Final Report

Asteroid Tracker

Tejas Dhanani

Advisor: Sourav Mukherjee

Date (dd/mm/yy): 10/12/21

Fairleigh Dickinson University Vancouver Campus
Bachelors of Science in Information Technology

1 Introduction:

According to United Nations Office for Outer Space Affairs (*unoosa.org/*), "Near-Earth Objects, or NEOs, represent potentially catastrophic threats to our planet. A near-Earth object is an asteroid or comet which passes close to the Earth's orbit." Researchers in astronomy and space objects observers have to study and research on near earth objects (NEOs) which are approaching earth. Therefore, it is hard for them to constantly go through the raw data on the website which *NASA's NEOWs (Near Earth Object Web Service) API* is providing. In order to solve this problem, I will be developing a mobile application using NASA's NEOWs API [API Link] where user can view NEOs closest approach date, speed and diameter and allow them to set a email notification by closest approach time which will help astronomy community and it's fan to get this data in a easy way.

2 Related Work:

There are some websites like `https://theskylive.com/near-earth-objects` which provide this data into raw format but it automatically displays the data of asteroids which are approaching in the future and it does not allow user to query the data by specific date. My approach would be to allow the user to configure the data by selecting the date and where a user can set an email notification timer for that event.

3 Approach:

I have solved this problem by creating a mobile application which can work on both platforms i.e. iOS and Android. I have used the following software architecture:

- React Native
- Node.js
- Javascript
- Firebase
- Expo

Use of fetch method to get the data from NEOWs API. Whenever each user opts for email notification for a specific event, that data will be posted into firebase and firebase cloud functions will help us sending the notification at a scheduled time.

A laptop with a Windows 10 Operating System to build the application and 2 phones (iOS and Android) would be used as it makes a good combination for testing and stability purposes. React Native programming language will be used as it provides us to build applications for both iOS and Android. React Native is faster because it can style UI components and it provides re-usability of code through it's components. Expo will be used to create a localhost server in order to test the app on mobile devices. Libraries like react-native-navigation to navigate between app's screens and firebase to store client's email address and triggering email jobs will be used.

4 Implementation:

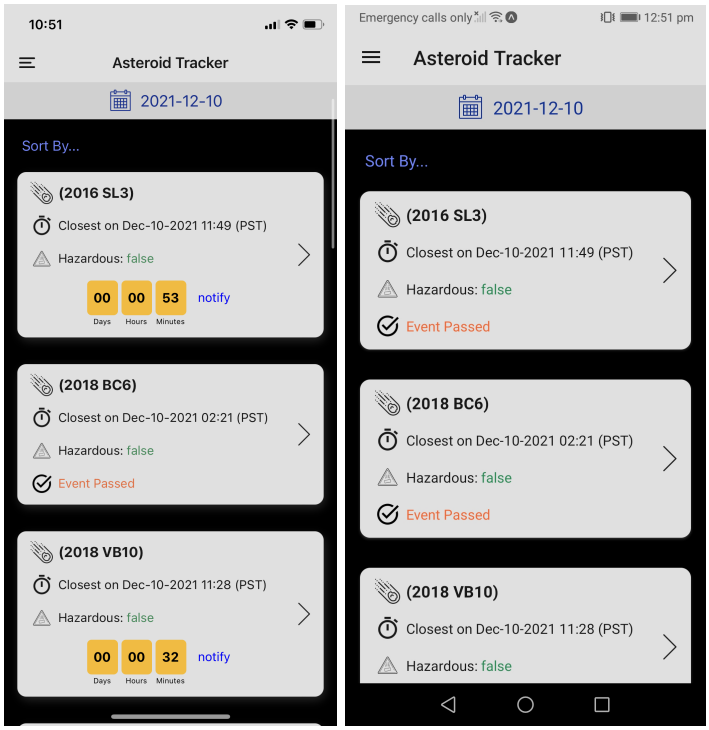
Asteroid Tracker application allows user to display all the (today's) Near Earth Objects (NEO) by default in a small view on the front page. The small view of each NEO consists of its:

- *name*
- *time*
- *hazardous-ness*
- *Event Timer*

The event timer object shows "Event Passed" if the Date.now() is greater than asteroid closest approach Date(), else it displays the countdown timer with the remaining time. It also has the button "notify" to set an email notification reminder for that event.

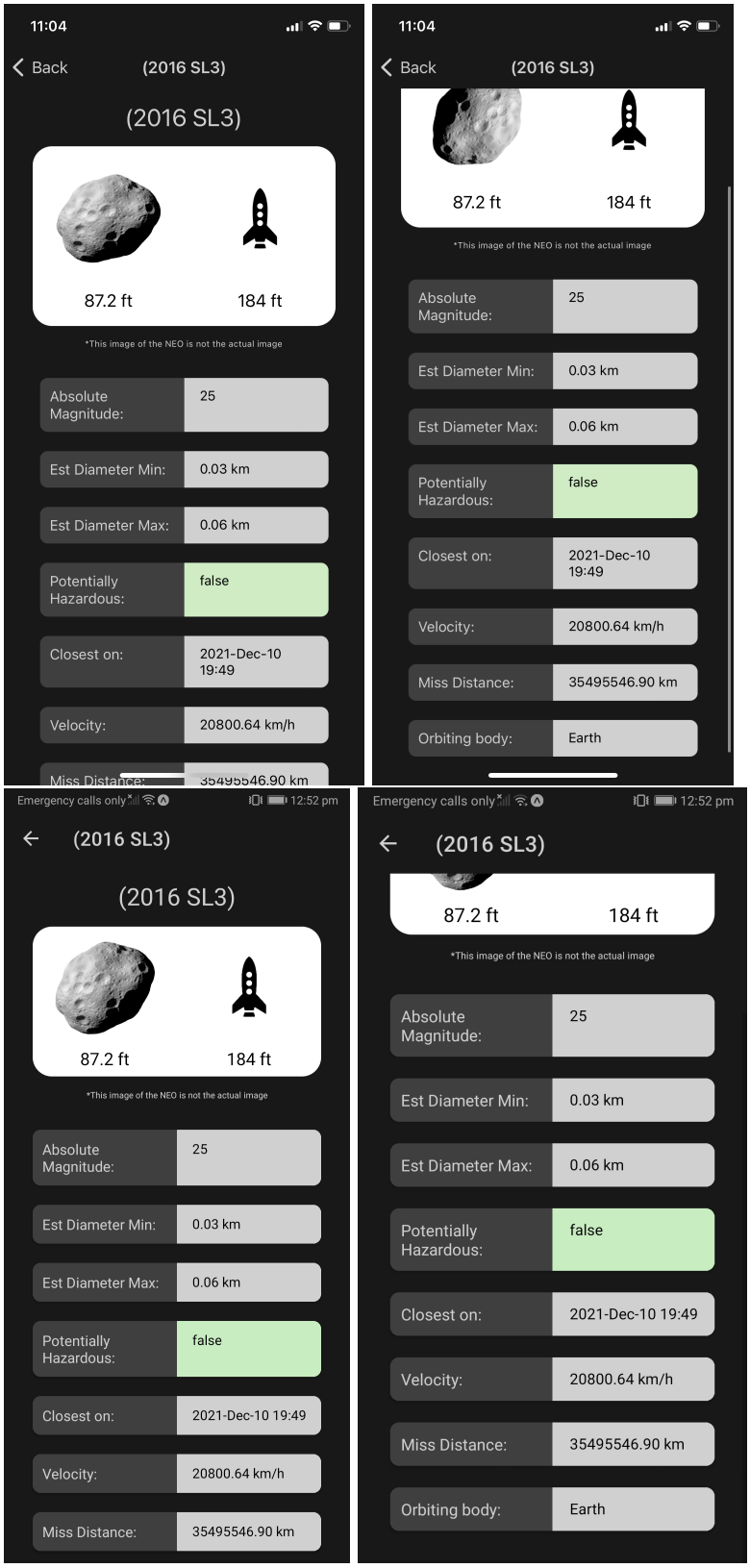
The front page or the loading page of the app contains of multiple small boxes where congregated information of all NEO's are present.

NOTE: There will be two types of images, one tested on iPhone iOS v15.1 and another tested on Honor EMUI Android v9.1.0

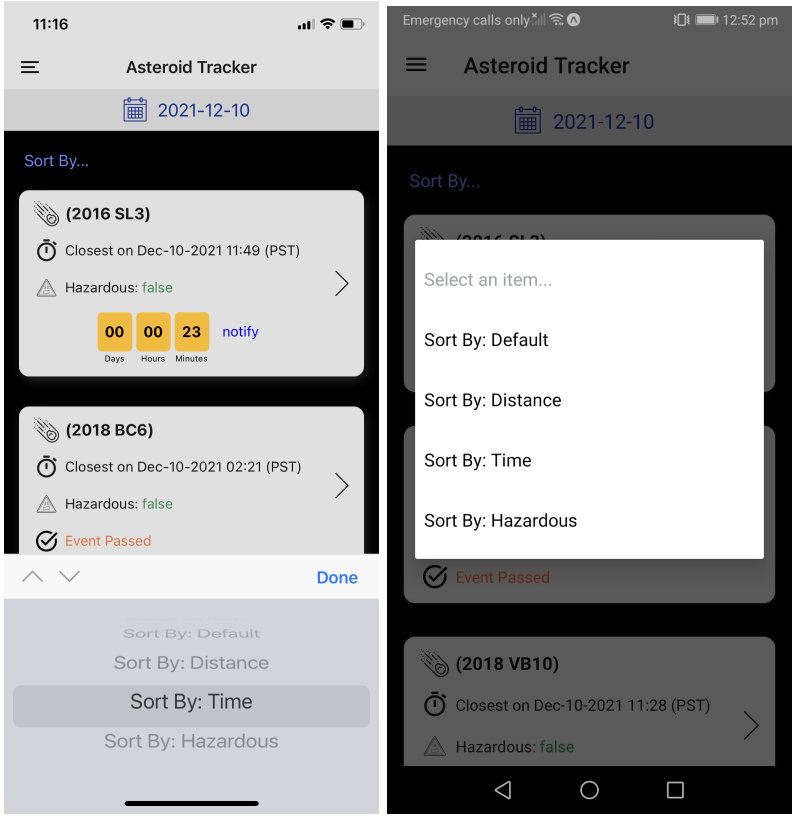


When one of the NEO is clicked by tapping once it open it's own page where it shows information about that NEO in more detail. Details include Absolute magnitude, maximum estimated diameter, minimum estimated diameter, hazardous-ness, closest approach date, speed, miss distance from Earth and the orbiting body. This page also has an asteroid image (taken from https://www.pngaaa.com/detail/849234) which is used for every NEO which means it is not the actual image of the asteroid. It gives dynamic look by randomly rotating the image by 360 degree when this page of each NEO loads.

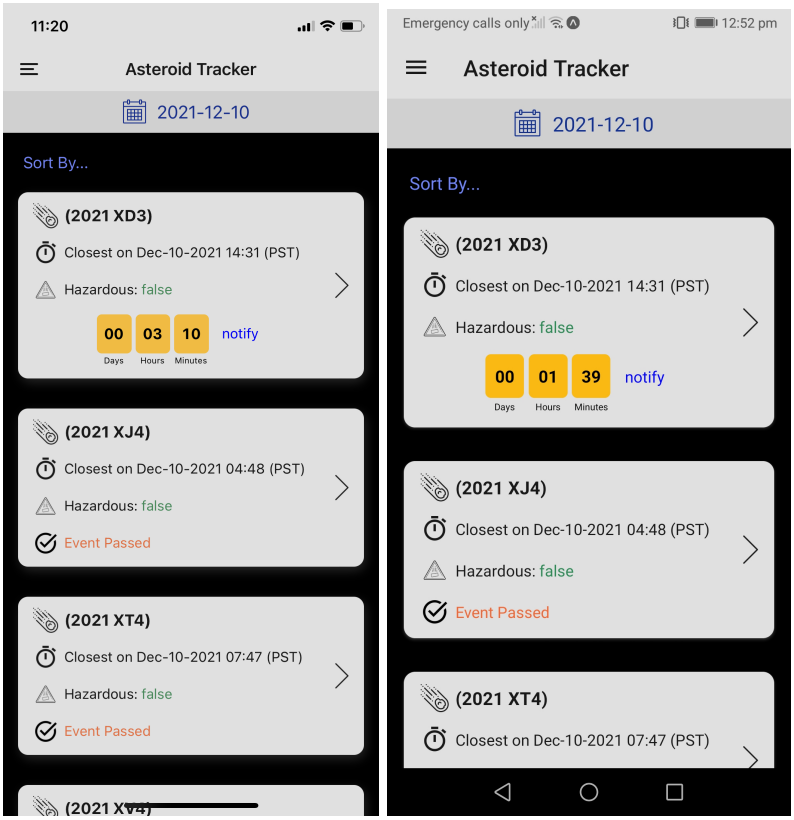
It also shows the static comparison between the size of the asteroid and a average size of space shuttle from NASA. Due to technical challenges, the feature of upscale-ing and downscale-ing the image of the space shuttle based on the size difference is not available at the moment but it might be available in future updates.



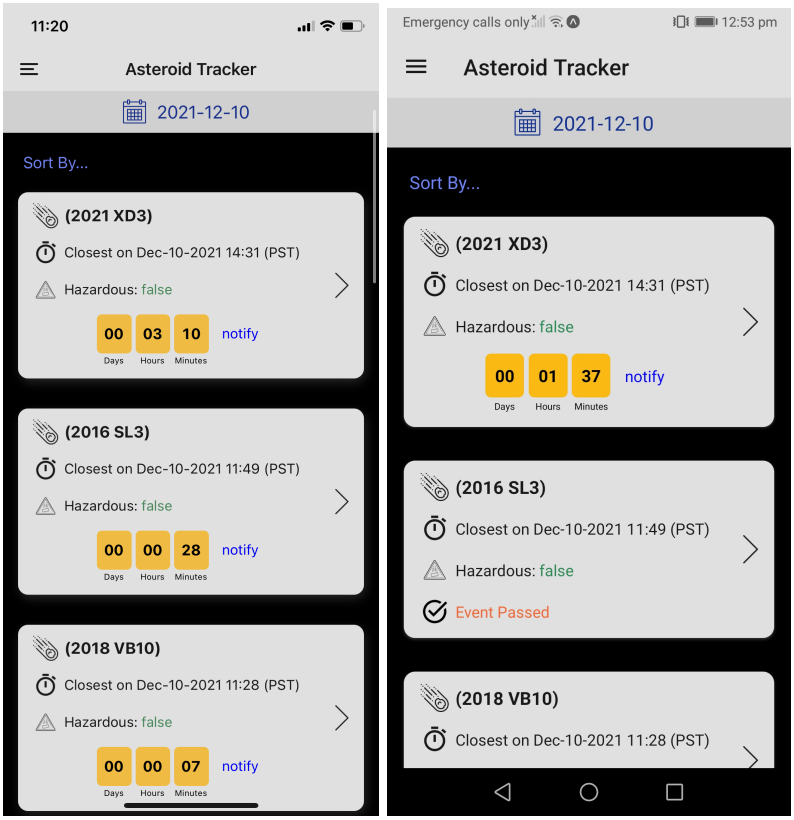
At this stage, the application also can sort the data by the following:



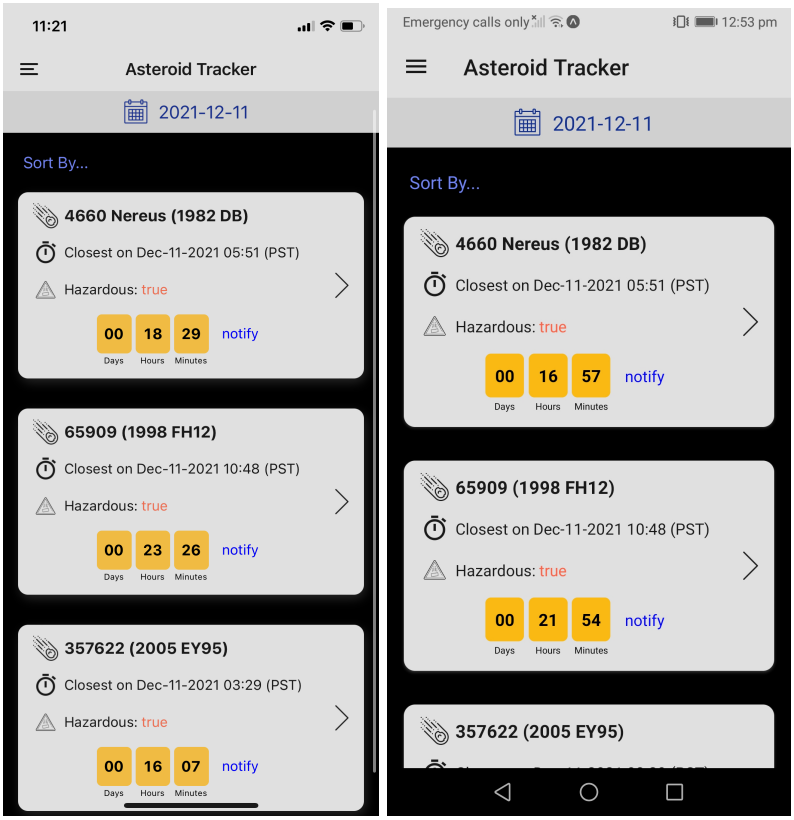
- Miss Distance (Closest first)



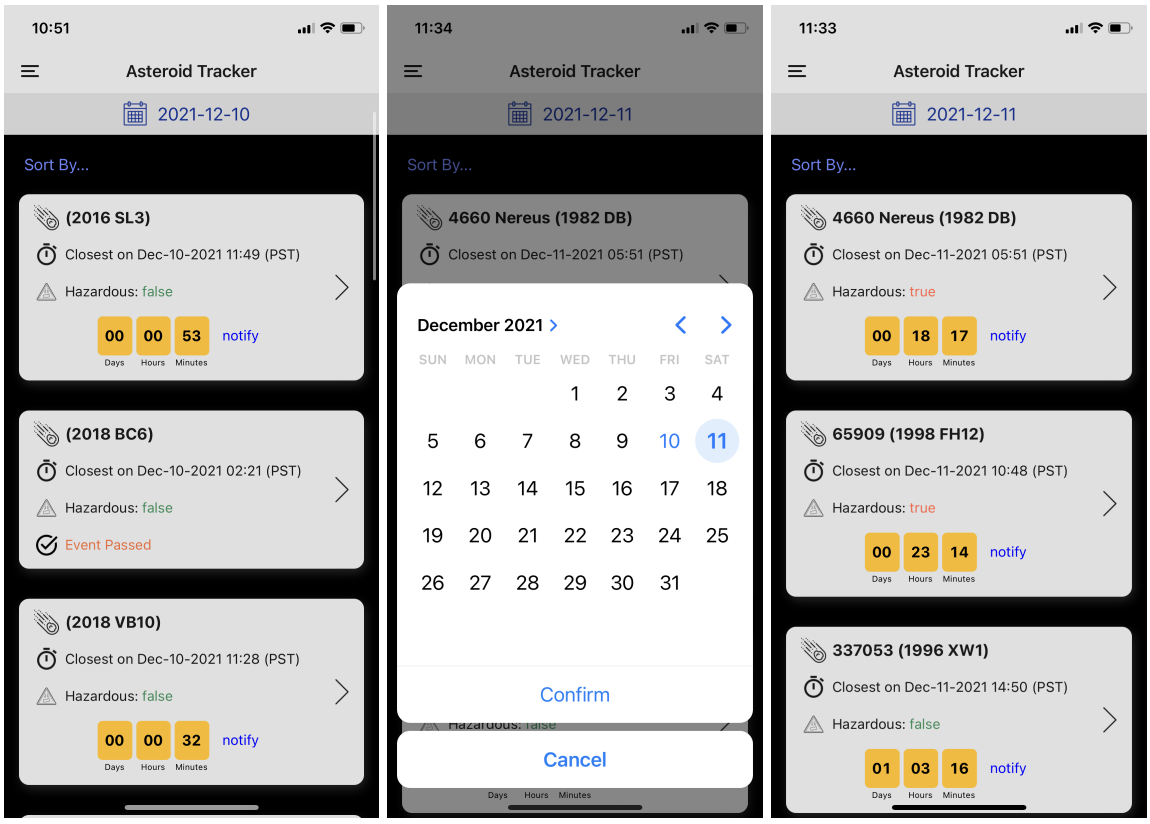
- Time

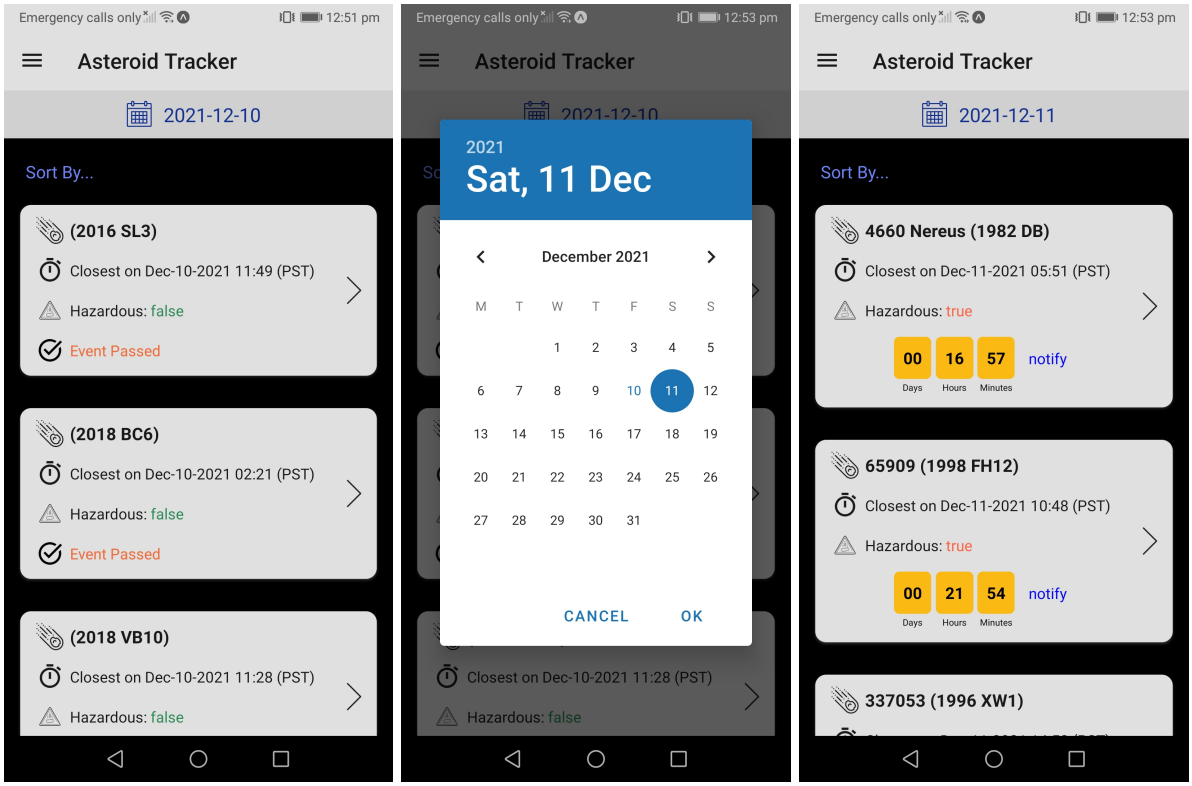


- Hazardous (shows objects with *true* hazardous attribute)

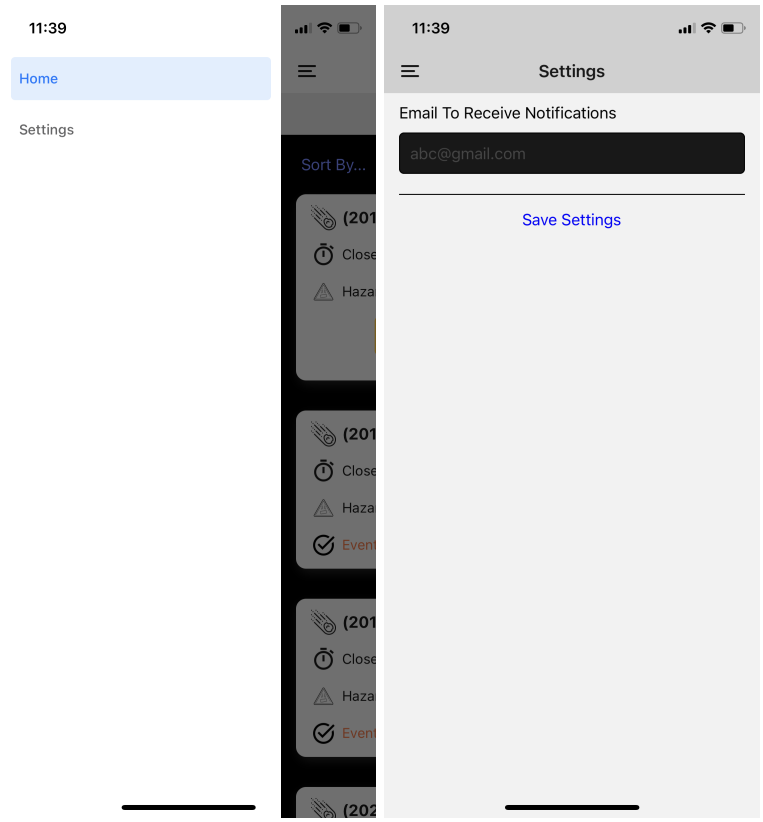


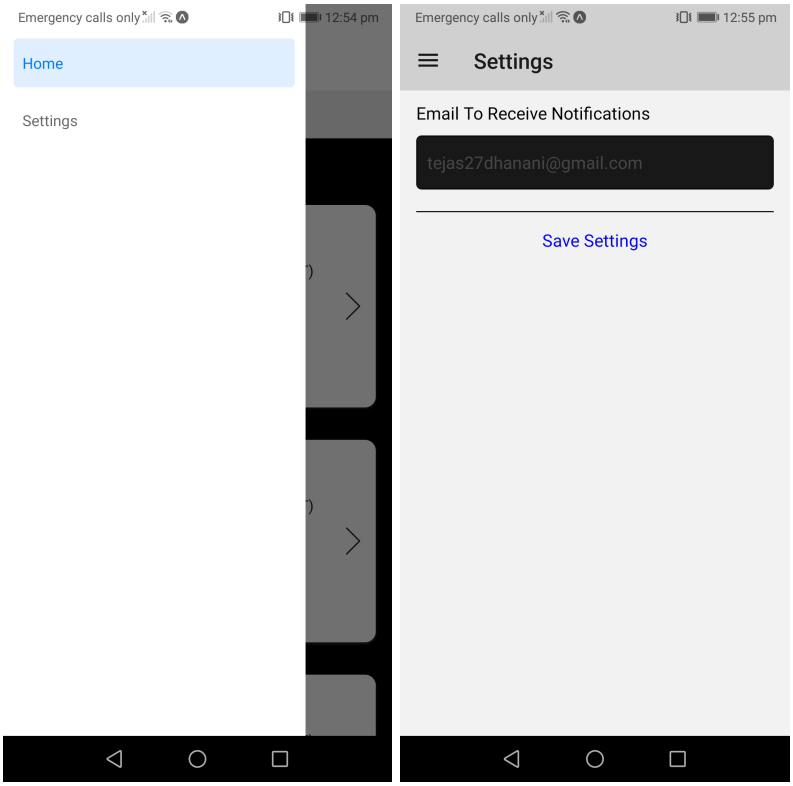
At this stage, the user can also select the date and retrieve the data. To open the calendar, the user have to tap on the date which is shown by default (here - 2021-12-10). In the below example, the user taps on 2021-10-11 and selects the date. The layout is updated and one can verify it by looking at the closest approach date which is second item on the NEO box.



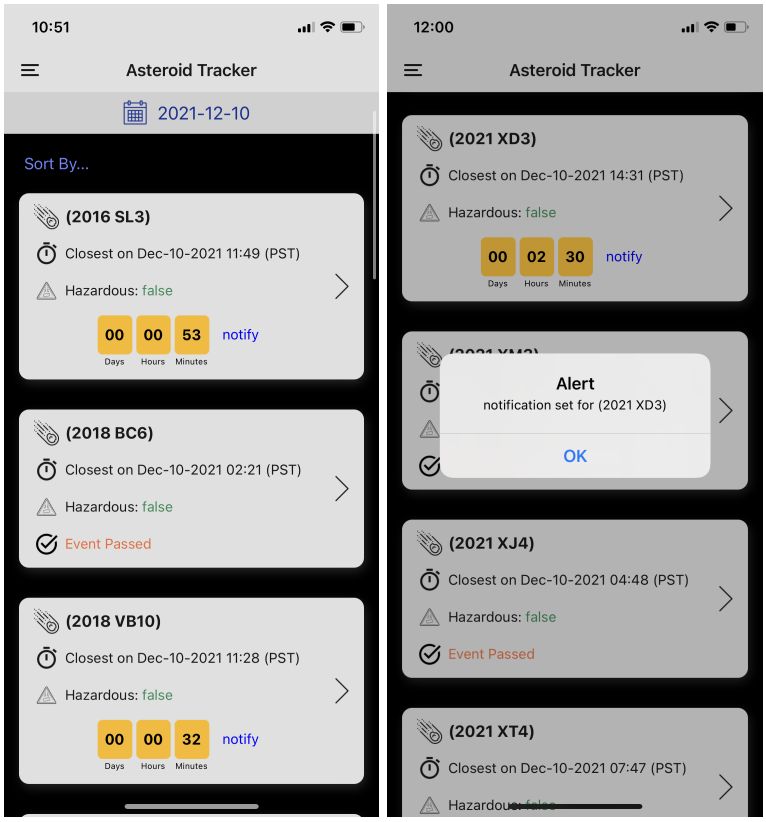


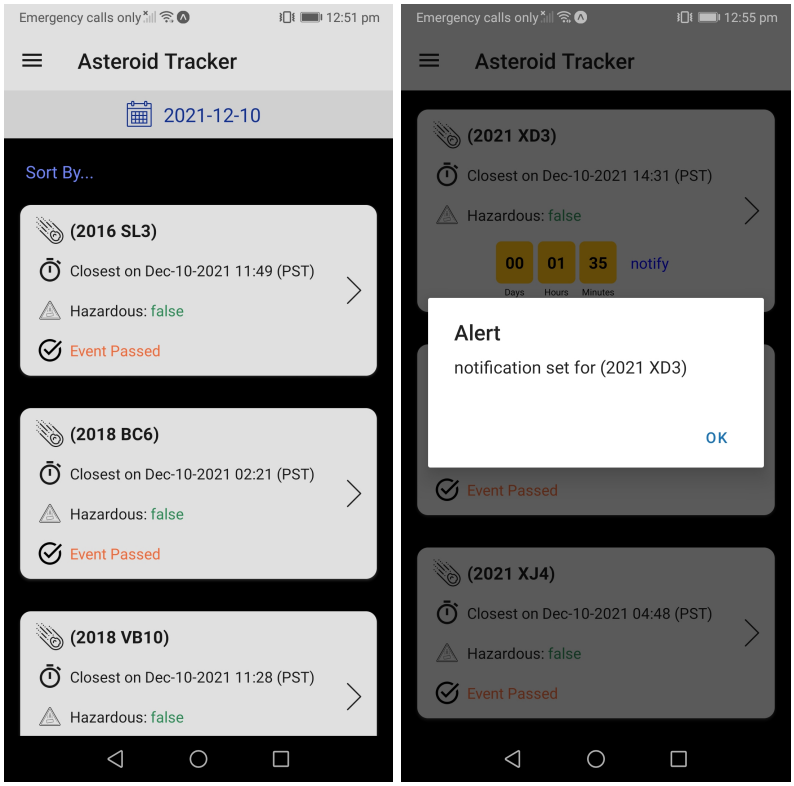
User can also click on the 'three lines' located on the top left corner of the front page to open the navigation drawer in order to find more options. The navigation drawer consists of two pages - (i) Home and (ii) Settings. Settings screen has an area for user to add email address and saving the changed settings. After user saves the updated email address, it saves to phone's local storage with the help of Async-Storage (https://react-native-async-storage.github.io/).



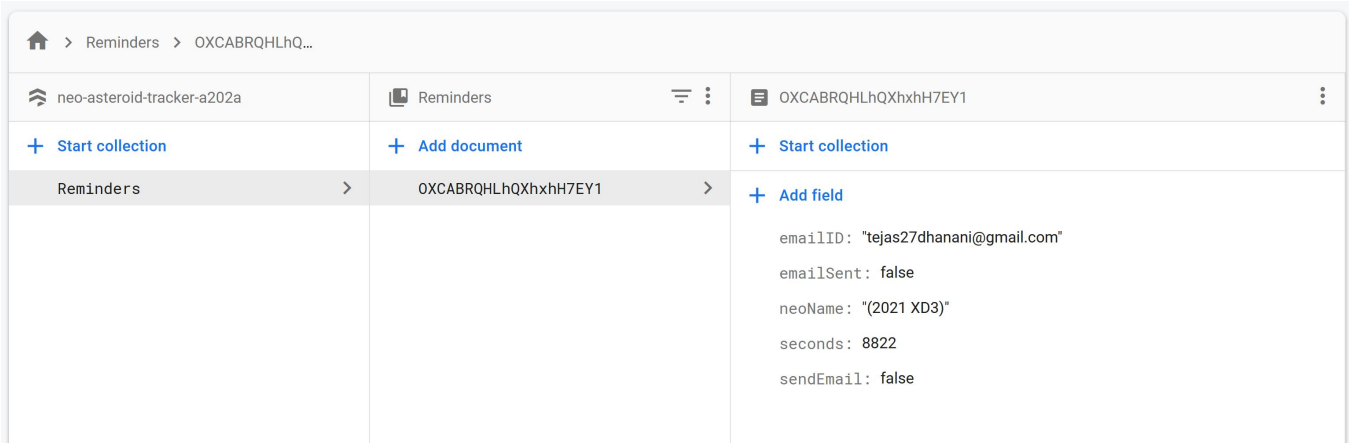


When the user clicks on the notify and if they have provided their email address on the settings screen, it will show a pop-up like "notification set for ${AsteroidName}".

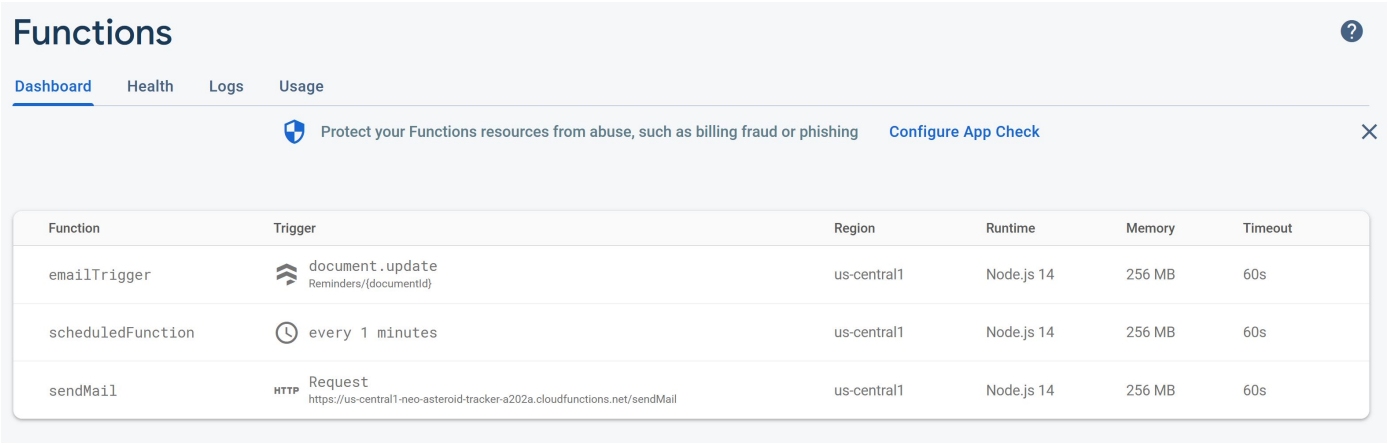




After the notification has been set, it creates an document in firebase firestore database.



By default, the `sendEmail` and `emailSent` fields are both `false`. And, `seconds` shows the number of seconds remaining to send an email. Now to update these fields and to send an email we will use firebase cloud functions.

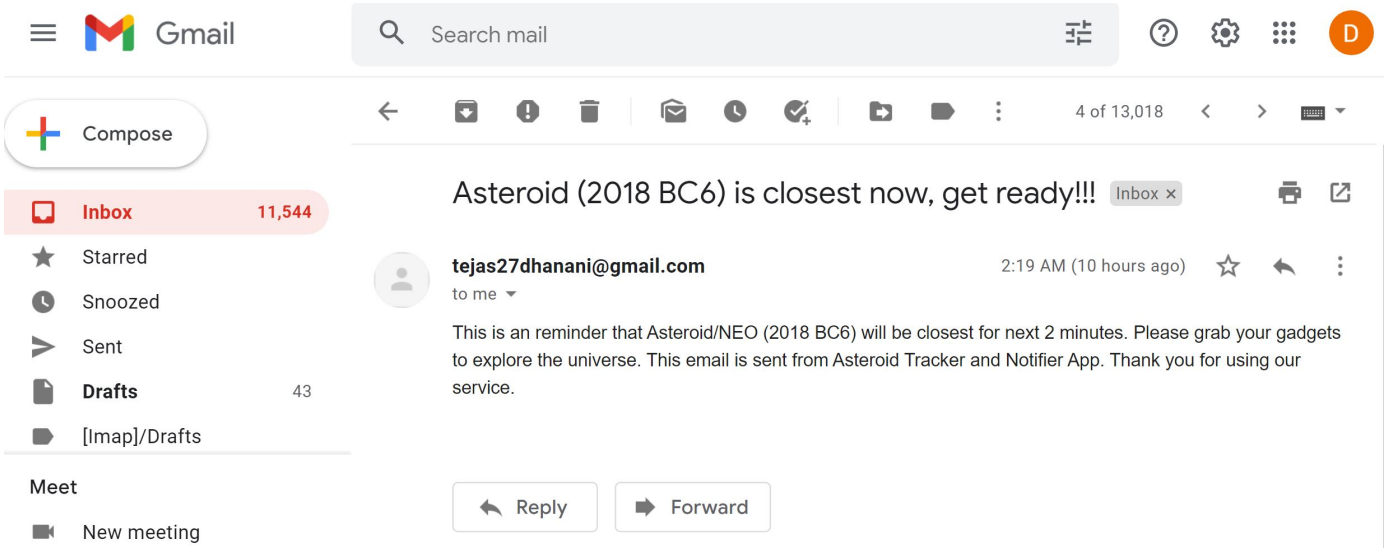


i. **scheduledFunction:** A pub/sub cloud function which will execute every 1 minute on the database to decrease the `seconds` field by 60 seconds. Also, it will check if the `seconds <= 120`, then it will set `sendEmail == true`

ii. **emailTrigger:** A firestore document.onUpdate() cloud function will run every document update. As the `scheduledFunction` is also updating the database every 1 minute, this function will also run every minute. But, on running it will check if {`sendEmail == true && emailSent == false`}, then it will trigger the `sendMail` cloud function.

iii. **sendMail:** A http onRequest cloud function which only runs if it gets a request from `emailTrigger` function. This will send email to the user using a node.js library called `nodeMailer` (https://nodemailer.com/about/).

Below is the example of how a user will receive an email about the event.



5 Challenges:

The couple of challenges which I faced while building this application are

(i) sending an email and the usage of cloud function: there is not much of a documentation available on the internet regarding this because as I am already using react native with expo. Expo has it's own dependencies connected with react native, so it was difficult for me to set those dependencies. But, I had to trial and error almost hundred times, to make it running.

(ii) While using expo with react native which allow us to build amazing applications for both the platforms (iOS and Android), there are reported bugs which cannot be fixed due to the flexibility it gives to code for multiple platforms. There would be some non-overlaps and developers just have to wait until the library providers fixes those issues. One of insoluble challenges which I faced was while developing `sort by button on the home screen` because it acts different on iOS and Android.

6 Evaluation:

This product will be considered successful if ease of use and user rating are good. Also, benchmarking the style and design of the application to proper use of spacing and colors.

7 Conclusion:

The final product is allowing the users to view all the NEOs name, closest approach date, speed, diameter and set a email reminder for closest approach time.

8 Citations:

- https://www.android.com/intl/en_ca/
- https://www.apple.com/ca/ios/ios-15/
- https://www.unoosa.org/oosa/en/ourwork/topics/neos/index.html
- https://api.nasa.gov/
- https://api.nasa.gov/neo/rest/v1/feed?start_date=2015-09-07&end_date=2015-09-08&api_key=DEMO_KEY
- https://theskylive.com/near-earth-objects
- https://reactnative.dev/
- https://nodejs.org/en/
- https://www.javascript.com/
- https://firebase.google.com/
- https://expo.dev/
- https://developer.mozilla.org/en-US/docs/Web/API/fetch
- https://www.microsoft.com/en-ca/windows
- https://github.com/wix/react-native-navigation
- https://react-native-async-storage.github.io
- https://www.hihonor.com/global/emui/