Department of Computer Science & Engineering
The University of Texas at Arlington

Detailed Design Specification
CSE 4317: Senior Design II
Spring 2023

Team Good Neighbors
State Farm Disaster Survery Drone

Faith Gutierrez
Mario Villatoro
Asim Regmi
Carlos Sanchez
Danielle Pham
Pratik Dhakal

Revision History

Revision	Date	Author(s)	Description
0.1	2.23.2023	FG	document creation
0.1	2.24.2023	AR, CS, FG, DP, PD, MV	First draft

CONTENTS

1 Introduction **6**

2 System Overview **6**

3 Cloud Function Layer Subsystems **7**
- 3.1 Layer Hardware . . . 7
- 3.2 Layer Operating System . . . 7
- 3.3 Layer Software Dependencies . . . 7
- 3.4 Subsystem Cloud Function VM Start . . . 7

4 Raspberry Pi Layer Subsystems **9**
- 4.1 Layer Hardware . . . 9
- 4.2 Layer Operating System . . . 9
- 4.3 Layer Software Dependencies . . . 9
- 4.4 Subsystem Upload images to cloud bucket . . . 9
- 4.5 Subsystem Trigger Cloud Function . . . 10

5 Object Storage Layer Subsystems **11**
- 5.1 Layer Hardware . . . 11
- 5.2 Layer Operating System . . . 11
- 5.3 Layer Software Dependencies . . . 11
- 5.4 Subsystem Object Storage . . . 11

6 Photogrammetry VM Layer Subsystems **12**
- 6.1 Layer Hardware . . . 12
- 6.2 Layer Operating System . . . 12
- 6.3 Layer Software Dependencies . . . 12
- 6.4 Subsystem VM Start . . . 12
- 6.5 Subsystem GPU download images . . . 13
- 6.6 Subsystem Start Photogrammetry Script . . . 13
- 6.7 Subsystem Send .fbx to Bucket . . . 14
- 6.8 Subsystem Shutdown VM . . . 14

7 Game Functions Layer Subsystems **15**
- 7.1 Layer Hardware . . . 15
- 7.2 Layer Operating System . . . 15
- 7.3 Layer Software Dependencies . . . 15
- 7.4 Subsystem . . . 15

8 Game Engine Layer Subsystems **17**
- 8.1 Layer Hardware . . . 17
- 8.2 Layer Operating System . . . 17
- 8.3 Layer Software Dependencies . . . 17
- 8.4 Subsystem . . . 17

9 React Web Application **19**
9.1 Layer Hardware 19
9.2 Layer Operating System 19
9.3 Layer Software Dependencies 19

10 MongoDB Layer Subsystems **20**
10.1 Layer Hardware 20
10.2 Layer Operating System 20
10.3 Layer Software Dependencies 20
10.4 Login Database 20
10.5 Claim Database 21

11 Appendix A **22**

List of Figures

1 System architecture 7
2 Cloud Function VM Start subsystem description diagram 8
3 Layer Raspberry Pi subsystem diagram 9
4 Layer Raspberry Pi subsystem diagram 10
5 Example subsystem description diagram 11
6 Subsystem Photogrammetry VM Description Diagram 12
7 Example subsystem description diagram 15
8 Example Game Engine description diagram 17
9 Example subsystem description diagram 19
10 Example subsystem description diagram 20

List of Tables

1 Introduction

In certain disaster scenarios sending a person to check on an insurance claim might be impractical or infeasible depending on the where damage is and how much damage occurred. The purpose of this project is to help State Farm develop and test the idea of creating something where a State Farm insurance Specialist or State Farm Agent would not have to be physically at the insurance claim site to validate the extent of the damage and file a claim. We going are going to be working with a team of electrical engineers who will build a drone that can fly around the disaster site and take photos of the site. Then we (the computer science engineering team) will develop an application that can use photogrammetry software to create a three-dimensional model and render that model into a virtual reality environment. Then, once in the virtual environment, the insurance agent will be able to walk around and take pictures of certain damage points as if they were at the insurance claim site in person. This will then be uploaded to a website that will automate the process of creating the insurance claim document.

2 System Overview

The Cloud Function layer is responsible for starting the VM instance by using the Google Cloud SDK. This layers considers that the Raspberry Pi makes a successful HTTPS POST request to the cloud function to trigger it since the cloud function isnât responsible for starting the VM instance if itâs not triggered The Raspberry Pi subsection involves the EE team sending the images to the cloud. In the Object storage layer, all the image files, 3D object file, claim documents, and game files for the VR system is stored. This layer only has one subsystem. This system also gets the communication from various other subsystem in different layers to access the files. The Photogrammetry VM Layer is where the photogrammetry process will take place. This VM will be hosted on google cloud. This layer will automatically take the pictures from the object storage when they are uploaded from the drone and turn those photos into a 3D .fbx file which will be used in the VR environment. The Game Functions Layer includes the functions that will be integrated into the game engine. This Layer will load the environment from the object storage, and will allow the user to interact with the virtual environment, examine and understand the damaged item or area in detail. The game functions includes subsystems: place pin, take screenshot and hone in on area, move around, and view map. The Game Engine Layer includes populate with images function that allows for the user to interact with the object so that they can retrieve more information regarding certain surface of the object. It also includes function that gives the user more information regarding the area that the object is located in using the map feature. The React Web Application layer is where the user will do all the work. This layer includes logging into the application giving the user access to claims in the database. This layer also allows users to view a list of claims that can be chosen to view in the virtual environment. The user will be able to create a PDF to view all the information about the claim that was chosen. The MongoDB Database layer includes the functions that will be integrated into the react website. This will allow the Agents to log in to the website and view the features only available to logged in users. This also includes the claims database which is how claims will be added to user accounts to to disperse claims between agents and also be able to see which agents worked on what claims.

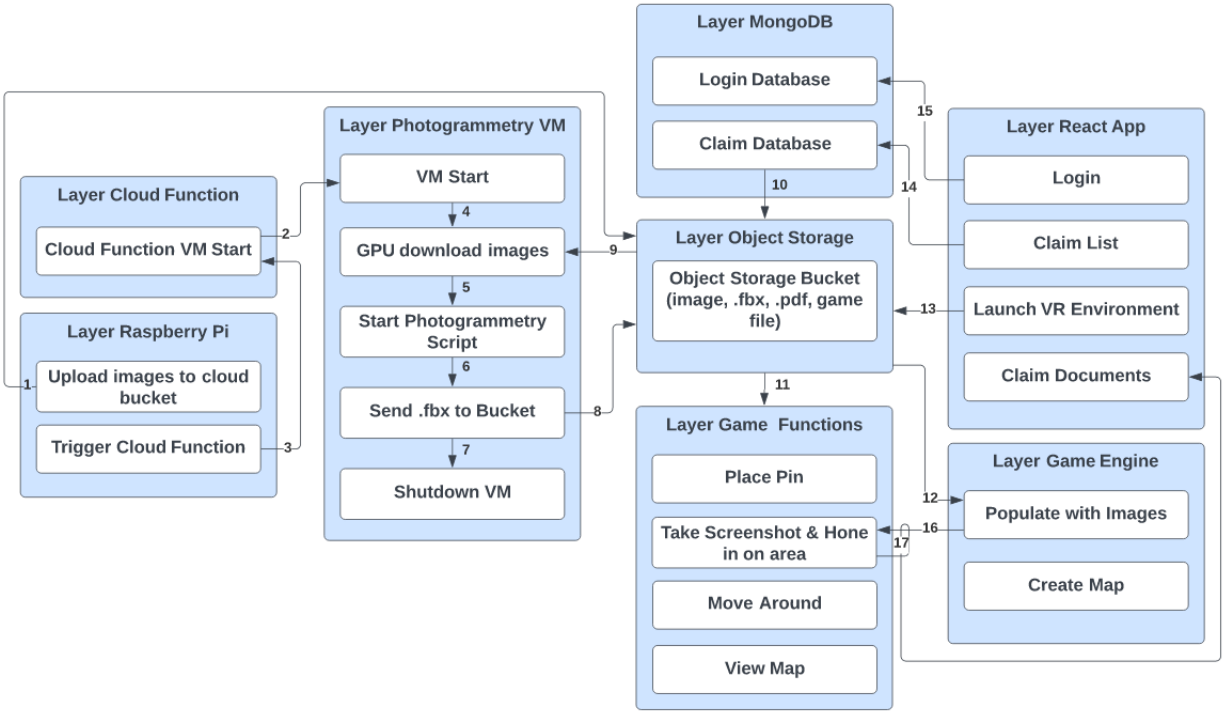

Figure 1: System architecture

3 Cloud Function Layer Subsystems

The Cloud Function Layer consists of a python function hosted on the cloud that starts the VM.

3.1 Layer Hardware

N/A

3.2 Layer Operating System

N/A

3.3 Layer Software Dependencies

The layer requires Python 3.0 or higher.

3.4 Subsystem Cloud Function VM Start

The Cloud Function VM Start is a python function that is hosted on the Google Cloud. It can be called by making a HTTP request to the the function URL. It starts the VM which then initiates the photogrammetry process.

3.4.1 Subsystem Hardware

N/A

3.4.2 Subsystem Operating System

N/A

3.4.3 Subsystem Software Dependencies

The subsystem requires Python 3.0 or higher.

Figure 2: Cloud Function VM Start subsystem description diagram

3.4.4 Subsystem Programming Languages

The subsystem used Python 3.0 or higher.

3.4.5 Subsystem Data Structures

N/A

3.4.6 Subsystem Data Processing

N/A

4 Raspberry Pi Layer Subsystems

This layer describes the subsystems for the Raspberry Pi layer and how it connects to the Google Cloud services.

4.1 Layer Hardware

This layer used a Raspberry Pi model 4B with 4GB of RAM.

4.2 Layer Operating System

The OS in the Raspberry Pi will be Raspberry Pi OS.

4.3 Layer Software Dependencies

This layer requires Python 3.0 or higher as well as the google-cloud-storage python library to connect to google cloud.

4.4 Subsystem Upload images to cloud bucket

This subsystem uploads the images from Raspberry Pi to the cloud storage

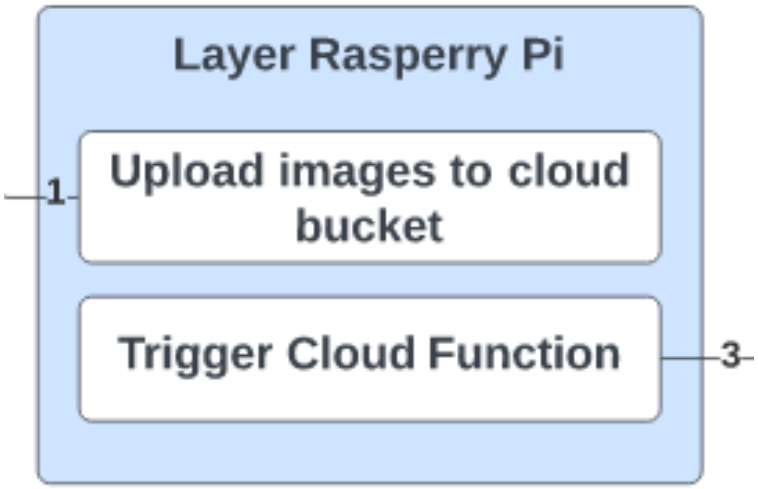

Figure 3: Layer Raspberry Pi subsystem diagram

4.4.1 Subsystem Hardware

The subsystem involves a Raspberry Pi Model 4B.

4.4.2 Subsystem Operating System

The subsystem requires Raspberry Pi OS.

4.4.3 Subsystem Software Dependencies

It requires Python 3.0 or higher as well as the google-cloud-storage python library to connect to google cloud.

4.4.4 Subsystem Programming Languages

The subsystem used Python programming language.

4.4.5 Subsystem Data Structures

N/A

4.4.6 Subsystem Data Processing

N/A

4.5 Subsystem Trigger Cloud Function

This subsystem makes an HTTP POST request to the cloud function.

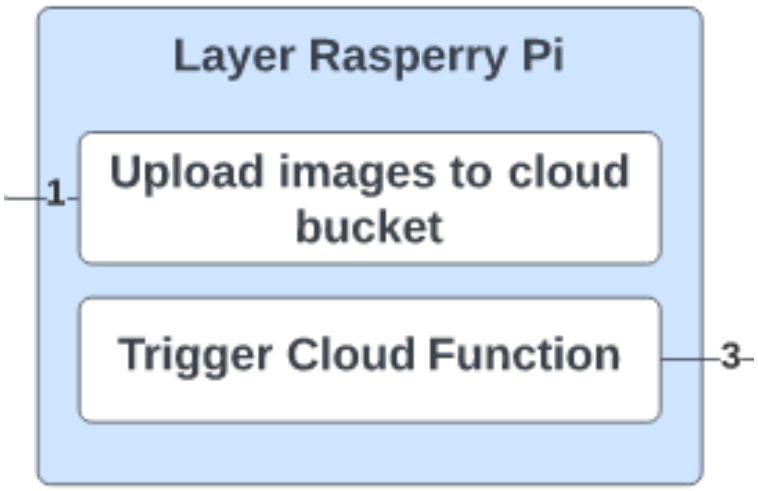

Figure 4: Layer Raspberry Pi subsystem diagram

4.5.1 Subsystem Hardware

The subsystem involves a Raspberry Pi Model 4B.

4.5.2 Subsystem Operating System

The subsystem requires Raspberry Pi OS.

4.5.3 Subsystem Software Dependencies

It requires Python 3.0 or higher as well as the google-cloud-storage python library to connect to google cloud.

4.5.4 Subsystem Programming Languages

The subsystem used Python programming language.

4.5.5 Subsystem Data Structures

N/A

4.5.6 Subsystem Data Processing

The response will be parsed into JSON after making the HTTPS POST request.

5 Object Storage Layer Subsystems

The layer consists of object storage bucket that stores the image and object files.

5.1 Layer Hardware

N/A

5.2 Layer Operating System

N/A

5.3 Layer Software Dependencies

N/A

5.4 Subsystem Object Storage

The subsystem consists of object storage bucket that stores the image and object files.

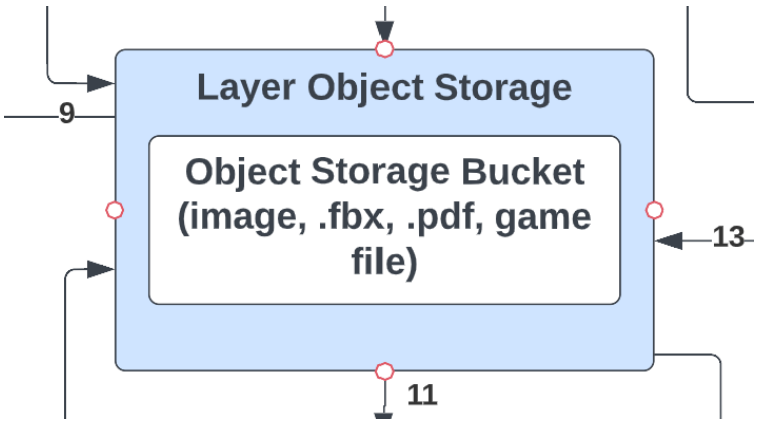

Figure 5: Example subsystem description diagram

5.4.1 Subsystem Hardware

N/A

5.4.2 Subsystem Operating System

N/A

5.4.3 Subsystem Software Dependencies

N/A

5.4.4 Subsystem Programming Languages

N/A

5.4.5 Subsystem Data Structures

N/A

5.4.6 Subsystem Data Processing

N/A

6 PHOTOGRAMMETRY VM LAYER SUBSYSTEMS

In this section, the layer is described in terms of the hardware and software design. Specific implementation details, such as hardware components, programming languages, software dependencies, operating systems, etc. should be discussed. Any unnecessary items can be ommitted (for example, a pure software module without any specific hardware should not include a hardware subsection). The organization, titles, and content of the sections below can be modified as necessary for the project.

6.1 LAYER HARDWARE

The Photogrammetry has a 32-core processor with Nvidia T4 graphics.

6.2 LAYER OPERATING SYSTEM

The Photogrammetry VM uses Windows as its operating system.

6.3 LAYER SOFTWARE DEPENDENCIES

N/A.

6.4 SUBSYSTEM VM START

This subsystem starts the VM instance. It gets triggered by the Cloud Function VM Start subsystem from the Cloud Function layer.

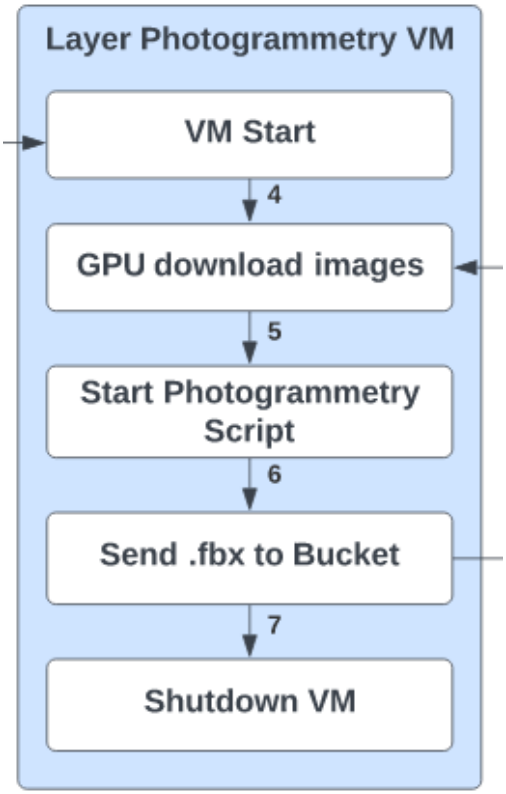

Figure 6: Subsystem Photogrammetry VM Description Diagram

6.4.1 SUBSYSTEM HARDWARE

This section boots up the VM.

6.4.2 Subsystem Operating System

This subsystem uses Windows.

6.4.3 Subsystem Software Dependencies

N/A.

6.4.4 Subsystem Programming Languages

Python.

6.4.5 Subsystem Data Structures

N/A

6.4.6 Subsystem Data Processing

N/A

6.5 Subsystem GPU download images

This subsystem simply downloads the images from the object storage bucket. It gets triggered once the VM has started.

6.5.1 Subsystem Hardware

The Nvidia T4 GPU downloads the images.

6.5.2 Subsystem Operating System

This subsystem uses Windows.

6.5.3 Subsystem Software Dependencies

N/A.

6.5.4 Subsystem Programming Languages

Python.

6.5.5 Subsystem Data Structures

The images are accessed as blobs in the google cloud storage and are downloaded into specific folders in the VM. The images in this case are unstructured data.

6.5.6 Subsystem Data Processing

The images are downloaded without any sorting algorithm with a simple loop that downloads each images with each iteration.

6.6 Subsystem Start Photogrammetry Script

This subsystem runs the photogrammetry script. It gets triggered once the GPU download images subsystem is done.

6.6.1 Subsystem Hardware

This section boots up the VM.

6.6.2 Subsystem Operating System

This subsystem uses Windows.

6.6.3 Subsystem Software Dependencies

This subsystem uses the photogrammetry software Reality Capture and CUDA 11 library.

6.6.4 Subsystem Programming Languages

Python.

6.6.5 Subsystem Data Structures

N/A

6.6.6 Subsystem Data Processing

N/A

6.7 Subsystem Send .fbx to Bucket

This subsystem sends the .fbx file to the object storage bucket. It also triggers the Shutdown VM subsystem.

6.7.1 Subsystem Hardware

N/A.

6.7.2 Subsystem Operating System

This subsystem uses Windows.

6.7.3 Subsystem Software Dependencies

N/A.

6.7.4 Subsystem Programming Languages

Python.

6.7.5 Subsystem Data Structures

The object file is uploaded as blob to the storage bucket.

6.7.6 Subsystem Data Processing

N/A

6.8 Subsystem Shutdown VM

This subsystem shuts down the VM. It is triggered by the .fbx file being send to the object storage bucket.

6.8.1 Subsystem Hardware

This section shuts off the VM.

6.8.2 Subsystem Operating System

This subsystem uses Windows.

6.8.3 Subsystem Software Dependencies

N/A.

6.8.4 Subsystem Programming Languages

Python.

6.8.5 Subsystem Data Structures

N/A

6.8.6 Subsystem Data Processing

N/A

7 Game Functions Layer Subsystems

the Game Function Layer is described in terms of the hardware and software specifications. Meeting these specifications will ensure that the virtual environment runs smoothly and provides a high-quality experience for State Farm agents to examine claims in a case. To create this virtual environment, the project is utilizing Unreal Engine 5, which is a powerful game engine that can be used to develop a wide range of interactive experiences. The end result will be an executable file that will be used to run the virtual environment on a compatible computer.

7.1 Layer Hardware

This layer requires DirectX 11 or 12 compatible graphics card.

7.2 Layer Operating System

This layer requires Windows 10 64-bit version or higher.

7.3 Layer Software Dependencies

This layer requires Unreal Engine 5.

7.4 Subsystem

The Game Functions Layer includes the functions that will be integrated into the game engine. This Layer will load the environment from the object storage, and will allow the user to interact with the virtual environment, examine and understand the damaged item or area in detail. The game functions includes: place pin, take screenshot and hone in on area, move around, and view map.

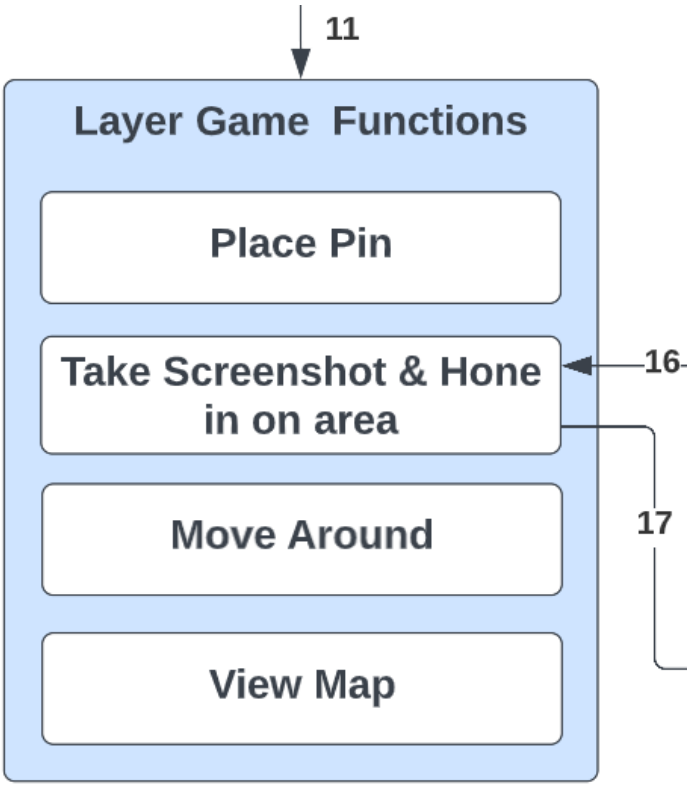

Figure 7: Example subsystem description diagram

7.4.1 Subsystem Hardware

A minimum of s 2.5 GHz quad-core Intel or AMD, 8 GB RAM Memory, and DirectX 11 or 12 graphics card is required for the subsystem. We are running Intel i7, 16 GB RAM, and RTX 2080.

7.4.2 Subsystem Operating System

Windows 10 64-bit version or higher is required by the subsystem.

7.4.3 Subsystem Software Dependencies

Unreal Engine 5 is required by the subsystem which installs libraries from DirectX End-User Runtime.

7.4.4 Subsystem Programming Languages

N/A

7.4.5 Subsystem Data Structures

N/A

7.4.6 Subsystem Data Processing

N/A

8 Game Engine Layer Subsystems

The purpose of this subsystem is to allow the State Farm agent to navigate the insurance claim area in a virtual environment setting. This allows the agent to garner a clear view of the claim area rather than just viewing the area with simple photographs

8.1 Layer Hardware

The Unreal Engine environment is running in a Intel i7, 16GB RAM, RTX 2080 graphics card

8.2 Layer Operating System

The game engine requires a Windows 10 x64 bits Operating System or higher

8.3 Layer Software Dependencies

Built in Unreal Engine libraries will be utilized. Various other scripts will be written as necessary.

8.4 Subsystem

The game engine will be populated with a 3D image, as well as various pictures that correlate to the real world position of a given picture. The virtual environment will map the insurance claim area it was based on.

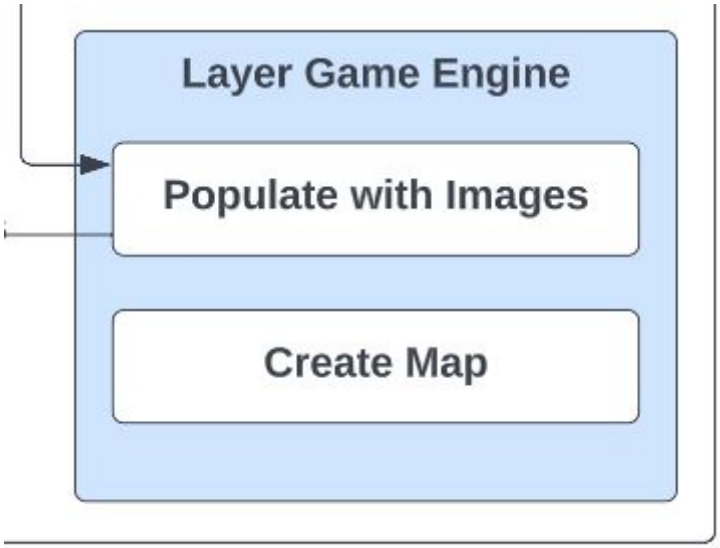

Figure 8: Example Game Engine description diagram

8.4.1 Subsystem Hardware

An Oculus headset will be required to access the Virtual Environment in Unreal Engine

8.4.2 Subsystem Operating System

The subsystem requires windows 10 x64 Operating System

8.4.3 Subsystem Software Dependencies

The subsystem will utilize built in libraries in Unereal Engine. Any other necessary libraries will be written by the team

8.4.4 Subsystem Programming Languages

The programming languages utilized for this subsystem will be C++

8.4.5 Subsystem Data Structures

N/A

8.4.6 Subsystem Data Processing

N/A

9 React Web Application

The react application is how we connect the user with the entire application. It was created with javascript and the react framework. Users will be able to see valuable information from web app and create the document that is the final step in the process of the entire project.

The React Web Application can be accessed using any computer, as long as the user has login credentials. This application was built using React and Node JS.

9.1 Layer Hardware

The react app does not have any hardware, but does use Node JS as the backend server to run the web app.

9.2 Layer Operating System

The react application can be accessed with any operating system.

9.3 Layer Software Dependencies

The react app uses Mongo DB, Express JS, React JS, Node JS, and Google Cloud Storage.

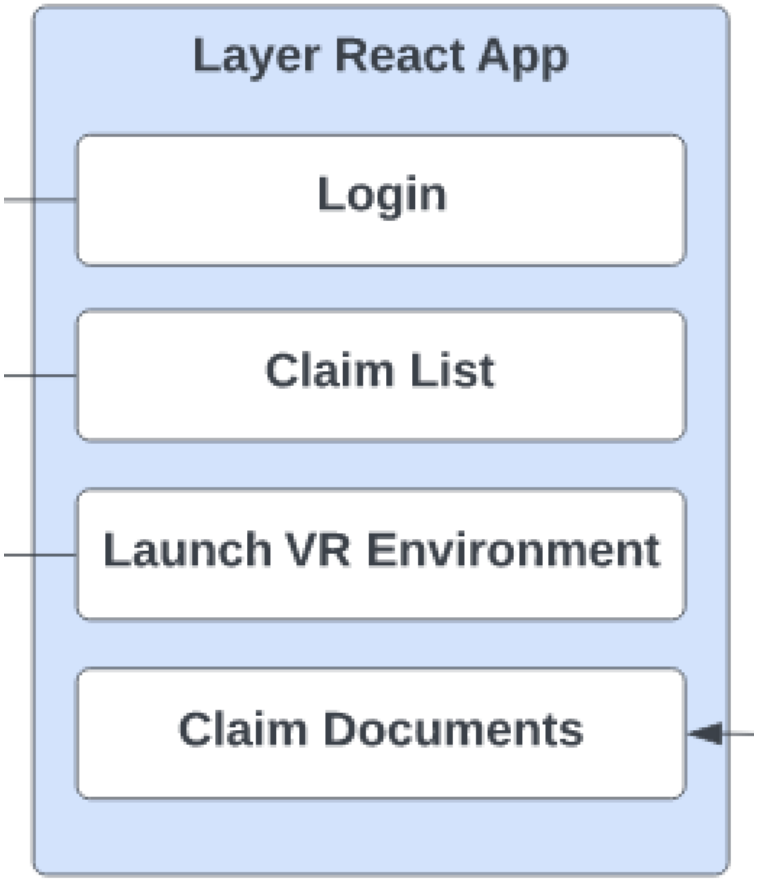

Figure 9: Example subsystem description diagram

10 MongoDB Layer Subsystems

Mongo DB is the database used for this system. User information and insurance claims are stored here.

10.1 Layer Hardware

The Mongo DB does not have any hardware.

10.2 Layer Operating System

No specific operating system is required to use the Mongo Database.

10.3 Layer Software Dependencies

Our database is hosted in Google Cloud Platform.

10.4 Login Database

Descibe at a high level the purpose and basic design of this subsystem. Is it a piece of hardware, a class, a web service, or something else? Note that each of the subsystem items below are meant to be specific to that subsystem and not a repeat of anything discussed above for the overall layer.

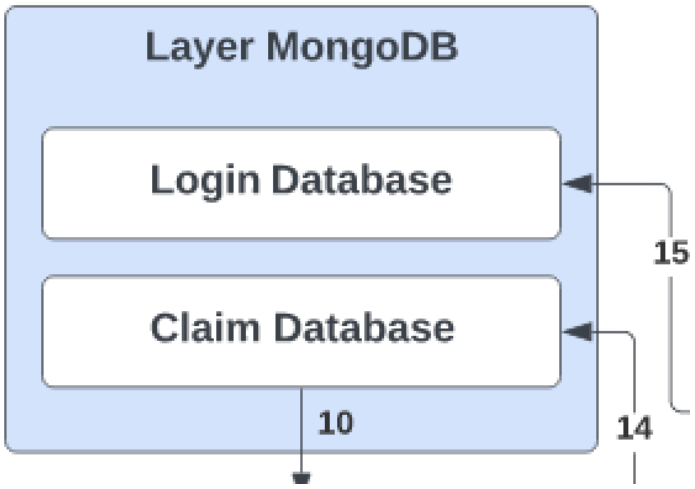

Figure 10: Example subsystem description diagram

10.4.1 Subsystem Hardware

No hardware is used for this subsystem.

10.4.2 Subsystem Operating System

No specific operating system is required for this subsystem.

10.4.3 Subsystem Software Dependencies

MongoDB frameworks is used for this subsystem.

10.4.4 Subsystem Programming Languages

JavaScript is used to make queries to the database.

10.4.5 Subsystem Data Structures

No special data structures are used in this subsystem.

10.4.6 Subsystem Data Processing

Queries are made to retrieve the information from the database and the react application will be populated.

10.5 Claim Database

This is a web service that stores claims created in the react application.

10.5.1 Subsystem Hardware

No hardware is used for this subsystem.

10.5.2 Subsystem Operating System

No specific operating system is required for this subsystem.

10.5.3 Subsystem Software Dependencies

MongoDB frameworks is used for this subsystem.

10.5.4 Subsystem Programming Languages

JavaScript is used to make queries to the database.

10.5.5 Subsystem Data Structures

No special data structures are used in this subsystem.

10.5.6 Subsystem Data Processing

Queries are made to retrieve the information from the database and the react application will be populated.

11 APPENDIX A

Include any additional documents (CAD design, circuit schematics, etc) as an appendix as necessary.

REFERENCES