Bug Report

Cycle 1 (06/04/2020)

Git branch with fixes: bug-fixes-1

1. *Screen:* Safe Zone Add. *Issue:* User unable to move the map behind the circle.
   - **Description**

   After the user inputted their desired safe zone address, the safe zone was drawn as a static circle, meaning when the user tried to move/drag the circle to the appropriate spot, nothing would happen. Not only that, but once the view of the circle was shown on the map, the map view would be changed to that region. And when the user touched the map behind the circle, it jittered and jumped endlessly. Below shows the various code snippets and corresponding UI.

   MapView UI code:

```
    <MapView
        style={styles.map}
        provider={PROVIDER_GOOGLE}
        zoomEnabled={true}
        zoomControlEnabled={true}
        showsBuildings={true}
        showsIndoors={true}
        showsUserLocation
        rotateEnabled={true}
        scrollEnabled={true}
        showsCompass={true}
        region={this.state.region}
        onRegionChange={(region) => { this.setState({region})}}>
    {SafeZoneAddress ? SZCircle(circle, radius) : null}
   </MapView>
```

   Map Circle UI code:

```
const SZCircle = (circle, radius) => {
   return (
   <MapView.Circle
   style={styles.fixedCircle}
```

```
        center={circle.center}
        radius={radius}
        strokeColor='transparent'
        />
    )
}
```

UI Screenshot (Before):

- **Resolution**

In this case there were actually two major issues that were disrupting our user experience. First, the static and interactive circle. Second was the jittering of the mapview. In order to resolve these issues, we looked to the github issues of the react-native-map library to try to seek out other developers with similar issues and perhaps possible workaround suggestions.

For the mapview, the source of this issue came from the state controlled region lines `region={this.state.region}, onRegionChange={(region) => { this.setState({region})}}`. In the background, it caused setState to be called continuously, which ultimately caused the jittering behaviour. A small piece of the logging is shown in the logs below:

```
 LOG  region change here: {"latitude": 53.36527074357852, "latitudeDelta": 0.0230412644424689,
"longitude": -6.544010639190674, "longitudeDelta": 0.020000562071799344}
 LOG  region change here: {"latitude": 53.36525433840296, "latitudeDelta": 0.023041273314497346,
"longitude": -6.543921455740929, "longitudeDelta": 0.02000056207180112}
```

```
 LOG  region change here: {"latitude": 53.36524033397982, "latitudeDelta": 0.02304128088815105,
"longitude": -6.543844677507877, "longitudeDelta": 0.02000056207180112}
 LOG  region change here: {"latitude": 53.36522833018488, "latitudeDelta": 0.02304128737984712,
"longitude": -6.543779298663139, "longitudeDelta": 0.02000056207180112}
 LOG  region change here: {"latitude": 53.36521852708316, "latitudeDelta": 0.023041292681398318,
"longitude": -6.543726660311222, "longitudeDelta": 0.02000056207180112}
 LOG  region change here: {"latitude": 53.36521112473956, "latitudeDelta": 0.023041296684617407,
"longitude": -6.543685756623745, "longitudeDelta": 0.020000562071799344}
 LOG  region change here: {"latitude": 53.365205923091935, "latitudeDelta": 0.02304129949769873,
"longitude": -6.543656922876835, "longitudeDelta": 0.020000562071799344}
 LOG  region change here: {"latitude": 53.3652027220777, "latitudeDelta": 0.023041301228815314,
"longitude": -6.543640159070492, "longitudeDelta": 0.020000562071800232}
 LOG  region change here: {"latitude": 53.3652019218241, "latitudeDelta": 0.023041301661606894,
"longitude": -6.543635465204716, "longitudeDelta": 0.020000562071799344}
```

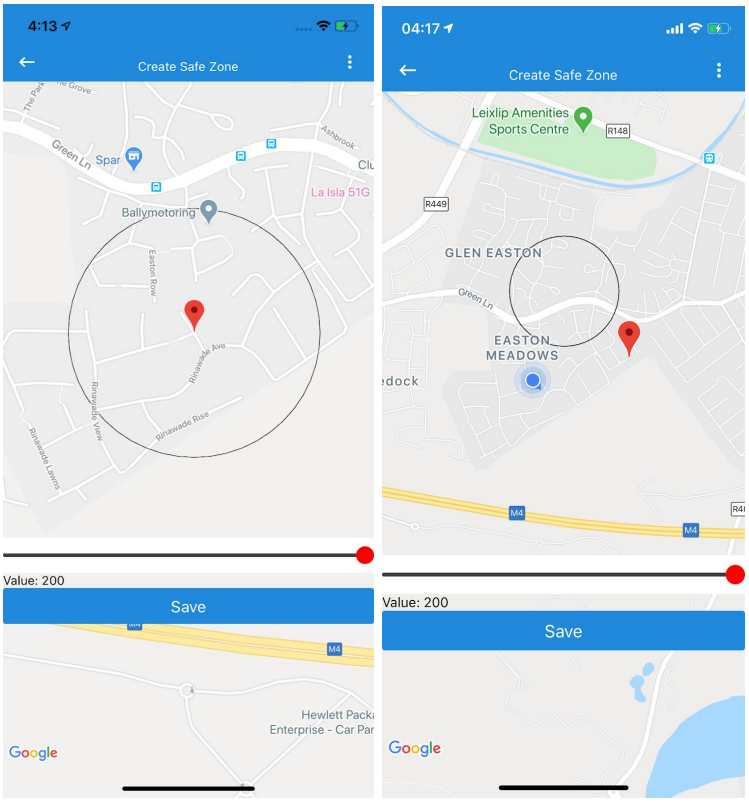
The solution we ended with was to adopt the Animated MapView and to wrap it within a **SafeAreaView**, avoiding unnecessary behaviour.

According to the react-native-maps documentation, there is no possible way to drag the circle or to smoothly move the maps underneath a circle overlay. Instead, we learnt that the only possible draggable component was a marker. This in turn led to an even better implementation. So the alternative was to take the inputted address, get the geographical coordinates and use it for the marker but to also use it for the circle's center point. That being when the user wants to adjust the exact point of their safe zone all they would need to do would be to drag and drop the marker, moving the circle along with it. Through using the key property within the circle component, react native recognises it as a new component and will re-render it. This allowed us to smoothly allow the user to adjust that circles' radius too. This was done through the following below code:

```
  <MapView.Marker
          style={{position:"absolute"}}
          ref={marker => { this.marker = marker }}
          key={(circle.center.longitude + circle.center.latitude).toString()}
          coordinate={circle.center}
          draggable={true}
          onDragEnd={(e) => {
            this.getNewAdress(e.nativeEvent.coordinate)
            this.setState({circle: {center: e.nativeEvent.coordinate},
            currentRegion:{latitude: e.nativeEvent.coordinate.latitude, longitude :
e.nativeEvent.coordinate.longitude, longitudeDelta: longitudeDelta, latitudeDelta:
latitudeDelta}})}}
              >
               </MapView.Marker>
               : null }
          {SafeZoneAddress ?
          <MapView.Circle
```

```
          key={(circle.center.longitude + circle.center.latitude +
radius).toString()}
               center={circle.center}
               radius={radius}
               strokeColor='transparent'
               />
               : null }
```

UI Screenshot (After):



2. *Screen:* Medical Profile Setup. *Issue:* Ongoing medical profile notification shows in foreground every minute. (iOS)
    - **Description**

Upon researching a way to display the medical profile within a user's lock screen, the first idea that came to mind was to create a longstanding notification. We quickly discovered that this was very much possible on the Android side but due to the iOS permissions, we need to uncover a work around. So for iOS on the first development cycle we created the notification using the react-native-push-notification library through the follow code:

```
if(Platform.OS === 'ios'){
   //on going notification isn't possible with ios. WORKAROUND -> schedule a
repeated notification.
```

```
   let ios_options = {
     date: new Date(Date.now() + (30000)), // in 30 secs
     repeatType: 'minute',
     id: '0',
     message: `Age: ${medical_data.age}\nSex: ${medical_data.sex}\nPrimary Carer:
${medical_data.doctor_name}\nKin: ${medical_data.kin}\nBlood type:
${medical_data.blood_type}\nConditions: ${medical_data.conditions}\nAllergies:
${medical_data.allergies}\nMedication: ${medical_data.medication}\nAdditional:
${medical_data.additional}`,
     color: "blue",
     alertAction: 'view',
     visibility: "public",
     title: "Medical Profile",
     playSound: false,
     foreground: false,
   };

   this.notif.scheduleNotif(ios_options)
 }
```

What this did was to repeat scheduled notifications every minute. We had hoped that this wouldn't create a new notification object if the user hadn't interacted with those previous, but it did, causing a bad experience through that repeated notification appearing every minute. Ultimately disrupting the user.

UI Screenshot:

- **Resolution**

Despite there being no real appropriate workaround to achieving an ongoing notification in iOS, instead we changed the repeated time from minute to day, and if the user was to click on the notification, it would be cancelled and a new one would be triggered using the same static id.

```
onNotif(notification) {
   const {navigate} = this.props.navigation
   console.log(notification);

     if(notification.userInteraction){
       if(notification.data.notificationType === "medical" && Platform.OS === "ios"){
         this.notif.cancelLocalNotifications({id:"0"})
firebase.database().ref(`medical/${this.state.userId}`)
         .on('value', snapshot => {

           const { additional, age, allergies, blood_type, conditions, doctor_name, kin, medication, sex } =
snapshot.val();

           let ios_options = {
             date: new Date(Date.now() + (30000)), // in 30 secs
             repeatType: 'day',
             id: '0',
             color: "blue",
             alertAction: 'view',
             visibility: "public",
             title: "Medical Profile",
```

```
            message:`Age: ${age}\n--------------\nSex: ${sex}\n--------------\nDoctor:
${doctor_name}\n--------------\nKin: ${kin}\n--------------\nBlood type:
${blood_type}\n--------------\nConditions: ${conditions}\n--------------\nAllergies:
${allergies}\n--------------\nMedication: ${medication}\n--------------\nAdditional: ${additional}`,
            playSound: false,
            foreground: false,
            userInfo: {
              notificationType: "medical",
             },
             data: JSON.stringify({notificationType: "medical"})
          }
         this.notif.scheduleLocalNotification(ios_options)
        })
    }else{
       navigate(notification.data.screen,{user_id:notification.data.sender_id})
      }
  }
}
```

To prevent the notification being displayed within the application foreground, we modified the UNUserNotificationCenter `willPresentNotification` function within the *IOS/AppDelegate.m* file, which is used to manage all notification-related behaviors in the app.By adding this check, it meant that only notifications which weren't of type medical would be shown in the foreground.

```
// Manage notifications while app is in the foreground
- (void)userNotificationCenter:(UNUserNotificationCenter *)center
willPresentNotification:(UNNotification *)notification
withCompletionHandler:(void (^)(UNNotificationPresentationOptions
options))completionHandler
{
 NSLog(@"User Info : %@",notification.request.content.userInfo);
 NSDictionary *userInfo = notification.request.content.userInfo; // read apns
payload
 NSString *notificationType = userInfo[@"notificationType"]; // check your
notification type
 if (![notificationType  isEqual: @"medical"]) { // we silence all notifications
which do not match our custom notification type
   NSLog(@"Type : %@",notificationType);
   completionHandler(UNNotificationPresentationOptionNone);
   return;
 }
  NSLog(@"Else Type : %@",notificationType);
 // custom notification type is displayed normally
 completionHandler(UNAuthorizationOptionSound | UNAuthorizationOptionAlert |
UNAuthorizationOptionBadge);
}
```

```
[1]
```

3. *Screen* :UserProfileScreen. *Issue*: User Profile picture not updating automatically

- **Description**

Originally when the user had selected a new image from their gallery/camera, it would trigger a change in the stored picture for that current user id within the cloud storage and also the corresponding url within the database that would be leveraged in order to download that very picture. However, when a user would attempt this, there would be a delay in between the time taken for the new picture to be uploaded, overwrite the old image, and display that picture on the user's profile (~60secs). The reasoning behind this was due to the slowness of uploading both the picture, retrieving its corresponding downloadable url (specifically this part) and then updating the state.

- **Resolution**

When the image is retrieved, rather than waiting for the upload to complete, we set the state of imagesource to be the uri of the uploaded picture when it is selected. That being when the imagesource is no longer null, that imagesource will be displayed rather than the old picture url for the user . There is no effect on the user end for this operation. As the new image transitions just as it would if uploading to cloud storage was quicker. Also, if the user was to exit the screen and reenter the screen, the url would have been loaded and completed by then and the new image would be displayed for the user as it appears in the storage.

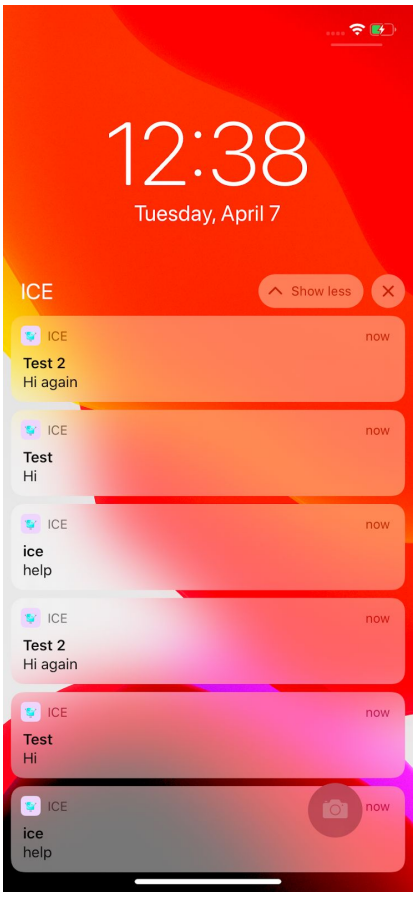
```
    <Avatar
      containerStyle={{alignSelf:"center", marginTop: 20}}
      size="xlarge"
      rounded
      onEditPress={()=>{this.changeProfilePic()}}
      source={ this.state.ImageSource !== null ? this.state.ImageSource
        : {uri: this.state.profile_pic_url }
      }
      showEditButton
      />
```

4. *Screen*: HomeScreen *Issue*: Notifications are retriggered on each app reload/refresh

- **Description**

On every reload occurrence, all existing notifications for the current user were reloaded and showed to the user. This was ultimately caused by the below code snippet, especially with the `**child_added**` database listener type. With using that type, it meant it would be triggered whenever a new child was added to that branch, however, if re-loaded or on first occurrence, it would first call on every child of the branch.

Screenshot (before):




```
firebase.database().ref(`notifications/${this.state.userId}`)
.on('child_added', snapshot => {

  const { msg, screen, sender_id, time, title } = snapshot.val();
  const { key: id } = snapshot;

  console.log("screen",screen.toString())
  let options = {
    id: id,
    autoCancel: false,
    bigText: msg,
    ongoing: true,
    priority: "high",
    visibility: "public",
    importance: "high",
    title: title,
    message: msg,
    playSound: true,
    vibrate: true,
    userInfo: {
```

```
                screen: screen.toString(),
                sender_id: sender_id.toString(),
              }, //for ios
              data: JSON.stringify({screen: screen.toString(), sender:
sender_id.toString()})
            }
            this.notif.localNotification(options);
    })
  }
```

- **Resolution**

The child_added database listener was best suited for this case, so we didn't want to change it. So instead we added the line:

```
firebase.database().ref(`notifications/${this.state.userId}/${id}`).remove()
```

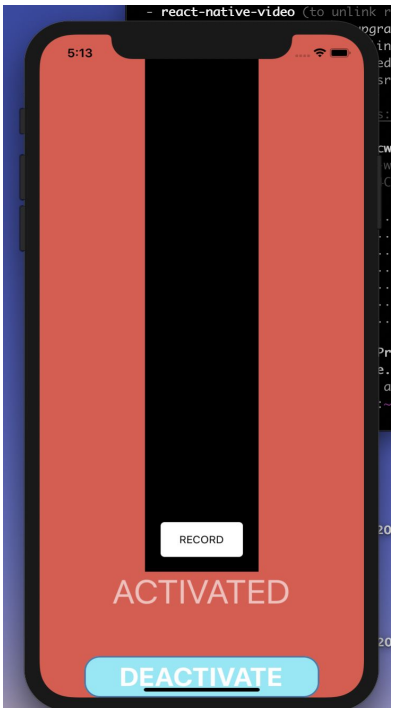
This meant that once a notification was created and shown, it would be removed from the database and not reshow to the users. However we did utilize that branch for the notification feed too. So for that aspect, we need to create a seperate notification_feed child node in our database which would not be affected by the popping of notifications.

5. *Screen*: SOS-Activated. *Issue*: Recording a SOS video required user initialization first.

- **Description**

The point of this screen was so that upon a user activating SOS mode, that being through either saying their safe word, shaking the device or pressing the SOS button, it would record the audio and camera of the user's device. However, originally, this required the user to then press record also, else no footage would have been captured. This impeded our use case. In an attempt to record automatically, we triggered the startRecord() function within componentWillMount, but we quickly discovered that direct recording wasn't directly possible when the camera component was rendered without any further interaction from the user.

UI Screenshot (Before):



- **Resolution**

We realised that the underlying issue to this was solely due to the app's access permissions. Without the user actually interacting with the camera component (via a touchable component), the permission check wouldn't be triggered and ultimately the camera would fail. As a workaround to the above, we manually triggered the `_requestPermissions()` function and when granted, the camera would be within a ready state, we manually set recording to be true and we could then trigger the startRecord() function. Whoever, checking those permissions and setting the recording state took a few milliseconds to complete, so we added a setTimeout function, which would start the recording process 50 milliseconds after being triggered, allowing for a smoother user experience. If retrieving the permissions failed or weren't granted, a simple Text component would be displayed to inform the user rather than an error being thrown.

Below shows the recording process with the timeout:

```
startRecording = () => {
   setTimeout(async () => {
       try{
       const options = {
         maxDuration: 4,
       }

       const { uri, codec = 'mp4' } = await this.camera.recordAsync(options);
       // / Preview
       this.setState({ recording: false, processing: true });
       const type = `video/${codec}`;
```

```
        const data = new FormData();
        data.append("video", {
          name: "mobile-video-upload",
          type,
          uri
        });
        this.setState({ processing: false });
        this.uploadVideo(uri);
      } catch (e) {
        // ignore
      }
    }, 50)
  }
```

Below: The camera will only start recording once the state is set and the camera is in a ready position.

```
       this._requestPermissions()
       if(this.camera){
         this.setState(
           { recording: true, processing: false, preview: false, uri: null, codec:
null },
           this.startRecording.bind(this)
         )
       }
```

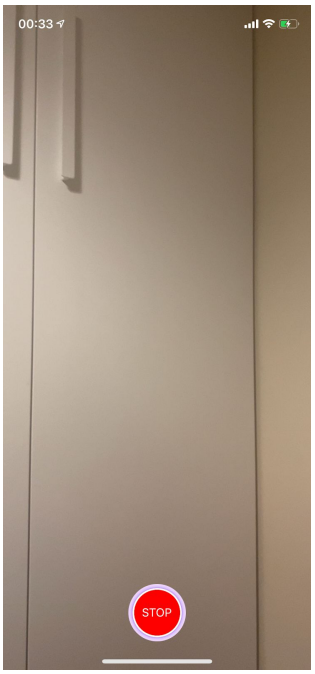
Below: Retrieving the device permissions

```
     _requestPermissions = async () => {
       console.log("Platform.OS",Platform.OS)
       if (Platform.OS === 'android') {
         const result = await
PermissionsAndroid.request(PermissionsAndroid.PERMISSIONS.CAMERA)
          this.setState({ permissionsGranted: true });
         return result === PermissionsAndroid.RESULTS.GRANTED || result === true
       }else{
         this.setState({ permissionsGranted: true });
         return true
       }
     }
```

Below: The camera component:

```
{ this.state.permissionsGranted ?
   <RNCamera
      ref={ref => {
       this.camera = ref;
     }}
     style={styles.preview}
     type={RNCamera.Constants.Type.back}
     flashMode={RNCamera.Constants.FlashMode.on}
     // onStatusChange={this.onCameraStatusChange}
     // onCameraReady={this.onCameraReady}
     androidCameraPermissionOptions={{
       title: 'Permission to use camera',
       message: 'We need your permission to use your camera',
       buttonPositive: 'Ok',
       buttonNegative: 'Cancel',
     }}
     androidRecordAudioPermissionOptions={{
       title: 'Permission to use audio recording',
       message: 'We need your permission to use your audio',
       buttonPositive: 'Ok',
       buttonNegative: 'Cancel',
     }}
   >
   </RNCamera> :
       <Text>Camera permissions not granted</Text>
   }
```

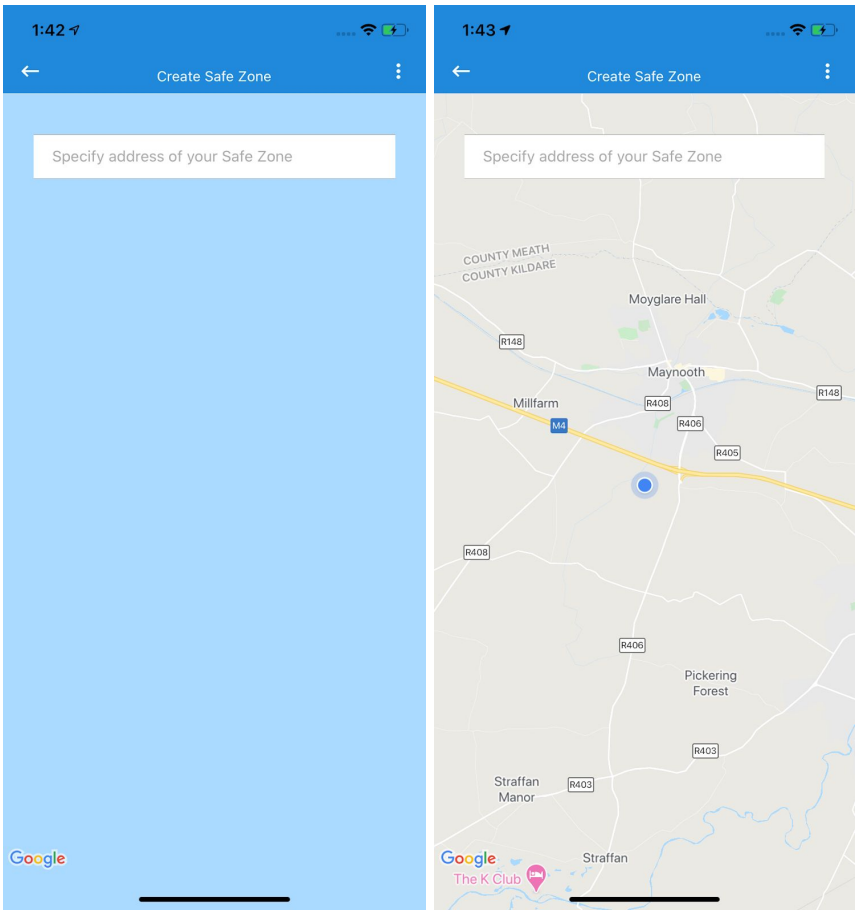
UI Screenshot (After):



6. *Screen*: SafeZoneAdd. *Issue:* Map initially showed sea coords did not refresh to show user location/region from (0,0) until manually refreshed:
    - **Description**

We had used navigation.geolocation to get the device current location but this wasn't great due to its lag in retrieving the device position quickly. We had initially set the state latitude and longitude to be 0, to act as placeholders for the actual data to replace it. But the unreliability of navigator.geolocation meant that sometimes it wasn't triggered at all until being reloaded, which ultimately left the mapview to appear blue, stuck on the coordinates (0,0).

UI Screenshots (1st shows the initial view and 2nd shows one after refresh):



- **Resolution**

This was quite a simple quick. By passing the coordinates from the parent screen to act as the default, it would give it enough time to load the map with the appropriate coordinates. So within the initialization of this.state, we adjusted latitude and longitude to:

```
latitude : this.props.navigation.getParam('latitude'),
longitude: this.props.navigation.getParam('longitude')
```

References and Guides:

[1]https://github.com/react-native-community/push-notification-ios/issues/63#issuecomment-583618575