



Mobile Final Technical Report

Team Members: Dylan Bieber, David McHugh

YCAS Radio Telescope Project
Senior Software Design Project II, Spring 2022
Prof. Donald J. Hake II

York College of Pennsylvania

Table of Contents

Abstract	3
Introduction	4
Background	5
Design	7
Admin Application	7
Home Screen	7
Status Screen	9
Sensor Screen	10
Weather Screen	10
Appointment Manager	11
Approval Dashboard	11
Connection Overview	12
Implementation	13
React Navigation	13
Firebase/ Notifications	14
TCP Protocol	14
Future Work	16
Admin Application	16
API Usage for data	16
MCU Error Flip	16
Command Encryption	16
Investigate Push Notifications	17
Live Communication to Radio Telescope	17
Live Photo of Radio Telescope	17
API Endpoints Point to Correct Database	18
References	19

Abstract

The York County Astronomical Society (YCAS) teamed up with York College to develop a remotely accessible, auto-locating, auto-tracking radio telescope for installation at the YCAS observatory at John C. Rudy County Park. Now in its fourth year of development, York College students have been working to implement all of the necessary functionality to run the telescope.

As a part of the overall project, the Mobile Team addresses the need for the York County Astronomical Society to interact with the radio telescope anywhere at any time directly from their mobile devices. The application connects to the ControlRoom App which will operate the telescope as long as specific safety parameters are met and shut down the telescope in the case of sensor errors, inclement weather, and invalid input related to the telescope's movement. The AdminApp has been developed for the administrators from YCAS to perform these major functions:

- Monitor the telescope in real-time
- Operate it remotely
- Receive notifications on telescope health
- Override sensors
- Check the health of the telescope from sensors in the ControlRoom App
- View local weather conditions from a weather station co-located with the telescope

Introduction

Professor Hake at York College reached out to Kerry Smith, from the YCAS, in January of 2017 to discuss the possibility of collaborating on a project to bring a radio telescope to the YCAS observatory in John C. Rudy County Park. This project is the culmination of work started by York College students as part of their senior projects as mechanical, electrical, and computer engineers in the summer and spring of 2018. In the fall and into the spring of 2018, Computer Science students joined this effort to create software for the telescope. These efforts will continue into this academic year. In the spring of 2020, the Mobile Team was tasked with building onto the existing AdminApp and creating a new application for public use. This effort was successful, as good progress was made in the AdminApp, and all functionality was completed in the new PublicApp.

In the fall of 2020, the Mobile Team was tasked with furthering the development of the AdminApp. This development included updating the app to function with the newest version of React Native, adding the necessary components to stream audio to the speakers that will be positioned by the radio telescope, encrypting the commands being sent from the Admin App to the ControlRoom App via the Middleman Service (MS), and small changes to design.

Fall of 2021, due to the departure of the sole team member working on the MobileApp in the Fall of 2020, and with no work performed on the MobileApp in

the Spring of 2021, there was no continuity between the outgoing team and the team coming in the Fall of 2021. Due to those factors, the React Native functionality of the app became deprecated to a point where more technical effort would be needed to revive the app than start from scratch. The 2021-22 team spoke with past members of the team, and Ed Nardo, a past mobile app team member, and current React Native developer came on as a mentor for this year's team. The team worked on multiple sprint schedules, based on continuing with the deprecated code base or refactoring with the latest version of React, including the pros and cons of each approach and presenting it to professors and the client afterward. After this discussion, the decision to rebuild this app from a fresh React Native app was made. The team was able to rebuild the basic app from previous years with minimal working functionality. Implementing a custom TCP protocol was a major implementation made to control the telescope. By the end of Fall 2021, the team had a working cross-platform mobile application that was able to drive the physical hardware.

Development has continued into the spring of 2022. Working off of the previous semester's, work the team built a working application that fits all specifications the YCAS admins had requested. The AdminApp has an intuitive user interface that allows admins to log in to use APIs. AES-256 encrypted TCP packets are sent to request the ControlRoom App move the telescope and return the status. The MobileApp receives current data on the telescope statuses including push notifications for any issues that may arise.

Background

The purpose of the AdminApp is to allow the York County Astronomical Society's administrators to remotely monitor and control the radio telescope

The AdminApp contains functionality for:

- Viewing current sensor statuses
- Overriding sensors for testing/ maintenance
- Executing ControlRoom App scripts remotely
- Moving the telescope to an absolute location
- Performing relative moves using a D-pad like device
- Viewing current weather conditions at the Control Room
- Delivering timely notifications for any issues from ControlRoom App

The AdminApp is powered by React Native[1], a javascript framework used to create cross-platform mobile applications that will work for both iOS and Android devices[1], and open-source packages to enhance features. The AdminApp utilizes APIs developed by previous members of the database team. The main use for the APIs will be for gathering sensor data and token authentication. More work on the backend will need to be made before appointments and user approval becomes functional on the app. The AdminApp uses AES 256 Encrypted TCP connections to communicate efficiently with the ControlRoom App to move the telescope and gather data.

Design

The AdminApp consists of multiple screens that are created using React Native and React Native Paper primarily [10]. Each screen is a single-page class structure screen, meaning that each contains:

- Constructor
- State variables
- Component lifecycle methods
- Render method containing HTML code, and a CSS styling section

Each page also utilizes React Native components to expand upon their base structure and incorporate more complex elements for ease of organization and interaction.

Admin Application

Login Screen

The Login Screen, as shown in Figure 1, is the first page that is seen when the application is opened. On this screen, there are input boxes for the user's email address and password. These values are stored using React Communities Async Storage package [12] after it is

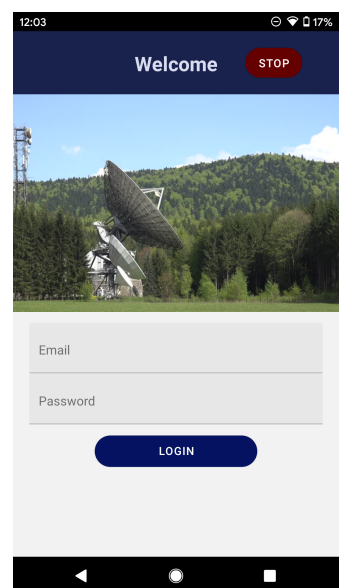


Figure 1: Login Screen

validated by the user token API. The email, password, and token are stored together locally on the phone or each admin. When admins reopen the app they will automatically be logged into the system. The login page also contains a stop button. This is a safety feature in case the admin gets logged out unexpectedly and the telescope needs to be stopped.

Home Screen

The Home Screen, as shown in Figure 2, is where users are directed automatically after logging in once. On this screen, the health of the telescope can be seen with the colored light at the top next to the “Status:” label, where green means everything is healthy, yellow means there is an unhealthy sensor, and red means there is an error. There is currently a stock photo of a radio telescope being shown, but this will be replaced with an image of the YCAS telescope or a live image of the telescope when the cameras are up and running.

Placed on top of the image are a few additional pieces of data, including the ambient outdoor temperature at the radio telescope location, the current azimuth and elevation positions of the telescope, the speed and direction of the wind, and, if an appointment is currently running, the names associated with that appointment.

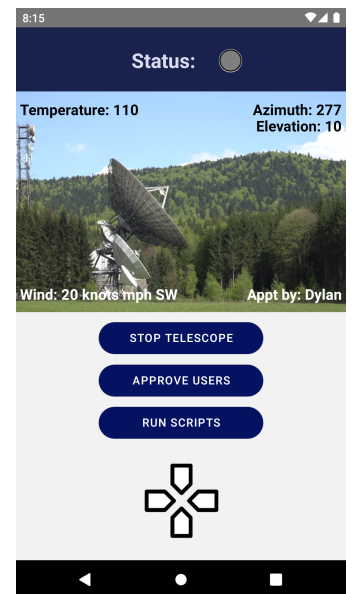


Figure 2: Admin application Home Screen

The other main functions within the application are reached through this page via various buttons. The “Stop Telescope” button, which issues a request to the ControlRoom App to stop the movement of the telescope, is shown in Figure 3.

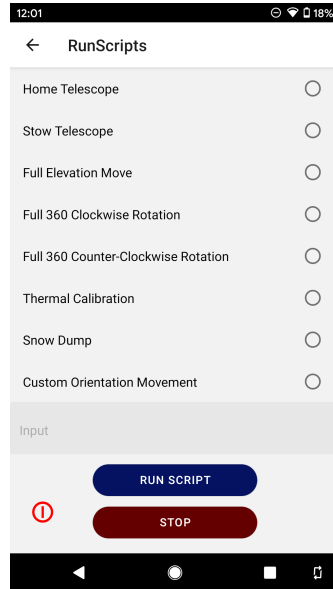


Figure 3:
RunScripts Page

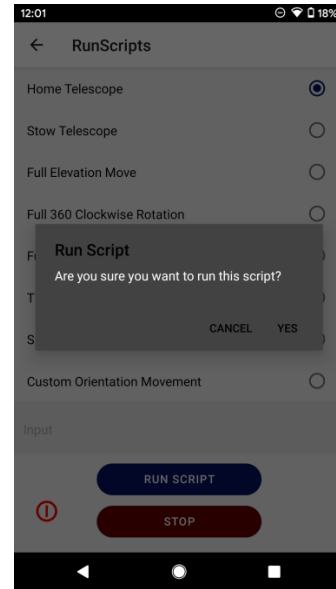


Figure 4:
RunScripts
Confirmation

The “Approve Users” button, takes the user to a screen to approve/deny new users. The “Run Script” button, shows a page for script selection as shown in Figure 3. Once a script is chosen, a pop-up is shown asking for final confirmation before sending the command, as shown in Figure 4. The script page also has a button to reset the MCU error bit, located next to the STOP button.

Below those buttons is a directional pad, which, when clicked, directs the user to a separate page where they can adjust the azimuth and elevation position, as shown in Figure 5. The user then uses a 4-direction control pad to control the telescope movement. In the center of the pad, the user can always stop the telescope. There is also another button to reset the MCU error bit, as seen on the lower left of the page.

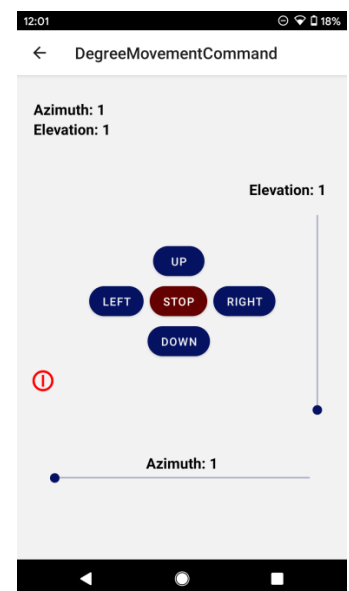
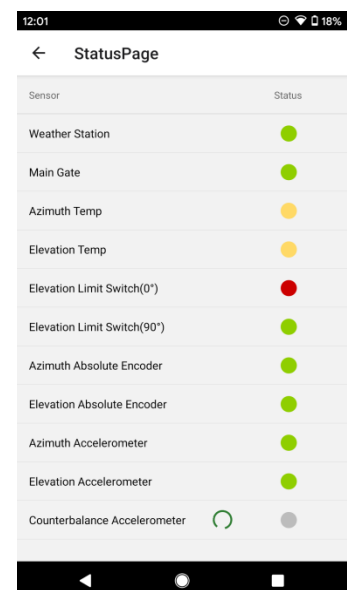


Figure 5:
Degree Movement
Commands

Status Screen

The status screen is shown when the user clicks “Status” at the top of the home screen. They will then be shown the overall health of the system with a series of

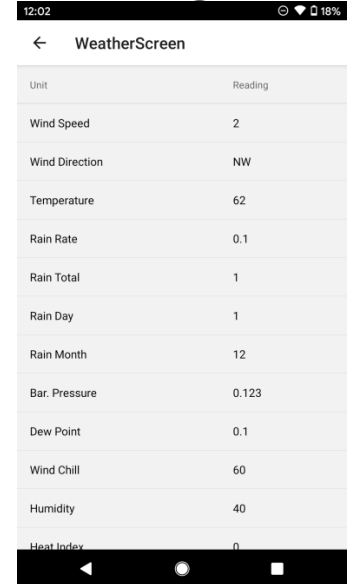


colored lights next to the names of different sensors, as shown in Figure 6.

Weather Screen

When the user presses either the wind speed or temperature on the home screen, they are taken to the weather page shown in Figure 7. This displays basic weather data gathered from the radio telescope weather station. This information is passed via TCP along with other telescope information. This will be replaced by API calls in the future.

Figure 6:
Status Page



The screenshot shows a mobile application interface titled "WeatherScreen". At the top, there is a status bar with the time "12:02" and battery level "18%". Below the title bar, there is a table with two columns: "Unit" and "Reading". The table contains the following data:

Unit	Reading
Wind Speed	2
Wind Direction	NW
Temperature	62
Rain Rate	0.1
Rain Total	1
Rain Day	1
Rain Month	12
Bar. Pressure	0.123
Dew Point	0.1
Wind Chill	60
Humidity	40
Heat Index	0

Figure 7:
Weather Screen

Appointment Manager

When the user taps on the name of the user with a current appointment running on the home screen, they are directed to the appointment manager screen. There are three tabs, one for current appointments, one for future appointments, and one for past appointments. This is shown in Figure 8, where the “Previous” appointments tab is chosen.

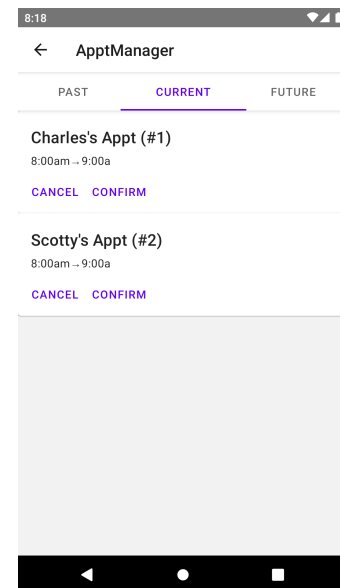


Figure 8:
Appointment
Manager

Approval Dashboard

The Approval Dashboard is reached by pressing “Approve Users” on the home screen. This is a place where admins can approve or deny new users, as shown below in Figures 9 and 10.

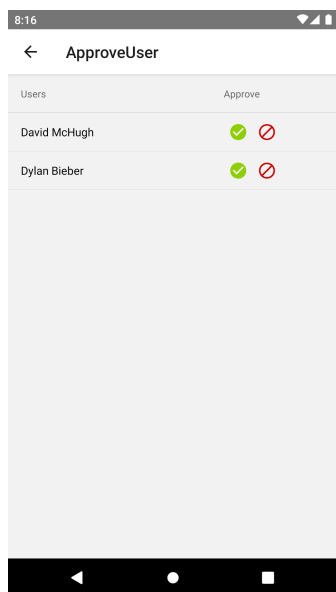


Figure 9:
User Approval

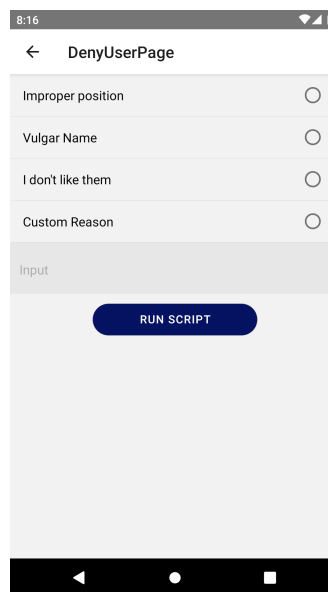


Figure 10: Deny
User Page

Connection Overview

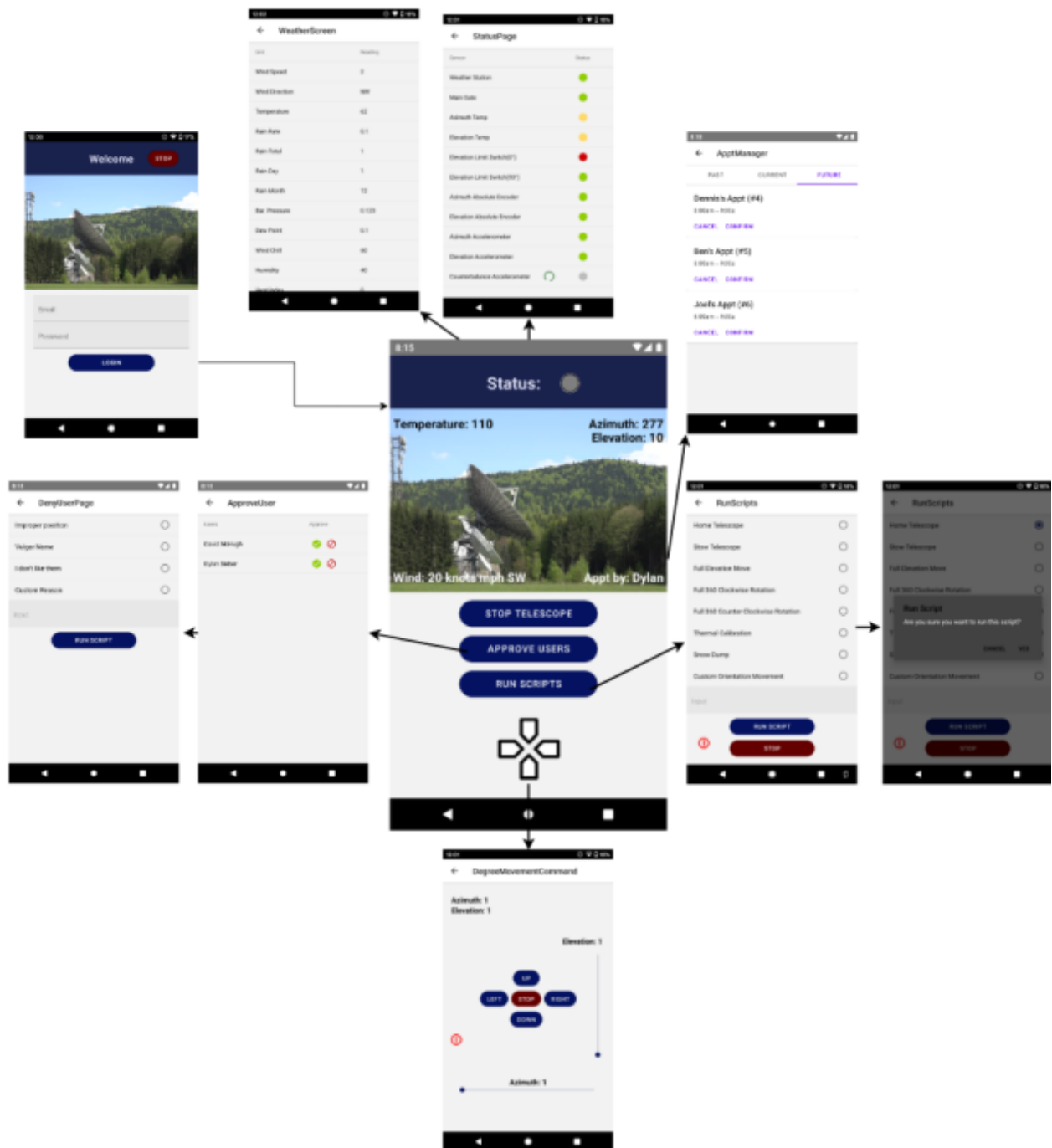


Figure 11:
Connection
Overview

Implementation

React Navigation

React Navigation is used in both applications to navigate between the different screens. React Native doesn't have a built-in idea of a global history stack like a web browser does -- this is where React Navigation comes in. React Navigation's stack navigator provides a way for the app to transition between screens and manage navigation history. If the app uses only one stack navigator, then it is conceptually similar to how a web browser handles navigation state - the app pushes and pops items from the navigation stack as users interact with it, and this results in the user seeing different screens. A key difference between how this works in a web browser and in React Navigation is that React Navigation's stack navigator provides the gestures and animations that one would expect on Android and iOS when navigating between routes in the stack[4]. React Navigation lives in "props", small code fragments that can be reused, and can be referred to by referencing the prop file. Navigating to a screen is built into React Native.

Firestore/ Notifications

Firestore is a collection of services for mobile apps, web pages, and Unity applications that offers features like hosting, storage, and cloud messaging[5].

Firestore is primarily used to implement push notifications to the admin.

The Mobile Team is utilizing the cloud messaging API for Firestore, so any device registered with the Firestore admin topic will receive a push notification in the case of an emergency. Each device is added to the topic of “admin” when the user logs in and is removed from the topic when the admin logs out. While subscribed to the topic “admin”, those phones will receive any notification pushed via Control Room to this specific topic. Currently, on the ControlRoom App, the push notification is called with a header and body to send. That information is sent using Firestore Admin dotNet [11]. This allows the use of the Google Firestore APIs from C# without much hassle. A test notification button was also added to the Control Rooms' appointment page, this page will be revamped.

TCP Protocol

This protocol is also explained in the final tech report for Control Room [9] as this was a joint effort. The best place to find the most relevant technical documentation is in the RemoteListener Documentation[7].

Through TCP in React Native, the AdminApp opens a TCP connection based on the IP Address and Port of the ControlRoom App, in

the constants' file. As long as port forwarding is enabled on the current router and the ControlRoom App is running, the RemoteListener.cs class will receive the connection. From there, an encrypted string is concatenated by the MobileApp and sent to the ControlRoom App. The ControlRoom App decrypts the string and parse it. Once parsed the ControlRoom App will either begin the requested action or return an error. The ControlRoom App will notify the MobileApp when a string has been received, parsed/started, and completed or failed. The estimated time may be sent back to the MobileApp if applicable. Once the action has been completed or failed, the ControlRoom App will destroy the connection after sending the success or failure message to the MobileApp.

For safety purposes, all connections run asynchronously however requests with the same priority cannot override each other. Only a command with a higher priority that is parsed may override a lower priority command. Pages that are opened on the MobileApp begin with a command to receive the most current data on the telescope's position. From there this command will be sent roughly every minute as long as the telescope isn't moving. Once the telescope is moving the request time will jump up to 5 seconds. If the MobileApp were to crash after sending a movement command that would cause no issue as the ControlRoom App handles all telescope movements. Once the MobileApp is relaunched all the current data will be updated again.

All Accepted Command Types

- A movement command
- A sensor initialization request
- An ALL_STOP command
- A script command
- Requesting current position data
- Overriding a sensor
- Homing routine
- Reset MCU Bit

All Return Types

- Success
- InvalidVersion
- InvalidCommandType
- InvalidCommandArgs
- MissingCommandArgs,
- InvalidScript
- InvalidRequestType

Packages/ Dependencies

[React-native-paper](#)

React-native-paper is a cross-platform material design tool for React Native. React paper is used in the admin app for buttons, lists, icons, and more. This package allows the admin to change the UI to the client's specifications.

[React-native-community/slider](#)

This is a React Native component used to select a single value from a range of values. The react-native-community/slider is an easy way to use a slider bar to gather a decimal value from the user, used specifically in the Degree Movement Page. React-native-paper does not have slider bars so this was a necessary extension

[Asyncstorage with React Native](#)

Asyncstorage is a built-in, React Native package that is used to store data locally on a mobile device; this offers the ability to store values within a map-like format on an individual user's phone. This is used on the admin app to store the

user's email address and password locally. A login page was added for API use and in order to ensure the users are remembered their information is stored after login. This is pulled once the app is opened and the admin automatically logs in.

[Axios](#)

Axios is a package allowing API calls to be completed in a simplistic manner. Axios is a Postman-like package for Javascript allowing the admin app to simply call the backend APIs using the user's locally stored token.

[React-native-tcp-socket](#)

React-native-tcp-socket is a simplistic way to send and receive commands using TCP. The admin app control over the telescope is all dependent on TPC commands, sent to the Control Room via the RemoteListener Class. TCP requests are sent to the Control Room and then await a response from the Control Room on whether the information was received, the status of the action, or data acquisition.

[CryptoJS](#)

JavaScript implementations of standard and secure cryptographic algorithms. The Mobile App and Control Room use the AES-256 encryption format, using the cipher-block chaining (CBC) mode. The Key and IV are stored asynchronously on the App and Control Room.

Future Work

Admin Application

TCP to API

Currently APIs are used but not the extent requested by the clients. APIs need to be used for everything listed in the issue below. The architecture is set up and ready but there will need to be minor changes made. We need to rely on TCP less for information. See the following GitHub issue where this is tracked:

https://github.com/YCPRadioTelescope/rtMobile_v2/issues/82.

Admin Requests Stored

Admin requests via TCP are not currently stored anywhere meaning it is lost once completed. Another table can be created to store admin commands on the ControlRoom App. The admins' email can be added to the command they send. The ControlRoom App can parse and store this data. It would be nice to have APIs so admins can view the most recent requests on the app.

[Shttps://github.com/YCPRadioTelescope/rtMobile_v2/issues/75](https://github.com/YCPRadioTelescope/rtMobile_v2/issues/75).

API Endpoints Point to Correct Database

Ensure that all endpoints are pointing to the correct database once we have the functioning radio telescope and production database. APIs will need to be pointed towards the production database. This will need to be changed to point to the production database once it is installed in the park. Along with this IP address, ports will need to be changed in order to create TCP connections.

Updating Package Dependency

Yarn packages, the libraries used within the mobile application ,will need to be updated regularly (4-6 months). See the following GitHub issue to keep track of when packages are updated:

https://github.com/YCPRadioTelescope/rtMobile_v2/issues/83.

Live Photo of Radio Telescope

YCAS admins have requested that the live feed of the telescope is to be shown on the app. This may be unnecessary due to the camera already having its own app.

References

- [1](2020). React Native, Retrieved November 28th, 2020 from <https://facebook.github.io/react-native/>
- [2](2020). Node, Retrieved November 28th, 2020 from <https://nodejs.org/en/>
- [3](2020). AWS, Retrieved November 28th, 2020 from <https://docs.aws.amazon.com/>
- [4](2020). React Navigation, Retrieved November 28th, 2020 from <https://reactnavigation.org/>
- [5](2020). Firebase, Retrieved November 28th, 2020 from <https://firebase.google.com/>
- [6](2020). Crashlytics, Retrieved December 5th, 2020 from <https://firebase.google.com/docs/crashlytics>
- [7](2021). Remote Listener Documentation from [RemoteListener Documentation](#)
- [8](2021). Control Room Final Tech Report from [Control Room Final Tech Report - Fall 2021](#)
- [9](2021). TCP for React Native, Retrieved December 9th, 2021 from <https://www.npmjs.com/package/react-native-tcp>
- [10](2021). React Native Paper, Retrieved December 9th, 2021 from <https://callstack.github.io/react-native-paper/>
- [11](2022). <https://github.com/firebase/firebase-admin-dotnet>
- [12](2022). <https://reactnative.dev/docs/asyncstorage>
- [13](2022). <https://callstack.github.io/react-native-paper/index.html>
- [14](2022). <https://openbase.com/js/@react-native-community/slider>
- [15](2022). <https://www.npmjs.com/package/axios>
- [16](2022). <https://www.npmjs.com/package/react-native-tcp-socket>
- [17](2022). <https://cryptojs.gitbook.io/docs/>