

REPUBLIC OF CAMEROON

\*\*\*\*\*

Peace-Work-Fatherland

\*\*\*\*\*

MINISTER OF HIGHER  
EDUCATION

\*\*\*\*\*

UNIVERSITY OF BUEA



REPUBLIQUE DU CAMEROON

\*\*\*\*\*

PAIX-Travail-Patrie

\*\*\*\*\*

MINISTERE DE  
L'ENSEIGNEMENT  
SUPERIEUR

\*\*\*\*\*

UNIVERSITE DE BUEA

## FACULTY OF ENGINEERING AND TECHNOLOGY

### COURSE TITLE:

INTERNET PROGRAMMING (J2EE) AND MOBILE PROGRAMMING

### COURSE CODE:

CEF440

## TASK 5 REPORT

### UI Design and Implementation

Course Instructor: Dr. Nkemeni Valery

### GROUP IV

S/N	Names	Matricules
1	ARREY ABUNAW REGINA EBAI	FE22A152
2	AWA ERIC ANGELO JUNIOR	FE22A162
3	FAVOUR OZIOMA	FE22A217
4	OBI OBI ETCHU JUNIOR	FE22A291
5	VERBURINYUY JERVIS NYAH	FE22A324

## Table of Contents

ABSTRACT.....	3
I. Introduction .....	4
1.1 Background.....	4
1.2 Problem Statement.....	4
1.3 Framework: User-Centered Design (UCD).....	4
1.5 Technology Stack .....	5
II. App Identity and Branding .....	6
a. App Name and Logo .....	6
b. Splash Screen.....	8
c. Design Language and Theme .....	8
III. Visual Design .....	10
a. Wireframes .....	10
b. High-Fidelity UI Designs.....	10
c. User Flow Diagrams.....	14
d. Design System / Style Guide .....	15
IV. Frontend Implementation .....	17
a. Technology Stack.....	17
b. Component Structure .....	17
c. Responsive Design .....	18
d. Animations and Transitions .....	19
e. Role-Based UI Behavior .....	20
V. Conclusion.....	21
VI. Appendices .....	22

# ABSTRACT

This report presents the user interface (UI) design and implementation of *Attendr*, a mobile-based attendance management system that leverages geofencing and facial recognition to ensure accurate, secure, and real-time tracking of student attendance in higher education institutions. In response to the limitations of manual and semi-digital attendance methods, the *Attendr* application aims to enhance transparency, reduce proxy attendance, and improve efficiency through automation.

The UI design process followed the **User-Centered Design (UCD)** framework, ensuring that the needs, preferences, and behaviors of the end-users (students, instructors, and administrators) were central to every stage of the design and development process. Through iterative design, prototyping, and user feedback integration, the team created an intuitive, responsive, and role-specific interface that supports seamless user interaction. The application was developed using **React Native** and **Expo**, ensuring cross-platform compatibility and optimized performance for mobile devices. This task focused on establishing the app's visual identity, translating mockups into functional interfaces, and implementing navigational flow tailored to diverse user roles. The final output is a visually cohesive, user-friendly, and scalable front-end that supports the application's core features.

## 1.1 Background

Accurate and efficient attendance management is a cornerstone of academic administration, yet traditional methods such as manual registers and RFID cards are fraught with challenges, including time consumption, human error, and susceptibility to manipulation. The proliferation of mobile devices and advancements in biometric and location-based technologies present an opportunity to revolutionize this process.

This application is conceived as a standalone mobile application that integrates facial recognition for biometric verification and geofencing for location validation, ensuring that only students physically present within a predefined classroom boundary can mark their attendance. The system is designed to operate independently of existing learning management systems, with potential for future integration. Its core functionalities include secure student authentication, real-time attendance recording, instructor dashboards for monitoring, and administrative tools for system configuration and user management.

## 1.2 Problem Statement

Current attendance systems suffer from:

- **Proxy attendance** (students marking attendance for absent peers).
- **Time-consuming manual processes** (long queues during roll calls).
- **Lack of real-time validation** (delays in reporting and analytics).
- **Security vulnerabilities** (unsecured data storage).

A **mobile-based solution** that integrates **facial recognition and geofencing** can eliminate these issues while improving accuracy and efficiency.

## 1.3 Framework: User-Centered Design (UCD)

The development of this mobile application is grounded in the user-centered design (UCD) methodology. UCD is an iterative framework that places end-users at the heart of the design process, involving them at every stage from requirements gathering to prototyping, testing, and refinement. This approach ensures that the system not only meets technical specifications but is also intuitive, accessible, and responsive to the real-world contexts in which it will be used.

Key steps in the UCD process include:

- **Stakeholder Analysis:** Identifying and engaging primary user groups (students, instructors, administrators) to understand their workflows, pain points, and expectations.
- **Requirement Gathering:** Conducting interviews, surveys, and usability studies to inform functional and interface requirements.
- **Prototyping and Testing:** Developing wireframes and interactive prototypes, followed by iterative usability testing with representative users to gather feedback and drive improvements.
- **Accessibility and Usability:** Prioritizing clear navigation, minimal check-in time, and support for users with varying levels of technical proficiency.
- **Continuous Feedback:** Implementing mechanisms for ongoing user feedback post-deployment to guide future enhancements.

## 1.5 Technology Stack

- **Frontend:** React Native (JavaScript) + Expo – Cross-platform development.
- **Facial Recognition:** TensorFlow.js / ML Kit – On-device face detection and matching.
- **Geofencing:** Expo Location + Google Maps API – Classroom boundary validation.
- **Backend:** Firebase / Node.js – Secure cloud storage for attendance records.
- **State Management:** Redux / Context API – Efficient data handling.

## II. App Identity and Branding

---

### a. App Name and Logo

#### ❖ **App Name:** Attendr

Attendr emphasizing on reliable and secure attendance tracking. The name reflects the app's purpose ensuring accurate and trustworthy attendance logging using modern technology like facial recognition and geofencing.

Uniqueness: Short, catchy, and easily searchable

Logo:



Symbolism

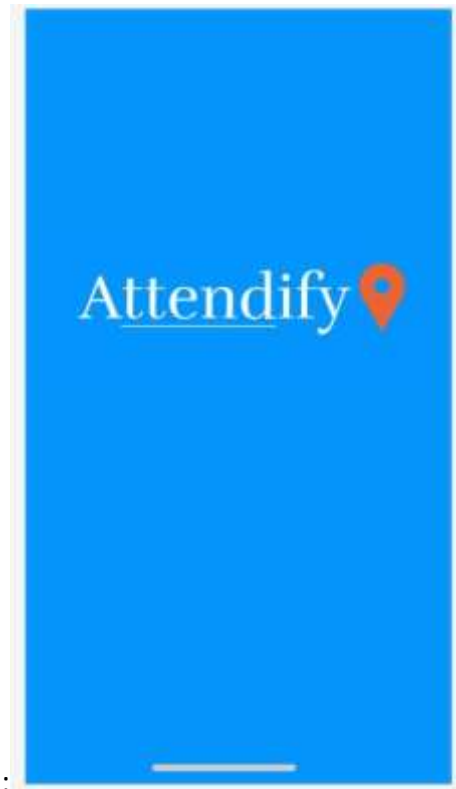
- **Concept:** The word "**Attendr**" in bold typography with a **red location pin** (□) replacing the dot over the "i".
  - **Symbolism:**
    - **Location Pin:** Represents geo-tagged attendance tracking (e.g., for classrooms/events).
    - **Red Accent:** Draws attention to actionable items (e.g., marking attendance).

Tagline: "*Mark Your Presence.*"

❖ **Color Palette Used**

Color	Hex Code	Usage
Soft Blue	(#0394FC	App header, navigation bars, primary buttons
Soft White	#F9FAFB	Backgrounds, content areas, text fields
Teal	#2DD4BF	Accent icons, progress indicators, secondary CTAs
Light Gray	#D1D5DB	Dividers, secondary text, borders
Red	#EF4444	Errors, absences, warning notifications
Green	#10B981	Confirmations, success states

## b. Splash Screen



The splash screen offers a smooth transition into the app, reinforcing branding with a minimalist layout.

### ❖ Design Tools Used:

- Figma (for wireframes, icon, and splash screen design)

## c. Design Language and Theme

### ❖ Typography

- Primary font: Inter ,Rufina
- Font hierarchy:
  - Headlines: Bold, 20–24px
  - Subheadings: Medium, 16–18px
  - Body: Regular, 14–16px

### ❖ Color Usage Guidelines

- High contrast between text and background ensures WCAG 2.1 accessibility.
- Navy Blue is used consistently for primary buttons and navigation.
- Teal and Green for success messages, Red for error states.
- Accent colors are used sparingly to avoid clutter.



## ❖ **Spacing and Layout**

- Uniform padding/margins (8/16/24 px grid)
- Rounded corners (8px) for a modern, friendly look
- Minimalist design with white space to reduce cognitive load
- Light/Dark Mode Support: Currently optimized for Light Mode, with Dark Mode support planned in future versions.

## ❖ **Accessibility Considerations:**

- Sufficient color contrast for text
- Large touch targets (min. 48x48 dp)
- Scalable fonts for readability
- Semantic roles used in code for screen readers

### a. Wireframes

Low-fidelity wireframes were created to map out the structure and layout of major user interface screens before applying branding or visual polish.

#### Screens Included:

- Splash screen
- Login Page
- Student Registration Page
- Check-In Page
- Attendance History Page
- Permission request Page
- Instructor dashboard
- Admin dashboard
- Settings Page

(Insert wireframes here as image files or embed a Figma/Balsamiq link)

#### Tools Used:

- Figma (for digital wireframes)
- Hand-drawn sketches scanned for early design validation

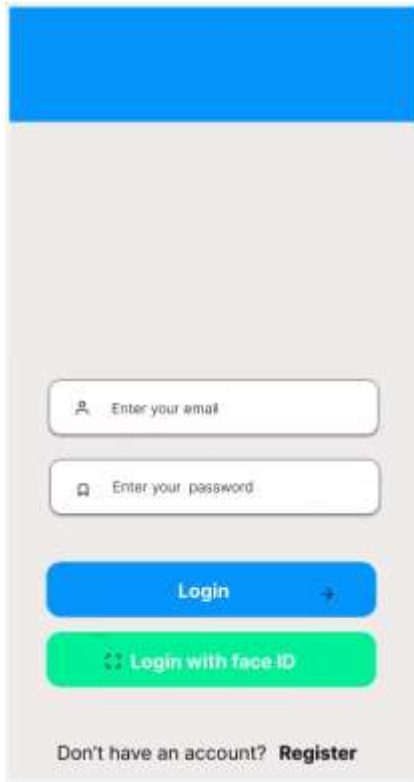
### b. High-Fidelity UI Designs

These mockups reflect the final look of the app, incorporating brand colors, typography, spacing, and interactive design elements.

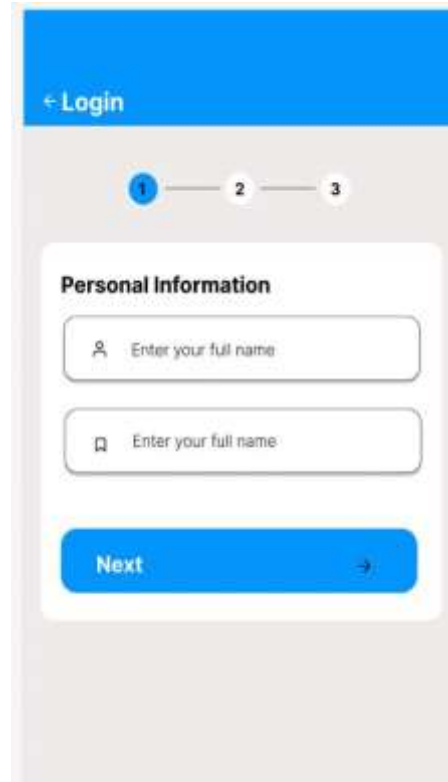
#### ❖ Included Screens:

##### Login / Register

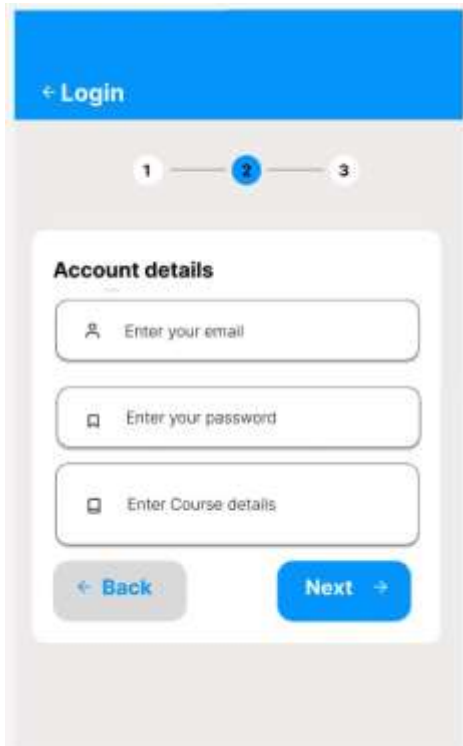
- Simple layout with branding
- Secure input fields
- “Login with Face ID” option



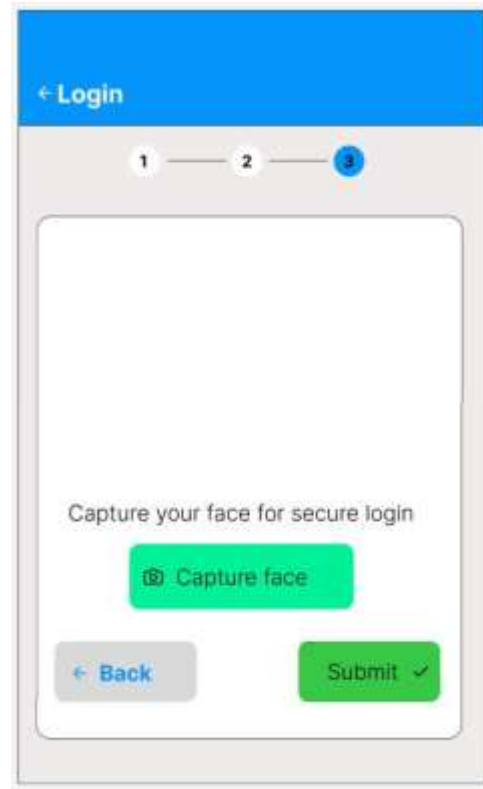
Initial login screen with a blue header. It contains two input fields: "Enter your email" and "Enter your password". Below them are two buttons: a blue "Login" button and a green "Login with face ID" button. At the bottom, it says "Don't have an account? Register".



Personal Information screen. It has a blue header with a back arrow and the word "Login". A progress indicator shows three steps, with the first step (1) highlighted in blue. The screen contains two input fields, both labeled "Enter your full name". Below them is a blue "Next" button with a right arrow.



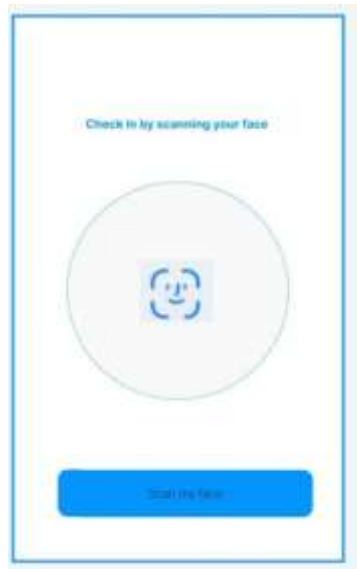
Account details screen. It has a blue header with a back arrow and the word "Login". A progress indicator shows three steps, with the second step (2) highlighted in blue. The screen contains three input fields: "Enter your email", "Enter your password", and "Enter Course details". Below them are two buttons: a grey "Back" button with a left arrow and a blue "Next" button with a right arrow.



Face capture screen. It has a blue header with a back arrow and the word "Login". A progress indicator shows three steps, with the third step (3) highlighted in blue. The screen contains a large white box for face capture. Below the box, it says "Capture your face for secure login". There are three buttons: a green "Capture face" button with a camera icon, a grey "Back" button with a left arrow, and a green "Submit" button with a checkmark.

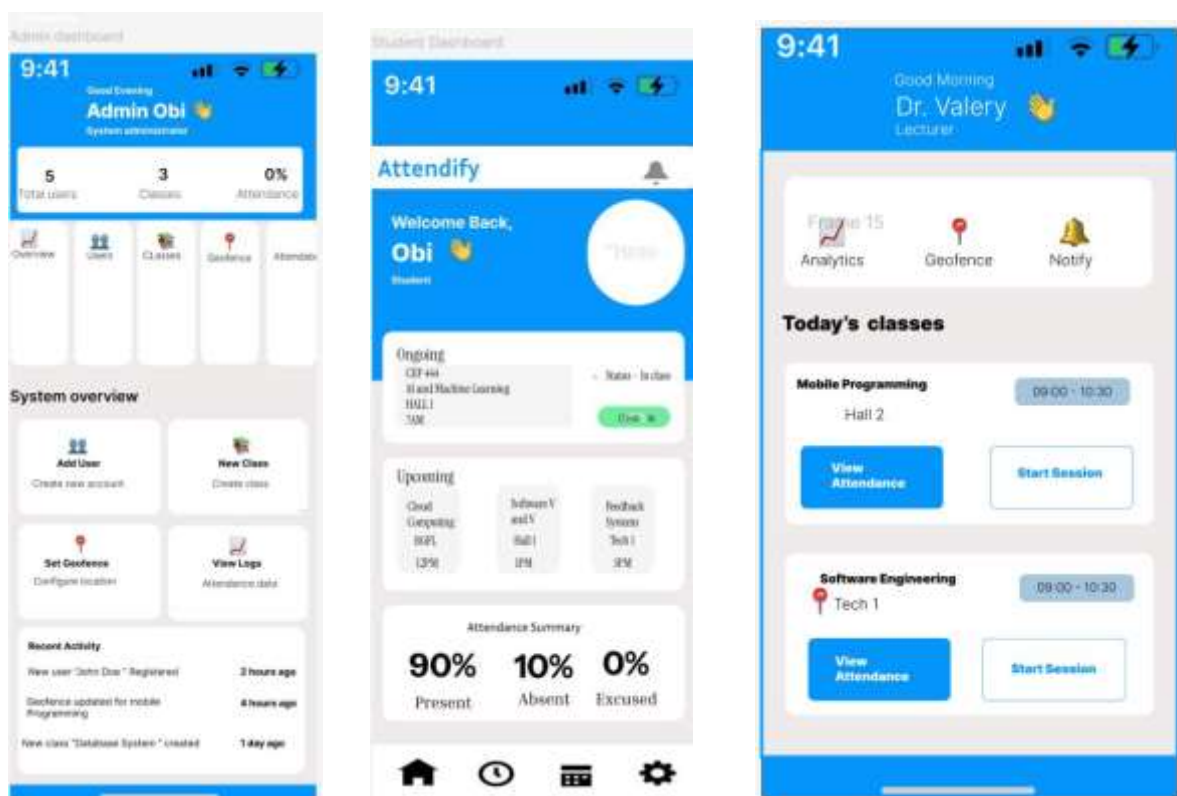
## Attendance Check-In

- Integrated camera view with a “Scan Face” button
- Geolocation indicator with radius compliance check
- Real-time status messages (e.g., “Face Detected”, “Location Verified”, “Attendance Recorded”)



## Dashboard

- Admin: Overview of all user activity, check-in logs, and system status
- Student: Personalized greeting, today’s attendance status, upcoming classes
- Lecturer: Enables instructors to view real-time attendance, manage overrides, and access analytic



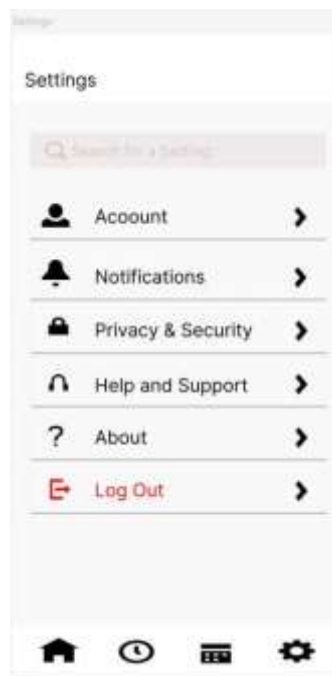
## Attendance History

- Calendar or list view
- Icons and color indicators for Present, Absent, Late, etc.
- Export button for admin (PDF/CSV)



## Settings / Profile

- Profile photo upload
- Manage Face Scan
- Update password, enable biometric login (optional)
- Role info (e.g., Student ID, Admin Email)



## c. User Flow Diagrams

These diagrams visualize how users interact with the system from login to task completion. Separate flows are defined for Admin and Student roles.

### ❖ Diagrams Include

- User Onboarding Flow
- Attendance Check-In Flow (with conditional location and face match logic)
- Admin Dashboard Navigation
- Notification & History Access
- Settings Flow

(Insert flowcharts or embed Lucidchart/Figma flow diagrams here)

### ❖ Highlights

- Role-based screen access:
  - Admins: Can access all dashboards, reports, and settings
  - Students: Limited to personal data, check-in/out, and notifications
- Conditional logic:
  - If user is outside geofence → show location error screen
  - If facial recognition fails → prompt re-scan

### ❖ Tool Used

- Figma's Flowchart Tool
- Lucidchart (for complex conditional logic)

## d. Design System / Style Guide

The design system ensures a consistent, accessible, and scalable user interface across all platforms.

### ❖ Colors

Type	Color Name	Hex Code	Usage
Primary	Navy Blue	#1E3A8A	App bars, buttons, key CTAs
Background	Soft White	#F9FAFB	Main backgrounds
Accent	Teal	#2DD4BF	Icons, progress indicators
Neutral	Light Gray	#D1D5DB	Borders, secondary elements
Success	Green	#10B981	Success states (e.g., "Check-In Successful")
Error	Red	#EF4444	Error states (e.g., "Face Not Recognized")

## ❖ Typography

Use Case	Font Size (sp)	Weight
Heading	20–24	Bold
Subheading	16–18	Medium
Body Text	14–16	Regular

- Fonts are scalable to support various screen sizes.
- High contrast for readability and WCAG 2.1 accessibility compliance.

## ❖ UI Components

Component	Description
Buttons	Rounded corners, filled (primary), or outlined (secondary), 48dp height
Text Fields	Underlined with helper/error text below
Cards	Elevation for grouping data like attendance logs
Modals	Used for confirmation actions (e.g., confirming check-in, face error)
Navigation Bar	Bottom tab bar (Student), drawer navigation (Admin)



## IV. Frontend Implementation

---

### a. Technology Stack

#### ❖ Framework

React Native: Chosen for its cross-platform capabilities, allowing the app to run on both Android and iOS from a shared codebase.

#### ❖ UI Libraries

- React Native Paper: Provides a Material Design system for consistent, accessible, and stylish UI components.
- React Native Vector Icons: Used for consistent icons across buttons, navigation, and alerts.
- Tailwind React Native Classnames (NativeWind): Enables utility-first styling for faster UI development.

#### ❖ State Management

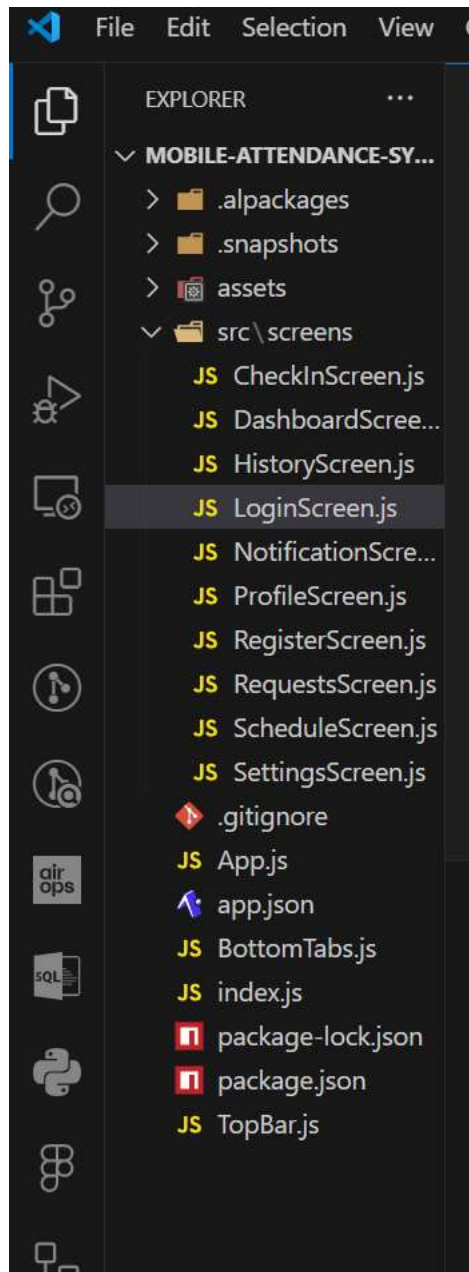
- Zustand: Lightweight and scalable state manager with simple hooks-based API, perfect for managing role, login state, and session info.
- React Context API: Used for global role-based UI access and theme settings.

#### ❖ Routing and Navigation

- React Navigation: For stack-based and tab-based screen navigation, with conditional flows based on user role.

### b. Component Structure

The React Native frontend codebase is structured in a modular and scalable format using reusable and role-based component segmentation.



## ❖ Component Design Principles

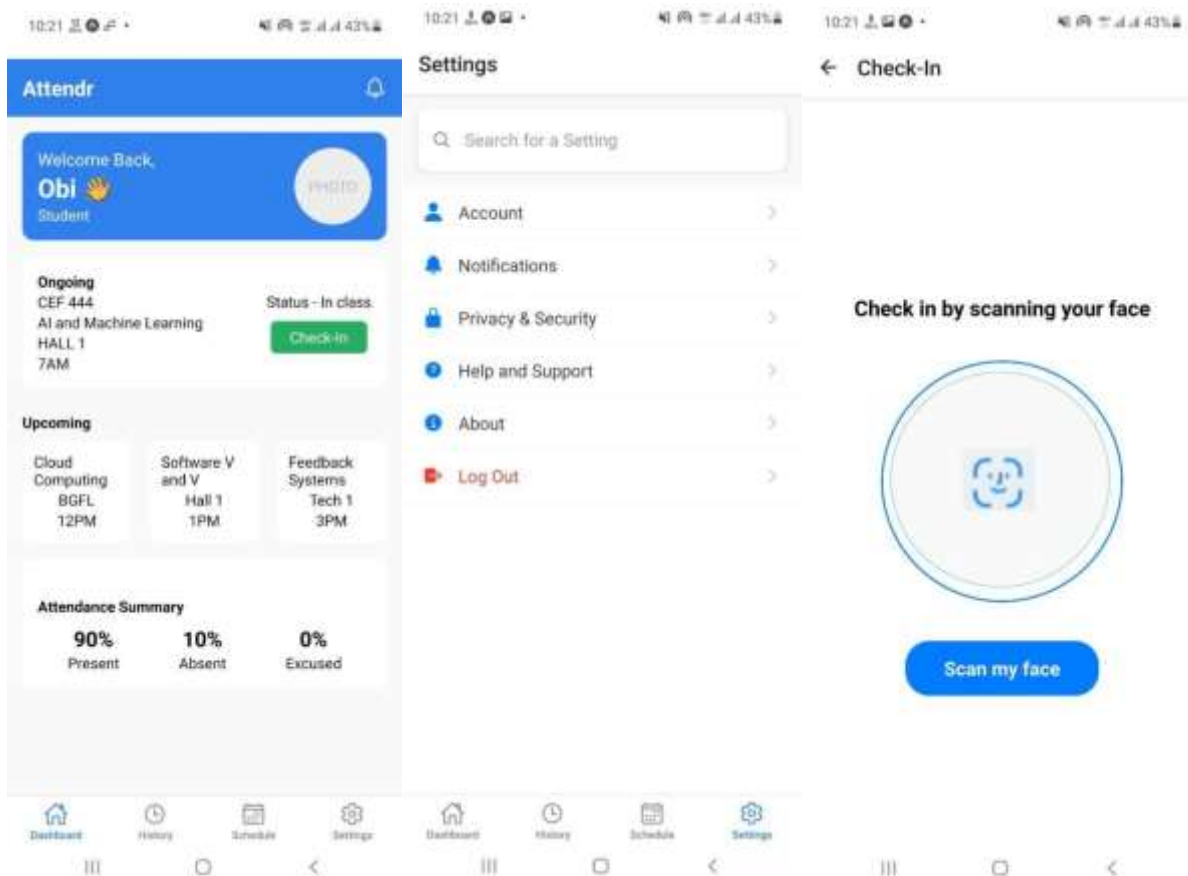
- Atomic Design: Button, Icon, Input as base components.
- Reusability: All layout blocks (cards, check-in prompts, alerts) are abstracted and reused across roles.
- Separation of Concerns: Each screen folder contains its own UI, logic, and subcomponents.

## c. Responsive Design

### ❖ Approach

- Flexbox-first layout using Tailwind (NativeWind) and Dimensions API to scale based on screen size.
- React Native Responsive UI utilities for adapting paddings, font sizes, and spacing.

- Conditional layout logic for tablets vs phones when necessary.
- Media Queries/Breakpoints: Not explicitly supported in React Native but mimicked using `useWindowDimensions()` and conditional rendering.



## d. Animations and Transitions

### ❖ Types of Animations Used

- Screen transitions (e.g., fade in/out, slide between tabs)
- Feedback animations for:
  - Attendance check-in success (Green checkmark + bounce-in)
  - Errors in location or face scan (Shake/wiggle icon or red pulse)
  - Button press effects (scale-in micro-interactions)

### ❖ Libraries

- React Native Reanimated: Used for performant animations with native threading.
- Lottie React Native: For success and error animations (e.g., animated checkmark or face scan ring).

## e. Role-Based UI Behavior

The app dynamically renders different screens and content based on the logged-in user's role (student, admin).

### ❖ Logic:

- Role is stored in global state using Zustand + AsyncStorage.
- On login, user role is fetched from backend and persisted.
- UI routes are conditionally loaded via React Navigation guards.

### ❖ Example Behaviors:

Feature	Student View	Admin View
Dashboard	Personalized greeting + check-in button	Overview cards with system metrics, user stats
Attendance History	List view of their own logs	Searchable list of all user logs
Settings/Profile	Update face scan, logout	Manage user roles, system settings
Notifications	Check-in reminders, alerts	System alerts, user flagging

To ensure a robust and user-friendly UI, the following considerations are critical, aligned with the SRS (NFR2: Usability, NFR4: Compatibility, NFR9: Mobile App):

1. **Accessibility:**
  - Adhere to WCAG 2.1 guidelines (e.g., high contrast ratios, screen reader support).
  - Use large, tappable buttons (min 48x48 dp) for touch interactions.
  - Provide text alternatives for icons and images.
2. **Responsive Design** (NFR5):
  - Optimize layouts for various screen sizes (Android 8+, iOS 12+).
  - Use relative units (e.g., %, vw, vh) for flexible UI scaling.
  - Test on both portrait and landscape orientations.
3. **Performance** (NFR7):
  - Ensure check-in process completes in  $\leq 5$  seconds by optimizing facial recognition (on-device processing where possible) and geofencing (cached location data).
  - Minimize API calls by batching data syncs.
4. **Security and Privacy** (NFR6, NFR8):
  - Encrypt biometric data at rest and in transit (NFR21).
  - Implement session timeouts after 5 minutes of inactivity.
  - Provide clear data deletion options with confirmation prompts.
5. **Visual Hierarchy:**
  - Use Navy Blue for primary actions and headers to guide attention.
  - Employ Teal for interactive elements (e.g., buttons, icons) to indicate interactivity.
  - Limit font styles to 2–3 (e.g., Roboto for Android, San Francisco for iOS) for consistency.
6. **Feedback Mechanisms:**
  - Provide immediate feedback for actions (e.g., Green checkmark for successful check-in, Red error for failures).
  - Use subtle animations (e.g., fade-ins, button presses) to enhance UX without overwhelming users.
7. **Frontend Implementation Suggestions:**
  - **Framework:** Use React Native for cross-platform compatibility (Android 8+, iOS 12+).
  - **Styling:** Leverage Tailwind CSS for responsive, utility-first styling.
  - **Libraries:**
    - Facial recognition: Use Face-API.js or a cloud-based service like AWS Rekognition.
    - Geofencing: Integrate Google Maps API or OpenStreetMap for boundary definitions.
    - Notifications: Use Firebase Cloud Messaging for push notifications.
  - **Structure:** Single-page application with navigation handled via React Navigation.
8. **Prototyping:**
  - Create low-fidelity wireframes using tools like Figma to validate layouts with stakeholders.
  - Test high-fidelity prototypes with sample users to refine usability.

This response provides a comprehensive UI design plan for the Mobile-Based Attendance Management System, addressing all components while adhering to the SRS and project requirements. The artifacts are structured to facilitate frontend implementation and ensure a user-friendly, secure, and efficient experience

## 6.1 Link to figma

<https://www.figma.com/design/VJNi8CLcwpDGk0YWG7ka4e/Attendance-management-system?node-id=0-1&m=dev&t=3BeyZFYZGGuMa7qA-1>