Operation RM
Northern Arizona University
Flagstaff, AZ

William Rogers (Team Lead):	wpr29@nau.edu
Isaac Faulkner:	lwf2@nau.edu
Andrew Milizia:	am5275@nau.edu
Nick Henderson:	nsh67@nau.edu

CS476 Technological Feasibility

November 10, 2023

Operation RM

Client:

General Dynamics Mission Systems

Mentor:

Italo Santos

Team Members:

William Rogers

Isaac Faulkner

Andrew Milizia

Nicholas Henderson

Overview:

The purpose of this Technology Feasibility document is to identify key technological challenges and design decisions. With this document the team can construct a plan for well structured, data-driven analysis to make informed decisions and avoid late development stage setbacks.

Introduction

Operation RM is undertaking the development of an easy to use Android application designed to enable tactical communication with a Remote Communication Interface (RCI). This aspiring operation is made possible through the client General Dynamics Mission Systems, an imperative player in the defense and technology industry.

General Dynamics has identified the need for a user-friendly and efficient method of communicating with their RCI system via Android mobile application. Operation RM's mission is to create an intuitive, feature-rich user interface that can incorporate essential features to meet the clients needs.

The visualized Android app will be an accessible communication hub that resembles the familiar layout of the clients pre-existing email application. It will contain an assortment of features such as sending and receiving files. Furthermore, the application will support audio and camera integration, granting the ability to add videos, audio messages, and photos and open them in the associated application.

An imperative functionality of this application is the ability of transmission and reception of files. The application will be able to connect to radio via a set of user-defined settings to send and receive files through the connection with the radio simulator, and radio modem using the supplied RCI interface via Wi-Fi. Multiple outgoing messages will be allowed to be queued up, with up to 10 messages containing up to 20MB of data each. The user will be notified of a successful transmission or any potential transmission issues, ensuring the user is constantly kept informed.

In this technological feasibility analysis the aim is to thoroughly examine the project's imperative design requirements and technological challenges. Starting with identifying these challenges, investigate potential solutions, evaluate alternatives, and lay the groundwork for the mobile application that resolves the problems identified by General Dynamics. This document will serve as the roadmap for systematically addressing challenges to ensure success.

Technological Challenges

Now that the client's desired functionality of the application has been established, namely a comprehensive Android application to connect to a radio modem and transmit data via an email queue, it is important to further understand each technological aspect of the project. It is crucial to ensure the feasibility of the potential technologies that will be used. This includes investigating challenges that will be encountered and what can be done to overcome them. Once each challenge has been overcome, it is important to integrate the suite of potential solutions together into one cohesive solution.

In the following section, the challenges associated with the development of the Android application will be investigated to provide an in-depth understanding of each challenge which will help in identifying the best potential solution possible. The major technological aspects that will be analyzed will be the development spaces, programming language, structure of the email queue, and virtual environment.

Programming Language

A project's programming language can make a significant difference in the overall structure and what utilities are able to be used. It is important to understand the benefits and drawbacks of each programming language when designing a project. For this project, choosing a programming language that supports Android development is a hard constraint. The overall effectiveness of the language with Android development will be assessed. In addition, the compatibility of the programming language with the Radio Control Interface (RCI) API will be evaluated. Finally, debugging capabilities will also be assessed. It is crucial to determine a programming language for further development and decision making.

Development Spaces

Developing an Android application can be done in a plethora of development spaces. It is important to consider the trade-offs of each one and how it will interact with the radio modem and its simulator. One element of consideration is Android emulator support due to Operation RM having possession of only two physical devices for testing. Having emulator support with the development space would allow for all team members to have the potential to test the code they are working on without needing to have a physical device with them. In addition, physical debugging for an external Android device would likewise be desired. The development space should also support the desired programming language. Finally, the cost for licenses should also be considered, as it is important that the development space can remain accessible to all members of the team.

Email Queue Structure

Developing an effective data structure for the email queue functionality is a necessity due to the potential of having up to 10 messages in the transmission pipeline at a given time. Managing the messages as they are staged for transmission and removed after transmission can be done in a variety of ways. Depending on the structure used, it will also be important to establish a method of scheduling the messages to be sent. This can depend on the criteria which could lead to a weighted schedule, overall fairness, or somewhere in between. Upholding organization of the messages to be transmitted is crucial for upholding the system's integrity and potential efficacy.

Virtual Environment

The radio simulator uses a C program running on Linux. Therefore, it is important to establish the best linux environment for testing the communication between the mobile application and the RCI. Currently, the simulator has been pre-configured on a virtual machine image, so compatibility with such would be desired. It must also be compatible with the selected development space in order to effectively integrate all of the project's technologies. Network support is also a necessity for the virtual environment since wifi will serve as the primary connection between the mobile application and the radio simulator.

Technology Analysis

There are many important considerations when developing the solution. Such considerations include the programming language, which will determine the structure and potential utilities that can be used. Another consideration is the development space, i.e. the development environment that will be used for programming and debugging. In addition, determining the email queue structure is also important for maintaining system integrity. Finally, the consideration of the virtual environment is crucial for ensuring ease of workflow for working with communications between the mobile application and the radio simulator.

Programming Language

It is important to identify a programming language that will be compatible with an Android application and has an easy to use debugger. It also must have compatibility with the given API from the client. This is integral to ensuring proper functionality of the application. In addition, data structure creation should be supported.

Desired Characteristics

- **Compatible with Android**. The programming language must be compatible with Android development systems. The more compatible the programming language is, the easier it is to utilize in development.
- **Easy Debugger**. The language should support an easy-to-use debugger to enable swift development. This will allow the identification of erroneous lines of code or other undesired behavior.
- **Compatibility with RCI**. The client-provided Radio Control Interface (RCI) API must be compatible with the chosen programming language, since the mobile application must be able to use the API for function calls.
- **Ability to use data structures**. Data structures will be a necessity for being able to queue messages for sending. It is important that the chosen programming language provides support for this.

Alternatives

There are a few programming languages that were investigated for this project, each with their respective strengths and weaknesses.

Java is a popular programming language first created in 1991, and has since been used for many applications. Java is often chosen as the language used in mobile applications. Java is a high-level object oriented programming language that supports many useful libraries and utilities.

Another option is Kotlin. Kotlin is a new language compared to Java, but there are many similarities. While Kotlin is newer, the documentation is definitely not as extensive as Java. However, Kotlin has an easy debugger to use. There is the potential that Kotlin will be challenging to connect to the API.

The last option is C. C is the default language for the API so it will be easy to connect. The problem with C is not typically supported for Android application development. C has a lot of what is needed to connect to the API, but it might be a bit more complicated to use to make an Android application.

Analysis

First, Java was evaluated. It is Android compatible since it is one of the most common Android development languages. It also has its own built-in debugger, Java Debugger(Jdb) which is straightforward to use. Java can communicate with the C API using the Java Native Interface. For data structures, Java already has a library of built in data structures, but implementing one from scratch would also be simplistic.

C was then analyzed. It is possible to use the language for Android development, but lacks the amount of support the other two languages have. C also has its own debugger known as the GNU debugger (GBD). The RCI compatibility with C is the best since it was written in C and can be directly used. It also has great data structure support through the use of structs and dynamic memory management.

Kotlin is the last language to be analyzed. It is another primary Android development language so it is easily compatible. Debugging is done with IntelliJ IDEA along with downloading plugins and another program to allow IntelliJ to debug Kotlin which is quite inconvenient and may cost a bit of time to get set up. Kotlin will likely have compatibility issues with the RCI due to it being a newer language which would potentially cause the team to struggle when working with the RCI. For data structures, Kotlin is very similar to Java and has numerous built-in data structures.

Chosen Approach

Each candidate was evaluated on a scale from 1-5 on the basis of how well the desired characteristics were met. This scale will be used for the analysis of the further challenges.

By taking the results of Table 1 into account, the most ideal language to use will be Java. With most of the code being the Android application, it will be best to focus on that. It might be a bit less compatible with the API, but overcoming this issue should be simple.

Kotlin is too new to use in this case because a firm understanding of the language is needed for the client and the team. Kotlin integration with the API might also be a bit harder to perform. So taking that into account as Java and Kotlin are about the same in Android app development .

Since most of the approach is to focus on Android app development, C is out of the picture because it has the least support with it, even taking into account the compatibility with the API.

	Java	Kotlin	C
Android Compatible	5	5	2
Easy Debugger	5	5	5
RCI API Compatibile	5	3	5
Data Strucutre Support	5	5	5
Total	20	18	17

Table 1. Programming Language Analysis

Proving Feasibility

Java will be used going forward for Android development, but if further evaluation concludes that the API is too difficult to integrate, reevaluation might be necessary. Java looks to be the best possible language to use because of its compatibility with both the API and Android app development. To prove this, a small test Android application will be developed along with testing some of the connection to the API.

Development Spaces

Having an intuitive platform that meets the needs for developing the Android application will be crucial to ensuring efficiency and ease of programming and debugging. Such needs include the ability for interactive debugging on the client supplied devices, and a polished development environment with code-completion. This makes the choice of development space one of the top priorities when determining the feasibility of such a project.

Desired Characteristics

There are many desired characteristics that are important to take into account when deciding on a development space:

- **Low cost for licenses**. This is important to ensure the ease of access of the application for the team members, regardless of financial status. This would also allow us to easily switch development devices with little or no cost.
- **Live debugging on an external device**. This is crucial to validate the solution using devices that fit the use-case provided to us by General Dynamics. In particular, the chosen development space would need to support live debugging with Android 12, 13, and compatible devices, specifically the Samsung S22. The client has provided two S22 phones for live debugging, so it is crucial the team is able to interface with them through the development space.
- **Supports an interactive simulator of an Android 12, 13, and 14 device for debugging**. While the client has provided two phones for live debugging, additional testing by members not in possession of the phones will need to be done via simulator.
- **Supports Java builds.** It is now established that Java will be used as the programming language. Therefore the development environment should support

Java. The development environment should also include code-completion for Java libraries for ease of use.

Alternatives

There are many potential solutions for the development space decision, each with their own benefits and considerations.

One potential solution would be to use the Android Studio IDE. Android Studio is a mobile development IDE built by Google and JetBrains, with the first release in 2014. Android Studio is built upon the IntelliJ IDEA IDE, a popular Java development environment. Android Studio is one of the most popular mobile development environments and is the only official mobile development environment for Android.

Another potential solution would be to use Visual Studio with Xamarin. Visual Studio is an IDE developed by Microsoft with an initial release in 1997. Since then, Visual Studio has become a massive IDE with support for many languages. Xamarin is an extension of Visual Studio also developed by Microsoft to support Android and iOS development.

One final solution would be to use the Eclipse IDE with ADT (ABAP Development Tools). The Eclipse IDE is a powerful Java IDE first created in 2001 by the Eclipse Foundation. The ADT extension can be added to Eclipse in order to support mobile Android development.

Analysis

Android Studio will be the first development space to be analyzed for the desired characteristics. Android Studio is a free development space which allows the team to easily begin development. It also supports live debugging by allowing a device to be connected to the development computer and uploading the app to that device. It also has a suite of interactive simulators ranging from different devices to versions which will be valuable for testing. Java is also supported with Android Studio which will allow development of the app with the chosen programming language.

Visual Studio with Xamarin is next. It is also free to develop with this development space. Live debugging is supported very similarly to Android Studio with connecting a device to the development computer. Visual Studio also does support simulator debugging similar to Android Studio. The main downfall of this development space is that it does not support Java which is the chosen programming language.

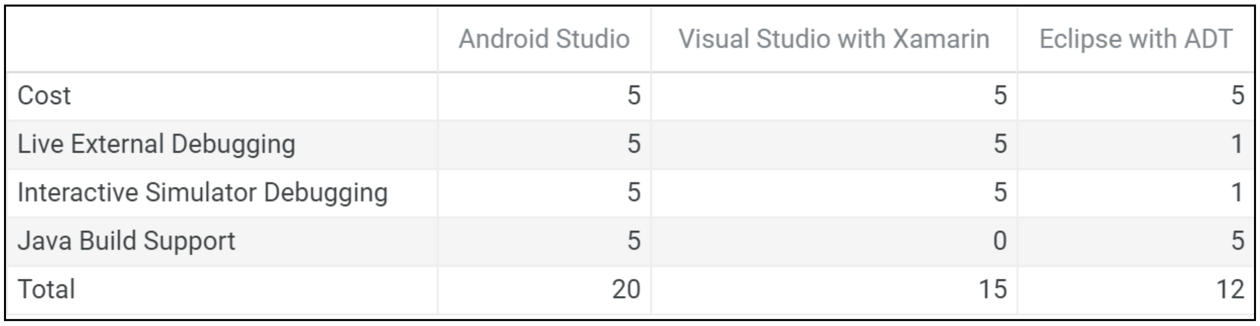
Eclipse is the last to be analyzed. It is also a free development space, just like the other two candidates. Unfortunately, it does not have clear instructions on whether live debugging is supported since there is limited information on this characteristic. This is the same for simulator debugging which really limits the testing functionality of the development space. However, Eclipse does support Java and contains numerous tools to aid in development.

Chosen Approach

Taking into account all of the requirements and needs, the ideal solution for a development space would be to use the Android Studio IDE according to the scores found in Table 2.

While Eclipse provides many advantages for Java development, and Visual Studio provides many advantages for cross platform development, for the purposes and requirements, Android Studio will work the best. In addition, Android Studio has a wide range of tutorials and documentation that provides many advantages beyond the basic requirements.

The following chart displays a comparison of the advantages and disadvantages of each of the candidates:



	Android Studio	Visual Studio with Xamarin	Eclipse with ADT
Cost	5	5	5
Live External Debugging	5	5	1
Interactive Simulator Debugging	5	5	1
Java Build Support	5	0	5
Total	20	15	12

Table 2. Development Space Analysis

Proving Feasibility

Android Studio will be continually evaluated for these desired characteristics going forward. One of the first tests that will be conducted is to create a sample Java application and be able to build it for Android 12, 13, and 14. Next, the interactive simulator will be used to ensure proper debugging functionality. The Android Studio IDE will also be tested for direct compatibility with the client-supplied Android devices. One demonstration to prove this requirement would be to create a sample application and connect the phone to the host device to ensure proper functionality.

Email Queue Structure

The email queue structure in terms of data structure and queue priority will be essential for implementation. Choosing the right queue structure will allow the efficient and logical handling of emails.

Desired Characteristics

There are many considerations when determining the best structure for the email queue.

- **Allow for 10 emails to be queued, each with a maximum size of 20MB each**. This means that the queue should be robust enough to be applied to relatively large file sizes, and handle a total of at least 200 MB of data.
- **Contain logical ordering and be predictable**. When a user queries an email, a logical ordering should take place, such that the user can understand what order the emails will be sent in. This can be in terms of size, priority, or time sent.
- **Handle failed file uploads**. If packets are misordered or lost when uploading the data, such as when a user experiences a poor connection to the radio modem, the queue should be able to handle skipping or removing a file upload if detected to be corrupted.
- **Minimal complexity to implement**. This will provide more time to focus on more complicated project components along with allowing the program to be easier to understand for the client for if they decide to add onto it in the future.

Alternatives

There are many solutions to this challenge, each of which has their own advantages and disadvantages.

First, a queue data structure could be used in conjunction with a linked list. In this data structure, each email that is queued up will logically be sent to the radio modem in the order that they are queued. A linked list will singly link each file in a chain.

Second, a stack data structure could be used. In this structure, the files will be logically sent with the last queued file sent first, after each is added to the stack.

Third, a heap structure. In this way, emails are added to the heap with consideration for their size. Then, the file with the minimum size on the heap will be executed first.

Analysis

The first data structure to be analyzed for meeting the desired characteristics is the queue. Regarding capacity, it is able to handle the maximum potential capacity since the linked list that wil structure the queue points to the location of the files that will be transmitted. A queue also has intuitive logical ordering since it is structured just like a line. Handling upload failures is another capability of the queue since the queue would just skip over the current email and go to the next in line. Making a queue out of a linked list is also a low complexity implementation of a data structure so it favors this characteristic.

The stack data structure is up next for analyzing it for the desired characteristics. For capacity, it is built into Java, but would not be able to contain the maximum size without having to change flags on application startup which hinders its effectiveness. Handling upload failures is done well with the stack since the corrupted upload would simply have to be popped off the top of the data structure. Regarding complexity, it is a little more complex to handle the push and pop functionality rather than a queue's enqueue and dequeue.

Lastly, the heap data structure was analyzed. The heap that is within Java can hold 1GB by default, so it also is effective at storing the maximum size of the email queue. Logical ordering is something the heap has as well with either the min heap or max heap configurations. For handling file upload errors with the heap, removing corrupted nodes and splicing the list together is a bit more complicated, but can be done. It has a higher potential for errors due to splitting and rejoining pieces of data. The heap is much more complicated to implement than the other data structure alternatives making it a poor choice under this regard.

Chosen Approach

After careful consideration of each constraint, the data structure that will be optimal for this project is the queue, shown by Table 3.

While the stack and heap were close contenders, the overall simplicity and effectiveness of the queue structure is what makes it the desirable choice. The heap would be a quicker data structure, but this is not a highly desired characteristic to consider. The queue is superior for this application over the stack because of memory capacity that is better suited for file transmission along with better failure protection.

The following chart displays how well the data structure options fit with the desired characteristics:

	Queue	Stack	Heap
Structure Capacity	5	3	5
Logical Ordering	5	5	5
Failure Protection	5	5	3
Complexity	5	5	3
Total	20	18	16

Table 3. Data Structure Analysis

Proving Feasibility

During the project's implementation and further client meetings, the effectiveness of the queue data structure will be evaluated. Currently, the queue does not need to have a priority field according to the client. To test the structure with the RCI, a test bed program will be developed to confirm that the desired characteristics are still met. After that, the structure will be implemented into the back-end of the application to ensure it properly integrates with the other technologies. These steps for testing will be used to prove that the queue data structure will be an effective solution to organizing the email queue.

Virtual Environment

The virtual environment is imperative for the communication between Operation RM’s application and the radio simulator. Selecting the best environment for this project will allow the team to ensure the ability to use all existing technologies and support for network needs.

Desired Characteristics

These are the most desired characteristics that are being considered in the selection of this environment.

- **Image support**. The client has provided an implementation of the radio simulator on a pre-existing image file. Therefore it is desired that the virtual environment can support this.
- **Compatibility with current technologies**. It is important that the virtual environment will be able to support communication between the mobile application and the radio simulator.
- **Network Support**. It is imperative to be able to connect the virtual machine to the Android app and communicate through the radio modem via wifi. This need is required to meet the needs of the client.
- **Cost effective**. In consideration of the team's financial resources the team strives to choose an environment option that meets all the requirements for the most economically feasible cost.

Alternatives

There are multiple environments that can be used to meet all of the requirements, each option presenting pros and cons.

The first option the team has investigated is VirtualBox. This environment supports a wide variety of image formats, making it versatile for importing and exporting images into a virtual machine. Furthermore, VirtualBox is compatible and works adequately with linux distributions. This virtual environment provides numerous support with various network modes and configuration allowing us to communicate with the provided radio via wifi. Another benefit of VirtualBox is that it is an open-source application and free to use which is a significant benefit for Operation RM's current funding. A possible disadvantage of VirtualBox would be its performance, it is generally reliable but does have some limitations compared to other commercial environments.

A second option that the team has researched is VMware. This machine supports a wide range of image formats, such as a VMDK format which allows files with .vmdk to be a complete and independent virtual machine. VMware also enables exceptional linux compatibility with their machine specific tools to enhance integration with linux systems. Furthermore, this machine provides advanced network support that could enable the team to connect via wifi. Consequently, this feature rich platform is for use at a high cost of 150-200 dollars per team member.

The third option explored by Operation RM was the use of Windows Subsystem for Linux (WSL). WSL does allow images to be imported into the system; however, it lacks a robust support system of formats and tools for manipulating images. This machine is a full linux environment with a wide array of linux distributions. WSL has limited networking control, it does have the capability to connect via wifi but lacks the necessary support for easy network configuration manipulation. Lastly, it is an open source application that can be done without cost by the team.

Analysis

VirtualBox is the first alternative that will be analyzed for the desired characteristics. Regarding image support, VirtualBox supports this by allowing for image importation and exportation and has image files that can be used for this. For Linux support with the radio modem simulator, it is compatible and allows the client to directly give the team an already set up image containing the simulator. Network support is another characteristic that VirtualBox has which will allow for effective testing of the app's connectivity functionality. For cost, VirtualBox is a free software that can be installed easily on personal computers.

VMWare is up next. Along with VirtualBox, it has image support making it simple for allowing each team member to work in a consistent environment. It also works well with Linux images which would allow for the simulator to be used on this platform. The network compatibility is similar to VirtualBox which would allow for the team to properly test the app being developed. The main drawback of VMWare is its cost. It does not stay within this financial bound requiring a high cost per team member that would not be economically foreseeable.

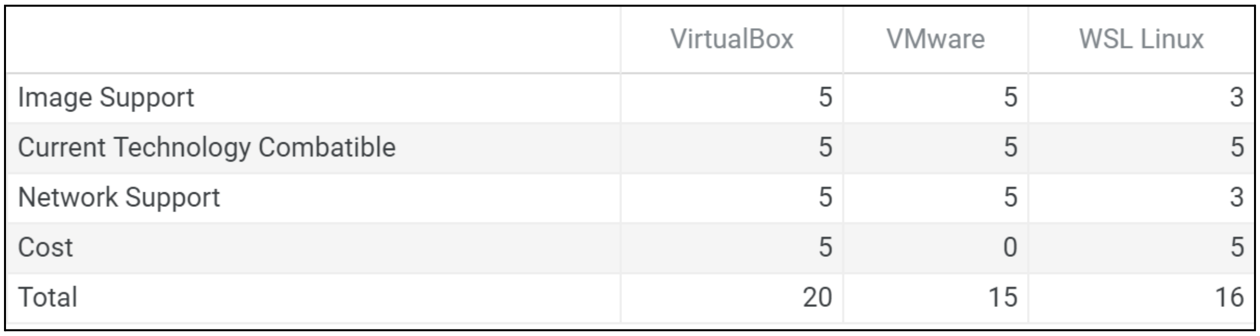
Lastly, Windows Subsystem for Linux (WSL) will be analyzed. It contains some image support, but is not nearly to the extent of the other alternatives. It is often not directly compatible with virtual machine image formats, such as VMDK or OVF. Consequently, this machine does not support the capabilities to effectively meet the team's needs for this characteristic. It satisfies the Linux compatibility as this environment is specifically a Linux subsystem. WSL does offer network support however, it has to have specific settings that require extra setup and configuration to ensure proper networking; consequently, this does not meet the full desire of the characteristic as it is not as user friendly as the previously mentioned machines. For cost, WSL is free to use and can be installed on a Windows personal computer along with the desired operating system.

Chosen Approach

After thorough research and consideration the Operation RM team decided that the VirtualBox environment would be the best option to meet the requirement needs which is shown by Table 4.

VMware and WSL were thoroughly considered, the VirtualBox environment is the most viable option. VMware has all the needed support features; however, the price of the environment is out of the team's economic resources. Meanwhile, WSL is a free environment but does not offer a rich enough support system that meets the needs of the team.

The following chart displays how well each environment met the criteria for the desired characteristic:



	VirtualBox	VMware	WSL Linux
Image Support	5	5	3
Current Technology Combatible	5	5	5
Network Support	5	5	3
Cost	5	0	5
Total	20	15	16

Table 4. Virtual Environment Analysis

Proving Feasibility

To ensure that this environment will meet all the needs of both the team and the client its image, compatibility with current technology, and network connectivity will be tested at General Dynamics. Following the test the environment will be implemented on each of the team's devices making sure that the team has a ubiquitous environment that is viable for each desired characteristic. These steps will ensure that the environment will be an efficient solution.

Technology Integration

Now that the solutions to the challenges have been selected, it is now time to show how they are going to be connected to make a complete application. The idea is that the user will have to communicate through an Android application for all of their communications. With that in mind, a programming language is needed to make the application. The language also has to be compatible with the API provided by the client. The Android application will have to talk to the radio modem and to do that, a virtual machine is required to run a simulator of the radio modem. This will then help the team be able to test the application without having the physical radio modem. The application will then get all the information the user needs though an email queue. The application will communicate with the radio modem using the General Dynamics RCI API. To develop the application, a good development software space is put in place.

System Diagram

Taking the solutions, it is important to integrate them into a comprehensive diagram used to envision how all these solutions will interact, shown by Figure 1 below. The diagram shows both the simulator of the radio modem and the Android app working in the Virtualbox environment. Eventually, the app will have to work outside the virtual environment and still communicate with the radio simulator. Then, it will show the current plan of action so the team stays on track. The diagram shows how the system will end up working as talked about above. It shows the radio modem simulator communicating with the Android application and the Android application taking the given information from the modem and using Java code to manage the email queue. The development will start inside VirtualBox to help get the app up and running. It also shows that the user can send data from the Android application to the radio modem.

The system diagram is represented below:

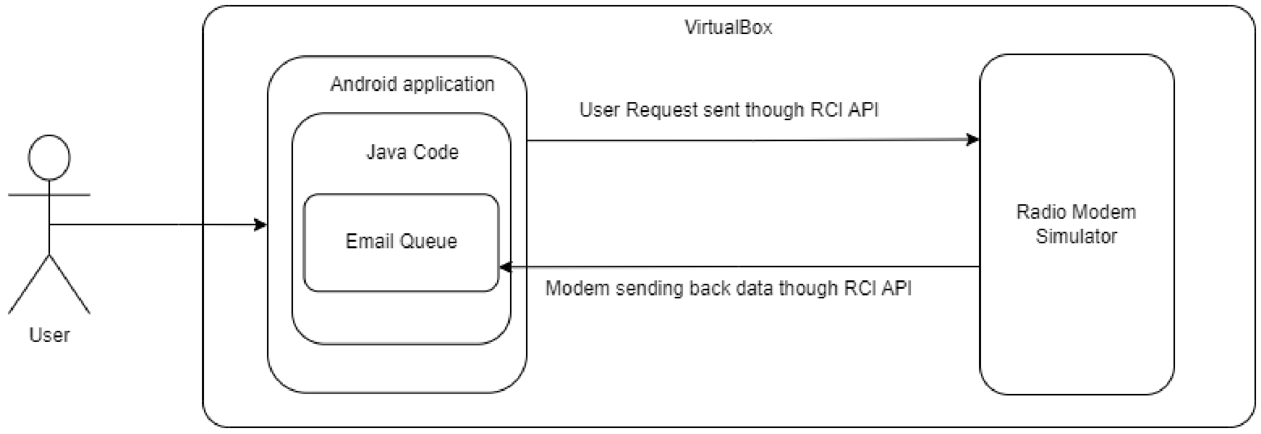


Figure 1. System Diagram

Conclusion

In conclusion, the project consists of developing software to make controlling a radio modem easier for the client General Dynamics Mission Systems. The client desires an Android application to communicate with the radio modem. There are multiple problems that have to be tackled to complete this project. The team has found a developmental system for the Android application and also solved the email queue system structure. A programming language to develop the Android application and connect to the API given by the client has been found. Along with this, the virtual environment to be used has been established. The team now has what is required to develop the project. The technologies planned for the development of the Android application will be able to get the application functioning with the radio modem simulator.

The team has figured out what technology would work for all challenges. Even if the chosen solutions are not as good as previously discussed, there are other options here that can be looked at before having to go through and find completely new ones. Android Studio was selected for the development space as it is probably the easiest to use to make an Android application. For the programming language, the team chose Java because it is the primary language for Android applications. For the email queue, the queue data structure was chosen as it will work the best overall. Finally, the virtual machine that was chosen is VirtualBox as it is free and gives the most options to use.

Overall, this project will assist General Dynamics in controlling their radio modem so they can quickly and reliably send data to the radio modem allowing for efficient, tactical communication.