Cloud MX Operations Center

Project Members

Muhammad Choudhury
mchoudhury2017@fau.edu

Nathan Dacosta
ndacosta2017@fau.edu

Ariful Islam
aislam2017@fau.edu

Tchaly Leandre
tleandre2017@fau.edu

Andy Nguyen
nguyena2017@fau.edu

Advisor
Hari Kalvah
kalva@fau.edu

Florida Atlantic University Department of Electrical Engineering and Computer Science
Spring 2020

Project Summary:

Dobbins Air Reserve Base has an outdated way of providing aircraft information to maintenance workers. Our sponsor would like to have a system where workers in the base can have easy and quick access to up to date information about the base and it's aircrafts. Our proposed solution is a mobile app that is supported by both IOS and Android devices that displays up to date information about the base and it's aircraft.

Table of Contents

1. Introduction

1.1 Problem Description (by Ariful)

The sponsor of this project is Brandon Rodts who is a Major in the US Air Force. He leads a Squadron at the Dobbins Air Force Base in Georgia. Major Rodts and his team uses spreadsheets to write down information pertaining to the aircrafts located on his squadron's hangar and the adjacent runway such as the maintenance status of each aircraft, where it is located, and the overall flight schedule.

The problem is that this is a very inefficient system; most of the members are not near a computer at all times and the spreadsheets need to be printed out and handed to the Major and other members several times a day. Using spreadsheets is also not useful for looking and editing in this environment as critical information is compacted into just a few cells and can sometimes be hard to view and can even lead to confusion. There is also the issue that the security of these spreadsheets and the database that stores all of their data is not very secure.

1.2 Problem Significance (by Ariful)

Our sponsors are members of the US Air Force which have the highest standards that it lays out for everyone in its ranks to ensure that the organization as a whole works properly. They require the most sophisticated technology and software that have exceptional encryption so that adversaries cannot intercept the data. This minimizes waste while maximizing their productivity. Creating a software solution to

replace using spreadsheets that store all relevant data while also being secure is something that a modern squadron of an Air Force would need in the modern day.

1.3 Goals and Objectives (by Ariful)

Our goal is to create a smartphone application that will give relevant real-time information about aircraft in the hangar or runway to anyone who has the application on their smartphone or tablet whether it be iOS or Android. This will allow the airmen to access the critical information wherever they are since they have a smart device on their person at all times. The application will also be encrypted because the information stored on these devices are not classified but are still sensitive data that cannot be shared with outside parties or cannot be leaked to other entities outside of the armed forces.

The application will contain the status of the aircraft, whether it has maintenance issues, who is the lead maintenance technician , and other comments about the aircraft. It will also contain the location of each respective aircraft on either the hangar or the runway, the overall weekly flight schedule for each aircraft, and if there are any physical problems with the hangar that need to be repaired. There will be separate admin and viewer modes in which the operation controller can update and view in the admin-mode whereas everyone else can only view data.

Our objective is to write the application in React Native so that we have both the iOS and Android apps based on only one universal codebase instead of two. We will also be using Amazon Web Services for the server and to store the data. We believe

that the creation of this smartphone application fulfills the needs of our beneficiaries tremendously as it solves all of their problems that they have discussed to us.

1.4 Related Research (by Nathan)

There are plenty of applications that function similarly to our project. Here are some patents that share the same concept needed for this project.

Patent number: US9119017B2

Title: Cloud based mobile device security and policy enforcement

Inventor: Amit Sinha

Assignee: Zscaler Inc

Date filed: Oct. 23, 2011

Date granted: Aug. 08, 2015

Number of claims: 19

Type of patent: Utility

Summary: This patent talks about a cloud-based security method for a mobile device. This security method provides a mobile configuration to the mobile device. This configuration provides communication between the mobile device and the external network like a database. This configuration also analyzes the communication to see if the mobile device meets the certain criteria such as meeting security requirements in order to access this external network. After connecting to the cloud-based security system, the user can transfer or receive data with an external network interface through this system. While the data information is being transferred to the user, the cloud-based security system monitors this data information to see if anything breaches security by

using data inspection engines. If data has been breached, then the security system will block the data to the mobile device and give a notification that data has been compromised.

Relevance: In this mobile application, security is the crucial point in the system requirement as this data is too confidential as one is dealing with military information. This cloud-based security system can help secure data from the database. The security system can also help with securing data once they are in the application. If the security system detects something illegal, the system will stop communicating with the mobile device. Also, it checks to see if the mobile device using the application is meeting security requirements. For example, this means that the cloud-based system could check for external hardware problems that the application may not keep track of like if a mobile device having computer virus on it or sending undesirable content to the database. This cloud-based security system could keep our data even more secure if we were to implement it into our application as we do not have a way in our proposed solution to deal with if the user sends undesirable content like a virus. If there is some malware that breaches our database, the cloud-based system would give a notification about what is happening, and we act accordingly. This security system will also enforce any of our security policies that we implemented like having a secure and safe connection to the database.

Patent number: US9537835B2

Title: SECURE MOBILE APP CONNECTION BUS

Inventors: Mansu Kim, Cupertino, CA (US); Joshua Sirota, Los Altos, CA (US); Suresh Kumar Batchu, Milpitas, CA (US)

Assignee: MOBILE IRON, INC., Mountain View, CA (US)

Date filed: Apr. 17, 2015

Date granted: Jan. 3, 2017

Number of claims: 20

Type of patent: Utility

Summary: The invention proposed by this patent can be implemented in many different ways such as a system or computer program. The main idea is to have a two-way secure connection bus which will allow an application to be managed while keeping the user experience run seamlessly. To establish a secure application connection bus, data needs to be transferred in encrypted form by storing the encrypted data in a storage location that can send and receive apps such as a database. With this method you can have several mobile applications on the same connection bus and leave out any other mobile applications that are irrelevant or unsecure. With each mobile app there will be an encryption for data coming out of it. For example, let's say there are two mobile apps connected to a secure connection bus, information coming from the first mobile app will have the first encryption and information coming from the second app will have the second encryption. The first and second encryption refers to the type of encryption that means that each app connected to the secure connection bus will have its own encryption and all encrypted data will go to the same storage location. In order to tell the difference between encrypted data the data will be assigned a data storage location identifier.

Here is the flow of how information will be transferred between two different mobile apps. First the information from the first app is generated and encrypted. Then it is assigned a data storage location identifier and then sent to the second app. The first app will then receive the second encryption information from the second app, which is then validated. When the second app gets the first encrypted information and data storage location identifier the source of it is validated which would be the first app. Then a second encrypted key pair is made and with the first encrypted information an encrypted payload is made becoming the second encryption. After that the second encryption is sent to the data storage location. That is the general flow of how information is transferred between apps connected to the secure connection bus. More can be added to it such as a management app that can send updated information to other mobile apps connected to it and more.

Relevance: Our project is the Cloud MX Operation Center that needs to be a mobile app and needs to be secure since information within the app is considered sensitive. With that this patent is very relevant because it proposes a secure connection bus that can connect to mobile apps. Our project needs to have an app where users can view the information and another version of the same app where admins can edit the information and both apps need to have a connection to a database where all information is stored. So one way to incorporate this patent into our project is by having a secure connection bus and have both versions of the app and a data storage location connected to that bus. This way the admin version of the app can securely send encrypted information to the viewer version of the app and the storage location which would be the database storing all the information related to the app.

Patent number: US8321302B2

Title: Inventory management system

Inventors: Donald G. Bauer, Richard J. Campero, Paul B. Rasband, Martin D. Weel

Assignee: Sensormatic Electronics LLC

Date filed: Jan. 23, 2003

Date granted: Nov. 27, 2012

Number of claims: 35

Type of patent: G06Q10/087 Inventory or stock management, e.g. order filling, procurement, balancing against orders

Summary: The patent is an advanced inventory management system. The system described contains sensors that can detect items as they come in or come out of the facility through their tags. Everything is inserted into the database including all related information of products so that other maintainers or workers will be notified if any items in the same class is depleted as they will need to be replaced. Along with data such as the size and weight of the items this database also includes data on whether there are defects in any of the items or if anything is misplaced.

Relevance: This patent is different from our project as we are creating a smartphone application that includes not only a database of aircraft of a squad the US Air Force but also the visualization of the aircraft and where they are in the hangar or the runway. Other differences are that our project will have an admin mode for admins and a view-only mode for most people and also our entire app will have enhanced security and encryptions since everything contained in the databases are classified information.

But this project is also similar to ours in that they both contain databases that contain information that is of interest to everyone involved. For example, our database will contain maintenance issues of every aircraft as well as who is the lead maintainer for each aircraft. It will also contain the flight schedules of every aircraft and to which bases or locations it will be flying or stopping by in. One thing about this patent that we can implement in our project is that the database advises or comes up with possible measures or actions that the maintainers should take. Our system can be designed to take in data like the facts that Aircraft X has its right wing dented and it hasn't flown it a few months and can then create a prompt for the maintainers notifying them that Aircraft X's wing must be fixed and that it should be flown afterwards. This is something that our sponsors will like since it will remind them of what should likely be done to these aircraft so there will be less thinking and burden upon them.

Patent number: US9858174B2

Title: UPDATABLE NATIVE MOBILE APPLICATION FOR TESTING NEW FEATURES

Inventors: Christian David Straub , Palo Alto , CA (US) ;

Yuliya Serper , Bellevue , WA (US)

Assignee: Oracle International Corporation, Redwood Shores, CA (US)

Date filed: Sep. 25, 2015

Date granted: Jan 2, 2018

Number of claims: 17

Type of patent: Utility

Summary: Application Development Frameworks (ADFs) provides a set of pre-defined code/data modules that are able to be used directly or indirectly in the development of an application. They may also include an Integrated Development Environment (IDE), code generators, debuggers, etc. which aid a developer in programming the logic of the application in a more efficient way. The reusable components and IDEs provided by an ADF greatly simplifies application development.

Oracle ADF includes libraries of standard-based Java Server Faces (JSF) that includes HTML5 and AJAX functionality. These components allow web deployed UI's to be developed with a degree of interactivity and functionality that was previously attributed to thick-client applications. This allows for data interactions, visualization, and encapsulated browser side operations easy to use and allows rich client application development easier. Oracle ADF provides a data-binding framework which simplifies binding UI to business services through drag-and-drop operations in the IDE. It keeps the independence of the business service from consuming interfaces, and makes the UI building processes decoupled from the business service layer implementation. This positions the application for implementation in a service-oriented architecture.

This is designed to solve problems relating to data binding for mobile apps.

Relevance: This patent will make implementing parts of the mobile application easier. Since the entire program doesn't have to be rebuilt, this makes testing new implementations much easier. Let's say you want to add, in the case of our project, a feature that allows the user to pull up specific details of the aircrafts, the Oracle ADF allows us to develop, implement, and test this feature without drastically affecting the

overall code of the program. This will let us know how the implementation works and whether it breaks the entire program or not.

2. System Design

2.1 Project Requirements (by Muhammad)

This section shall breakdown the objectives of the Cloud MX Operation Center. These requirements are a high level description of what the Cloud MX Operation Center will be at the end of it's development. These requirements are organized into three sections: functional, usability, and safety requirements. Functional requirements are what the Cloud MX Operation Center's system will be able to accomplish. Usability requirements are what users will need in order to interact with the system. Safety requirements are meant to make sure that the system is safe to use by the users and the organization responsible for it.

Functional Requirements (by Muhammad):

F.1. The system shall display up-to-date information related to aircrafts.
F.2. The system shall display up-to-date information related to relevant buildings.
F.3. The system shall display up-to-date information related to maintenance workers.
F.4. The system shall contain a database where all information is stored.
F.5. The system's UI will have a digital representation of the Dobbins Air Reserve Base that users may interact with to see information.

Usability Requirements (by Muhammad):

U.1. The system shall be a mobile application where all related information may be viewed
U.2. The system shall be supported by both IOS and Android devices.
U.3. The system shall need an internet connection inorder to use it.
U.4. Admins can make changes to the database through the mobile applications.
U.5. Regular users can only view information in the mobile application no changes can be made within it.

Safety Requirements (by Muhammad):

S.1. The system shall be encrypted.

S.2. The system shall have admin and user privileges.

2.3 Project Requirements Not Met (by Muhammad)

While developing new requirements were found however we had to make sure the requirements that we already defined were met. These requirements could be implemented to further improve the overall system:

F.7. The system shall display the status of equipments
F.8. The system shall contain a password recovery feature
U.10. Admins can add equipment to the system through the mobile application
U.11. Admins can update equipment from the mobile application

2.4 Project Requirements Added/Changes (By Nathan)

While developing the project we realized that we can further improve the system by adding the following functionalities:

F.6. The system shall display the maintenance history of aircrafts

We've improved usability:

U.6. Admins can add new users to the system through the mobile application
U.7. Admins can add new admins to the system through the mobile application
U.8. Admins can add new aircrafts to the system through the mobile application
U.9. Admins can remove aircrafts from the system through the mobile application

As a result of improving some of the usability the following safety requirement was removed for being obsolete:

S.3. Users will be manually placed within the system, and the system will not provide a sign up option.

2.5 System Structure (By Nathan)

_____As stated before the Cloud MX Operations Center will be a mobile application containing information related to aircraft maintenance. The app itself is broken down into several screens and it will be connected to a back-end server where all information is stored. There will be a sign-in, map, building, maintenance history, add user/admin, aircraft, add/remove aircrafts, and overall aircrafts screens.

The sign-in screen is what users will see when they first open up the Cloud MX Operations Center. Its purpose is to verify users attempting to enter the app, if the user inputs the proper sign in information they will be permitted into the app and if not they will remain in the sign-in screen.

The map screen will contain a graphical representation of the Dobbins Air Reserve Base and this is the screen that users will see once they have successfully signed into the system. This screen is the home screen of the app and acts as a hub area for all information the app contains. Users can tap on a building on the map of the base to see more information about it. For example if a user taps on the Maintenance Bay on the map they will be taken to the building screen to see more information about that specific location. The building screen will show the status of the Maintenance bay. The same goes for any other buildings that the user taps on. The map screen also contains buttons that will lead users to the add admin, add user, add aircraft, and remove aircraft screens which are self explanatory what those do and can only be accessed by admins. The map screen will also show if an aircraft is currently inside a building by displaying the serial number of the aircraft right under the name of the building or parking spot.

The map screen can also lead users to the overall aircraft screen which shows a list of all the aircrafts serial numbers. The serial numbers of each aircraft that would be listed serves as a link to that particular aircraft’s aircraft screen if the user taps on the serial number of an aircraft. The individual aircraft screen will show more in depth information about the aircraft the serial number is assigned to such as status, remarks, parts needed, flight schedule, estimated time of completion, lead maintainer assigned to it, location, and sortie type. The individual aircraft screen also contains a button that will lead users to the maintenance screen of the chosen aircraft.

This system will have two types of users: admins who would be those with leadership roles and normal users who would be on base staff. Admins will have the power to go into the app and make changes to information displayed while normal users can only view the information displayed. This is how information displayed will be updated, by having admins enter the app and make the necessary changes. For admins making changes will be simple, they will be able to make changes directly in the app and when they are done they will press the update button in order to push the new data into the database via API calls. Whenever an admin is updating an aircraft's information the moment they click on the update button the new data is reflected in the database and a copy of that change is saved into the maintenance history portion of the database.

With that the description of the Cloud MX Operations Center design is complete. The method of how all this will be accomplished will be discussed in section 3 which goes over the implementation.

2.6 Block Diagram (By Andy)

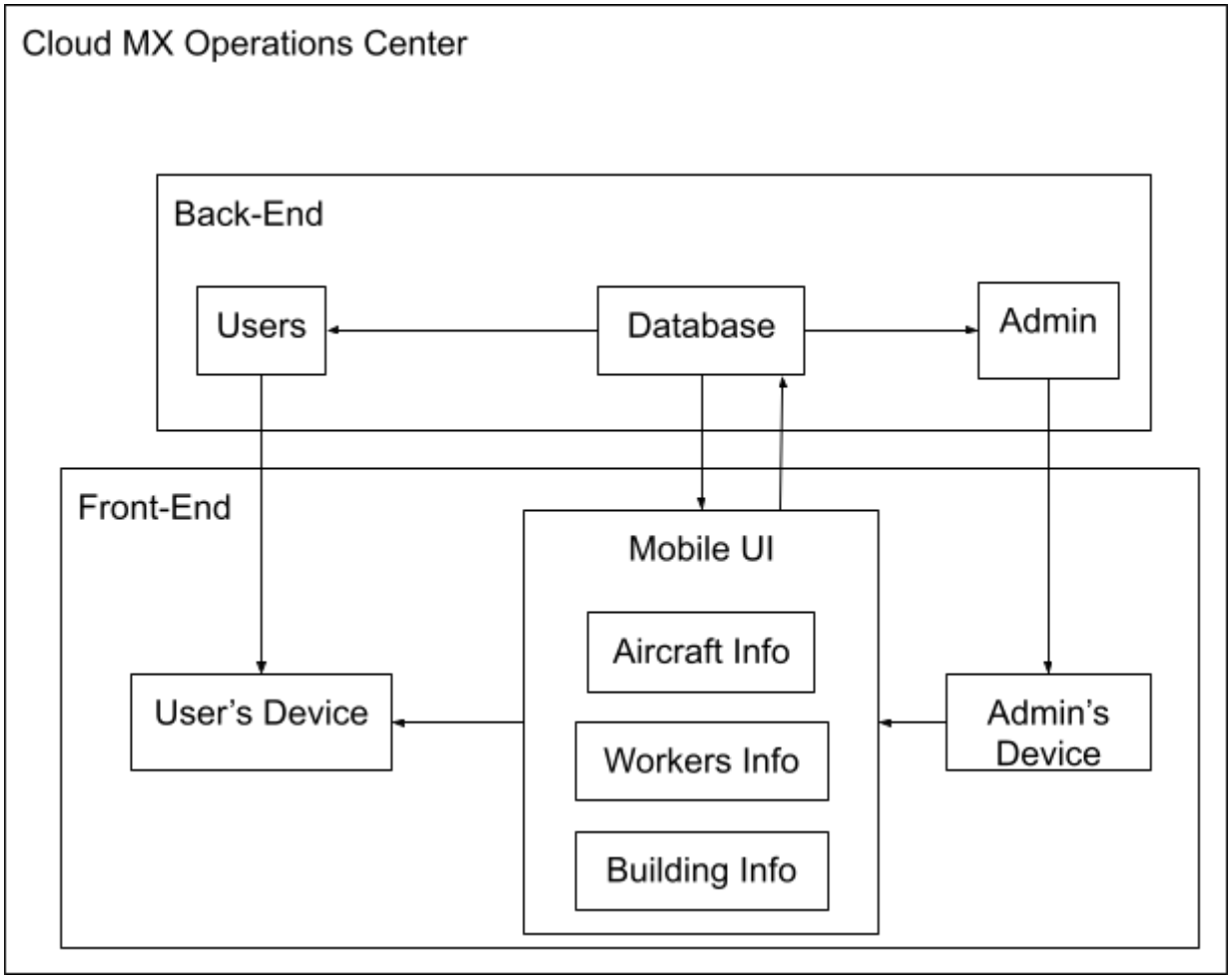

Figure 1: Block Diagram

The back-end section deals with how information displayed by the system is edited and how users are added to the system. The arrow from the Database box pointing to the Admin box shows that the Admin(s) will already be a part of the Cloud MX Operations Center system. The arrow pointing from the Admin to the Admin's Device shows how admins view the mobile ui from their device. The arrow point from the admin's device to the mobile UI shows that admins can edit information within the mobile ui. The arrow pointing from the mobile ui back to the database shows that changes made by the admin within the mobile ui is sent to the database to update it. The arrow from Database to Users represents how each user of the system is manually added into the system by the Admin. Traditionally a sign up screen would be provided,

but since the information presented by this Could MX Operations Center is considered sensitive extra care must be taken to ensure security. The back-end subsystem will be worked on by Tchaly Leandre, Nathan Dacosta, and Andy Nguyen. Specifically Tchaly will set up the database, Nathan will handle how the IOS version of this system will connect to the database, and Andy will handle how the Android version of this system will connect to the database.

The front-end section deals with how information is displayed to the user. If a user has been added to the system by the admin they can use their device to see the Mobile UI. The arrow from User to User's Device represents the transition from back-end to front-end and how users that are a part of the system may see information stored with the database. This is what the arrow pointing from the Database to Mobile UI means, information from the Database is sent to be displayed in the Mobile UI. The Aircraft Info, Building Info, and Worker Info boxes represent the types of information that the Mobile UI will present. The arrow from the Mobile UI pointing to the User's Device represents the Mobile UI being displayed on the user's device. The arrow also means that this is a one-way interaction, the user will not be able to change any information within the Mobile UI, the Mobile UI is purely for viewing purposes. The front-end system will be worked on by Muhammad Choudhury and Ariful Islam. Specifically Muhammad Choudhury will work on how to display the information from the database for IOS devices while Ariful will work on how to display the information from the database for Android devices.

2.7 State Diagrams (By Andy)

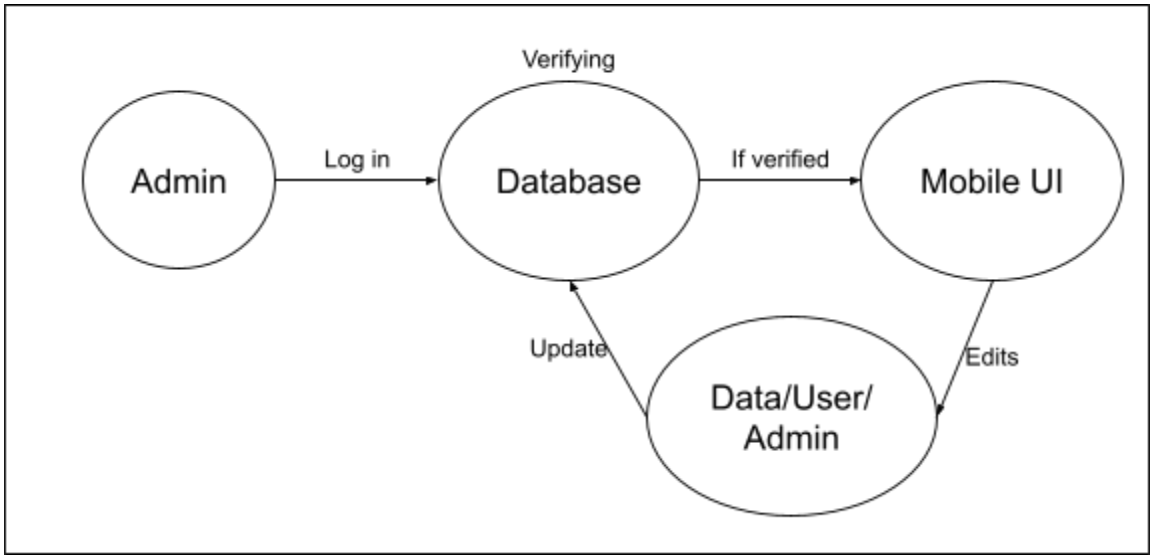

Figure 2: Back-End State Diagram

Back-End State- The above diagram shows the process of how the Admin interacts with the Mobile UI to change data within the database. An admin logs in to the mobile ui, the system will first verify if the admin used the correct information to login by checking the database. If the login information matches what is in the database then the admin is allowed into the mobile ui where they can edit information and add or remove users or other admins from the system. Any changes made will be sent to the database to reflect the changes.

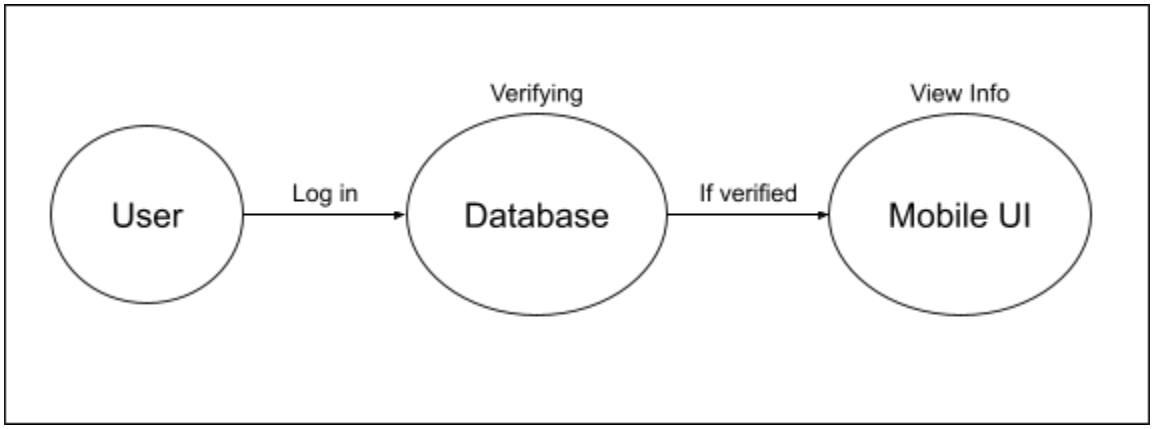

Figure 3: Front-End State Diagram

Front-End State- The figure above shows the process of how users will interact with Mobile UI in order to view information provided by the app. The user inputs their login information then the information is sent to the database to verify if it is correct. If the login information provided matches the one for the user in the database they are permitted to view the mobile ui. This is where users can view information that is stored within the database of the system.

2.8 Use Cases (by Nathan)

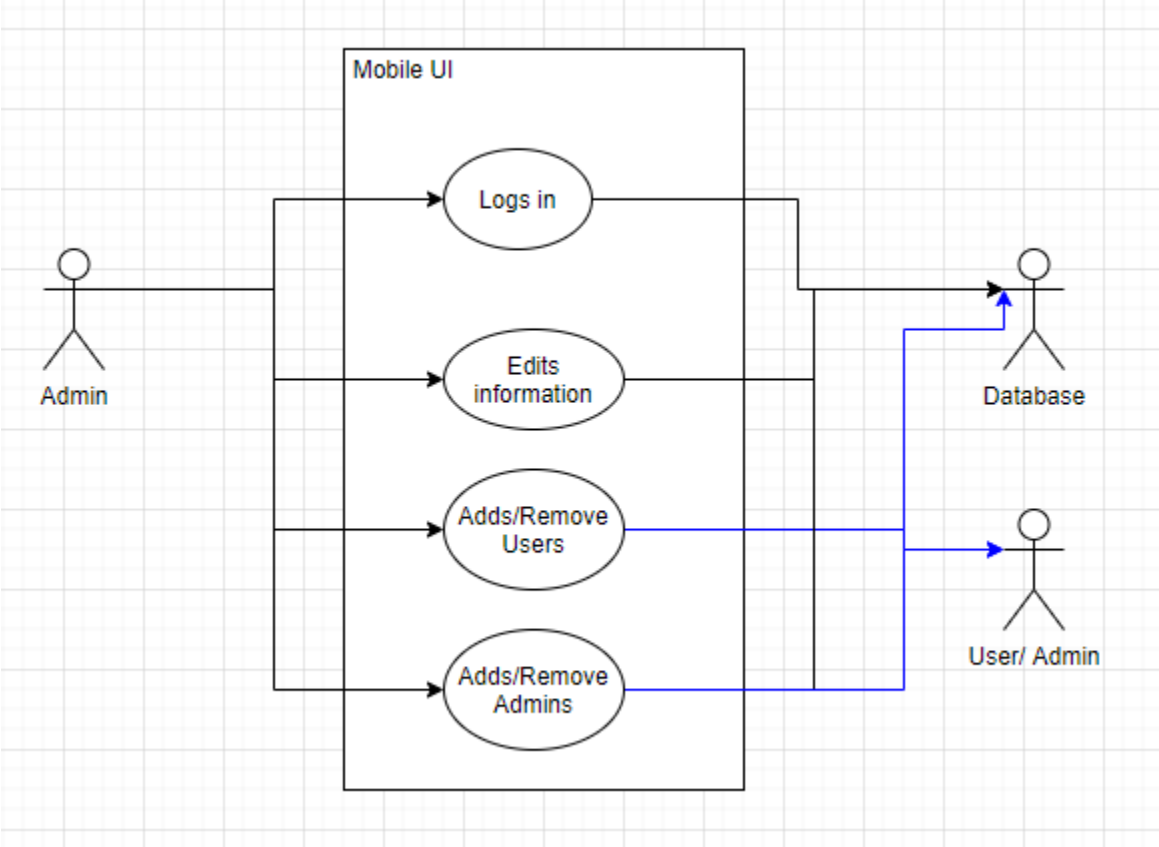

Figure 4: Use-Case 1

The figure above shows a use case in the perspective of the Admin. The Admin can log into the Mobile UI which is sent to the database to verify. If verified the Admin can edit information, add/remove users and or admins. All changes are sent to the database to

reflect the changes and the users and or admins previously not added before are now part of the system or if existing are removed if deemed so. This case shows the flow of the Back-End state diagram

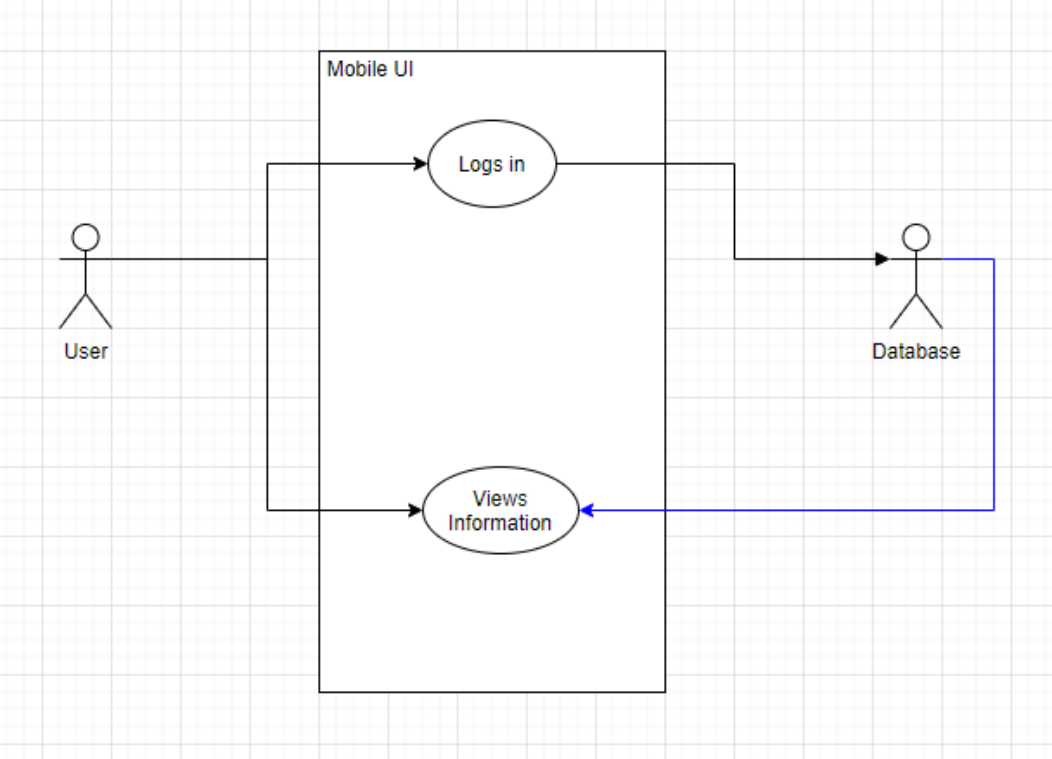

Figure 4: Use-Case 2

The figure above shows a use case in the perspective of the user (anyone who isn't an admin). The user can login to the Mobile UI and if verified may view information that is stored within the database. This case shows the flow of the Front-End state diagram.

Requirement	Subsystem	Description
F.1.	Back-End/Front-End	The overall aircraft screen and individual aircraft screen will display up-to-date information about aircrafts by connecting to the AWS back end server.

F.2.	Back-End/Front-End	The building screen will show up-to-date information about the statues of building by connecting to the back-end server
F.3.	Back-End/Front-End	Information about maintenance workers assigned to aircrafts will be in the individual aircraft screen.
F.4.	Back-End	All information about the system will be stored within the AWS server.
F.5.	Front-End	The map screen will contain the digital representation of Dobbins Air Reserve Base.
F.6.	Front-End/Back-End	The app will display the maintenance history of an aircraft. Data displayed will come from the database.
U.1.	Front-End	The system will be written to be a mobile application.
U.2.	Front-End	The system will be written in React Native which is a framework that runs mobile apps for both Android and IOS devices.
U.3.	Back-End	The system will require the internet in order to connect to the external server with AWS.
U.4.	Back-End/Front-End	Admin users have their own viewing mode of the app where they can make changes.
U.5.	Back-End/Front-End	Normal users will have their own viewing mode of the app where they can only view information
U.6.	Back-End/Front-End	Admins will be able to add new users through the app
U.7.	Back-End/Front-End	Admins will be able to add new admins through the app
U.8.	Back-End/Front-End	Admins will be able to add new aircrafts through the app
U.9.	Back-End/Front-End	Admins will be able to remove aircrafts through the app
S.1.	Back-End	AWS provides encryption for data being sent to and from servers within it.

S.2.	Back-End	Those with admin privilege can make changes to information within the app while those without it may only view information.

Table 1: Summary of Requirements and Solutions

3. Implementation

3.1 Hardware (by Tchaly)

This project is mostly software and assumes that the users have the necessary hardware. The only hardware needed is a mobile device such as a phone or tablet regardless of it being iOS or Android. It is preferable that the phone or tablet that is being used to run this application is no more than 6 or 7 years old as older devices may not have the capability to run applications as efficiently as newer mobile devices. However, generally speaking all mobile devices that run on the iOS and Android operating systems will be able to run this application.

3.2 Software (by Tchaly)

This project will be mostly software and will need a lot of software components. Cloud MX Operations Center system needs to be displayed on mobile devices for both IOS and Android devices. The mobile devices that will run this application must be updated to one of the more recent versions of Android and iOS for the application to work properly. Information displayed needs to be secured as well since they are considered to be sensitive information. There are two aspects to the software portion of this system. There is a front-end and back-end. The front-end is concerned with how information is displayed on the app of the system and the back-end is about the functionality of the app and how it will connect to the systems database. The team that is working on this project will be divided into two sub teams and each sub-team will work

on either the front-end or the back end portions of the app because that is the most efficient approach.

Front-end: (By Nathan)

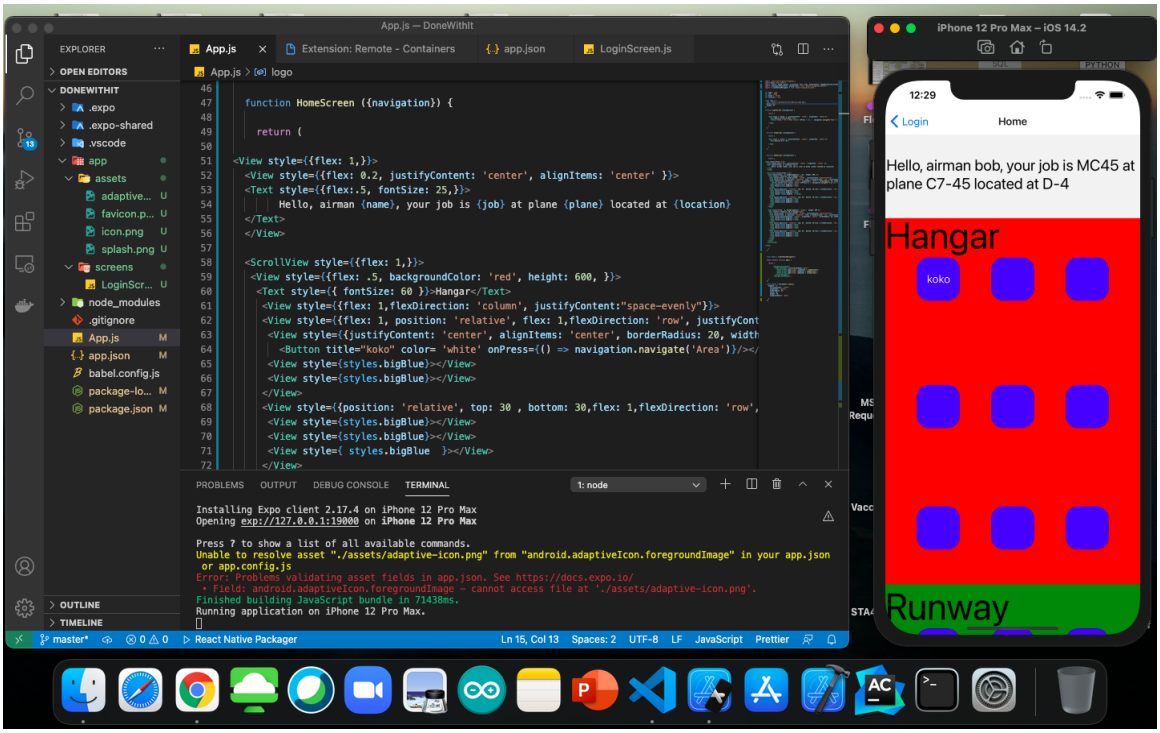

Figure 5: Example of Front-End (beta version of app)

The following team members will be responsible for this portion of the implementation: Muhammad Choudhury and Ariful Islam. Currently we believe that creating a baseline code for the app that supports both IOS and Android might cause problems with connecting to the database. More research will be needed for this approach. We will be coding the application using React Native, which is an open-source framework written in JavaScript which is a language that all members of this team have experience programming in. The benefit of React Native is that it allows us to code just once and then afterwards can be deployed to work on both Android and iOS devices. This option is more optimal than having two separate codebases for

Android and for iOS as it will cut the team workload in half while maintaining the same throughput. Muhammad Choudhury and Ariful Islam will be responsible for creating the front-end portions of the application, which includes everything the end user sees and also what he interacts with. Muhammad Choudhury will specifically be resposible for making sure the app is funcional on iOS devices whereas Ariful Islam will be resposible for making sure the app is functional Android devices. Both of the front-end members will be using Visual Code Studio to code the application in JavaScript using the React Native framework. Muhammad will be using the iOS simulator from XCode and Ariful will be using the Android simulator from Android Studio to test changes in the app in real-time while changes are made to the code.

Back-End: (by Andy)

The following team members will be responsible for this portion of the implementation: Tchaly Leandre, Nathan Dacosta, and Andy Nguyen. The back-end portion of the system has a lot of aspects, the database portion and how information from the database connects to the IOS and Android versions of the app. The database we plan to create will be done by using AWS because it provides encryption for data being sent from and to it which is important for our sponsors, it also has back-end support with React Native which will make it easier to connect the mobile app with the external server. The database itself which will store all information related to the app will be implemented by Tchaly Leandre. The other aspect to back-end is how the database will connect with the IOS and Android apps for the system. Nathan Dacosta will be responsible for how the IOS app will connect to the database and Andy Nguyen will be responsible for how the Android app will connect to the database. This will primarily be done through API calls made in Lambda and connect through API Gateway, both are provided by AWS.

Database Structure (by Tchaly)

All information that the app will display will come from a database and any changes performed to the information within the app by the admins are reflected in the database.

- Admin Table
    - Contains the names, username, and passwords of all admins
    - Only admins can make someone an admin
    - Anyone list in this table have the power to make edits to information provided by the app
    - Admins can be added by other admins from the app
- User Table
    - Contains the names, username, and passwords of all users except for admins
    - Anyone listed in this table can only view the information displayed by the app
    - Users can be added by an admin from the app
- Aircraft Table
    - Contains all information about aircrafts: serial #, status, last flight, sortie type, remarks, estimated time of competition for maintenance (ETIC), parts ordered (MICAPS), location, and lead maintainer's name
    - When an admin makes changes in the individual Aircraft Info screen the changes will be reflected in this table.
- Building Table
    - Contains all information about buildings and parking spots for the aircraft.
    - When an admin makes changes in the Building Info screen the changes will be reflected in this table.
- Maintenance History Table
    - Contains all maintenance information for all aircrafts
    - Has the same fields as the Aircraft Table
    - When the admin updates the aircraft information the changes are inserted into the maintenance history table as a new field
- Maintenance Team Table

    - Contains the name of the lead maintainer and serial number of the aircraft that they are assigned to

- Aerospace Ground Equipment (AGE) Table
    - Contains the name and status of any equipment used for aircraft maintenance

API Details (by Andy)

API calls are how the app and the database communicate with each other. It allows the app to display data from the database and send changes to the database.

- /admin
    - Check if the username/password combination is in the admin table
- /admin/create
    - Add a new admin in the admin table
- /aircraft
    - Select everything from the aircraft table
- /aircraft/add
    - Add a new aircraft in the aircraft table
- /aircraft/remove
    - Remove a specific aircraft in the aircraft table
- /aircraft/update
    - Update a specific aircraft information in the aircraft table
- /equip
    - Select everything in the AGE table
- /equip/add
    - Add new equipment in the AGE table
- /equip/update
    - Update a specific equipment in the AGE table
- /location
    - Get the serial number of a specific location
- /maintenance-history
    - Select everything in the maintenance history table
- /users
    - Check if the username/password combination is in the user table
- /users/create
    - Add a new user in the user table

3.3 User Interface (by Muhammad)

The following are all the screens that the app contains.

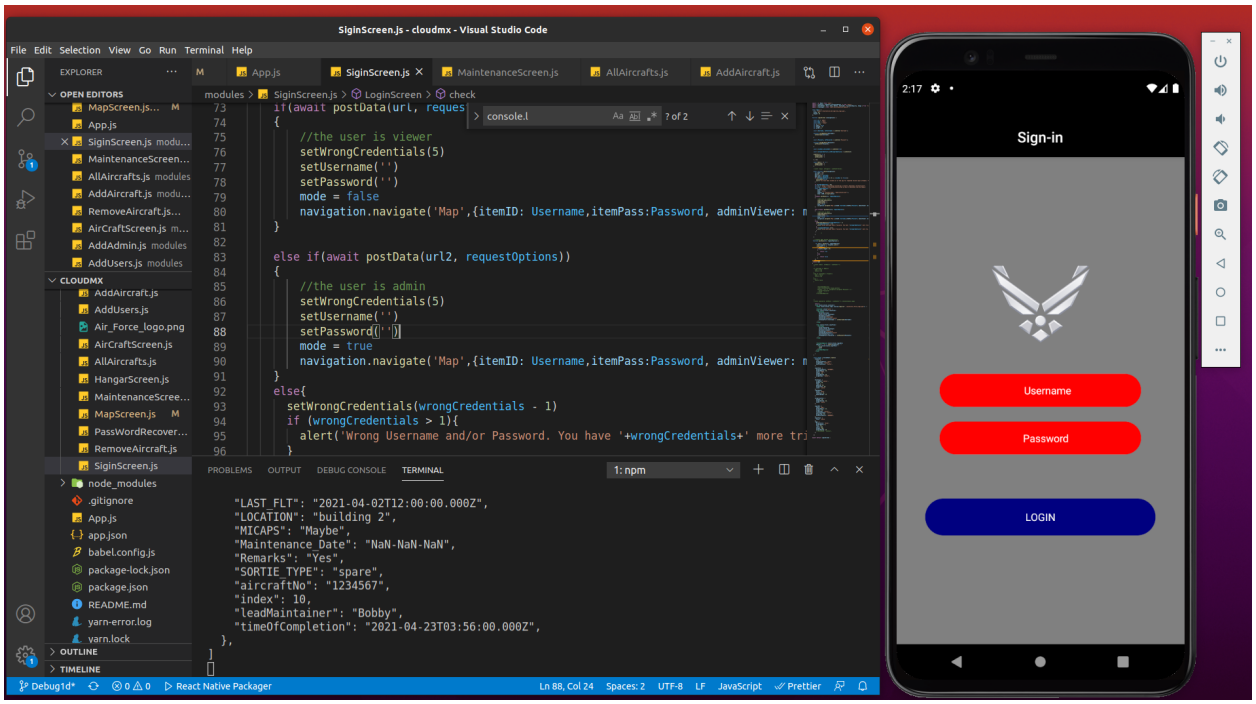

Figure 6: Sign-in Screen

Sign-in Screen: The user must insert their username and password to log in. They will enter the app on the home screen as an admin or viewer depending on which database their username and password belongs to

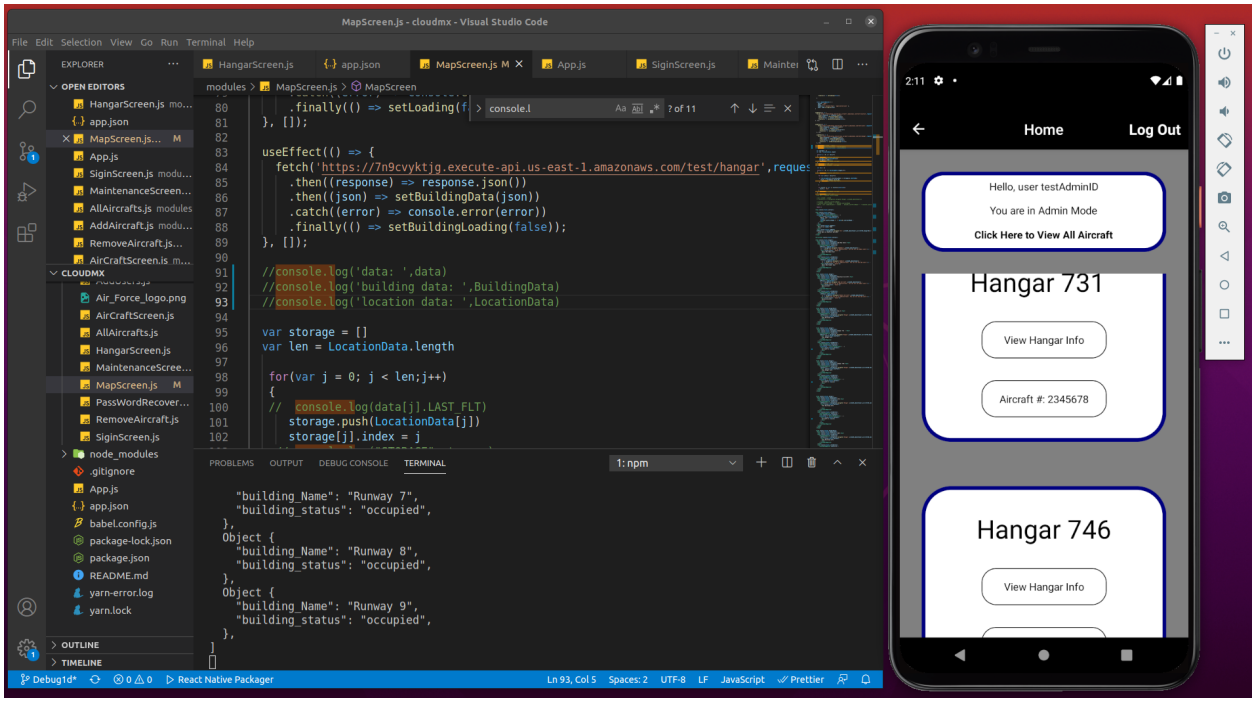

Figure 6: Home Screen (Map Screen)

Map screen: All the facilities of the map on the Dobbins base will be displayed including the serial numbers of the aircraft in each hangar or runway spot if an aircraft is currently within them.

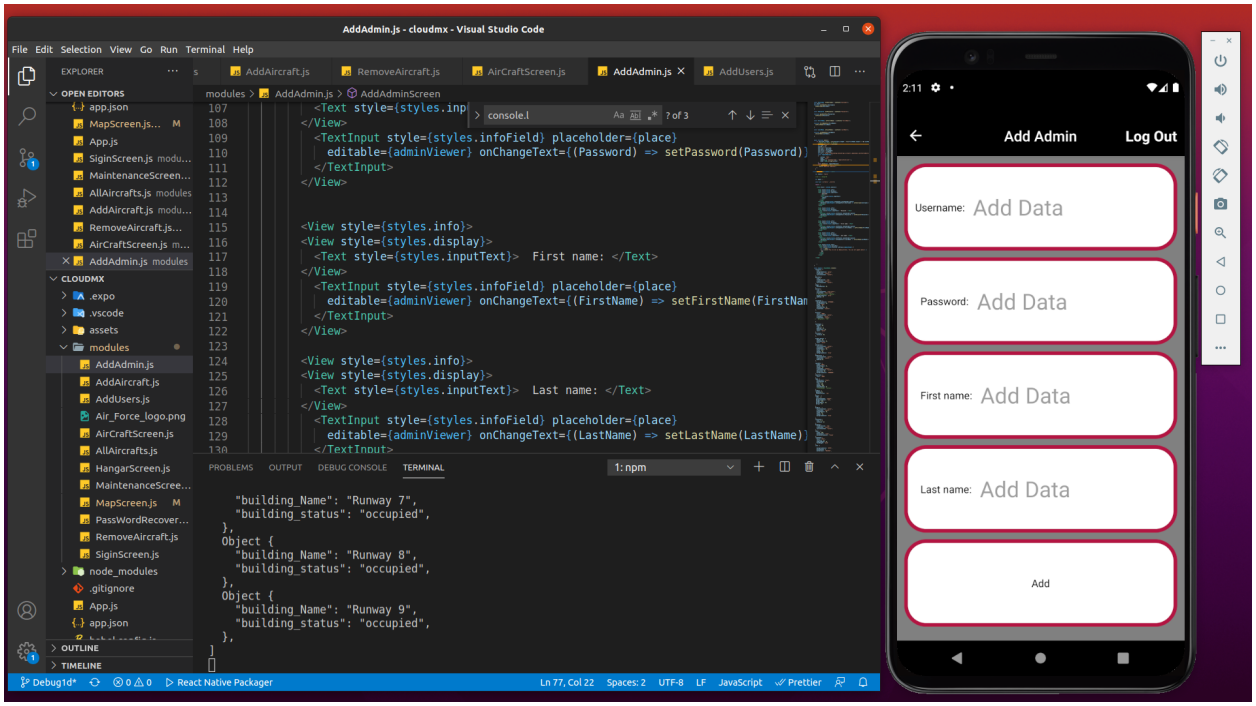

Figure 6: Add Admin Screen

Add Admin Screen: Admins can insert new admin users into the system. Only the username and password fields are required

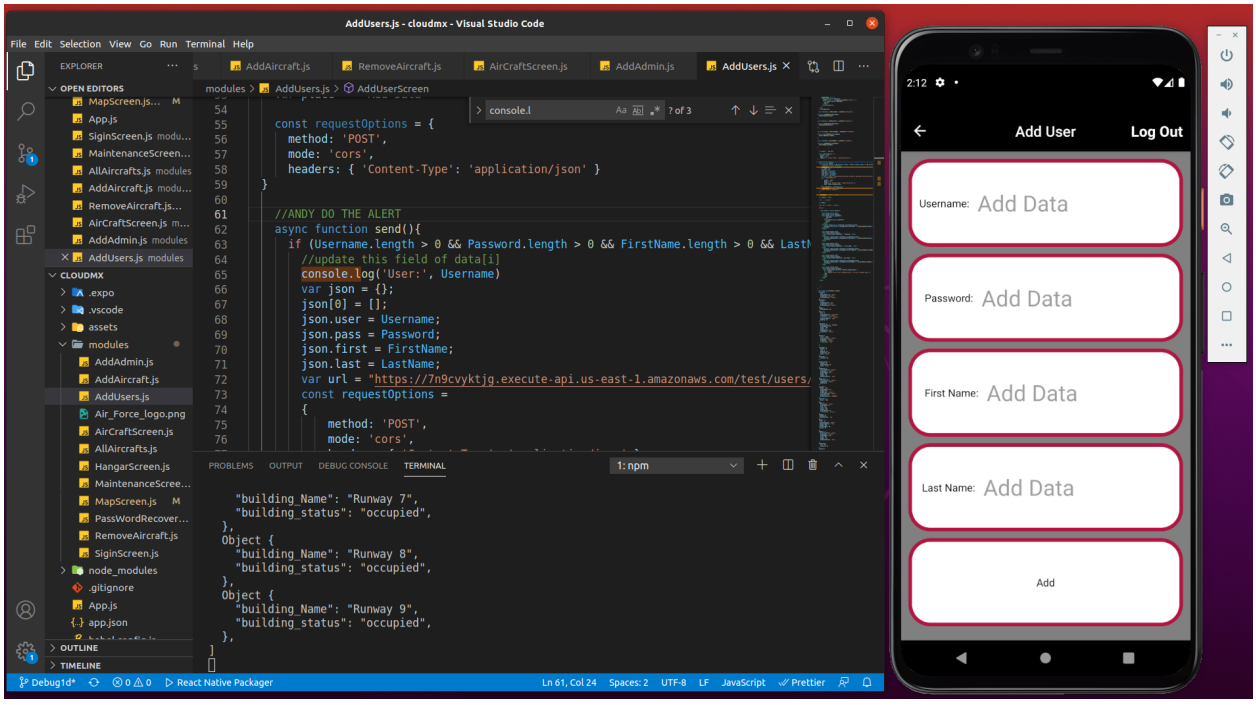

Figure 7: Add User Screen

Add Admin Screen: Admins can insert new viewer users into the system. Only the username and password fields are required

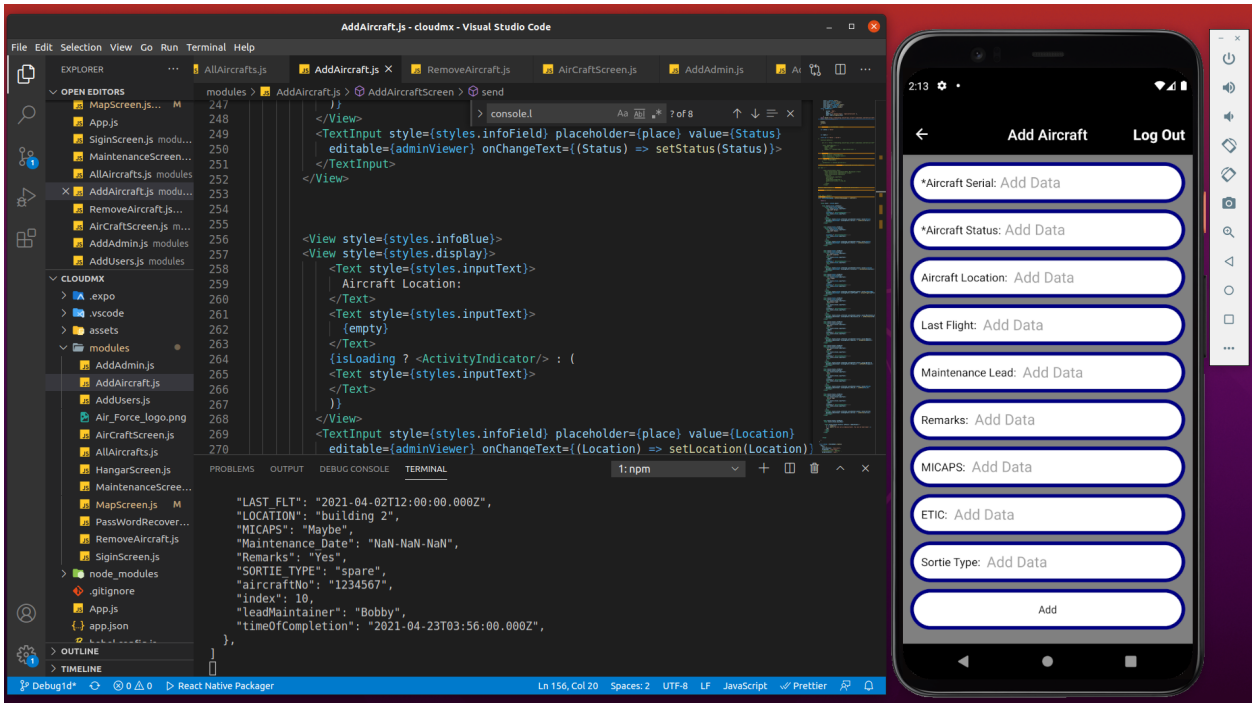

Figure 8: Add Aircraft Screen

Add Aircraft Screen: Admins can add new aircraft into the system. Serial number and aircraft status are required for adding an aircraft but other fields can be added as well

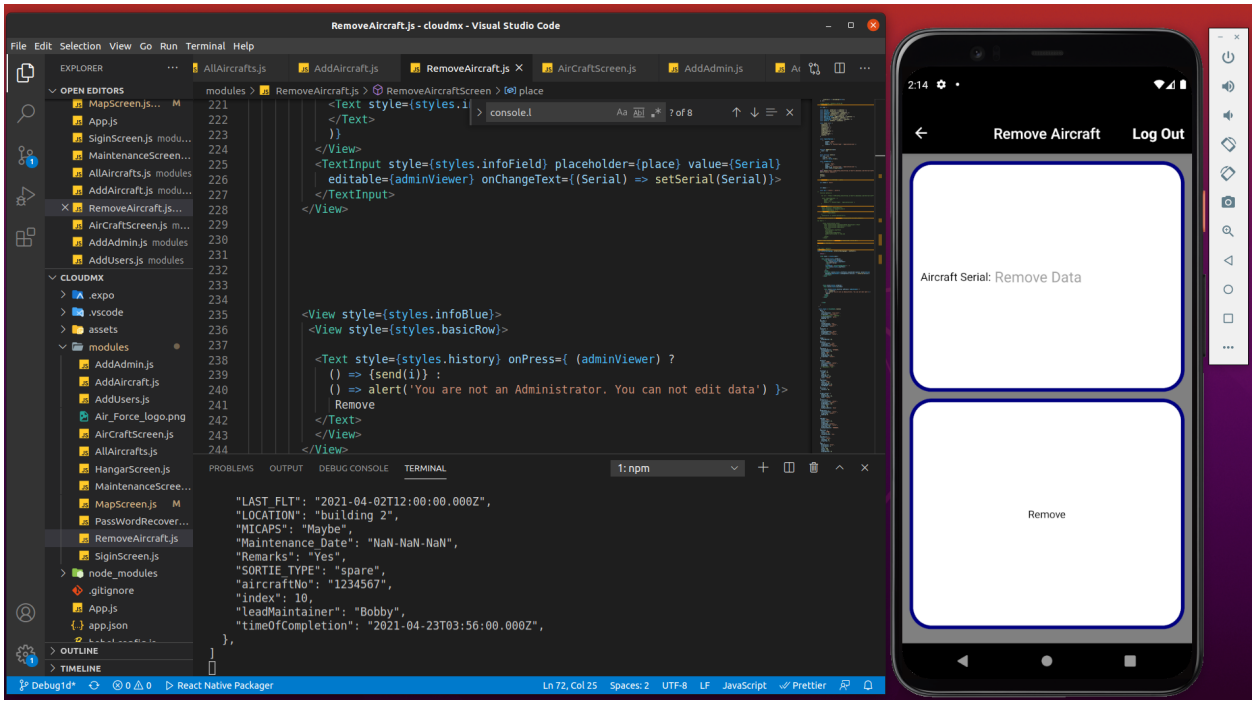

Figure 9: Remove Aircraft Screen

Remove Aircraft Screen: Admins can remove an aircraft from the system by typing in only the serial number of the aircraft only

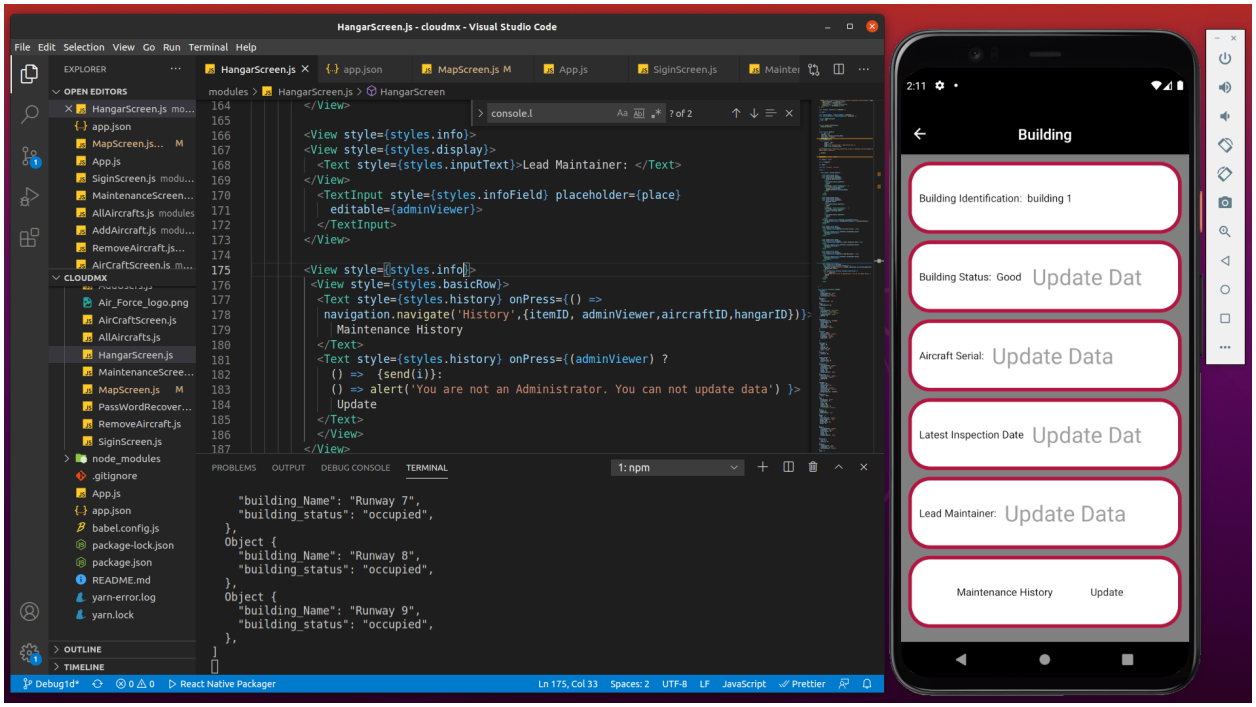

Figure 10: Building Screen

Building Screen: Shows all the information of each individual building that is clicked on. Admins can update any field of any building.

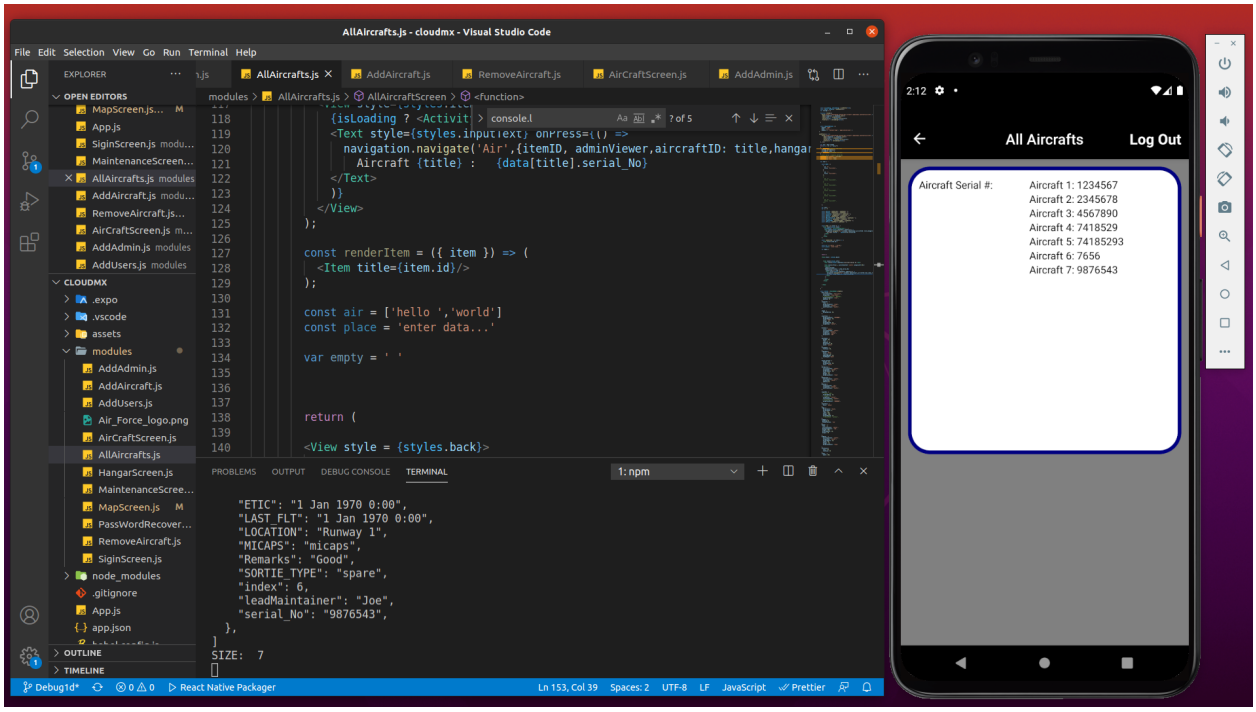

Figure 11: All Aircraft Screen

All Aircraft Screen: Shows all the aircrafts that are in the system along with their serial numbers in a list. Clicking on any aircraft will send the user to the page for that aircraft.

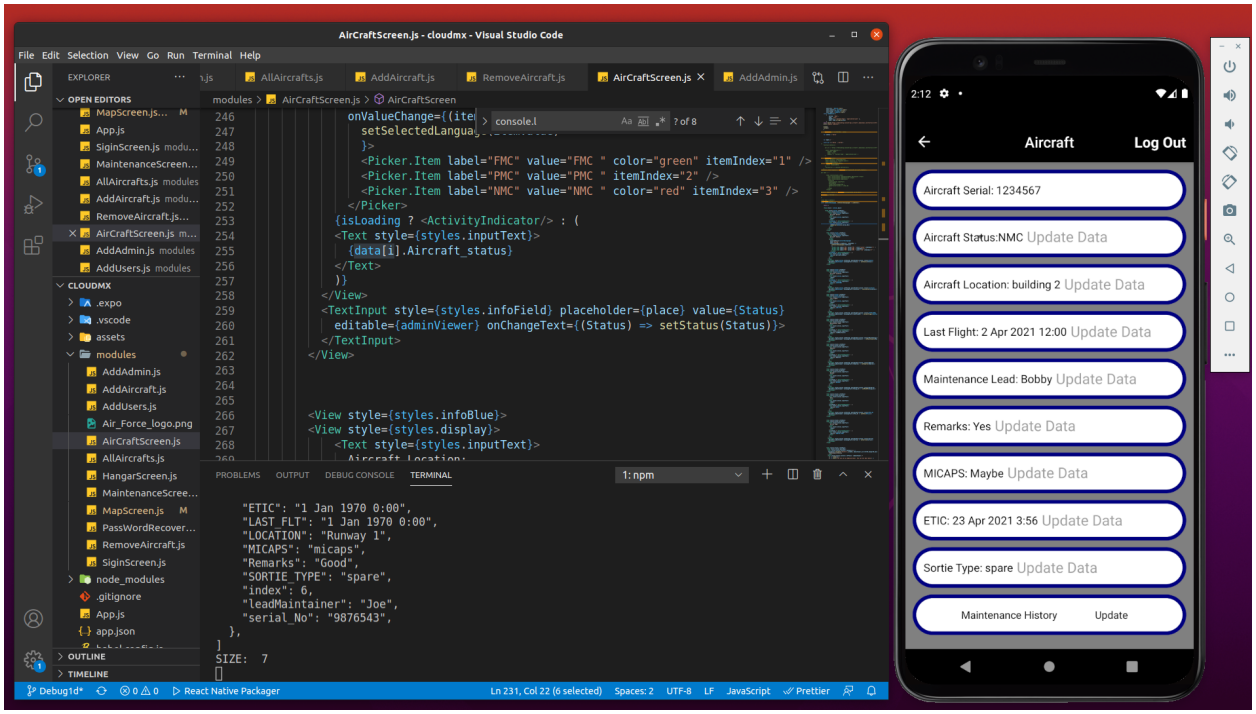

Figure 12: Individual Aircraft Screen

Individual Aircraft Screen: Shows all the information of each individual aircraft that is clicked on. Admins can update any field of any aircraft. Clicking on the maintenance button will take the user to the maintenance history page of that aircraft.

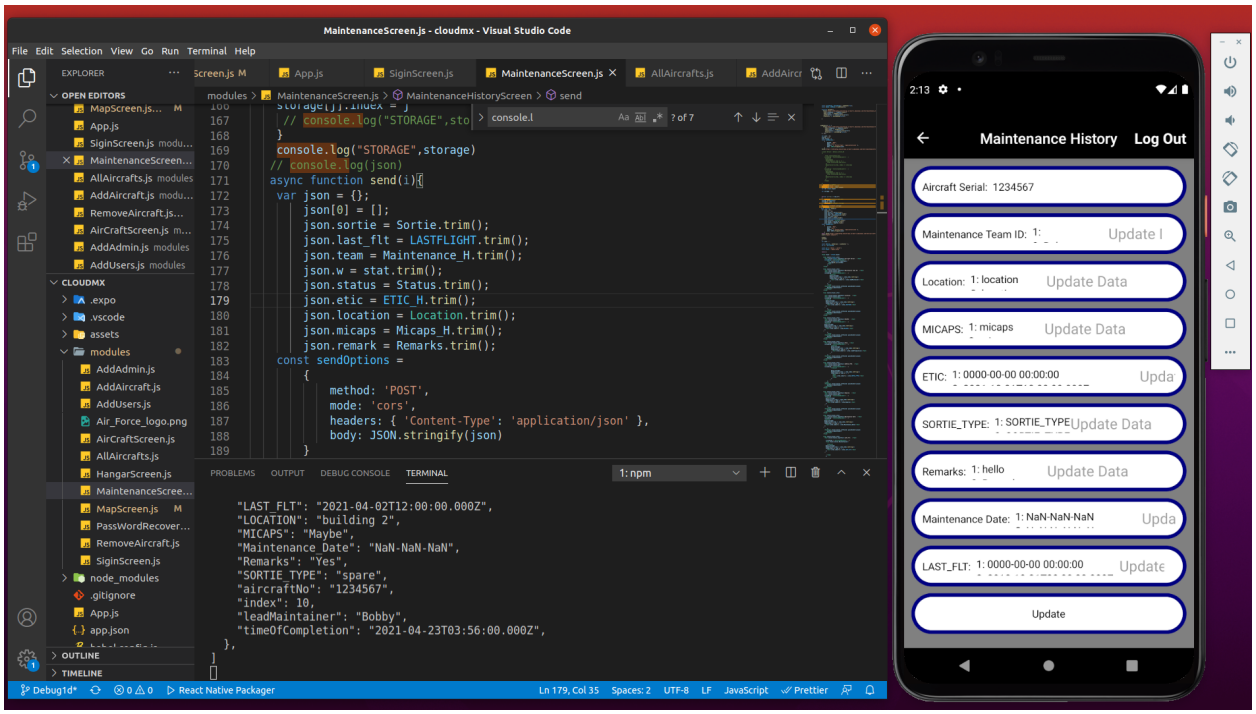

Figure 13: Maintenance History Screen

Maintenance History Screen: Shows the detailed maintenance history of each aircraft. Each field appears as a list because each aircraft is periodically updated by the crew

3.4 Testing (by Nathan)

Throughout the development of this system numerous testing has been done. Testing was done to each screen to see if it functions properly.

The sign-in screen was tested by filling the username and password with the correct credentials. After the screen was developed a test username and password was added to the user table and the admin table in the database. The username and password created for admins was testAdminID and testAdminPassWord respectively. The username and password used for normal users was testUserID and testUserID respectively. After adding the test credentials to the database we use it in the sign-in screen. The sign-in process was successful and we were taken to the map screen which is supposed to happen. The map screen will also have text at the top saying whether the current user is in viewing mode or admin mode.

The mobile application can distinguish between a normal user and an admin based off of which credentials used to sign-in. The purpose of this is to allow only admins to make changes to the database through the app while preventing normal users from doing the same. To test if the modes are working correctly we signed in using normal user credentials and attempted to do things that only admins should be able to do such as changing data or attempting to access the screens that only admins can access. If a user tried to do any that was listed previously they would get a notification saying that they are in viewing mode and that they cannot make changes to the data from the app. This test was successful, normal users are unable to make any changes to data from the app while admins can free do so.

Now to how we tested how the app responds to an admin making changes to the data. We had a test admin make changes to the data from the app and we checked the database if the changes were reflected correctly after the test admin pressed the update button. This took the most amount of time because we had to be through and account for all scenarios. We also had to be careful of how other aspects of the database are affected, this is mainly the case for Maintenance History. The way the api calls are set

up when an admin makes changes to an aircrafts information it not only updates the aircraft table within the database but a copy of the change is inserted into the maintenance history table. Fortunately after much testing we managed to successfully update both the aircraft table and the maintenance history table. We also had to make sure that the data from the database is presented in the app in a way that users can understand. For example, in MySQL a date and time variable is stored within a data type called datetime which stores information in the following format: YYYY-MM-DD hh:mm:ss. However, the sponsor requested for that kind of information to be presented through the following format: DD-(month initials)-YYYY hh:mm. Doing this was difficult because we had to make sure that the format that admins were using to change data can be converted to a value that can be stored in the database. The conversion was done through the api call in which it takes the admin's input and converts it into something that the database and store properly.

We have been successful in our testing of the app and have concluded that it works as it should.

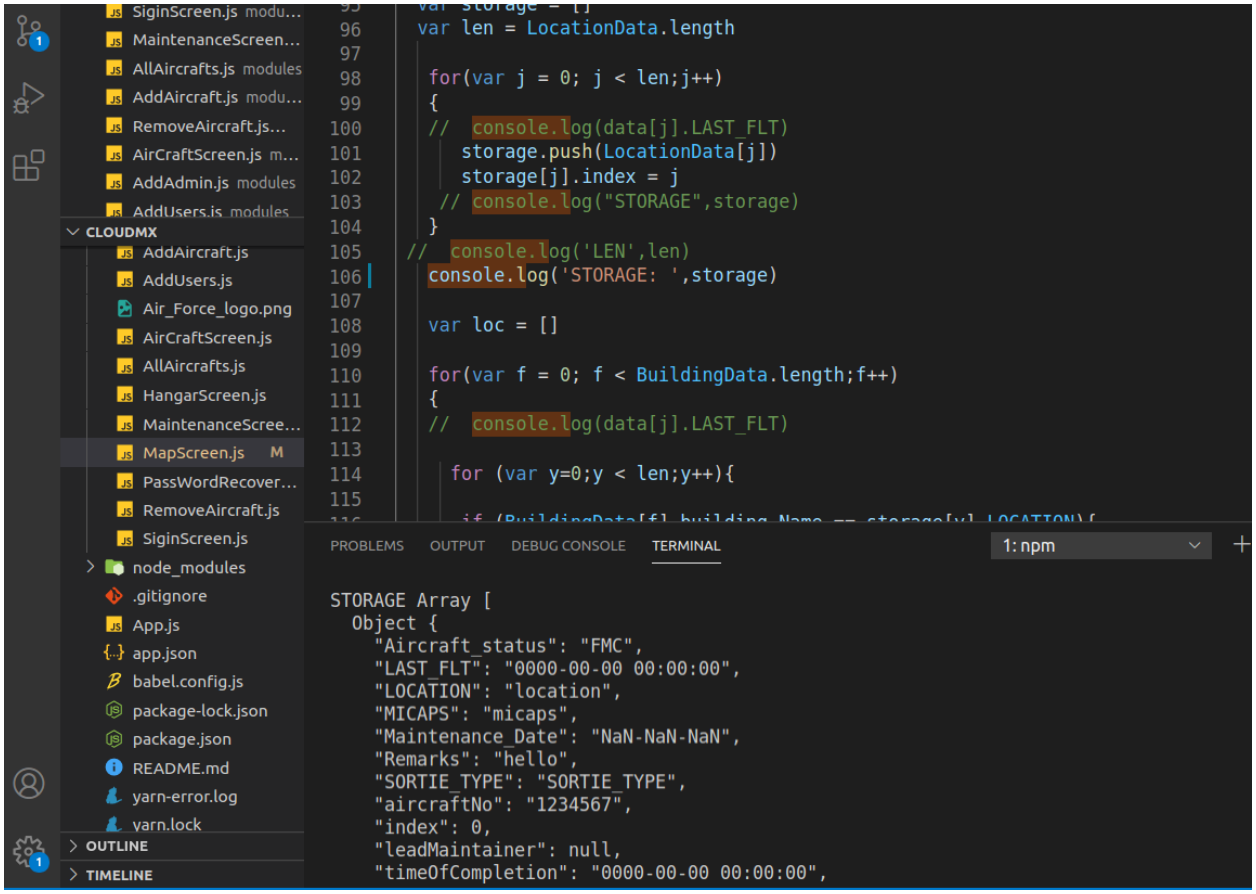

Figure 12: Example of Testing

Console Logging was done for debugging purposes. Information such as the contents of an array or variable would be displayed on the terminal in Visual Studio Code

3.4 Lessons Learned

By Muhammad

I am personally glad that this project was given to us because it really tested and pushed my ability as to what I can do. This is very easily the most complex application that I have worked on; it is also the first software that I have taken apart in creating that will actually be used in an actual organization. And I also learned the skills of communicating with other members in a dynamic way which included helping other members when they needed it. There are many cases throughout the two semesters where I felt like I was overcome with some of my other classes and also with applying to

jobs for when after I graduate, but I always placed this project ahead of everything else, and I feel proud and accomplished looking at the current end product.

By Ariful

I learned a lot about mobile development and the services provided by Amazon Web Services and I got the opportunity to get insight on how air reserve bases perform aircraft maintenance such as appreciations that they use. For example, we learned that aircraft status is labeled under three terms: FMC, PMC, and NMC which means fully mission capable, partially mission capable, and not mission capable respectively. I also learned a lot about coordinating teams and the importance of effective communications. I'm used to working on my own so I never had to articulate my thoughts about the design of a project with others.

By Tchaly

This project was very eye-opening for me. This is the first time I had to make a database that was actually being used and not just for a class. I do not believe its structure is one-hundred percent perfect, and could do some ironing out, but given the scope of this project, it did not hamper the functionality of the application. I see where I can improve in database design and this experience will definitely be very helpful if I decide to pursue a career in Database Design. Other than this, having experience with working while being a part of a team is always helpful, so it goes without saying that this experience was very fruitful for me.

By Andy

I learned a lot from this project as I learned about the services on Amazon Web Services (AWS) such as Amazon API Gateway and Amazon Lambda. Making the REST API was tough at first, but it got easy as I continue to work on the API. The Amazon Lambda functions were the hardest part as I have to create functions to do certain functionality like querying the database, or adding an aircraft to the database. Then, the function will have to send data to the mobile device such as the JSON data of a table. This experience gave me a better appreciation of working on the back-end of a mobile

application. In addition, working with my group members improves my communication skills that could be used in the future.

By Nathan

I learned quite a bit from this project. While going through the sign-in screen especially, I started to understand the intricacies of having to connect a front end to the back end of the app itself. While I wasn't specifically tasked with that job, I did end up debugging and breaking the application (testing for all possible scenarios) to the best of my ability. Another thing that I learned from the development of the app is React-Native. Having experience in React-Native now will be a great tool to add to my development toolbox, as well as gave me insight into making a pleasant and functional application on multiple platforms.

4. Project Management (by Ariful)

4.1 Project Workload Distribution

This section will break down how work is divided among the five of us. Since this project requires the app to run on both IOS and Android devices we made Muhammad Choudhury and Nathan Dacosta in charge of the IOS aspect becuase they have IOS devices to test the app on and get an accurate simulation of how the app interact with IOS devices. Similarly, Ariful islam and Andy Nguyen are in charge of the Android aspect because they have access to devices that can accurately simulate how the app would interact with an Android device. The following is a more detailed list of the responsibilities for each team member:

- Ariful Islam (BSCS)
- Tchaly Leandre (BSCS)
- Muhammad Choudhury (BSCS)/(BSCE)
- Andy Nguyen (BSCS)
- Nathan Dacosta (BSCS)

System	Task	Responsible
Back-End	Create database	Tchaly Leandre

Front-End	Create the sign-in screen	Ariful Islam
Back-End	Update database with information related to sign-in screen	Tchaly Leandre
Back-End	Create queries for api call	Tchaly Leandre
Back-End	Connect sign-in screen to database	Andy Nguyen
Front-End	Create Map screen	Muhammad Choudhury
Front-End	Create Building screen	Muhammad Choudhury
Back-End	Update database with information related to building screen	Tchaly Leandre
Back-End	Create queries for api call	Tchaly Leandre
Back-End	Connect Building screen	Nathan Dacosta
Front-End	Create Overall Aircraft screen	Ariful Islam
Back-End	Update database with information related to overall aircraft screen	Tchaly Leandre
Back-End	Create queries for api call	Tchaly Leandre
Back-End	Connect Overall Aircraft screen	Andy Nguyen
Front-End	Create Aircraft screen	Muhammad Choudhury
Back-End	Update database with information related to aircraft screen	Tchaly Leandre
Back-End	Create queries for api call	Tchaly Leandre
Back-End	Connect Aircraft screen	Nathan Dacosta
Front-End	Create Maintenance History screen	Muhammad Choudhury
Back-End	Create queries for api call	Tchaly Leandre

Back-End	Connect Maintenance History screen	Andy Nguyen
Front-end	Create Add User/admin screen	Ariful Islam
Back-End	Create queries for api call	Tchaly Leandre
Back-end	Connect Add User/admin screens	Nathan Dacosta
Front-end	Create Add/Remove Aircraft screen	Muhammad Choudhury
Back-End	Create queries for api call	Tchaly Leandre
Back-End	Connect Add/Remove Aircraft screen	Andy Nguyen

5. Conclusion (by Andy)

If this project will have a mostly local impact because we were asked to create the app specifically for the Dobbins Air Reserve Base. If the project is successful it can improve the maintenance of aircrafts for the base. However, we do not know if other air reserve bases have outdated ways of conducting aircraft maintenance. If so then this project can have a global impact if it were to be expanded to other bases. We kept in mind the potential of the project to be used on other bases and it can be applied to other bases with little changes to it.

The main ethical and professional consideration for our project was security. Our sponsor requested for the app to provide encryption to the app since information within it is considered sensitive. We did include encryption to the best of our abilities, however we cannot boast that it is military grade. It is our hope that if the project were to be used, that the information it would contain remains in the right hands. If our project should be used, we highly recommend that the encryption be upgraded to military grade.

Considering that this is an application made for members of the US Air Force, our group understood that we needed to apply a considerable amount of effort and

sophistication into this project. It would be unwise to give the armed forces that protect us a middling application that barely functions so we each underwent a large amount of research of React Native and AWS, which is what the app uses for a backend server, so that we could give the airmen wing the best product possible. It is our hope that the sponsor and anyone else that would use this project will be happy with it and serve them well.

6. References

“AWS Free Tier.” *Amazon*, 2014, aws.amazon.com/free/?trk=ps_a131L0000085EJvQAM.

This is where we learned about AWS

React Native, Facebook, reactnative.dev/.

This is where we learned about React Native.

“React Navigation .” *React Navigation Blog RSS*, reactnavigation.org/docs/navigating.

This is where we learned how to navigate between pages in React Native.

“Expo CLI.” *Expo Documentation*, docs.expo.io/workflow/expo-cli/.

This is where we learned about Expo and the Expo CLI (Command Line Interface)

“Chapter 3 Tutorial.” *MySQL*, dev.mysql.com/doc/refman/8.0/en/tutorial.html.

This is where we learned how to create queries for MySql.

“Run Apps on the Android Emulator : Android Developers.” *Android Developers*, Google, developer.android.com/studio/run/emulator.

This is where we learned how to use the Android Studio Emulator

"Running Your App in the Simulator or on a Device : Apple XCode." *Apple Developer Documentation*, Apple, developer.apple.com/documentation/xcode/running_your_app_in_the_simulator_or_on_a_device.

This is where we learned how to use the XCode Emulator

"Tutorial: Create a Calc REST API with Two AWS Service Integrations and One Lambda Non-Proxy Integration." *Amazon Web Service*, Amazon, docs.aws.amazon.com/apigateway/latest/developerguide/integrating-api-with-aws-services-lambda.html.

This is where we learned about how to connect the REST API to Amazon Lambda functions.

"Cloudmx" *Github*, https://github.com/ndacosta2017/cloudmx

This is where the source code for our project is located.