STUDENT REGISTRATION SYSTEM

PRESENTED BY: DAVID FODAY AND ALPHA DANIEL NGEBEH

COURSE TITLE: MOBILE SOFTWARE DEVELOPMENT WITH REACT NATIVE AND JAVASCRIPT.

BIOMETRIC DATABASE DEVELOPMENT.

INSTITUTION: NJALA UNIVERSITY

TITLE:

STUDENT REGISTRATION SYSTEM

ABSTRACT:

Registering students might sound simple, but it's a complex process with multiple moving parts. You need a way to collect and organize student information and process payments. But that's only a small part of what a student registration system should do.

Student Registration System is a mobile application built using React Native and C# Asp.net to provide an efficient and user-friendly student registration system for educational institutions. The application aims to simplify the process of student registration, including user authentication, course enrollment, and academic information management. By integrating local storage and cloud-based data storage, the system will provide easy access to student data anytime and anywhere, improving both student and administrative experiences.

The old system is no longer adequate due to the associated problems. The new system must be designed to correct shortcomings. The new system is in modular forms, which make the work easy, review and amendments can be done, without any effect on the others. Documentation will be presented in a simple form in such a way that computer literate will understand the application of the package, it is also designed in such a way that it will generate the expected output (Student Registration). Adequate security is given to the new system against illegal manipulation, due to the limited number of personnel directly involved in the schoolwork,

OBJECTIVE:

- To create a file for the academic records of the student
- Simplify the student registration process by providing a mobile solution.
- Ensure seamless course enrollment and academic information access for students.
- Provide user authentication and role-based access for students, faculty, and administrators.
- Store and retrieve data using secure and scalable storage solutions.

- ❖ Enhance the user experience through a clean, intuitive UI/UX design.
- ❖ Reduce data loss since the system will ensure that all the respective files are updated.
- ❖ To create a database per the registration procedure of the institution
- ❖ Increase the data retrieval time of information/reports for easier analysis.
- ❖ Reduce data redundancy.

TARGET AUDIENCE:

- ❖ This document is mainly for the developers and the technical and academic staff.
- ❖ Students seeking an efficient registration process.
- ❖ Educational institutions looking for an automated system to manage student enrollment and data.
- ❖ Administrators and faculty members responsible for managing student records.

TECHNOLOGIES USED:

Frontend: React Native, JavaScript

Backend: C# Asp.net

Database: Firebase Firestore (or SQLite for local storage)

State Management: Context API or Redux

Authentication: Biometric Authentication

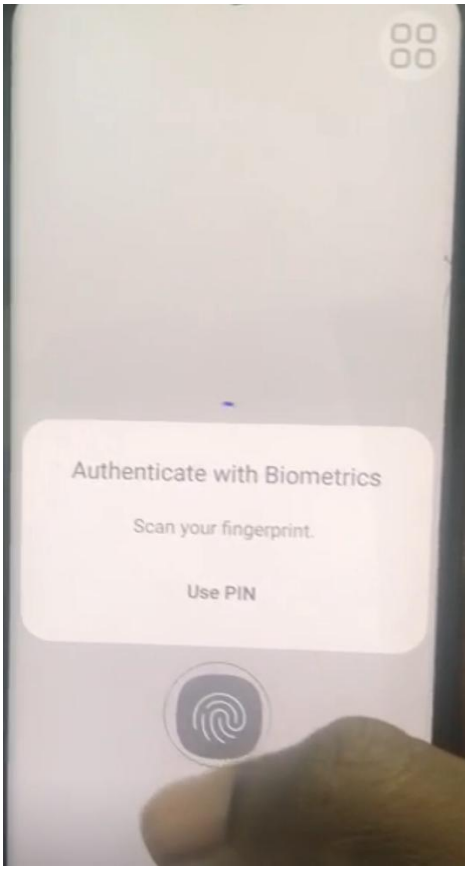
APIs: RESTful APIs for backend communication

Storage: AsyncStorage for local storage, Firebase Firestore for cloud-based data.

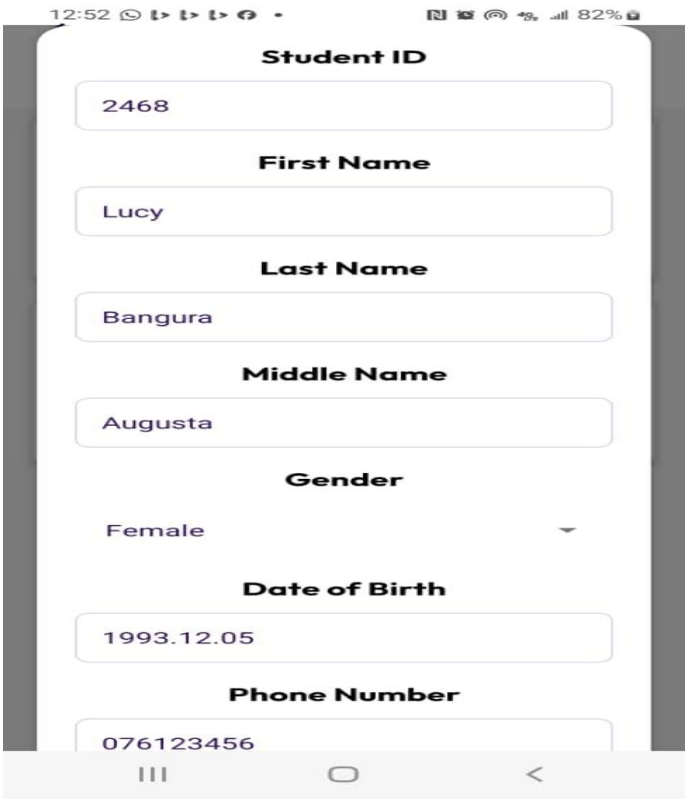
APPLICATION DESIGN AND STRUCTURES

Wireframes: Wireframes will be created for the following major screens:

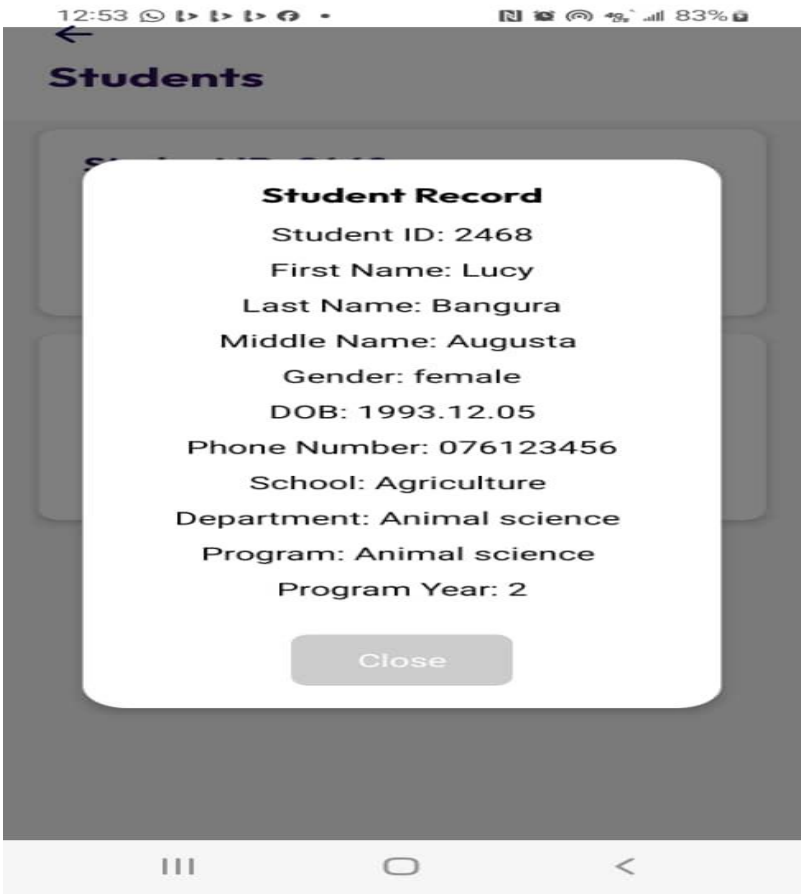
1. **Login Screen:** Authentication screen with Biometric fingerprint for login and registration.



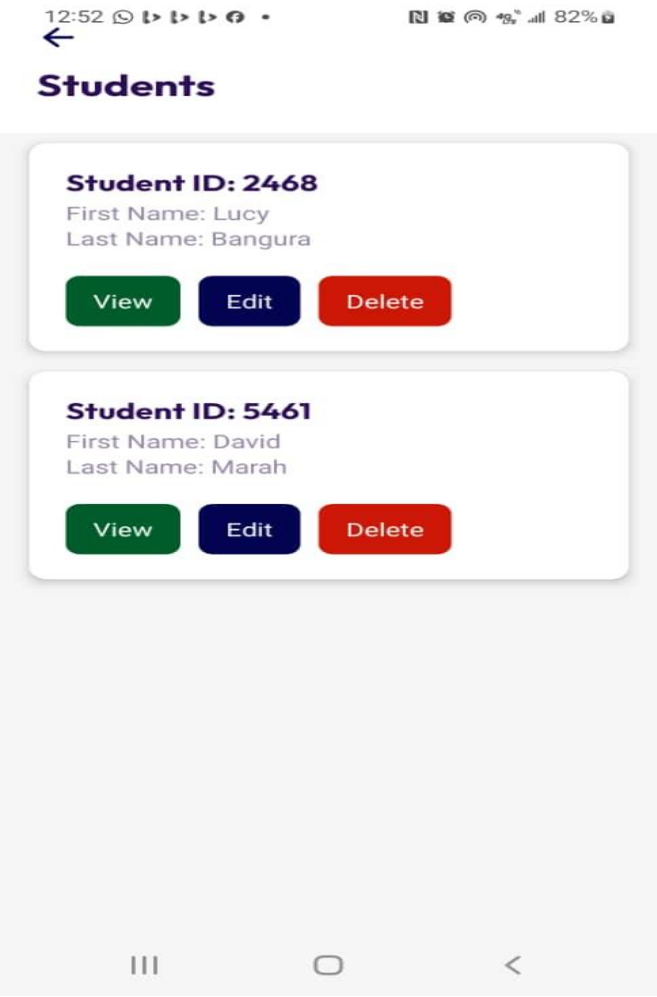
2. **Registration Screen:** Form to collect student details for account creation.



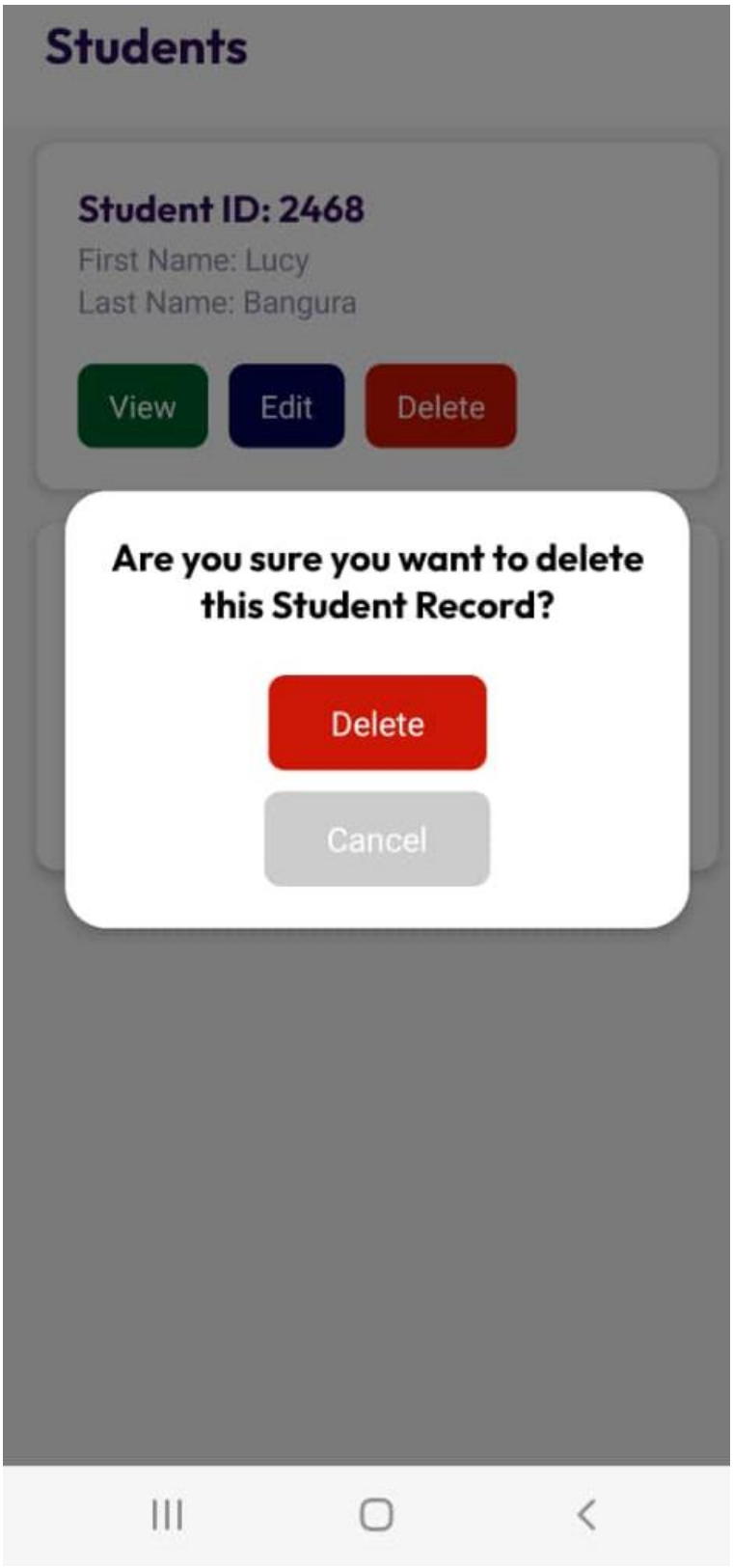
3. **Dashboard:** Student dashboard displaying student records information.



4. **View and Edit Screen:** Allows students to view and edit their details.



5. **Profile Delete Screen:** Displays and allows deleting of student details.



6. **Admin Dashboard:** Interface for administrators to manage student data and course offerings.

User Interface (UI): The UI will be designed to be minimalistic, focusing on easy navigation and clear data presentation. The color scheme will be consistent and visually appealing to ensure a positive user experience. Key UI elements include:

- Clean and simple forms for registration and enrollment.
- A responsive dashboard with easy access to student details and courses.
- Visual indicators for data loading, successful operations, and error handling.

User Experience (UX): The UX will be centered on ease of use:

- **Onboarding:** Simple onboarding for new users with guided prompts.
- **Navigation:** Intuitive navigation between screens using React Navigation.
- **Error Handling:** Clear and informative error messages for invalid input.
- **Responsive Design:** Ensure the application adapts to different screen sizes.

DEVELOPMENT

Setup:

1. **Development Environment Setup:**

- Install Node.js and React Native CLI (expo-cli).
- Initialize a new React Native project using Expo:

```
npx expo-cli init StudentRegistration
cd StudentRegistration
npm start
```

2. **Dependencies:**

- Install required libraries like React Navigation, Firebase, and state management tools:

```
npm install @react-navigation/native react-redux firebase @react-native-async-storage/async-storage
```

Components:

- **Reusable Components:** Form components (e.g., InputField, Button), Layout components (e.g., Header, Footer), and UI elements (e.g., Card, List).
- **Screen Components:** Modular screen components for login, registration, dashboard, and profile.

State Management:

- **Context API/Redux:** Implement global state management to handle user authentication, course data, and student profile information.

Navigation:

- **React Navigation:** Implement stack and tab navigation to allow users to switch between different screens seamlessly.

API Integration:

- **Firebase Firestore:** Integrate with Firebase Firestore to fetch and store student and course data. RESTful APIs can be created for advanced backend features.

Data Storage:

- **Local Storage:** Use AsyncStorage for temporary storage of user preferences and data.
- **Firebase Firestore:** Store persistent student and course data securely in the cloud.

Authentication:

- **Biometric Fingerprint Authentication**: Implement user login and registration with Biometric Fingerprint Authentication supporting fingerprint authentication.

Error Handling:

- **Form Validation**: Implement form validation to ensure required fields are filled out correctly.
- **Error Boundaries**: Handle unexpected errors gracefully and provide user feedback.

TESTING

Unit Testing:

- **Jest:** Write unit tests for critical components such as the registration form, login screen, and API calls.
- **Mocking Dependencies:** Use mocking libraries like jest.fn() to simulate API responses and test components in isolation.

Integration Testing:

- **End-to-End Testing:** Use tools like Detox or Cypress to ensure different parts of the application work together seamlessly.
- **Manual Testing:** Perform manual testing on different devices and emulators.

User Testing:

- **User Feedback:** Conduct user testing with a small group of students and administrators to gather feedback on usability and performance.
- **Iterations:** Use feedback to improve the application and fix any identified issues.