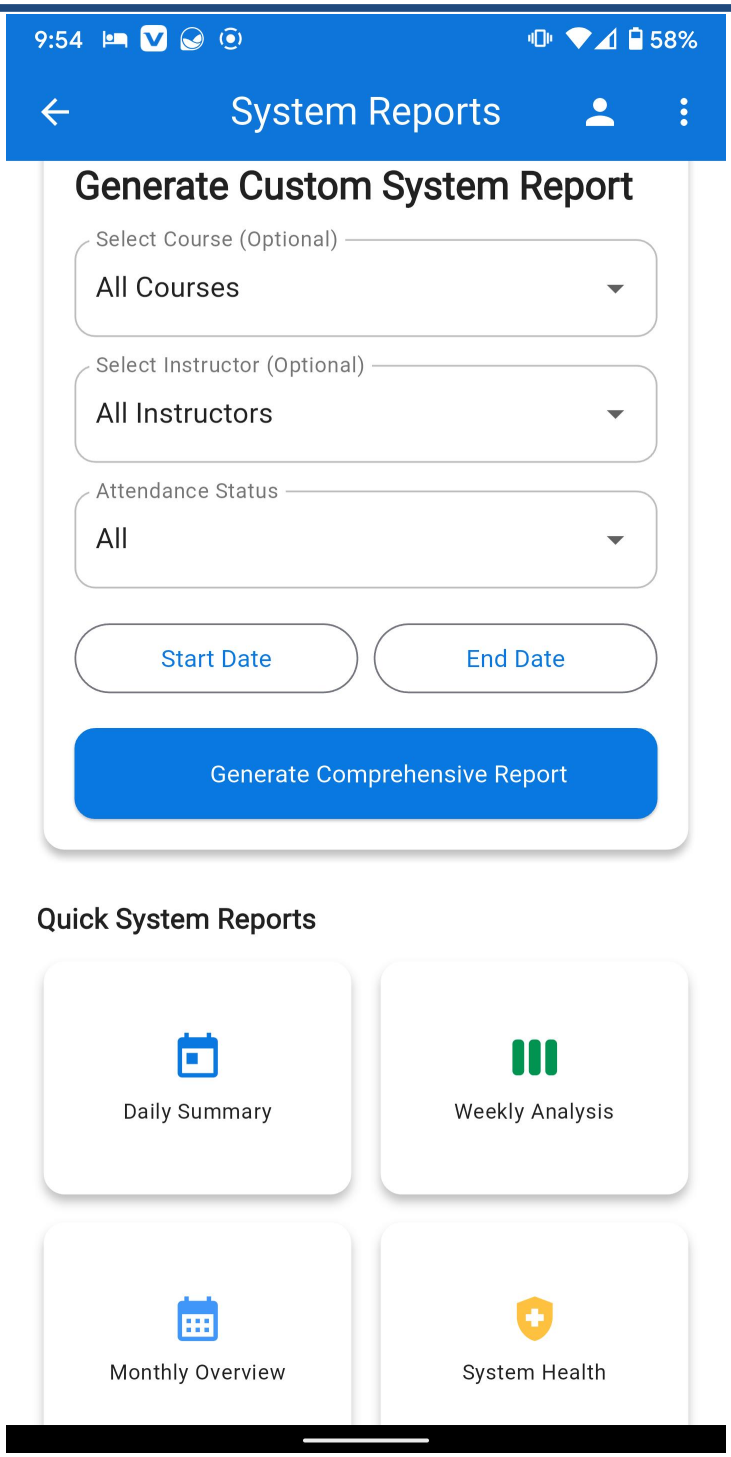

9:53
59%
Course Management
Search courses by code or name...
CE
CEF420 - Software Engineering
Instructor: Dr. Nkemeni Valery
CE
CEF430 - Database Systems
Instructor: Prof. John Doe
CE
CEF440 - Internet
Edit
Manage Enrollments
Delete

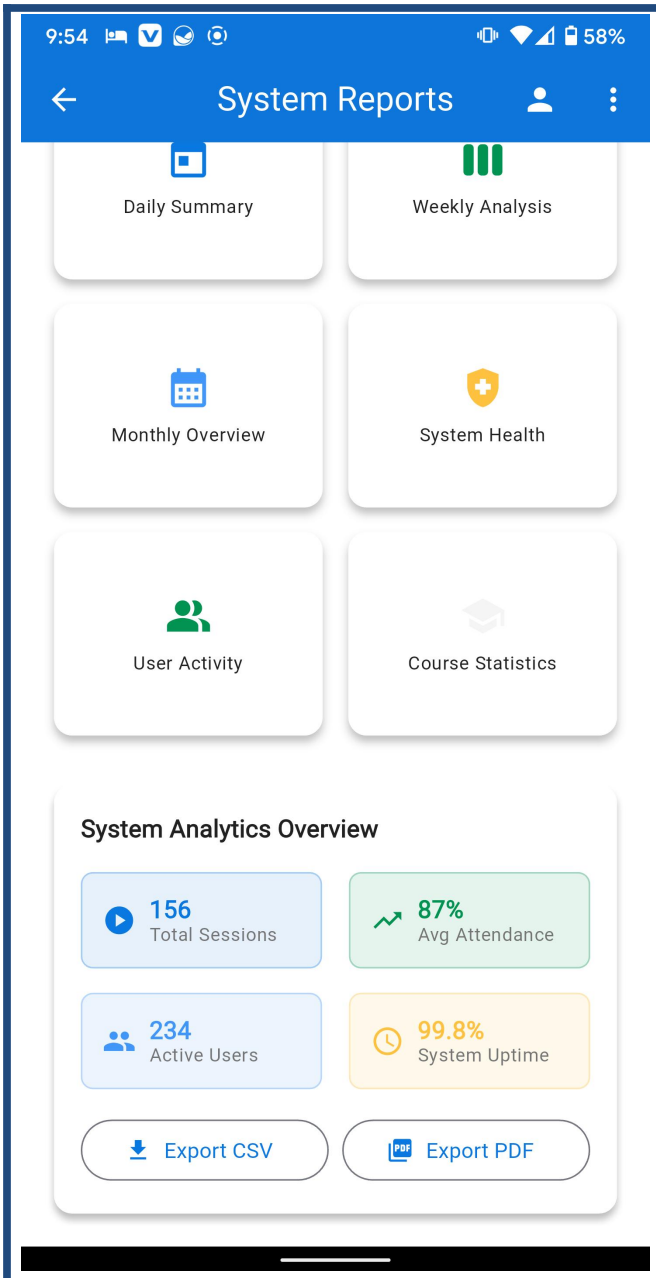

9:53
59%
User Management
Search by name, email, or ID...
Edit User
Full Name
Billa Sophia
Email
billa.sophia@ub.edu.cm
ID (Matricule/Staff)
FE22A176
Role
STUDENT
Cancel
Save
Email: eyong.godwill@ub.edu.cm
STUDENT
ACTIVE
N
Nebota Ismael Owamba
ID: FE22A256

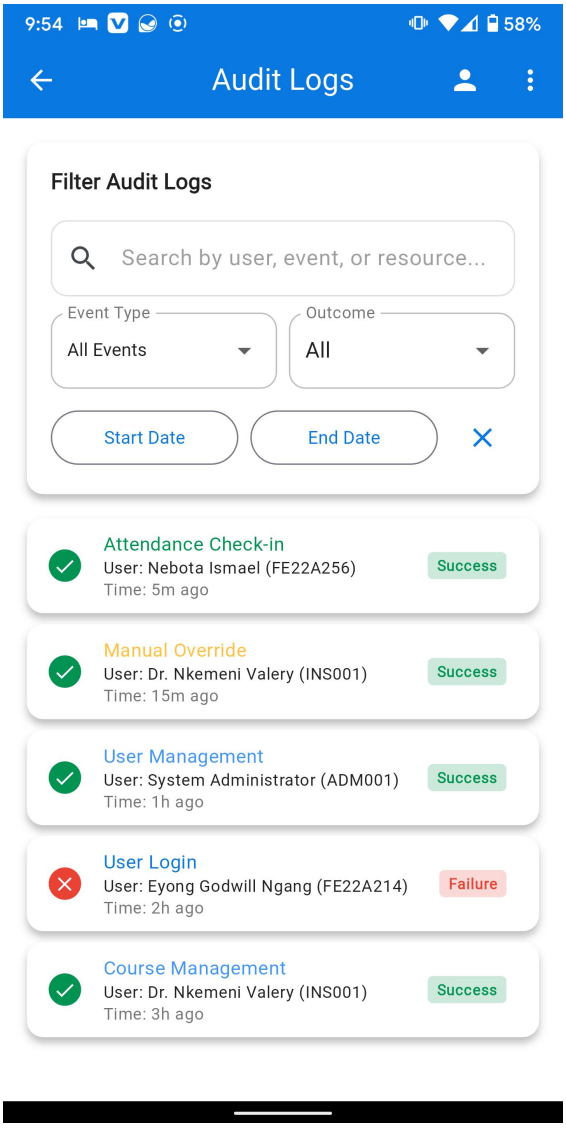

9:53
59%
Course Management
Edit Course
Course Code
CEF440
Course Name
Internet Programming and Mobile
Description
Advanced course on web and mobile application development
Instructor
Dr. Nkemeni Valery (INS001)
Cancel
Save

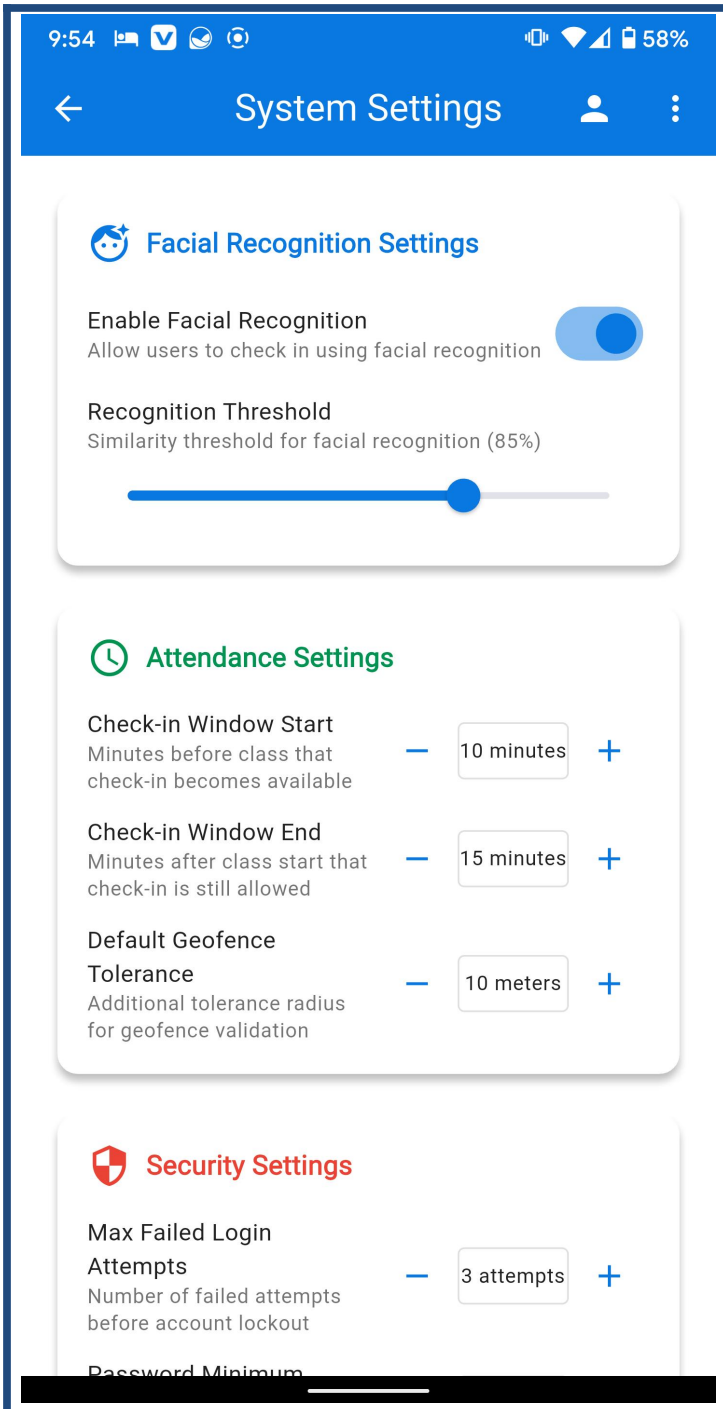

9:54
59%
Geofence Manageme...
Search geofences by name...
Computer Lab A
ACTIVE
Radius: 30.0m
FET Amphitheater 101
ACTIVE
Radius: 50.0m
Lecture Hall B
ACTIVE
Radius: 40.0m


9:54
59%
Geofence Manageme...
Search geofences by name...
Edit Geofence
Geofence Name
Lecture Hall B
Latitude
Longitude
4.164
9.2812
Radius (meters)
40.0
Tip: Use GPS coordinates app or Google Maps to get precise location coordinates.
Cancel
Save


9:54
58%
System Reports
Generate Custom System Report
Select Course (Optional)
All Courses
Select Instructor (Optional)
All Instructors
Attendance Status
All
Start Date
End Date
Generate Comprehensive Report
Quick System Reports
Daily Summary
Weekly Analysis
Monthly Overview
System Health


9:54
58%
System Reports
Daily Summary
Weekly Analysis
Monthly Overview
System Health
User Activity
Course Statistics
System Analytics Overview
156
Total Sessions
87%
Avg Attendance
234
Active Users
99.8%
System Uptime
Export CSV
Export PDF


9:54
58%
Audit Logs
Filter Audit Logs
Search by user, event, or resource...
Event Type
All Events
Outcome
All
Start Date
End Date
Attendance Check-in
User: Nebota Ismael (FE22A256)
Time: 5m ago
Success
Manual Override
User: Dr. Nkemeni Valery (INS001)
Time: 15m ago
Success
User Management
User: System Administrator (ADM001)
Time: 1h ago
Success
User Login
User: Eyong Godwill Ngang (FE22A214)
Time: 2h ago
Failure
Course Management
User: Dr. Nkemeni Valery (INS001)
Time: 3h ago
Success



Framework: Flutter 3.1.0+ (Dart 3.1.0+)

State Management: Riverpod with code generation

Local Storage: Hive with type adapters

Navigation: GoRouter with authentication guards

UI Framework: Material Design 3

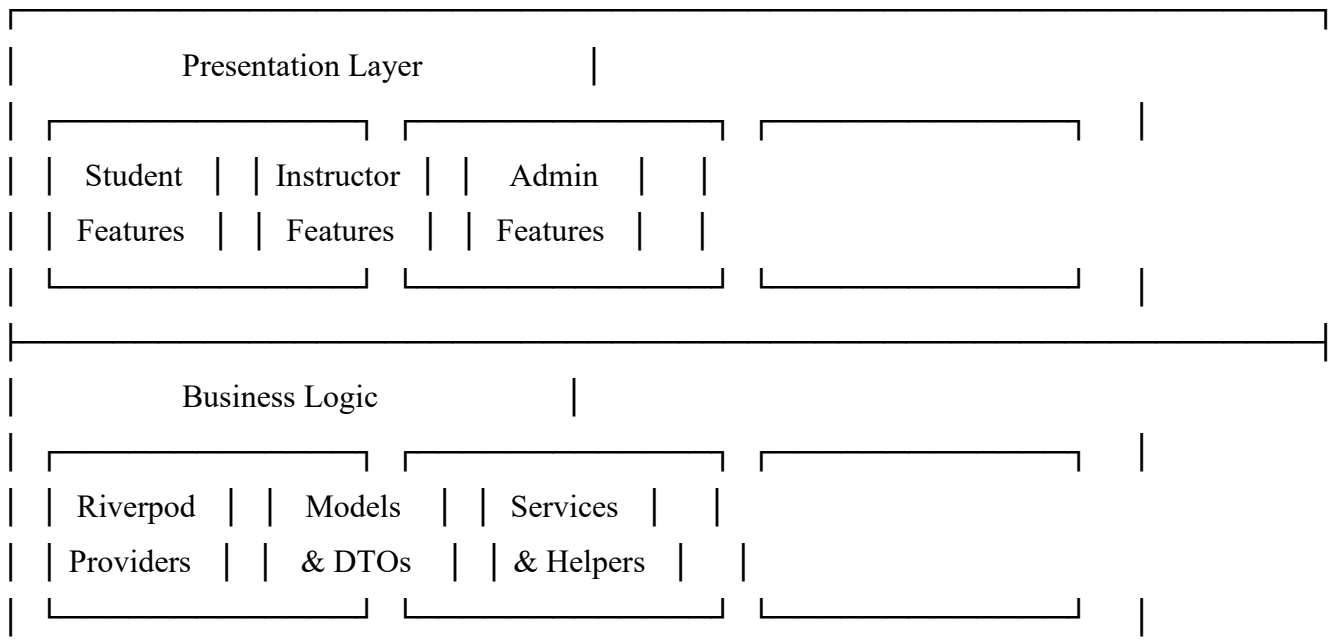
Responsive Design: flutter_screenutil

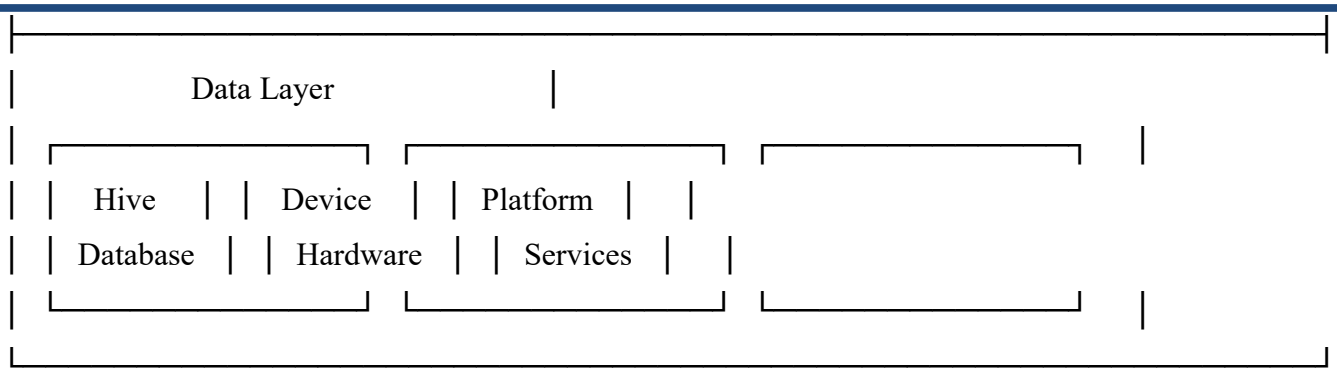
Camera Integration: camera plugin

Location Services: geolocator plugin

Permissions: permission_handler

5.2 Architecture Pattern





6. Core Features Documentation

6.1 Authentication System

File: lib/core/providers/auth_provider.dart

The authentication system implements role-based access control with the following capabilities:

// Demo Credentials

Student: FE22A256 / any password

Instructor: INS001 / any password

Administrator: ADM001 / any password

Features:

Secure login with credential validation

Password reset functionality (UI implemented)

Session management with automatic timeout

Role-based navigation and feature access

Persistent login state using Hive storage

Security Measures:

Password hashing simulation (bcrypt-ready)

Session token management

Automatic logout on inactivity

Input validation and sanitization

6.2 Data Models

Location: lib/core/models/

The application uses strongly-typed data models with Hive type adapters:

User Model (user.dart):

```
@HiveType(typeId: 0)
class User extends HiveObject {
  @HiveField(0) String id;
  @HiveField(1) String fullName;
  @HiveField(2) String email;
  @HiveField(3) String matriculeOrStaffId;
  @HiveField(4) UserRole role;
  @HiveField(5) UserStatus status;
  @HiveField(6) bool hasFacialTemplate;
  @HiveField(7) DateTime createdAt;
  @HiveField(8) DateTime updatedAt;
}

enum UserRole { student, instructor, admin }
enum UserStatus { active, inactive }
```

Course Model (course.dart):

```
@HiveType(typeId: 1)
class Course extends HiveObject {
```

```
  @HiveField(0) String id;
  @HiveField(1) String courseCode;
  @HiveField(2) String courseName;
  @HiveField(3) String description;
  @HiveField(4) String instructorId;
  @HiveField(5) DateTime createdAt;
  @HiveField(6) DateTime updatedAt;
}
```

Session Model (session.dart):

```
@HiveType(typeId: 2)
class Session extends HiveObject {
  @HiveField(0) String id;
  @HiveField(1) String courseId;
  @HiveField(2) DateTime startTime;
  @HiveField(3) DateTime? endTime;
  @HiveField(4) String geofenceId;
  @HiveField(5) SessionStatus status;
  @HiveField(6) DateTime createdAt;
  @HiveField(7) DateTime updatedAt;
}

enum SessionStatus { scheduled, active, ended }
```

AttendanceRecord Model (attendance_record.dart):

```
@HiveType(typeId: 3)
class AttendanceRecord extends HiveObject {
  @HiveField(0) String id;
  @HiveField(1) String studentId;
  @HiveField(2) String sessionId;
  @HiveField(3) AttendanceStatus status;
  @HiveField(4) DateTime? checkInTimestamp;
  @HiveField(5) String? overrideJustification;
  @HiveField(6) String? overrideBy;
  @HiveField(7) bool get isOverridden;
```

```
  @HiveField(8) DateTime createdAt;
  @HiveField(9) DateTime updatedAt;
}

enum AttendanceStatus { present, absent }
```

Geofence Model (geofence.dart):

```
@HiveType(typeId: 4)
class Geofence extends HiveObject {
  @HiveField(0) String id;
  @HiveField(1) String name;
  @HiveField(2) double latitude;
  @HiveField(3) double longitude;
  @HiveField(4) double radius;
  @HiveField(5) bool isActive;
  @HiveField(6) DateTime createdAt;
  @HiveField(7) DateTime updatedAt;
}
```

6.3 Local Storage System

File: lib/core/services/hive_service.dart

Implements offline-first architecture with encrypted local storage:

```
class HiveService {
  static late Box<User> userBox;
  static late Box<Course> courseBox;
  static late Box<Session> sessionBox;
  static late Box<AttendanceRecord> attendanceBox;
  static late Box<Geofence> geofenceBox;
  static late Box<dynamic> enrollmentBox;
  static late Box<dynamic> settingsBox;

  static Future<void> init() async {
    await Hive.initFlutter();
    // Register adapters and open boxes
  }
```

}

Features:

Type-safe data access

Automatic data persistence

Encryption for sensitive data

Offline-first capabilities

Data integrity validation

6.4 Demo Data System

File: lib/core/services/dummy_data_service.dart

Provides realistic demo data for development and testing:

Demo Users:

Students: 5 demo students with varied attendance patterns

Instructors: 2 demo instructors with course assignments

Administrators: 1 system administrator

Demo Courses:

CEF440: Internet Programming and Mobile Programming

CEF430: Computer Networks

CEF450: Software Engineering

Demo Sessions: Realistic session data with various statuses
Demo Attendance: Comprehensive attendance records with patterns
Demo Geofences: University of Buea campus locations
6.5 Student Features
Dashboard (lib/features/student/screens/student_dashboard.dart):

Active session display with real-time updates

Quick check-in access for eligible sessions

Attendance statistics overview

Navigation to key features

Facial Enrollment (lib/features/student/screens/facial_enrollment_screen.dart):
// Key Features:
- Guided enrollment process with clear instructions
- Real-time camera preview with face detection guides
- Image quality validation and feedback
- Privacy consent and data usage transparency
- Re-enrollment capability for data updates
Check-in Process (lib/features/student/screens/check_in_screen.dart):
// Dual Verification Process:

1. Location Acquisition & Geofence Validation

2. Facial Recognition & Template Matching

3. Attendance Record Creation

4. Real-time Feedback & Confirmation

// Performance Target: < 5 seconds end-to-end

Attendance History (lib/features/student/screens/attendance_history_screen.dart):

Comprehensive attendance records display

Advanced filtering by course and date range

Visual status indicators and statistics

Export capabilities (future enhancement)

6.6 Instructor Features

Dashboard (lib/features/instructor/screens/instructor_dashboard.dart):

Course overview with session statistics

Active session monitoring

Quick access to session management

Performance analytics

Session Management (lib/features/instructor/screens/session_management_screen.dart):

// Session Lifecycle Management:

1. Session Creation with Geofence Assignment

2. Real-time Session Monitoring

3. Student Check-in Tracking

4. Session Termination and Finalization

Real-time Monitor (lib/features/instructor/screens/real_time_monitor_screen.dart):

Live attendance tracking during sessions

Student list with real-time status updates

Summary statistics and visualizations

Manual override access

Manual Override (lib/features/instructor/screens/manual_override_screen.dart):

// Override Capabilities:

- Attendance status modification with justification
- Audit trail creation for accountability
- Historical override tracking
- Validation and confirmation workflow

Reporting (lib/features/instructor/screens/instructor_reports_screen.dart):

Custom report generation with flexible filters

Quick report templates for common needs

Export functionality (CSV/PDF simulation)

Historical report access

6.7 Administrator Features

Dashboard (lib/features/admin/screens/admin_dashboard.dart):

System-wide statistics and health monitoring

Management module quick access

Real-time system status indicators

Administrative action shortcuts

User Management (lib/features/admin/screens/user_management_screen.dart):

// CRUD Operations:

- User Creation with Role Assignment
- User Profile Editing and Status Management
- Account Activation/Deactivation
- Password Reset Administration
- Advanced Search and Filtering

Course Management (lib/features/admin/screens/course_management_screen.dart):

// Course Administration:

- Course Creation and Editing
- Instructor Assignment Management
- Student Enrollment Administration
- Course Status and Lifecycle Management

Geofence Management (lib/features/admin/screens/geofence_management_screen.dart):

// Location Boundary Management:

- Geofence Creation with GPS Coordinates
- Visual Map Interface (future enhancement)
- Radius and Tolerance Configuration
- Active/Inactive Status Management

System Reports (lib/features/admin/screens/admin_reports_screen.dart):

Comprehensive system analytics

Multi-dimensional filtering capabilities

Export functionality for data analysis

Trend analysis and insights

Audit Logs (lib/features/admin/screens/audit_logs_screen.dart):

// Security and Accountability:

- Complete activity logging
- Advanced search and filtering
- Event categorization and analysis
- Compliance and security monitoring

System Settings (lib/features/admin/screens/system_settings_screen.dart):

// Configurable Parameters:

- Facial Recognition Thresholds
- Geofence Tolerance Settings
- Security Policy Configuration
- Data Retention Policies
- Feature Toggle Management

7. State Management Architecture

Riverpod Implementation with code generation:

```
// Provider Examples:
@riverpod
Future<List<Map<String, dynamic>>> studentActiveSessions(StudentActiveSessionsRef ref) async {
  // Provider implementation
}

@riverpod
Future<Map<String, dynamic>> sessionDetails(SessionDetailsRef ref, String sessionId) async {
  // Provider implementation with parameters
}
```

Key Providers:

authProvider: Authentication state management

currentUserProvider: Current user session

studentActiveSessionsProvider: Student dashboard data

instructorCoursesProvider: Instructor course data

adminSystemStatsProvider: System statistics

sessionDetailsProvider: Session-specific data

8. Security Implementation

8.1 Data Protection

// Encryption for sensitive data

- Biometric templates encrypted at rest

- Secure token storage using platform keychain
- Input validation and sanitization
- SQL injection prevention

8.2 Privacy Compliance

// Data handling practices

- Minimal data collection principle
- Explicit user consent for biometric data
- Data retention policy implementation
- User rights management (access, correction, deletion)

8.3 Access Control

// Role-based security

- Authentication guards on all routes
- API endpoint protection simulation
- Feature-level access control
- Audit logging for sensitive operations

9. Performance Optimization

9.1 App Performance Targets

Cold Start: < 3 seconds

Check-in Process: < 5 seconds average

Real-time Updates: < 5 seconds latency

UI Responsiveness: < 200ms interaction feedback

9.2 Optimization Techniques

// Performance strategies

- Lazy loading of non-critical data
- Image compression for facial enrollment

- Efficient state management with Riverpod
- Memory management for camera operations
- Background processing for data synchronization

10. Testing Strategy

10.1 Test Coverage Areas

// Unit Tests
- Data model validation
- Business logic verification
- Provider state management
- Utility function testing

// Widget Tests
- UI component behavior
- User interaction simulation
- Screen rendering validation
- Form input testing

// Integration Tests
- End-to-end user workflows
- Cross-feature interaction
- Performance benchmarking
- Error handling scenarios

11. Development Workflow

11.1 Setup Commands

```
# Complete project setup
make setup

# Development with hot reload
make hot-reload

# Code quality checks
make format lint test
```

Production builds

make build-all

11.2 Code Generation

Generate all code (Hive adapters, Riverpod providers)

flutter packages pub run build_runner build --delete-conflicting-outputs

Watch mode for development

flutter packages pub run build_runner watch --delete-conflicting-outputs

12. Deployment Configuration

12.1 Android Configuration

Minimum SDK: 23 (Android 6.0 Marshmallow)

Target SDK: Latest stable Android API

Permissions: Camera, Fine/Coarse Location, Storage

Features: Camera hardware, GPS location, Front camera

12.2 iOS Configuration

Minimum Version: 12.0

Permissions: Camera, Location, Photo Library, Microphone

Features: Camera hardware, Location services

Privacy: Usage descriptions for all sensitive permissions