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

System Requirements Specification
CSE 4316: Senior Design I
Fall 2022

Good Neighbors
Disaster Survey Drone

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

Revision History

Revision	Date	Author(s)	Description
0.1	11.13.2022	FG	document creation
0.2	11.14.2022	AR, CS, DP, FG, MV, PD	complete draft

Contents

List of Figures

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

This section describes the overall structure of our software system.

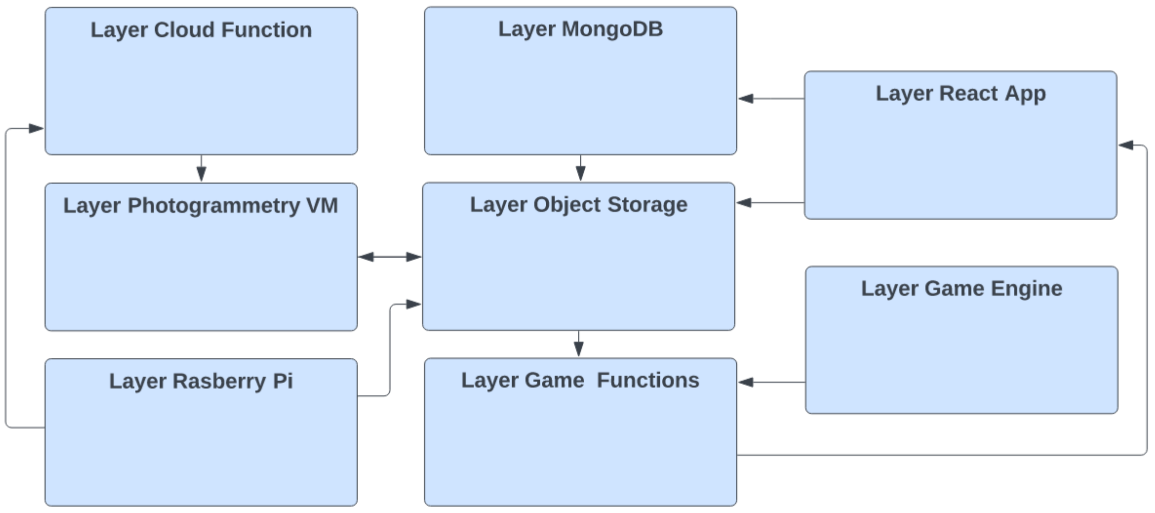

Figure 1: A simple architectural layer diagram

2.1 Cloud Function Layer Description

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

2.2 Raspberry Pi Layer Description

Each layer should be described separately in detail. Descriptions should include the features, functions, critical interfaces and interactions of the layer. The description should clearly define the services that the layer provides. Also include any conventions that your team will use in describing the structure: naming conventions for layers, subsystems, modules, and data flows; interface specifications; how layers and subsystems are defined; etc.

2.3 Object Storage Layer Description

In this 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.

2.4 Photogrammetry VM Layer Description

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. This layer assumes that Google Cloud is not be offline.

2.5 Game Functions Layer Description

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.

2.6 Game Engine Layer Description

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..

2.7 React Web Application Layer Description

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.

2.8 MongoDB Layer Description

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.

3 Subsystem Definitions & Data Flow

This section breaks down our layer abstraction to another level of detail.

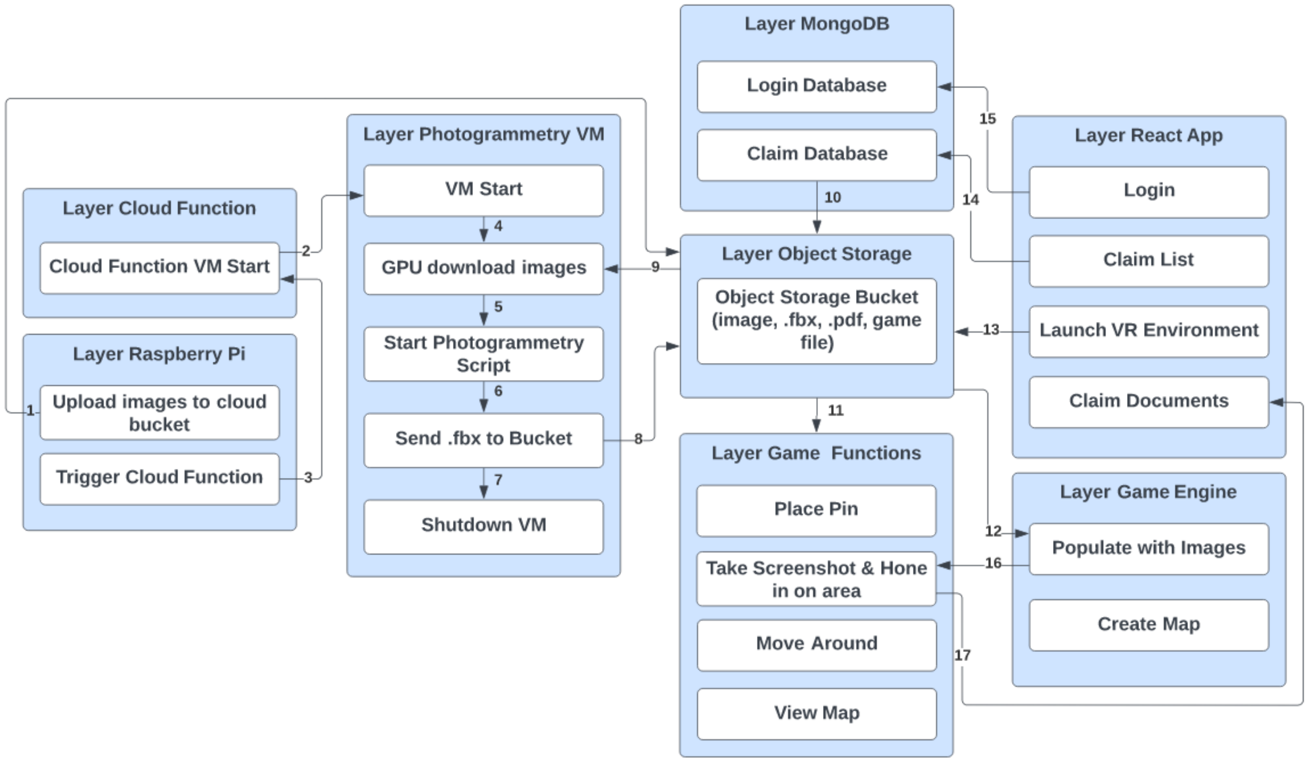

Figure 2: A simple data flow diagram

4 Cloud Function Layer Subsystems

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 isn't responsible for starting the VM instance if it's not triggered.

4.1 Subsystem Cloud Function VM Start

The Cloud Function VM Start subsystem acts as a micro process to start the VM instance whenever images are uploaded via the Raspberry Pi.

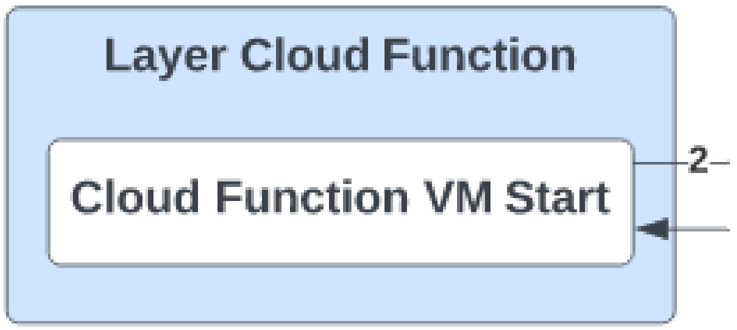

Figure 3: Cloud Function VM Start subsystem description diagram

4.1.1 Assumptions

The Cloud Function gets triggered successfully without throwing any HHTPS error status code.

4.1.2 Responsibilities

The cloud function is responsible for starting the virtual machine so that the photogrammetry process can be started.

4.1.3 Subsystem Interfaces

Table 2: Cloud function VM start interfaces

ID	Description	Inputs	Outputs
#2	Cloud function calls Cloud SDK to start the instance	N/A	N/A
#3	HTTPS POST Request to trigger the cloud function	N/A	N/A

5 Raspberry Pi Layer Subsystems

The Raspberry Pi Layer is responsible for uploading the images to the cloud storage as well as triggering the cloud function to start the VM. This layer has two subsystem which will run through a script that is on the Raspberry Pi.

5.1 Subsytem Upload images to cloud bucket

The upload images to cloud bucket subsystem is responsible for uploading the images from the drone to the google cloud storage bucket. It communicates with the cloud bucket through gsutil application to upload the images.

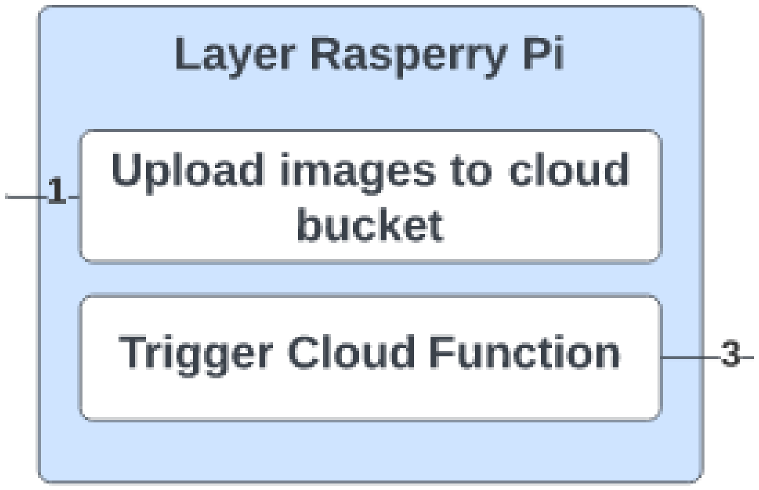

Figure 4: Layer Raspberry Pi subsystems

5.1.1 Assumptions

The raspberry pi is online and connected to internet to initiate the upload to the cloud.

5.1.2 Responsibilities

This subsystem is responsible for uploading the images to the cloud storage bucket.

5.1.3 Subsystem Interfaces

Table 3: Upload images to cloud interfaces

ID	Description	Inputs	Outputs
#1	uploading images to cloud storage	N/a	N/a

5.2 Subsytem Trigger Cloud Function

5.2.1 Assumptions

The images are successfully transferred to the cloud.

5.2.2 Responsibilities

This subsystem is responsible triggering the cloud function which starts the VM.

5.2.3 Subsystem Interfaces

Table 4: Trigger cloud Function interface

ID	Description	Inputs	Outputs
#3	triggering the cloud function	N/a	N/a

6 Object Storage Layer Subsystems

In this 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.

6.1 Object Storage Bucket

The Object Storage Bucket subsystem stores the following files: image, .fbx, .pdf, and game file.

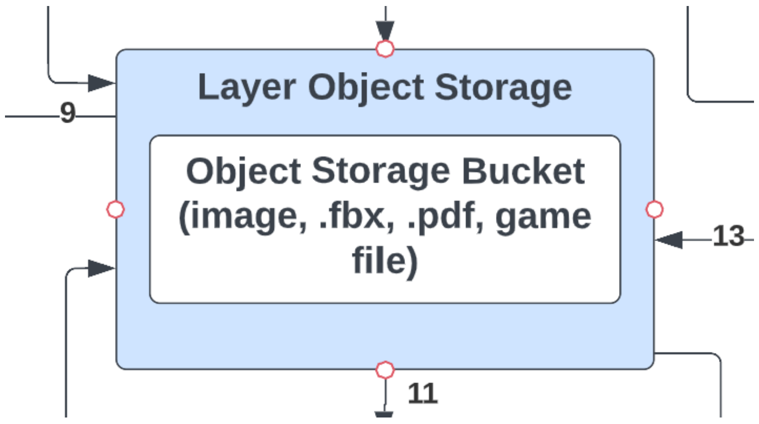

Figure 5: Layer Object Storage subsystem diagram

6.1.1 Assumptions

Other layers communicate with the object storage using gsutil application.

6.1.2 Responsibilities

This subsystem is responsible for storing the images sent from the drone as well as the .fbx file sent from VM. Similarly, this subsytem also stores the claim documents and game file.

6.1.3 Object Storage Bucket Interfaces

Table 5: Object Storage Bucket interfaces

ID	Description	Inputs	Outputs
#1	uploading images from Rpi to cloud storage	N/a	N/a
#8	uploading .fbx file from VM to cloud storage	N/a	N/a
#9	sending the images from cloud to GPU for processing	N/a	N/a
#10	React App updating the claim documentation (pdf) on cloud storage	N/a	N/a
#11	Game Function accessing the files from object storage	N/a	N/a
#12	Game Engine access images from Object Storage	N/a	N/a
#13	VR Environment accessing the files from object storage	N/a	N/a

7 Photogrammetry VM Layer Subsystems

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. This layer assumes that Google Cloud is not be offline.

7.1 Subsystem VM Start

This subsystem starts the VM instance. It gets triggered by the Cloud Function VM Start subsystem from the Cloud Function layer. Once the VM has started, it triggers the GPU download images subsystem.

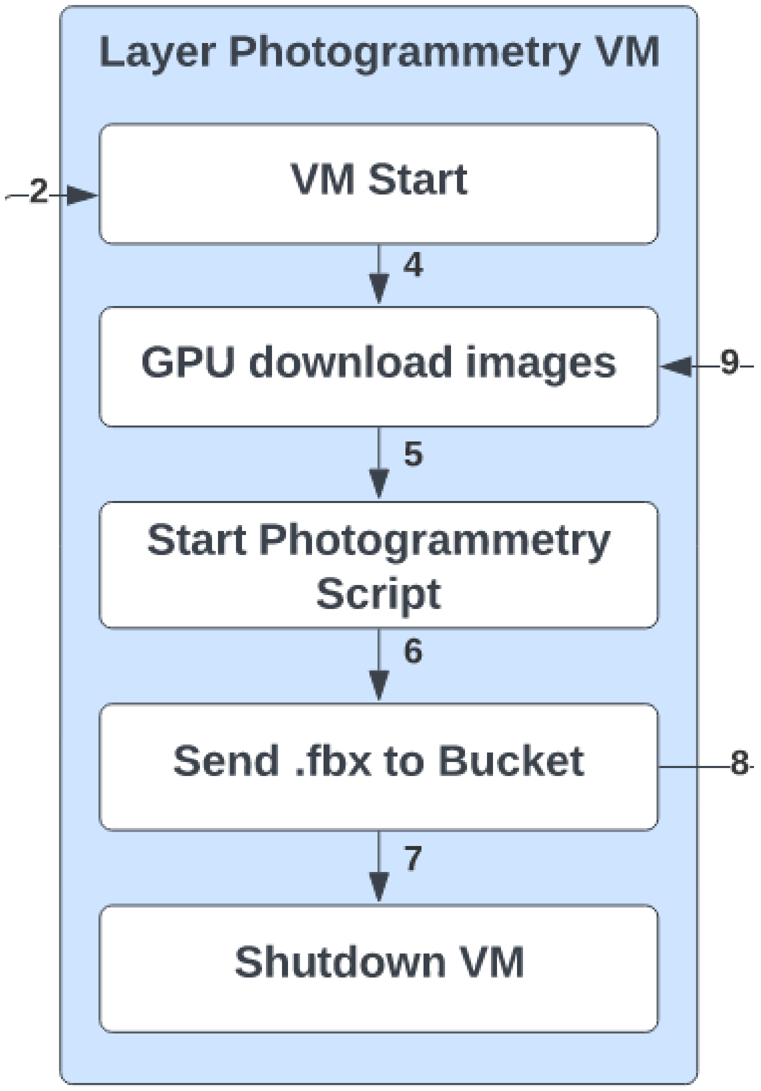

Figure 6: Example subsystem description diagram

7.1.1 Assumptions

The subsystem assumes that the drone has uploaded images to the object storage bucket.

7.1.2 Responsibilities

This subsystem is responsible for starting the VM instance. Its important that we run the VM instance for only enough time to do the photogrammetry process to avoid wasting money. It is also responsible for triggering the GPU download images subsystem.

7.1.3 Subsystem Interfaces

Each of the inputs and outputs for the subsystem are defined here. Create a table with an entry for each labelled interface that connects to this subsystem. For each entry, describe any incoming and outgoing data elements will pass through this interface.

Table 6: VM Start interfaces

ID	Description	Inputs	Outputs
#2	The cloud function start triggers the Photogrammetry VM to start	N/A	N/A
#4	The VM starting triggers the GPU to download the images	N/A	N/A

7.2 Subsystem GPU download images

This subsystem simply downloads the images from the object storage bucket. It gets triggered once the VM has started. Once the images are done downloading, it triggers the start photogrammetry script subsystem.

7.2.1 Assumptions

The subsystem assums that images have been uploaded from the drone into the object storage bucket.

7.2.2 Responsibilities

This subsystem is responsible for downloading images from the object storage bucket and triggering the start photogrammetry script subsystem.

7.2.3 Subsystem Interfaces

Each of the inputs and outputs for the subsystem are defined here. Create a table with an entry for each labelled interface that connects to this subsystem. For each entry, describe any incoming and outgoing data elements will pass through this interface.

Table 7: GPU download images interfaces

ID	Description	Inputs	Outputs
#4	The VM Start subsystem triggers the GPU to download images	N/A	N/A
#9	The GPU downloads images from the object storage bucket	images	N/A
#5	Once the GPU has finished downloaded images, it triggers the photogrammetry script to start	N/A	N/A

7.3 Subsystem Start Photogrammetry Script

This subsystem runs the photogrammetry script. It gets triggered once the GPU download images subsystem is done. And once the script is done, it sends the .fbx file to the object storage bucket.

7.3.1 Assumptions

The subsystem assumes that the GPU has downloaded all of the images taken from the drone.

7.3.2 Responsibilities

This subsystem is responsible for running the photogrammetry script which takes the pictures take from the drone and turns them into a .fbx file using the photogrammetry software RealityCapture. It is also responsible for sending that .fbx file to the object storage bucket.

7.3.3 Subsystem Interfaces

Each of the inputs and outputs for the subsystem are defined here. Create a table with an entry for each labelled interface that connects to this subsystem. For each entry, describe any incoming and outgoing data elements will pass through this interface.

Table 8: Start Photogrammetry Script interfaces

ID	Description	Inputs	Outputs
#5	Once the GPU has finished downloaded images, it triggers the photogrammetry script to start	N/A	N/A
#6	Once the photogrammetry script has ran, it triggers the Send .fbx to bucket subsystem	N/A	N/A

7.4 Subsystem Send .fbx to Bucket

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

7.4.1 Assumptions

The subsystem assumes the photogrammetry process has completed.

7.4.2 Responsibilities

This subsystem is responsible for sending the .fbx file to the object storage bucket so that it can be accessed later from the react app. It is also responsible for triggering the VM to shutdown.

7.4.3 Subsystem Interfaces

Each of the inputs and outputs for the subsystem are defined here. Create a table with an entry for each labelled interface that connects to this subsystem. For each entry, describe any incoming and outgoing data elements will pass through this interface.

Table 9: Send .fbx to Bucket interfaces

ID	Description	Inputs	Outputs
#6	Once the photogrammetry script has ran, it triggers the Send .fbx to bucket subsystem	N/A	N/A
#7	Once the .fbx file has been sent to the object storage bucket, it triggers the Shutdown VM subsystem	N/A	N/A
#8	The .fbx file gets sent from the Photogrammetry Layer to the Object Storage Layer	N/A	.fbx file

7.5 Subsystem Shutdown VM

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

7.5.1 Assumptions

The subsystem assumes that the photogrammetry process is complete and the .fbx file is in the object storage bucket.

7.5.2 Responsibilities

This subsystem is responsible for shutting down the VM so that it does not waste money while no processing is happening.

7.5.3 Subsystem Interfaces

Each of the inputs and outputs for the subsystem are defined here. Create a table with an entry for each labelled interface that connects to this subsystem. For each entry, describe any incoming and outgoing data elements will pass through this interface.

Table 10: Shutdown VM interfaces

ID	Description	Inputs	Outputs
#7	Once the .fbx file has been sent to the object storage bucket, it triggers the Shutdown VM subsystem	N/A	N/A

8 Game Functions Layer Subsystems

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.

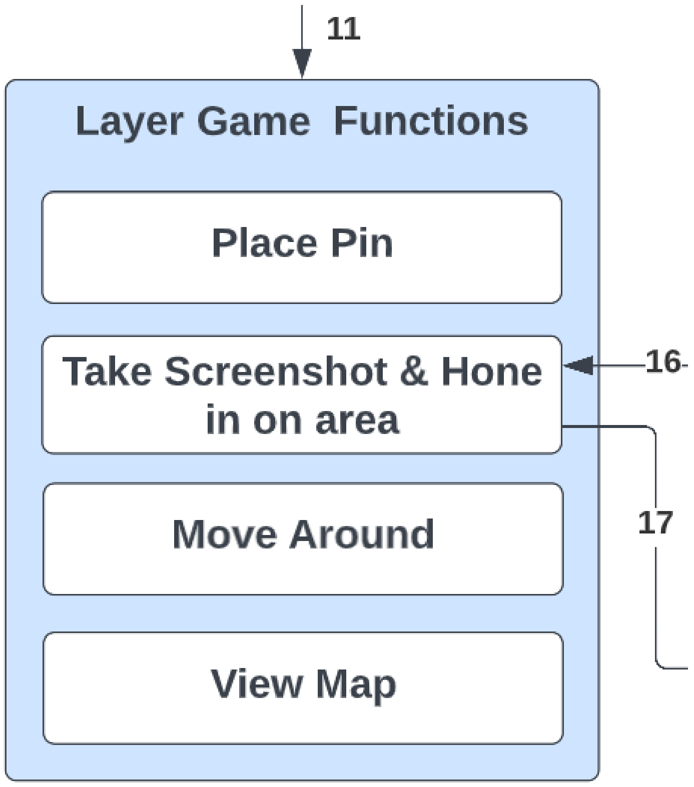

Figure 7: Example subsystem description diagram

8.1 Subsystem Place Pin

This subsystem allows the user to place a pin the the virtual environment.

8.1.1 Assumptions

The subsystem assumes that images have been uploaded from the drone into the object storage bucket, and that there will be target areas that need the users attention.

8.1.2 Responsibilities

This subsystem is responsible for marking areas that are of value to the client's claim. A pin will be placed in the virtual environment, noting that it is an area of importance or needs to be reassessed.

8.1.3 Subsystem Interfaces

Table 11: Place Pin interfaces

ID	Description	Inputs	Outputs
#1	Place Pin	User inserts pin in desired location	Pin in place in virtual environment

8.2 Subsystem Take Screenshot & Hone in on Area

This subsystem will allow the user to take screenshots in the virtual environment and zoom in on the damaged items or area. When zoomed into the damaged item or area, an image of the target area will populate the screen. The screenshot images will automatically be uploaded to the 'Claims Document' on the State Farm website, where agents will be able to conveniently access and review.

8.2.1 Assumptions

The subsystem assumes that images have been uploaded from the drone into the object storage bucket.

8.2.2 Responsibilities

This subsystem is responsible for sending the screenshot images into the React App where agents will be able to access it on the website, and getting real images of damaged items and areas.

8.2.3 Subsystem Interfaces

Table 12: Take Screenshot & Hone in on Area interfaces

ID	Description	Inputs	Outputs
#1	Take Screenshot	User will use 'print screen' to take a screenshot	screenshot will be placed in 'claims document' in React App
#2	Hone in on Area	User will use scroll button to zoom into an item or area	item or area of attention will be viewed at a larger scale & actual image of the damage will be populated on the screen

8.3 Subsystem Move Around

This subsystem allows the user to move around the virtual environment in first or third person by manipulation of the mouse and WASD keys.

8.3.1 Assumptions

The subsystem assumes that images have been uploaded from the drone into the object storage bucket.

8.3.2 Responsibilities

This subsystem is responsible for allowing the user to walk through the virtual environment that is uploaded from the object storage. This will determine what objects the user is able to interact and collide with.

8.3.3 Subsystem Interfaces

Table 13: Move Around interfaces

ID	Description	Inputs	Outputs
#5	Move Around	mouse and WASD keys	user movement in virtual environment

8.4 Subsystem View Map

This subsystem allows the user to view a digitalized map or location of the target area.

8.4.1 Assumptions

The subsystem assumes that images have been uploaded from the drone into the object storage bucket.

8.4.2 Responsibilities

This subsystem is responsible for determining the location of the damage and being able to view it on a digitilized map.

8.4.3 Subsystem Interfaces

Table 14: View Map interfaces

ID	Description	Inputs	Outputs
#6	View Map	User loads map	Location on map is shown

9 Game Engine Layer Subsystems

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.

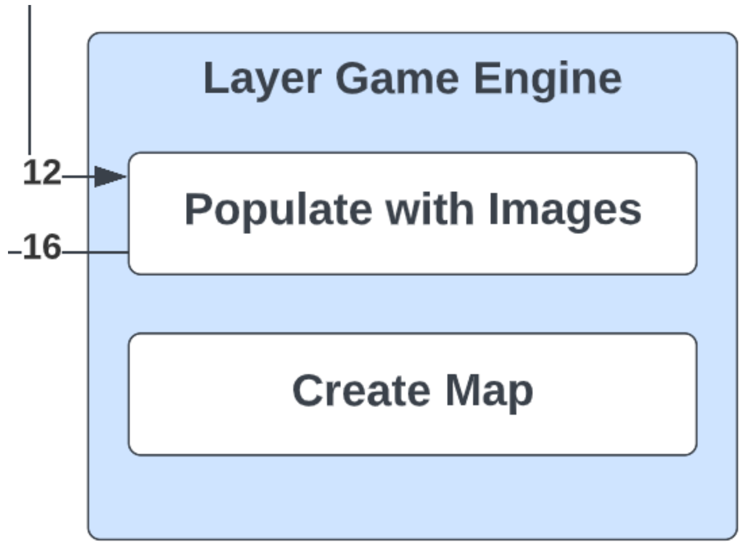

Figure 8: Game Engine Layer Subsystems Diagram

9.1 Subsystem Populate with Images

This subsystem allows the object inside the engine to be populated with images.

9.1.1 Assumptions

This subsystem assumes that the section of the object selected by the user has a corresponding picture associated to it.

9.1.2 Responsibilities

This subsystem is responsible for allowing to click on a particular section of the object and retrieving a corresponding picture associated with it.

9.1.3 Subsystem Interfaces

Table 15: Populate with Images interfaces

ID	Description	Inputs	Outputs
#1	Populate with Images	User clicks on a desired location on the object	An image corresponding to the area is retrived

9.2 Sub System Create Map

This subsystem creates a 2D map of the area surrounding the user and places in in the top left corner of the screen

9.2.1 Assumptions

This subsystem assumes that all the objects in the environment can be mapped onto a small 2D plane to be placed on the corner

9.2.2 Responsibilities

This subsystem is responsible for giving the user more information regarding the surrounding they are in using the Map feature

9.2.3 Subsystem Interfaces

Table 16: Create Map

ID	Description	Inputs	Outputs
#1	Create Map	User Opens the Game	A 2D map of the area is displayed on the top left corner of the screen

10 React Web Application

The React Web Application layer includes the user interface allowing the state farm agent to login and retrieve environments from the database. This layer will allow the user to access the virtual environment and view the screenshots taken in the virtual environment.

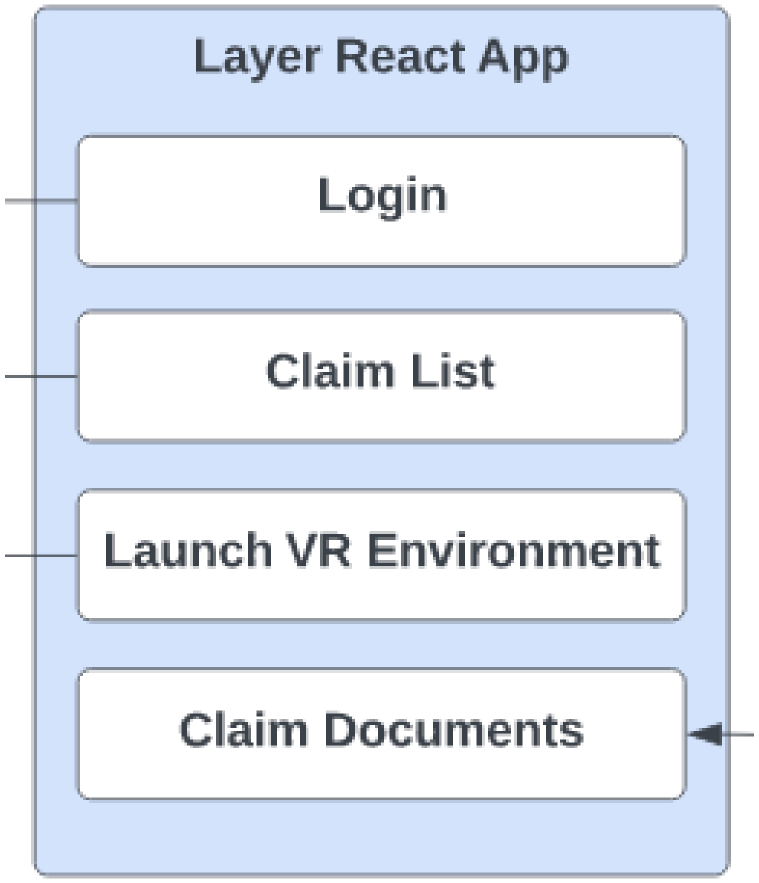

Figure 9: React App Layer subsystem

10.1 Subsystem Login

This subsystem will allow the user to log into the react web application.

10.1.1 Assumptions

This subsystem assumes the user has credentials stored in the database.

10.1.2 Responsibilities

This subsystem is responsible for getting the user access into the application.

10.1.3 Subsystem Interfaces

Table 17: Login interfaces

ID	Description	Inputs	Outputs
#1	Login	Username and Password	Access to the website

10.2 Subsystem Claim List

This subsystem will retrieve a list of claims from the database.

10.2.1 Assumptions

This subsystem assumes there are claims available in the database.

10.2.2 Responsibilities

This subsystem is responsible for retrieving the files from the database so the user can just click on the claim.

10.2.3 Subsystem Interfaces

Table 18: Claim List interfaces

ID	Description	Inputs	Outputs
#1	Claim List	Folder with all files	Populated buttons with claim information

10.3 Subsystem Launch VR Environment

This subsystem will trigger a pop up screen, showing the virtual environment for the chosen claim. This will allow the user to enter the virtual environment.

10.3.1 Assumptions

This subsystem assumes there is an .fbx file available in the object storage bucket.

10.3.2 Responsibilities

This subsystem is responsible for opening the correct virtual environment and allowing the user to enter it.

10.3.3 Subsystem Interfaces

Table 19: Launch VR Environment interfaces

ID	Description	Inputs	Outputs
#1	Opening Pop Up Screen for VR	.fbx file	Screen with the virtual environment

10.4 Subsystem Claim Documents

This subsystem will create a PDF file with all the information the user has retrieved from the virtual environment.

10.4.1 Assumptions

This subsystem assumes that the user has already interacted with the virtual environment.

10.4.2 Responsibilities

This subsystem is responsible for creating a PDF file.

10.4.3 Subsystem Interfaces

Table 20: Claim Documents interfaces

ID	Description	Inputs	Outputs
#1	Creating PDF	Information about the claim	PDF file

11 MongoDB Layer Subsysystems

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.

11.1 Login Database

The login database within the MongoDB Layer is responsible for connecting the agents to the website and display the proper information for the particular agent. It communicates with the React Web Application Layer and the Layer Object Storage.

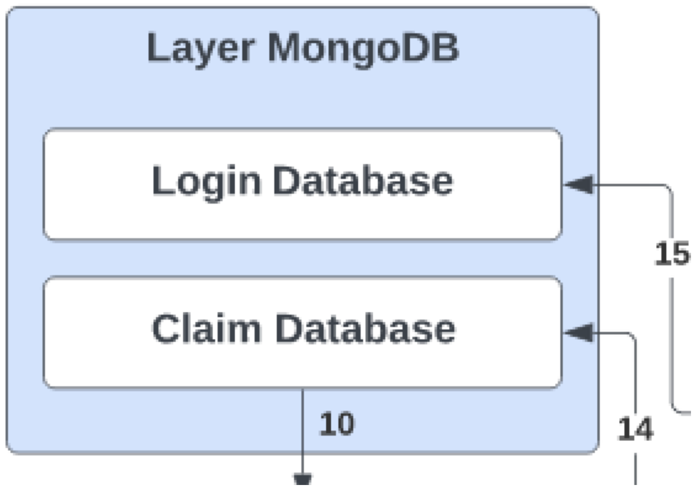

Figure 10: Example subsystem description diagram

11.1.1 Assumptions

The login databases assumes an account has already been created for the agent and will login with already provided credentials.

11.1.2 Responsibilities

The responsibilities of this subsystem include protecting the users password and personal information, displaying the correct information based on which user is logged in, linking claims between accounts so the users can open the external application to use the VR environment.

11.1.3 Subsystem Interfaces

The interfaces of the MongoDB Layer include the login functions of the react app, the claims list in the react app, and the layer object storage.

Table 21: Claim List interfaces

ID	Description	Inputs	Outputs
#1	Login Database	User information	n/a
#2	Claim Database	Claim files and IDs	Claim Files

11.2 Login Database

This subsection will control all of the account information

11.2.1 Assumptions

Assumes that all user information will be given by admin

11.2.2 Responsibilities

The responsibilities of this subsystem include storing and retrieving login information for when users log in.

Table 22: Claim List interfaces

ID	Description	Inputs	Outputs
#1	Login Database	User information	n/a

11.3 Claim Database

This subsection will control the storing of information related to the claims.

11.3.1 Assumptions

Assumes that all user information will that claims will be retrieved from the cloud to be uploaded mongoDB.

11.3.2 Responsibilities

The responsibilities of this subsystem include storing and retrieving claim information.

Table 23: Claim List interfaces

ID	Description	Inputs	Outputs
#1	Claim Database	Claim files and IDs	Claim Files

References