CA400 Final Year Project - Technical Specification

Project Title: Protego
Members: Emma Ablitt (17356661), Mikolaj Adamski (17441622)
Supervisor: Renaat Verbruggen
Submission Date: 07/05/2021

Contents

---

1. Introduction

1.1 Overview

The following document outlines the process of developing the application Protego, a cross platform personal safety application. The motivation behind the application will be discussed within this document, as will the decisions made throughout the design phase. Throughout the development of Protego, we faced many challenges and had to be creative with our solutions. Such obstacles are discussed further down this document. Included at the end of this document are the possible developments envisioned for this application in the future.

1.2 Motivation

In researching an idea for a final year project, many articles discussed the necessity of owning a mobile phone. Security was found to be the primary reason for owning a smartphone, with studies showing a large percentage of people feel more secure when they have their mobile phone[1]. With this interesting topic in mind, we discovered the DCU SafeZone application. The design and productivity of the application is of a high standard, but the functionality is only available on the DCU campus. This inspired an application that took this idea and broadened it to the entire general public.

The key desire of this application was simplicity. Nobody knows how they will react in a dangerous or high stress situation, the easier a person can access help in these situations, the better. After some research, the idea of a “big red button” became the inspiration behind the SOS Mode. Some discussion ensued about the possible situations that may arise in which a user would trigger SOS Mode. This led to multiple options being available to trigger SOS Mode, including voice activation and shaking the device.

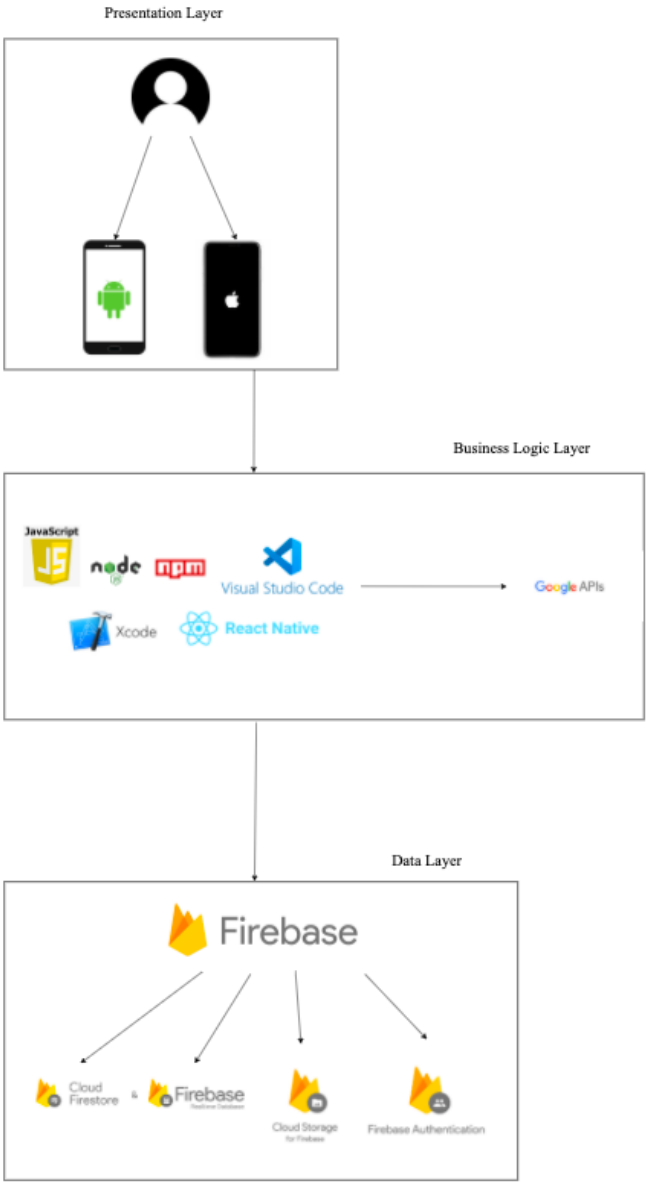
1.3 Glossary

- *API:* Application Programming Interface, which is a software intermediary that allows two applications to talk to each other.

- *Backend:* The ‘backend’ refers to the software of the application, which the user does not see.
- *React Native:* A Javascript framework used to develop applications for Android, iOS, Web and Windows.
- *Firebase:* a platform developed by Google to provide tools for tracking analytics, app crashes and which acts as a server for applications.
- *Firebase Cloud Storage:* A storage service built for Firebase.
- *Firebase Real-time Database:* A database system which stores and syncs data from users in real-time with the application database.
- *Javascript:* The language in which Protego will be developed.
- *UI:* User interface. The design the user interacts with.

2. System Architecture

2.1 System Architecture Diagram



The above image represents the architectural structure of our system, which consists of three layers. At the top of the diagram lies the Presentation Layer. This layer includes the areas of the application that the user interacts with. This includes the User Interface of the application and the functionalities they interact with.
The middle layer, the Business Logic Layer includes the software and components used to build the application. This includes React Native which was the framework used, Visual Studio Code which was the code editor used. Also included are the multiples APIs and libraries used throughout the code.
The last layer is the Data Layer. This layer consists of the applications backend. This layer is largely run by Firebase. Firebase will manage the database, the cloud storage and the authentication of a user.

2.2 Presentation Layer

Since our original design choices, the user interface of the application has changed multiple times. Originally the application was going to feature a white background. When researching and discussing the scenarios in which Protego may be used, most of them required a level of subtlety from the user, not to alert any nearby dangers of their actions. To assist this a dark background was implemented to provide less light bounce away from the user.

Our first design output had the SOS button as the sole home screen. A large “red button” with the rest of the functionality in a tab bar at the bottom of the page. After further thought, it was decided that the other functionality of the application, specifically the Fake Call button should be as easily accessible to the user. This led to the final five bubble displays on the home screen.

Originally the Protego logo was going to be a larger display. It was going to consist of an image with the name intertwined. When it was first implemented, it was decided that the large image was unnecessary and distracted away from the buttons on the home screen. When dealing with emergency situations, as many distractions should be eliminated as possible and the design be kept simple.

2.3 Business Logic Layer

From the onset, React Native was the framework to be used developing the front-end of this project. React Native allowed us to develop for both Android and iOS. Initially, this was difficult as neither developer had any experience using the framework. However, after some introduction videos and test projects, it became more familiar.

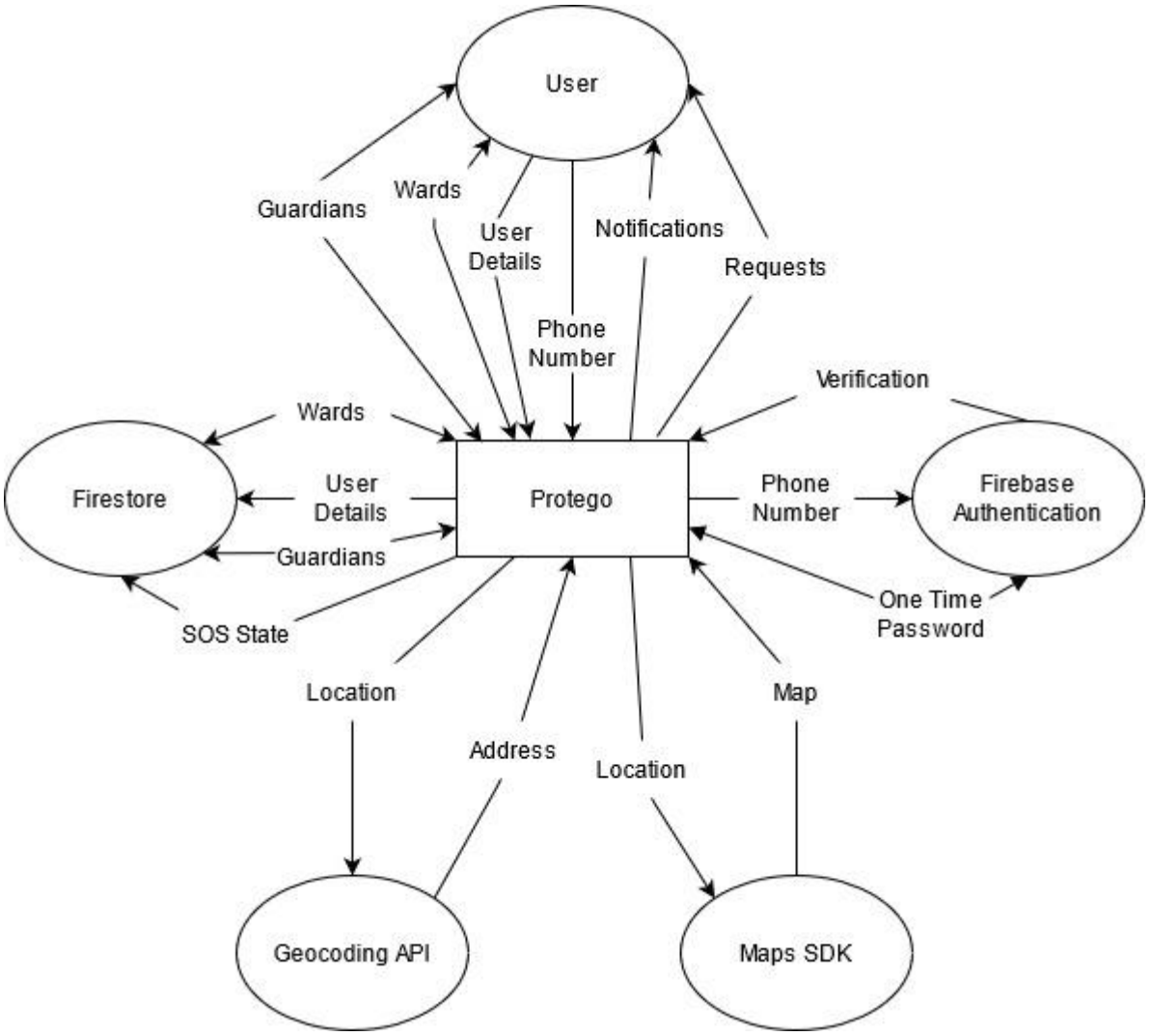
From an iOS perspective, originally Atom was the intended code editor. But, Visual Studio Code was found to have more built in features and is an overall more powerful code editor. XCode was required to build and run the application. Both Visual Studio Code and XCode have built in debuggers and auto generated testing.

2.4 Data Layer

With previous experience in using Firebase, it was always the ideal choice. Firebase is easily integrated into Android Studio and iOS. It consists of many resources which we made use of, including Firebase Database - for storing all important information, and Firebase Authentication - for registering and authenticating users.

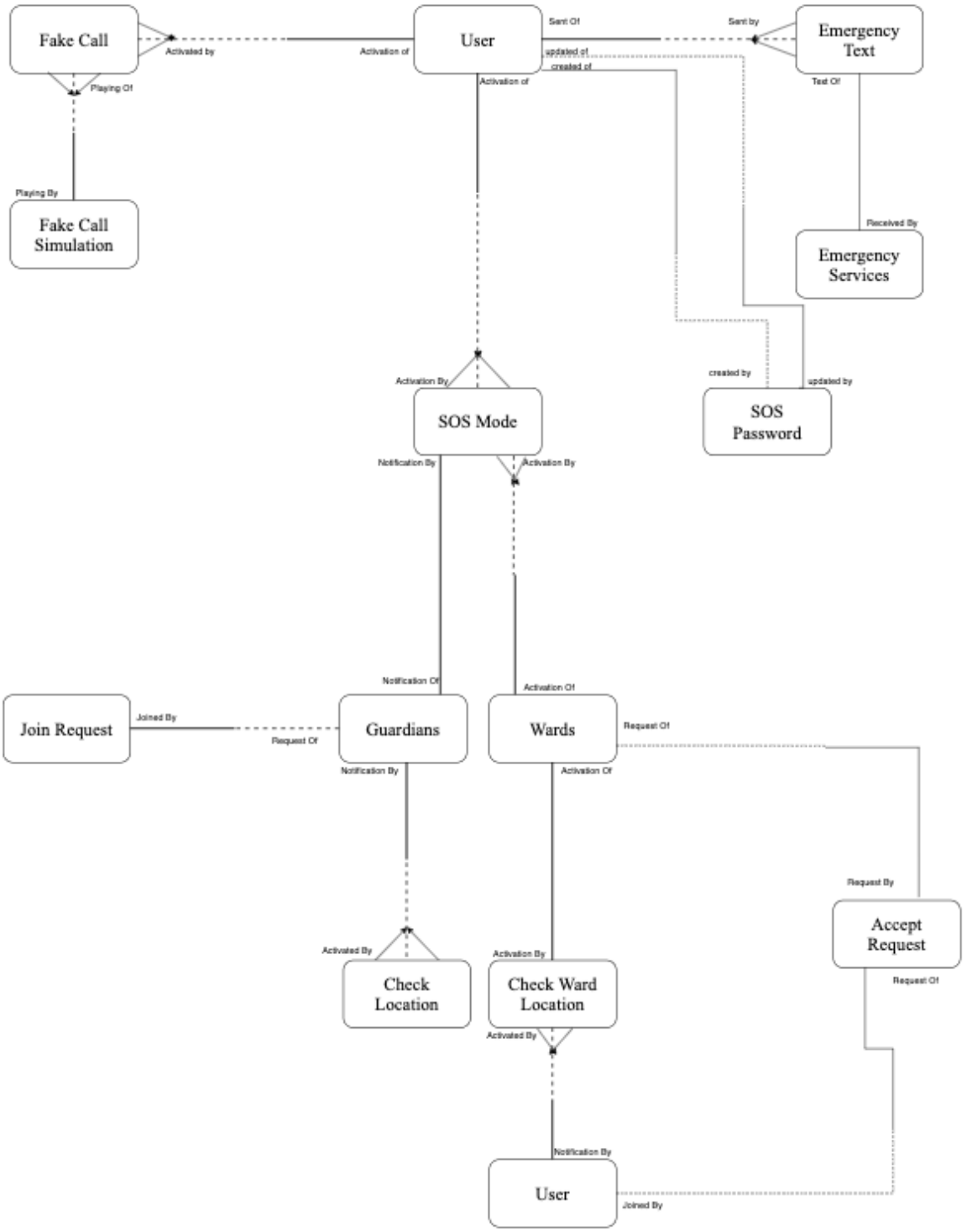
3. High Level Design

3.1 Context Diagram



The above Context Diagram shows the overall boundaries of the system. It shows the requirements of each entity involved in the application. It is a high level view of the system shown as a single process. It is made clear what details of the user each entity requires and how they connect to each other.

3.2 Logical Data Flow Diagram



The Logical Data Flow diagram above focuses on the different business activities and the relationships between them. It shows when and where these relationships interact with each other.

4. Problems and Resolutions

During the course of our project we have found ourselves facing a number of issues. Some of these issues were more difficult to resolve than others. Some of the issues that arose include:

1. Background Location

When we began implementing SOS Mode, the main feature is a user's Guardian being able to track their location. This meant a user's location must be available to the application, even if their mobile device is locked or the application is inactive. When researching through the available resources that React Native provides we came across the library '@mauron85/react-native-background-geolocation'. The library's description suited our needs perfectly and it was well instructed. However, when we first started implementing live location using this library, it kept timing out after 30-40 seconds. The error provided notified that the library was losing the GPS location. This stopped the application from closing. We discovered the reason the library was timing out was because of a forgotten permission. The access permission ACCESS_BACKGROUND_LOCATION, wasn't included in our code. We had not accessed for the user's permission but we were trying to request their location. As soon as we added the permission, the background location stopped timing out.

```
check(PERMISSIONS.ANDROID.BACKGROUND_LOCATION)
  .then((result) => {
    switch (result) {
      case RESULTS.UNAVAILABLE:
        console.log(
          'ANDROID UNAVAILABLE BACKGROUND_LOCATION
        );
```

2. Accessibility

When designing this application we wanted to provide all users with the reassurance that help was available at the click of a button. Accessibility is an

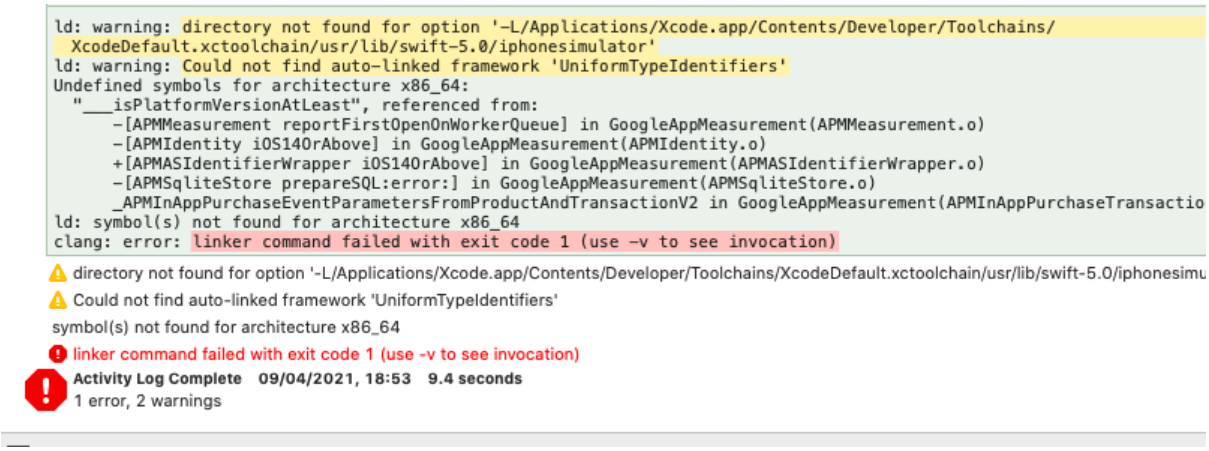
important part in making an application inclusive for all users. A lot of research was carried out to determine the best ways to make the application more accessible.

After the first implementation of the application, it was clear that it wasn't very accessible. If a user was visually impaired, they would find it very difficult to use this application.

In order to assess and improve the accessibility of Protego, we used the Google Accessibility Scanner together with TalkBack. The scanner suggested some changes and we acted accordingly. With TalkBack, we checked if the labels were configured correctly. Originally when we provided labels, on some devices in the emulator, they were being read out twice. We later discovered that this was due to a mismatch between the versions of the devices and TalkBack. As soon as both were updated to the latest version, the labels were only read out once.

3. XCode

On working on the iOS version of the application, after initializing Firebase within the application, XCode provided an error when building the project.



With a lot of online research and discussions with other developers, there was no solution found. A lot of possible solutions were tested, simple options including cleaning the build and other options such as changing build settings, editing the podfile and carefully scanning through the App Delegate files.

Eventually after restarting XCode multiple times, it was decided it wasn't a coding error but a settings one. Having been through the settings multiple times also, XCode was eventually deleted and redownloaded as there was no solution found for this error.

5. Testing

5.1 Adhoc Testing

Throughout the development process, we performed Adhoc Testing. When a new error occurred or if part of the application wasn't behaving properly, we first carried out our own crash testing, locating where the issue was.
If this failed, we would run the debuggers and local testing provided within Visual Studio Code and XCode.

5.2 User Testing

Due to the restrictions of Covid-19, our user testing was restricted to those within our households. Developers were present while the testers used the application, this gave the users the opportunity to ask questions if necessary. Users were asked to walk through the application, taking their time while exploring all of the functionalities. Afterwards they were asked to provide feedback.

All testers were provided with a Plain Language Statement and asked to complete a consent form before starting the testing process.

Since this testing didn't provide a huge number of responses, we created a survey. We were able to send this survey to a wider range of users and therefore receive a preferable level of feedback.

5.3 Integration Testing

Integration testing was important for this application. After the individual unit testing of tracking a user on their map or triggering SOS Mode, it was important that these functionalities worked seamlessly together. If a user triggered SOS Mode successfully but then the map aspect failed, the application would be

severely lacking in functionality. This caused integration testing to be of vital importance.

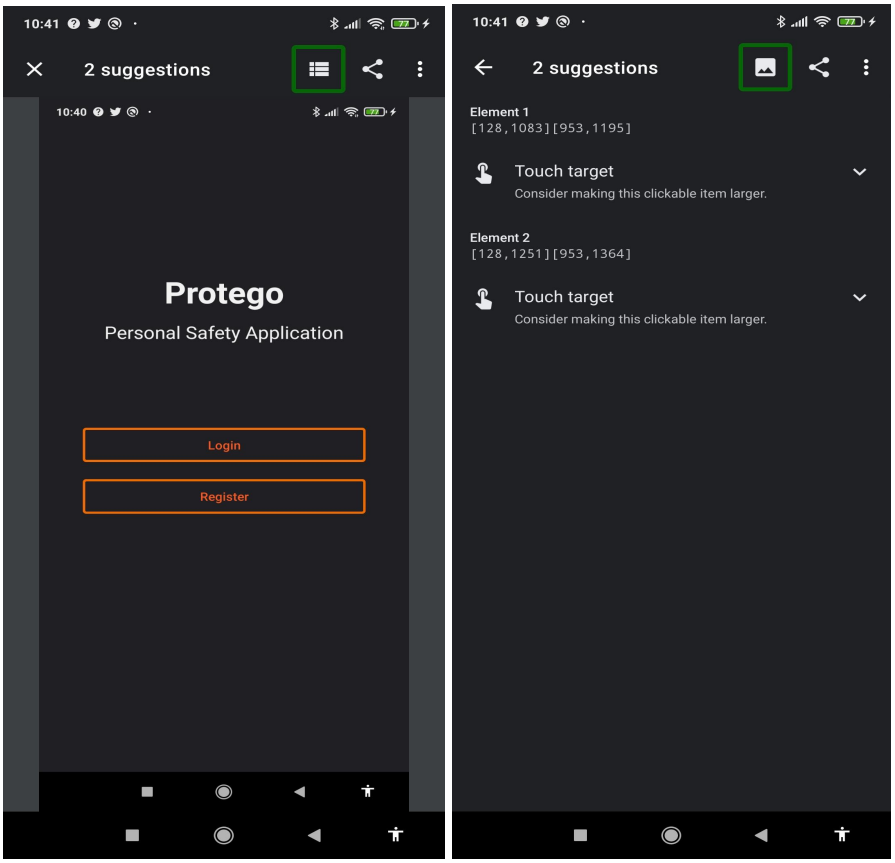
5.4 Accessibility testing

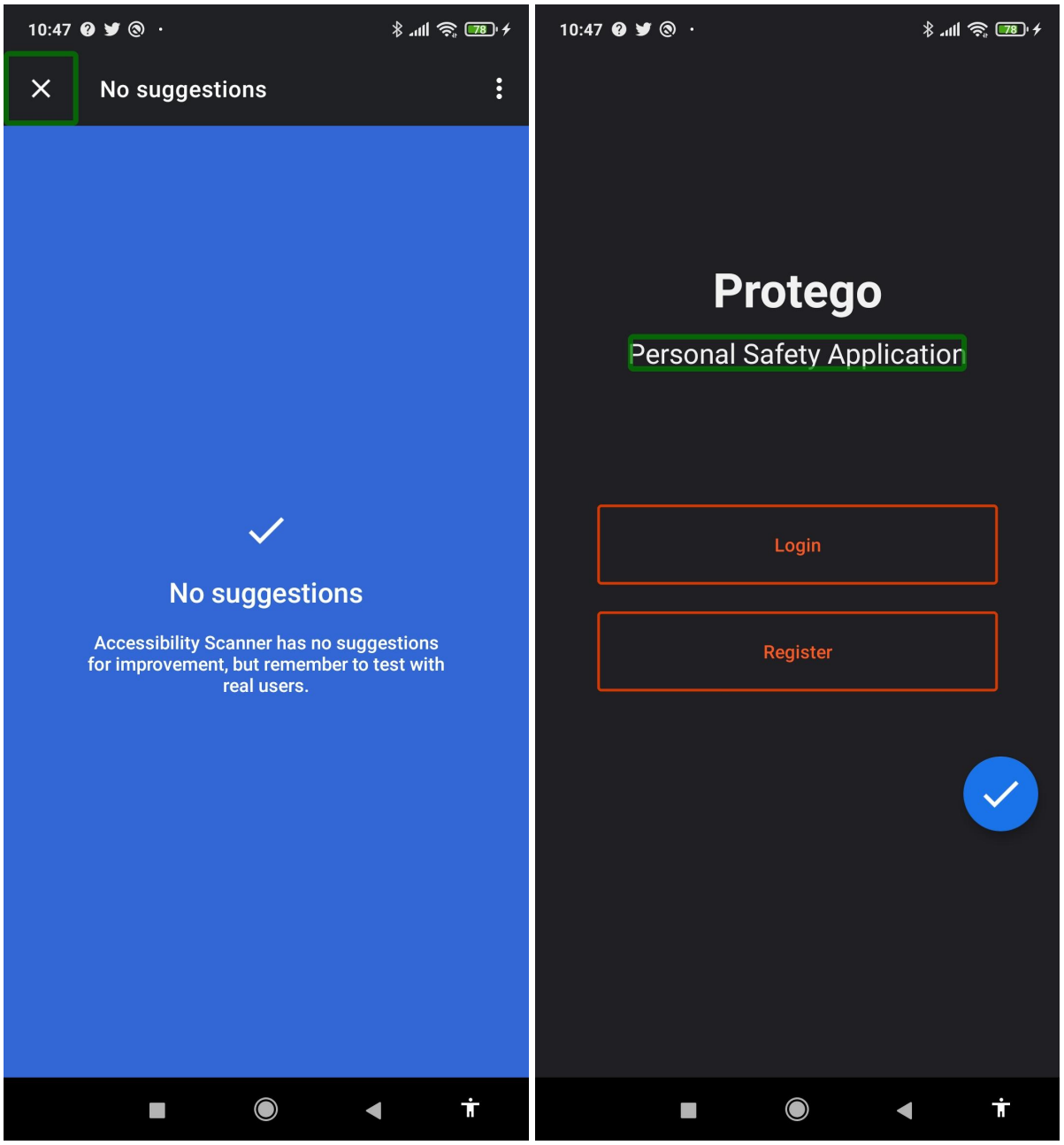
In order to make our application more accessible we consulted the React Native documentation [2]. Since we are using React Navigation in our project, we consulted its documentation to improve accessibility too[3].
As mentioned earlier, Google Accessibility Scanner and TalkBack were used together to assess and improve the accessibility of Protego.

Here are some of the changes that were suggested and implemented into the app.

5.4.1 Make clickable items larger

When testing the application with the scanner we were informed that some of our buttons were not big enough to be considered optimal for accessibility.
By increasing the size of elements, we make it easier and faster to interact with them.





5.5.2 Make elements screen reader friendly

When using Protego with TalkBack activated, we have noticed that some of the elements were not responsive or the screen reader output was not fully understandable. To achieve this, all accessible elements were updated to include the `accessible={true}` property. Additionally, the `accessibilityLabel`

property was also configured, which informs users that are using the VoiceOver functionality about the element that they have selected.

```
<Input
  accessible={true}
  accessibilityLabel="Enter phone number"
  placeholder="Phone number"
  value={guardianNumber}
  keyboardType="phone-pad"
  onChangeText={(text) => {
    setGuardianNumber(text);
  }}
  leftIcon={
    <Icon name="phone" type="font-awesome" size={24} color="#f95a25" />
  }
/>
```

```
<Button
  accessible={true}
  accessibilityLabel={'Add Guardian'}
  accessibilityRole={'button'}
  title="Add Guardian"
  onPress={() => navigation.navigate('Add Guardian')}
  titleStyle={styles.titleStyle}
  buttonStyle={styles.buttonStyle}
  type="outline"
/>
```

6. Future Development

6.1 Voice Commands/Activation

In the initial project design, it was planned to incorporate voice recognition which would allow the user to activate SOS mode with their voice. Although this feature was not able to be carried out during the time period of this project, it is a possible idea for further development of the application.

6.2 Improve notification system

Currently, the only notification implemented is that which is sent to a guardian when a ward activates SOS mode. However, the only way for a guardian to know if the ward has deactivated SOS mode, is for them to check the 'Wards' tab in the guardian manager. The implementation of a push notification would be a much more effective way conveying that.
Additionally, the addition of a request push notification would also be very beneficial. This way, users will know that they have received a request, and whether or not their request has been accepted or rejected.

6.3 Emergency call

The emergency text feature was implemented for users who are unable to make phone calls. In the future, the option to make a call could also be implemented.

7. Appendices

---

[1]. J Nasar, P Hecht, and R Wener. 2007. 'Call if you have trouble': Mobile phones and safety among college students. International Journal of Urban and Regional Research.
[2] https://reactnative.dev/docs/accessibility
[3] https://reactnavigation.org/docs/stack-navigator/