InfraDrone Android Application

FINAL REPORT

Team Number 07
InfraDrone
Mitra, Simanta
Evan Snitkey, Blake Agey, Yangxiao Wang, David Schmadeke
sdmay18-07@iastate.edu
https://sdmay18-07.sd.ece.iastate.edu

Contents

1 Introduction

1.1 PROJECT STATEMENT

For this project, we are to design an Android application to display data that has been collected by a drone. The types of data to be displayed are images, reports, and 3D models. The application should also have the capability to function on mobile VR devices. The VR component of this project will be used to view the 3D models as well as to navigate through the application itself.

1.2 PURPOSE

As drones become more advanced and capable, their uses can become more engineering focused. With high-end drones having the ability to carry advanced imaging equipment we can use their mobility to get data and images from otherwise hard to reach viewpoints. Once the data is collected it would be useful to be able to analyze the data anywhere, especially on site. We hope to create an application that can be used to quickly and easily access the data collected from these drones from a mobile source.

1.3 GOALS

Since this project is based on creating a mobile application, by the end of the project we hope to become familiar in visual studio as well as app development in general. We also want to be able to deliver a fully functioning application that meets all of the requirements given to us, and we want to meet those requirements on a timely basis. By working as a team we hope to gain experience in teamwork, communication, and knowledge sharing, not only within the team but with our clients as well.

2 Deliverables

Our final deliverable should be a fully functional application that displays the data collected by a drone. The application should meet the following requirements:

- Similar display features as current InfraDrone web portal.
- Standard InfraDrone style GUI and design.
- Data should be extracted from AWS folder structure.
- User friendly 3D model viewing on application.
- Ability to view images, reports, and locations on application.
- VR visualization using Google Cardboard system.

3 Project Requirements/Specifications

3.1 FUNCTIONAL

Functional Requirements for project:

- Create an application equivalent to display features of Web-Portal
- Apply the company's GUI and design styles
- Navigate a user friendly application
- View images, reports, and locations
- Interact with user friendly 3D models
    - Display three types of 3D models
    - Ability to rotate and zoom 3D models
    - User can click on points on 3D model to view notes/annotations
- Application will extract the necessary data from Amazon Web Services folder structure
- Ability to use Google Cardboard or Daydream System for VR visualization

3.2 NON-FUNCTIONAL

- secure login and logout of user's account
- application works for current and future versions of Android devices
- Friendly user interface
- optimized loading times for application

3.3 STANDARDS

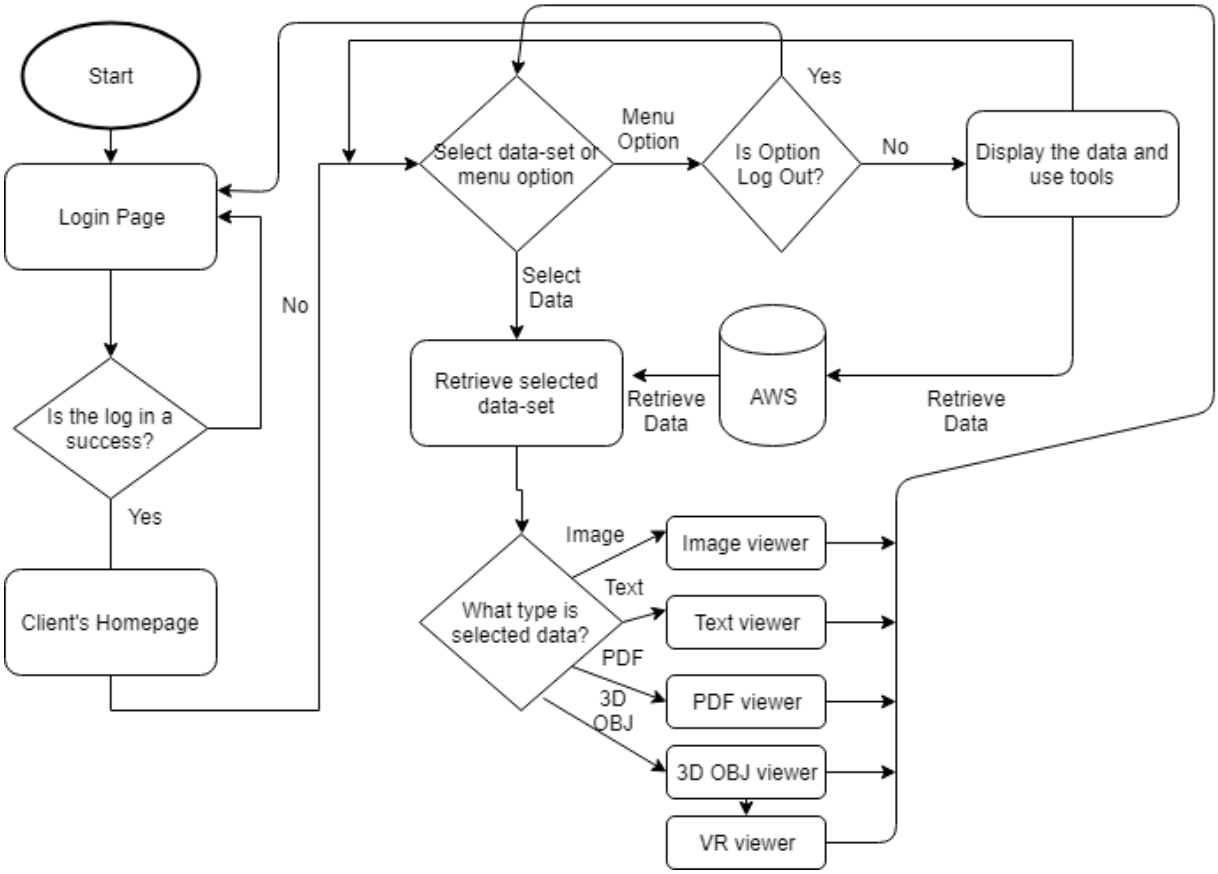
Our team plans to follow standard industry practices to reduce defects in code as well as improve overall quality. If we find any changes that need to be made to the production process, the team will meet and review the changes. We plan to use regression testing to ensure that new components do not mess with the previously working ones. We will also try our best to do code reviews of each other's work to further reduce the chance for errors. Our team will also work to adhere to the standards created by organizational standards like IEEE and ABET. So far, none of the practices we plan to use would be unethical to these organizations.

4 Design

4.1 Previous Work

As a part of SE 329 (software project management) and SE 339 (software architecture), Evan Snitkey and Blake Agey worked together in groups on two android applications. One of them was a caloric nutrient calculator application and the other was an online shopping application. Both were developed in Java using the Android Studio IDE. Yangxiao Wang also took those two courses and worked on an android shopping application and a JavaScript based online application. David Schmadeke and Yangxiao Wang enrolled in CPRE 338 (Android development) which will help in developing our base Android application.

4.2 Current Revised Project Design



The figure above is our current revised project design. It shows the different paths a user can take within the application and shows the connection to AWS for user data and security. The design is almost identical to the current existing web portal (as requested by the client), other than a few main concepts. The first and obvious one being that this design is for an Android application. The next differing concept being the VR functionality within the application. There is a button on the 3D Obj viewer activity, that when clicked, it will process the currently rendered obj image and place it into a VR view for the user without having to re-render it. The other differing concept from the web portal application is the requirement of having to download the data files when you wanted to view them since this is a requirement inside of Android. With this design, we achieved a streamlined, fluid Android application that users can view all their data from and serves the same functionalities as the online web portal with some newly added features such as VR.

4.3 Implementation Details

1. After we did some research and obtained some advice from our client. We decided to use Xamarin to develop our application for IOS and Android simultaneously. We followed a tutorial to create a simple app by using Xamarin. However, we ran into some problem starting our own app.

    a. Strengths
    Xamarin provided cross platform development based on the one language C#. That means we can just use the same IDE, language, and APIs for IOS, Android, and Windows phone apps.

    b. Weaknesses
    The most difficult part for us is the different language and platform. Xamarin is using C# and all our team member are familiar with Java instead of C#. And Xamarin does not work as we expected. We thought we can write only one app for multiple platform, however, we need to create IOS and Android app individually. Although there are some parts like UI and backend can be shared.

    c. Results
    Also we do not have the access to the Apple Developer Program, we cannot test any IOS related things. With the various problems and difficulties, we decided to create a working Android app first. As result, we are developing our app on Android Studio with Java.

2. We planned to copy the web app's UI to our mobile app as the client required. When we jumped into the UI design stage, we realized that we cannot just copy

the design from web app. The mobile devices have limited screen compared to a computer. We may redesign the UI and keep the web app's color and logo design.

a. Strengths
   It can save a lot time for us to have an existing UI design, we do not have to start over from scratch.

b. Weaknesses
   We will have many restrictions and limitations to match the web app's design.

c. Results
   After re-thinking the UI design, we decided to use a navigation bar to replace the section button on the web app.

3. The backend development was a tough task. There were many problems we faced and solved. The most important problem would be that we do not have direct access to client's AWS account. We created our own AWS account for testing. But the client does not have the same setting as we do. To solve this problem, we reached our client frequently and tried to duplicate their settings.

   a. Strengths
      Using the AWS is more convenient than deploy our own server and database.

   b. Weaknesses
      Our team does not have experience with AWS. And the AWS is already set up for client's web app. And we do not have direct access to client's AWS account.

   c. Results
      Although our clients changed the structure of their data storage during our development process, we successfully pull the data and create our own data structure for the app.

4. After numerous attempts, we realized that to parse the object file without any third-party library is not feasible. It is too difficult for a 4-person team. Therefore, we decided to use a third-party library to parse the data, and use Google Cardboard API to display the object with VR view.

   a. Strengths
      External library helped us parse the files and display it on mobile devices.

   b. Weaknesses
      We should start this part earlier with sample data instead of waiting for

completion of backend. Another problem could be the documentation of the third party's code. They might not be documented very well.

c. Results
We tried SDK Viewer 808 api, Rajawali3D api and JPCT-AE. The first two failed and JPCT-AE succeeded. It took us too much time to figure out the compatibility issue with the failed api. And JPCT's poor documentation made us struggle for a while. After all, we solved all of the problems and successfully used JPCT to display object in VR view.

4.4 Assessment and Testing of Proposed methods

Technologies that can be done in different ways in this project include 3D object viewing, displaying data from AWS S3, working with the federated identity user pool, and cross-platform development. We started developing in the Xamarin framework using C#, but we quickly realized that it would take much longer to develop in C# since we did not have nearly as much experience in C# as Java and the VR API's were only available for Java and C/C++. We decided to develop in Java via Android Studio and push cross-platform development off as a stretch goal for next semester. For 3D object viewing, we tried out a library called ModelViewer-080, however we found that we would have to rewrite most of the code to get .mtl textures to work within the 3D object files. We switched to a different library called Rajawali which is much easier to use.

4.5 Testing Process and Results

The testing process throughout the project has included both hardware testing and software testing. For hardware testing we used our own android phones to load and test the performance and functionality of the application on a dummy account. This was a very useful because it was a real simulation of the application and how it would perform. Some limitations to hardware testing was the amount and diversity of devices we were able to test on. There are a lot of phones on the market and we could only test on a select few.

We also included software testing in our project by using Android Studio's built in emulator. The emulator was very useful in testing a wide variety of devices since you could choose from a many device to emulate. Having a large testing pool was important since many phones are different sizes, images and screen swipes will behave differently on different devices.

The result of our testing processes was a overall successful since we had very practical testing solutions readily available. The wide variety of test cases through the

integration of both hardware and software testing was also very useful in testing the overall functionality of our application.

4.6 Validation

As a group, we have selected David Schmadeke to be our quality assurance expert. With this title, he will be expected to extensively test all developed code to ensure that everything we create works as it is intended. We plan on creating and attending many technical information sharing meetings during the latter half of the semester to address testing and project validation as well. To further the success in our validation process, we will be meeting with our clients weekly to demo our week's progress and note any improvements that we can make going further.

During our second semester as we started to finish the project we decided to include a survey as part of our attempt to further test and validate our application. The survey was sent to our contact at InfraDrone. We want at least 10 people to test the application. We hope that this survey will give us information about how well we designed the navigation, GUI, VR, and ability to effectively display data.

4.7 Related Products and Literature

Our application is one of many applications in the market. More specifically our application is a data visualization application that should present data in a fast and easy form for the user to analyze. Our application is not however intended for everyone. This app is intended specifically for employees and clients of InfraDrone, so it is unique in that way.

5 Challenges

From the discussion with our client, we found that we can potentially face three challenges/risks that may slow or hinder your plan.

- Extract data from AWS (Amazon Web Service)
- 3D Obj viewer
    - SDK Viewer 808 api (could not parse long values)
    - Rajawali3D api (failed to parse client Objs)
    - 3D Objs not rendering in JPCT-AE rendering view
- Lighting and shadow issues within JPCT-AE Obj renderer
- Rendering large Objs on mobile takes 30+ seconds

- Texture rendering issues
    - JPCT-AE api required bitmap image types to be applied as textures to 3D Objs
- Google Cardboard and JPCT-AE are different api's and datasets (transitioning the data from JPCT-AE to Google Cardboard)
    - Head tracking within VR using the JCPT-AE data

Extracting data from AWS was tricky because we had no experience working with the structure. This was also difficult for us because the client's AWS structure continually changed throughout the past year. We started 3D Obj viewing development with an api we found called SDK Viewer 808. This viewer could not parse in long values from Obj files, so we had to scrap work on this api and find a new one. The next api we found was called Rajawali3D - this one we could not get to work or even find out why it wouldn't work. Finally, we found an Android mobile gaming api called JPCT-AE. This api actually could render the Obj files, however it took between thirty and sixty seconds and at first the Obj was not in the view of the camera.

Our client's Obj files are complex and large (20+ megabytes). Each of the objects are different physical sizes and have vastly different central points. This means that after rendering an object, they could be at different locations and distances from the camera. We created a basic algorithm to centralize and scale the objects and place them right in front of the camera. JPCT-AE includes a way to serialize an Obj file to make it load faster on Android, however it would have to be applied to every Obj file on the client's server - which at this time is out of scope and unreasonable. After rendering an object, the object would be dark and have weird shadows. We were able to fix this by testing different lighting amounts to find the perfect one and setting the location of the light source right on top of the camera - this would eliminate all shadows.

Once we had Objs successfully rendering within view in the JPCT-AE api, the textures wouldn't render. We struggled with this for a long time because no errors would be thrown, and the program would continue like nothing was wrong. After doing some in depth nested debugging, we came to the realization that the api only allowed bitmap file types for rendering. This was annoying because it took so long to fix this small issue - we just converted our jpeg texture files to bitmap at runtime before adding them to the objs. This could have been easily fixed if the api had better documentation or if an error message would have popped up saying why the texture was not added.

Because our application required VR functionality, we decided to go with Google Cardboard as our VR api. The tricky part about this was that we needed to be able to take the rendered obj from JPCT-AE and display it in VR. The two apis had different data structures, so combining the two included some data manipulation and translation. The biggest issue with this was the VR head tracking because we had to use the Google Cardboard data to tell JPCT-AE how much to rotate the object and in what direction. The last big issue we had involved how long it took to render the objects. We wanted a

seamless design so that one button click would redirect the object from non-VR (JPCT-AE) to a Google Cardboard VR view, but we didn't want to wait another 30+ seconds for the object to render again just for VR. We were able to pass the world object from the non-VR view to the VR view, so it would only take less than 3 seconds to load the post rendered obj in VR.

6 Conclusions

The main goal of this project is to be able to deliver a functioning application that can easily display images and 3D models, collected from a drone and stored on AWS. The final design of the application should have the ability to view images and models in a VR environment. Our client is InfraDrone (renamed to InfraLytiks). We met with our client on a regular basis to ensure we are on the right track as well as to express any concerns or difficulties with the project. We had a lot challenges, although, we positively contacted or client and solved the problems.

Overall, we successfully implemented most of the desired features. The UI and navigation of the android application follow the current design of InfraDrone's website. The data viewing in regular mode(non-VR) works fine. The VR part is the only remaining issue. The whole VR function works fine, however, the object in VR view cannot rotate correctly when user look up and down while there is no problem with left and right rotation. Other than the VR issue, we are confident to say that our project is successful. We learned a lot from this project. All the challenges and problems we encountered and overcame became our valuable experiences. Those lessons could help us a lot in our future career path.

7 References

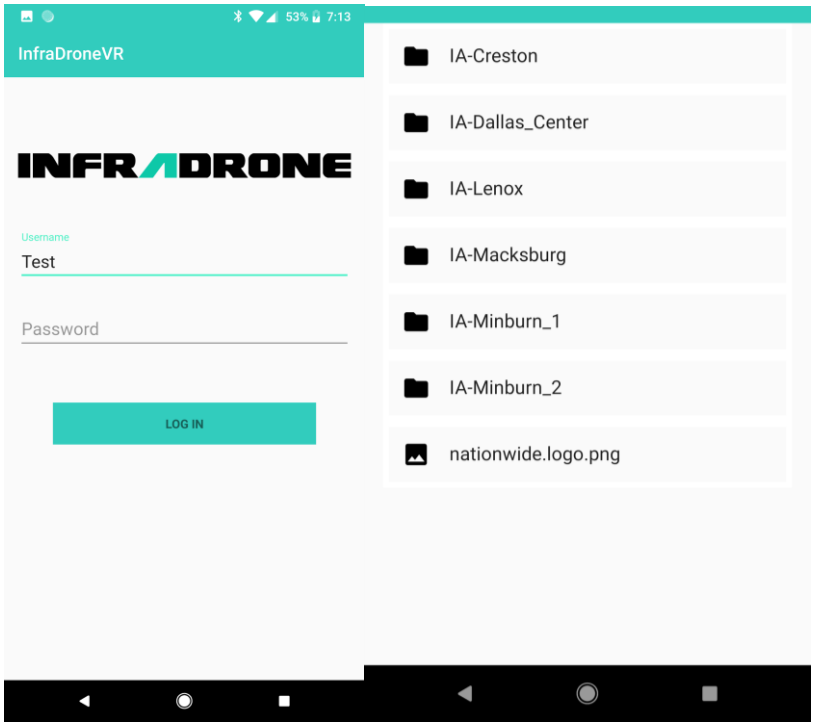
1. Vidyadharan, Akash. “InfraDrone VR Phone Application.” Iowa State University. Ames, IA.
2. InfraDrone. “InfraDrone Web Portal.” *Login*, ec2-34-206-210-136.compute-1.amazonaws.com/Active/index.php.
3. “Rajawali.” *GitHub*, github.com/Rajawa
4. Google. “Google VR.” *Google VR*, Google, vr.google.com/.
5. “Amazon Web Services (AWS) - Cloud Computing Services.” *Amazon Web Services, Inc.*, aws.amazon.com/.
6. “jPCT-AE.” jPCT 3D engine http://www.jpct.net/jpct-ae/

8 Appendix I - Operation Manual

This application is designed for the InfraDrone and their clients. This appendix will cover step by step instruction on going through the application. We have split the manual up into three components so that it is easier to understand and follow. These components are the login sequence and navigation, viewing pdfs and images, and finally viewing 3d objects using VR and non-VR means.

8.1 Logging In and Navigating Application File Structure

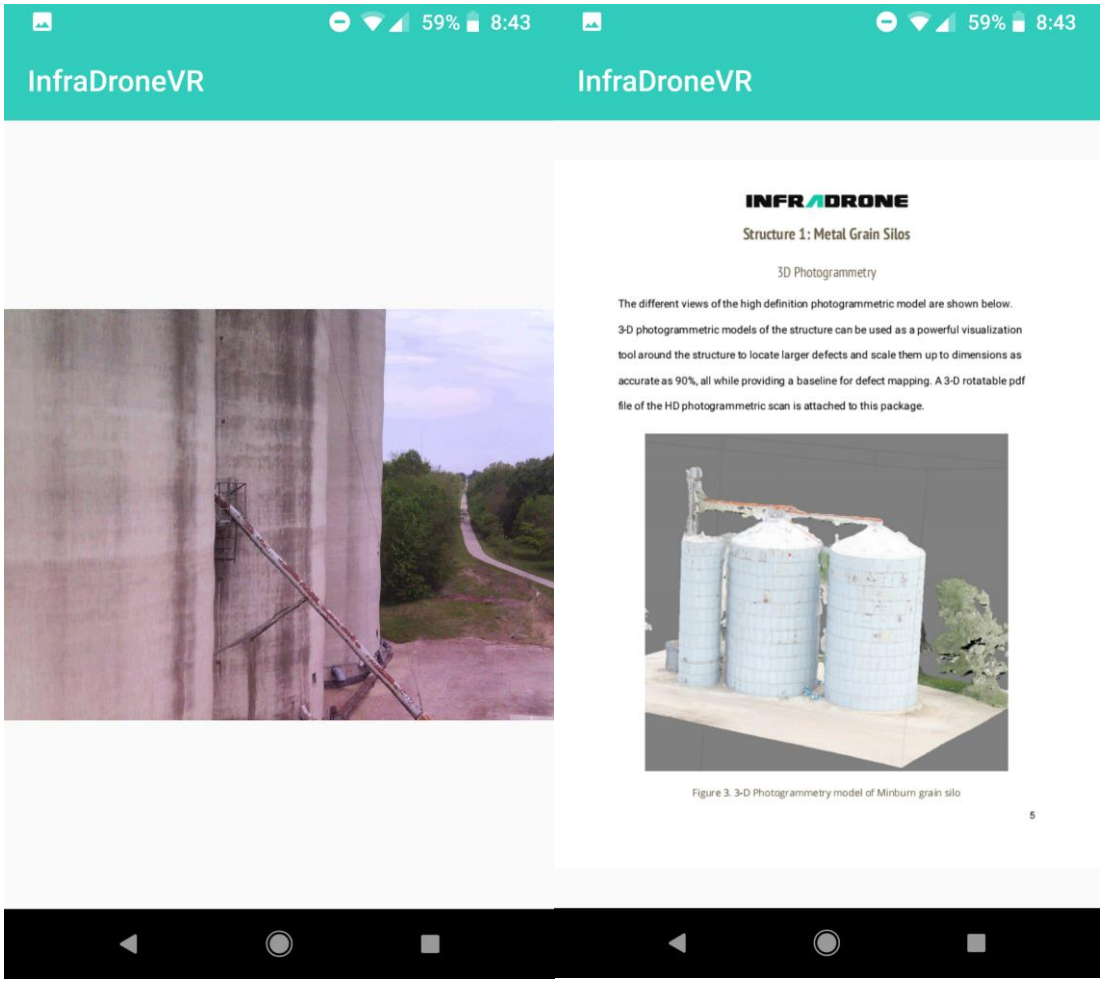
The login screen will appear the first-time users open the application. User can also choose to save username and password. When login successful, the folder view will appear. Different file types would have different icons. User can choose to go to certain location, street address, date that associate with the data.



8.2 Viewing Images and PDFs

After navigating to an image, clicking on the file will download the image, once downloaded you can then view the image in portrait mode. Once a file is downloaded you can go back to the file and view it again without downloading it again. When you click on an already downloaded file you will be prompted, just click open when asked. Each image should be sized to fit on the screen, so no adjusting is required. There are no pinch to zoom features included in the image viewing. Once you have finished viewing the image you can hit the back button to return to the previous folder and then navigate to any other part of the application.

Another data type that is available are PDFs. To view a PDF, navigate to a PDF file then click on it to download it. Once downloaded you can view the PDF in portrait mode. To navigate through the PDF, you can simply swipe left or right to go to the next or previous page. The PDF viewer has no pinch to zoom features. Below are examples of what the Image and PDF viewers look like.



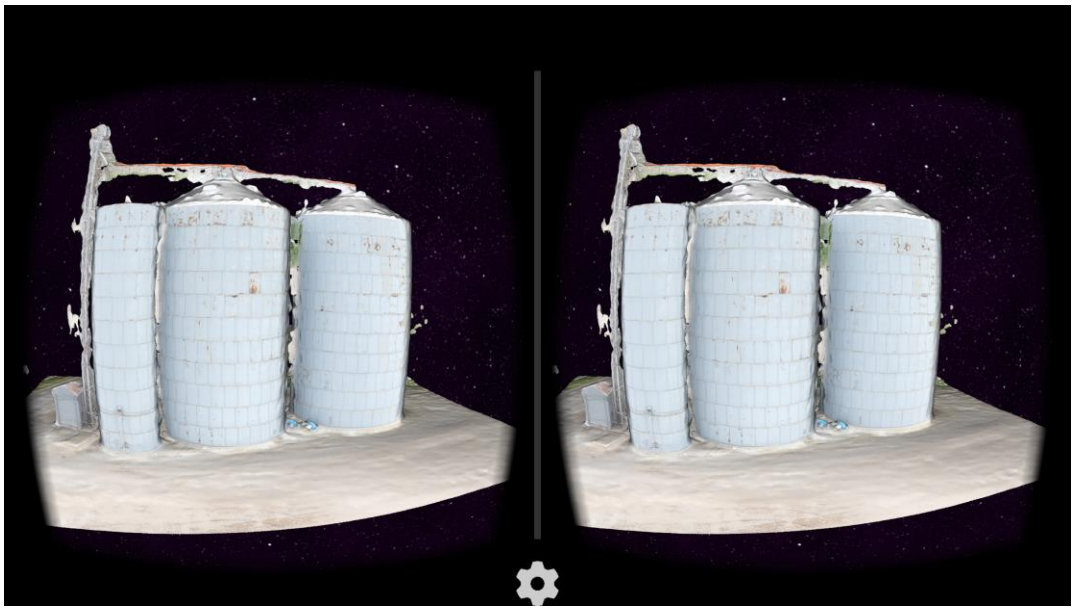
8.3 Viewing Obj Files

Upon clicking of an object file (.obj), if the object has never been rendered before, then the object is downloaded from the AWS database and then parsed and given textures. After about 30 seconds to a minute the object file is rendered. The user is then able to rotate the object with touch movement controls. If the user wants to see a VR version of the object, they can press the button with the icon of an eye on it in the bottom left corner.

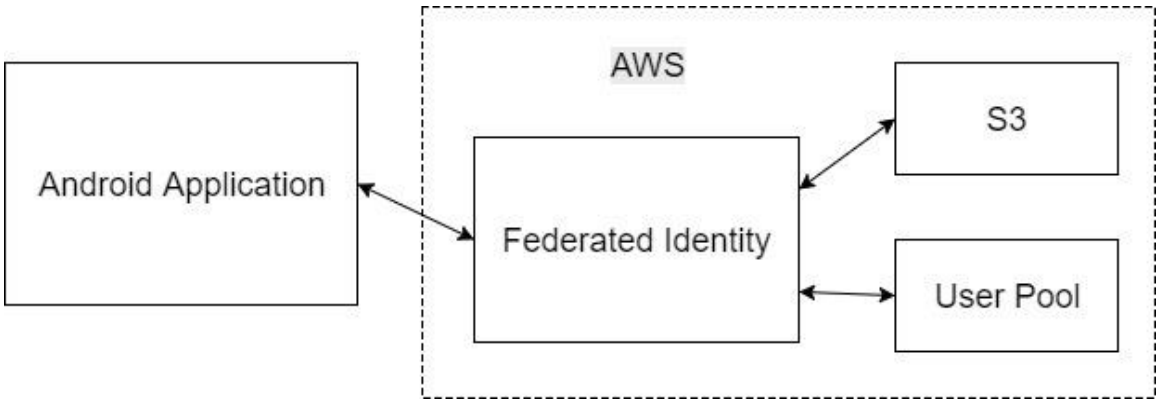
Note: If the object has already been rendered once before, a pop-up message will appear and ask if the user would like a new version downloaded or if they wanted to render the previously downloaded version of the object file.

After clicking the VR button, the current Android activity will switch from the non-VR Obj viewer to the VR viewer. This switch will take approximately three to five

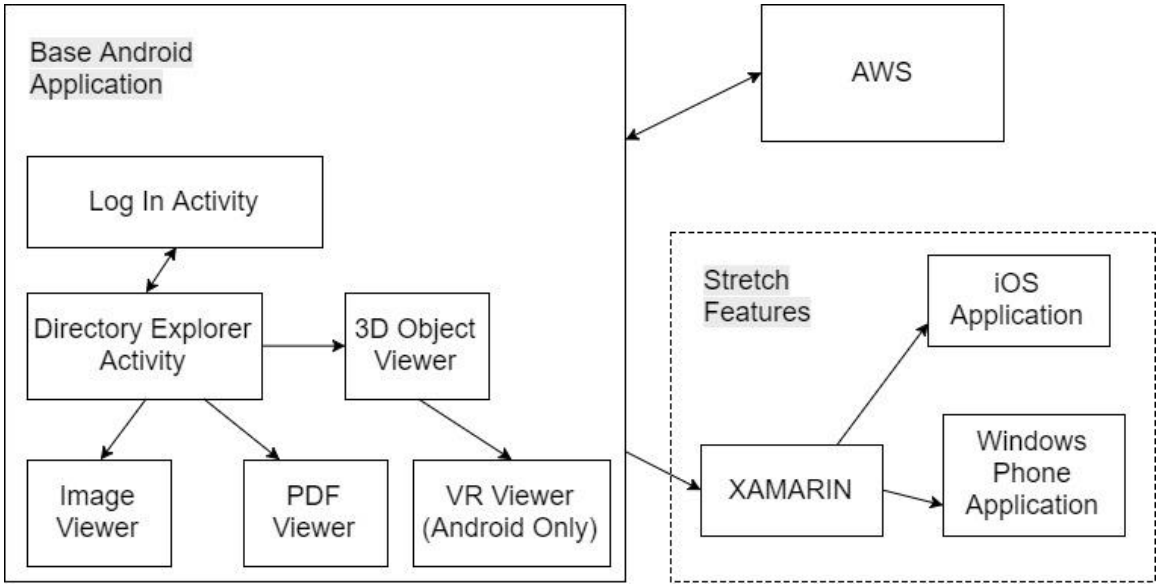
seconds. During this time, it is recommended to set up your Google Cardboard device. You can do this by un-attaching the Velcro strap on the top of the Google Cardboard device, sliding your VR compatible Android device into the view port, and finally, reattaching the Velcro strap. As you move your head to the left and right, the object will rotate in real time. To go back to the file structure, just press the back button.



9 Appendix II - Initial Design Versions



The figure above was the first initial design for our system block diagram. It has stayed the same throughout the duration of the project development time because the structure was given to us by our client. The Amazon Web Server platform and its components were already set up for the company's already existed web portal application. Because our task was to create an identical copy of their web portal application on Android, the design outside of the Android application could not be changed.



The image above is our initial proposed system event diagram. It is comprised of two key components - our Android application and AWS. AWS contains a federated identity component which is used to authorize user pools. The user pool is a group of users that are associated with our application. They will be using these services to log in and once logged in, they will have access to their data stored in AWS S3. All the client data will be viewable for the client within the application.

The only difference between the current system event diagram and the initial one is the platform availability (as noted above by 'stretch features'). After first semester concluded, our client said that it would be unnecessary to create a whole new application from scratch for iPhone or Windows Phone because rarely any of our client's employees use phones other than Android. Due to this, we dropped the stretch features and focused all our work into one main application for Android.

The proposed event diagram is like the block diagram; however, it includes more in-depth information regarding the Android application and stretch features. The Android application design structure is very similar to the current web portal. The user will start by logging into the application when it starts up. After this, they will be redirected to their directory explorer which will display all the user's data in AWS S3. After navigating to an image or PDF, they will have the option to view them in a larger window using the viewer feature. They can also view 3D objects via the 3D object viewer. The 3D object viewer will include a VR button which will display the 3D object in VR. Our stretch feature is cross-platform development; this will be explored more during semester two when we have the Android application completed.