Project Report

Unstable Bluffs Detection Monitoring System

Joubert Marion O. Trias
Professor Donyanavard
CS 250-02
9 December 2022

Table of contents

1. User Manual

a. Description of the system

This is a detection monitoring system to monitor bluffs in the Del Mar area and alert any authorities when an unstable bluff has been detected. It is a needed system because unstable bluffs are present in the Del Mar area and would potentially endanger a person's life for the people on the beach near the bluffs. This system will help avoid any danger for those people. Major features include high definition and wifi enabled cameras with night vision, a central processing system, and an alarm.

b. User Requirements

Users will be able to interact with the system in the central processing station which will act as the main hub for controlling the cameras and processing images. The hub will be able to view live images and videos from the cameras and older images from the cloud storage. It will also have the ability to alert the authorities of highly dangerous bluffs. It will include its own hardware such as a computer system to interact with the cameras, a cloud storage system where to store the data of the images, the high-definition and built-in wifi cameras to record and provide live-feed of images and videos, and alarm that will provide a loud wail that will reach the distance beyond the bluffs. The UI of the system should be similar to Windows UI. Constraints may include not compatible with any mobile systems in terms of viewing live footage or old ones but still viable to receive any warnings via alarm.

2. Design Documents

a. Requirements

i. Functional Requirement

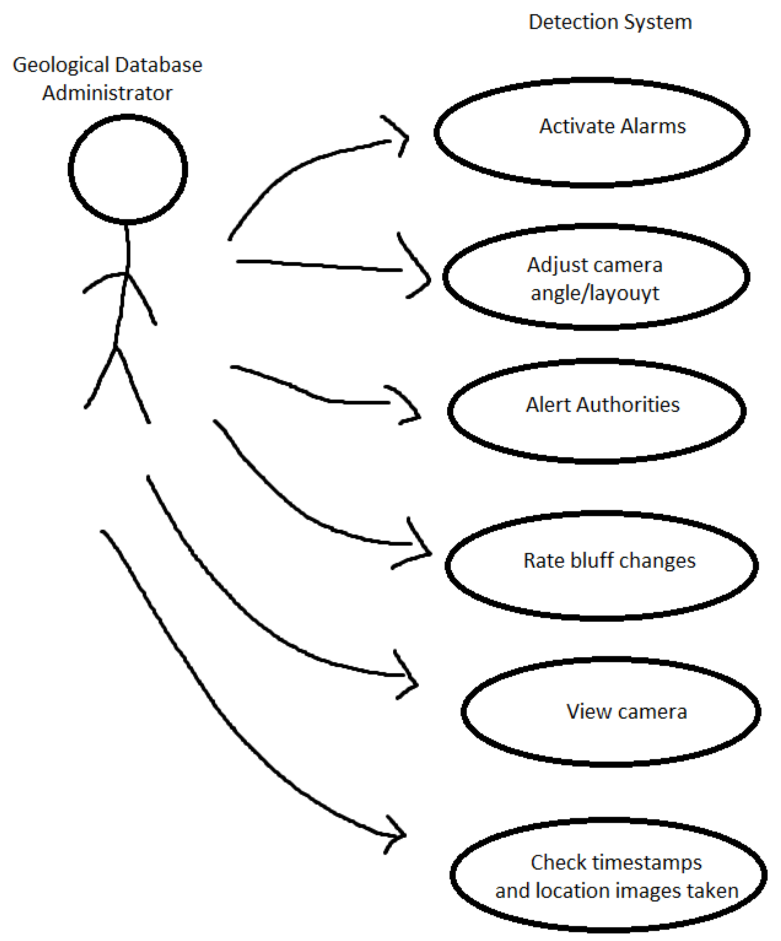
We will introduce the functionality of the system which is alerting authorities, rating bluff changes, activating alarms, viewing cameras, and checking timestamps and geolocation. An administrator can activate alarms in the event of the bluffs becoming unstable for unforeseeable events such as earthquakes or tsunamis and be able to immediately alert the authorities. Administrators can change the camera layout in the center to which it is more convenient or need adjusting to observe more or better on the bluffs. Only the administrator can rate the images taken from the bluff changes as they are more experienced in this type of field. The ability for both users to view the cameras and see where it was taken from for both commercial and educational purposes but the administrator still holds the hand of the hardware.

ii. Non-Functional Requirement

Data can be leaked and can be used for malicious purposes or disrupt the cloud storage as a whole with security. Security should be tight to prevent any security vulnerabilities. Portability can also be an issue, this will be constrained as it relies on the Windows system. Although this can be accessed to any computer but in the event that it cannot, the cloud storage that is filled with data is still secured and functional as we store it from a cloud instead of an onsite server. This cloud would contain a two month's worth of hourly data. This would be kept in a secure cloud storage which can only be accessed for administrative purposes to study and prevent unstable bluffs.

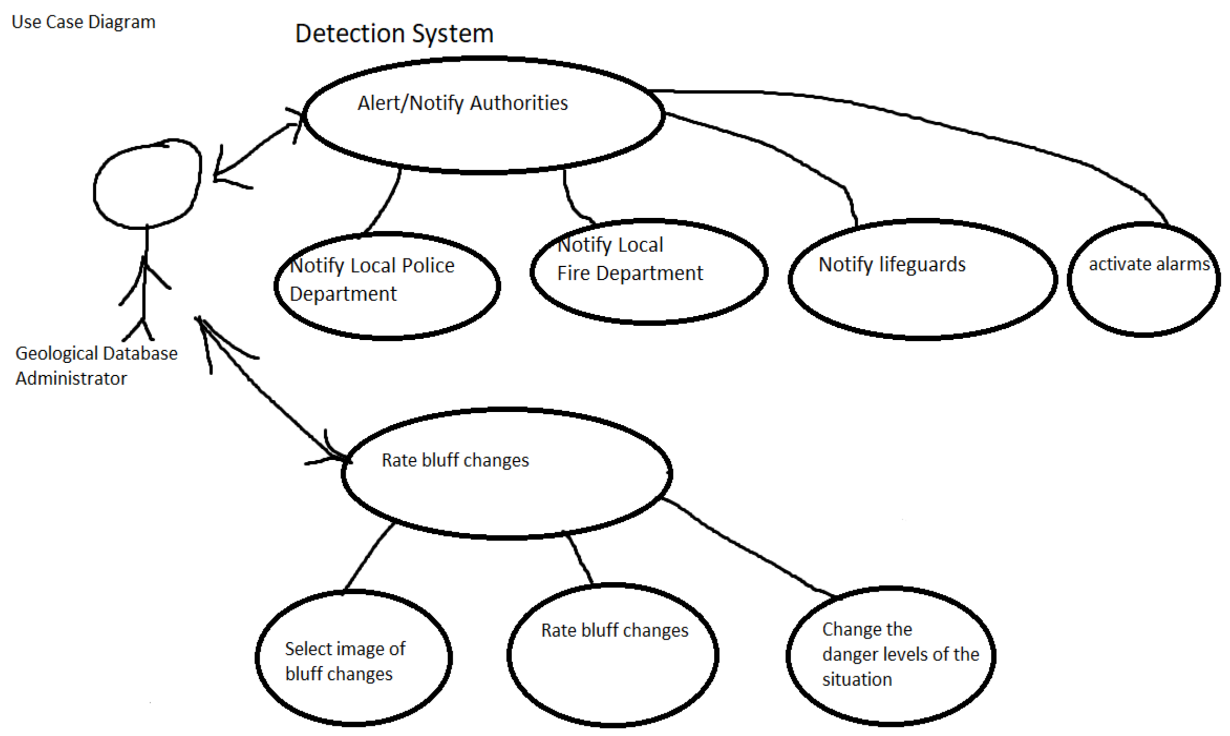
b. Use Case Diagram

i. General Use Case


Detection System
Geological Database Administrator
Beach goer
Activate Alarms
Adjust camera angle/layouyt
Alert Authorities
Rate bluff changes
View camera
Check timestamps and location images taken

This is the generalized use case diagram, the functionalities on this diagram have been described in the functional requirement above. This image will portray how users can interact with the system without going too in depth as it is only intended for clients

ii. Specific Use Case


Use Case Diagram
Detection System
Alert/Notify Authorities
Notify Local Police Department
Notify Local Fire Department
Notify lifeguards
activate alarms
Geological Database Administrator
Rate bluff changes
Select image of bluff changes
Rate bluff changes
Change the danger levels of the situation

In this specified case, we go deeper into the generalized case diagram in terms of alerting and notifying authorities and rating bluff changes. The Geological Database Administrator will serve as the main actor which can trigger alerting authorities which branches into notifying the following authorities such as the local police, fire department, and lifeguards that will end up activating the alarms. Software cannot simply evacuate everyone so it is up to the users or people around the area to evacuate once the alarms have been activated. Another specified case would be rating bluff changes in which only the administrator can trigger. They have to select the image of bluff change, rate image, change danger levels in which we should keep an eye on this one or start evacuation procedures.

c. Validation

In this validation, we will test if the system meets the client's requirements.

Validation Acceptance Testing

Is the entire system connected?

Answer: Yes, the entire system will be connected to via IPs. All together they are connected to the network infrastructure that will send and receive data from the cameras and to the central processing station to be rated.

Test

Camera connected to Local Test Facility
Local Test Facility connected to Network Infrastructure

Network Infrastructure connected to Central Processing Station
Central Processing Station connected to the Alarm System
Central Processing Station connected to the Database

Expected Result: Entire system is online and fully functioning.
Is the client satisfied? Yes/No

Can the images be stored in a database and remain there for a certain amount of time?

Answer: Yes, the data of the images will be all stored in the database including ones that are a month old.

Test

Camera takes image of bluff
Camera sends image to Local Test Facility then to Network Infrastructure
Central Processing Station accepts image from Network Infrastructure and rates
Stores data of the image to database
Hold data for 1-2 months

Expected Result: Camera was able to take a bluff and sends it to the system for rating then stored in the database.
Is the client satisfied? Yes/No

Are security features up to par to your expectations?

Answer: There will be security features in the event of breach or rupture of bluffs. An alarm system will be present on those beaches that will alert those people. It will also notify the following authorities that will aid in the

Test

Activate Alarm System
Did the alarm go off? Yes/No
Did authorities respond? Yes/No

Expected Results: Authorities responded and the Alarm system went on.
Is the client satisfied? Yes/No

Does the camera consist of the necessary features?

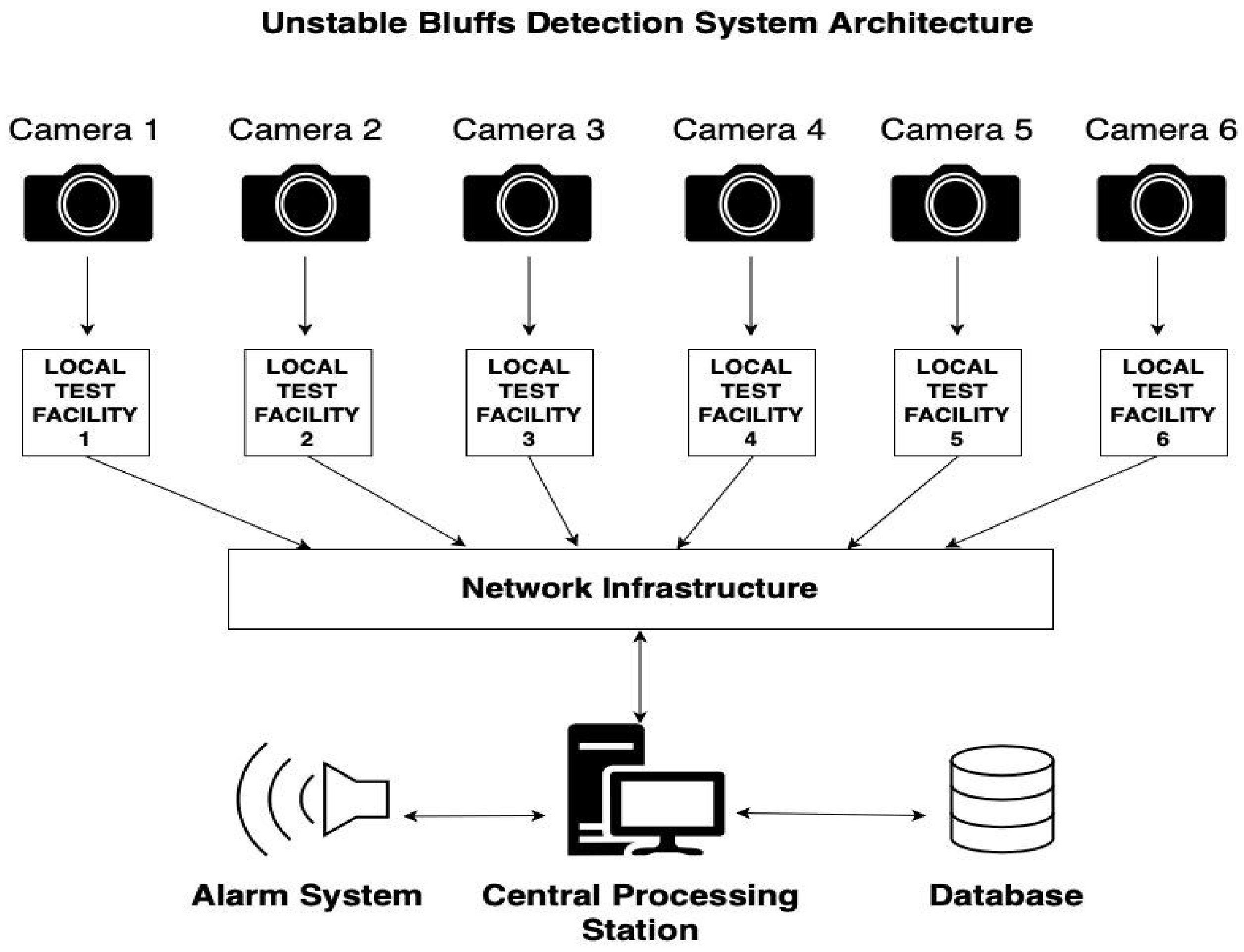
Answer: Yes, the cameras will be wifi-enabled to connect with the system, able to use night vision to monitor bluffs during the night. It is high definition as well.

Test

Camera takes an image of a bluff
Camera takes an image of a bluff in the night

Expected Result: Captures the images that are high definition and one in night vision feature. Is the client satisfied with the results of the image? Is this the feature they want or want to add more? Yes/No

d. Architecture Diagram



The Detection System consists of 6 10-megapixel wifi-enabled cameras each with an internal timer. Each camera is connected to a local test facility (LTF) which accepts images then transmits them to the central site processing system (controlling computer) through a network infrastructure. The network infrastructure consists of IP networks, wireless access points, routers, and repeaters. The central site processing system will be able to send and adjust internal timing intervals to each camera and receive images from each camera. The central site processing system also has a database to store and retrieve images and an alarm system to send alerts to lifeguards, rail traffic controllers, and first responders when the bluff rating reaches a 4 or 5 indicating imminent danger.

e. Security

In terms of the security, any of the things in our system can be breached or at risk of attacks or corruption. We must know the computer security elements which are confidentiality, integrity, and availability.

i. Confidentiality

Confidentiality is to protect data or information from unauthorized personnel. It must not be compromised by any unauthorized person and can only be seen from authorized parties such as our clients and/or administrators. In terms of our system, our data must be protected against malicious use. Any of the hardware can be breached and the lines in between causing a leak or unauthorized access of the system. The most prime target would be the data of the images stored in our database. It is a potential vulnerability where an actor can breach into that data. It also must protect the administrators and clients data as they would need it to access the database and central processing station.

ii. Integrity

Integrity of the data is also a vulnerability in our system. As mentioned previously, an actor can breach the database, network infrastructure, or any of the hardware and the connections in between can tamper with the data. Possibly, they would change the data into something inaccurate and make false readings which would result in false alarms.

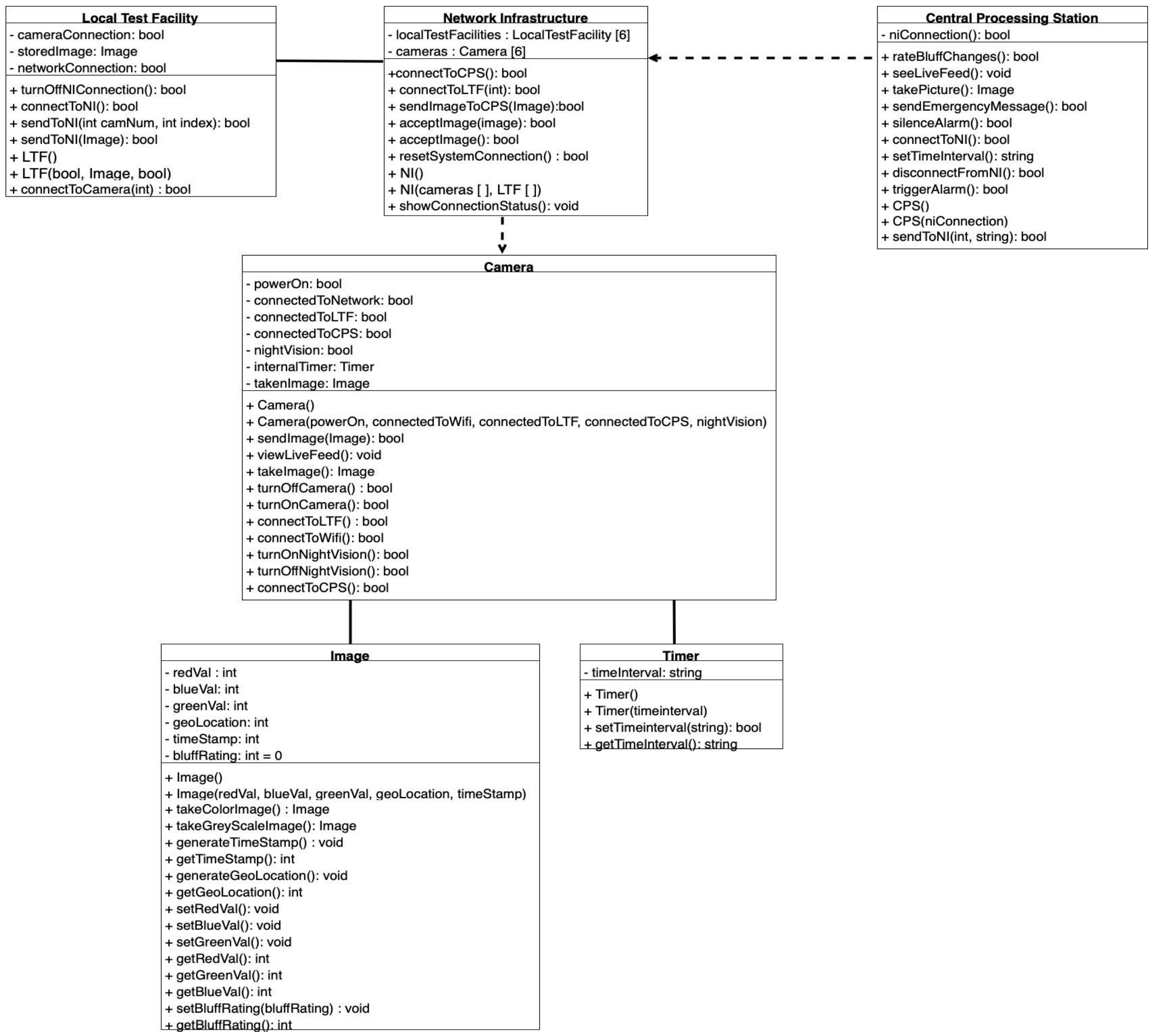
iii. Availability

Availability allows for authorized users to access the data of the system whenever they need it. A vulnerability that can occur is denial of service. Users will be unable to access the system when a denial of service occurs. Although this may or may not be intentional, it can disrupt the service of the users which makes it dissatisfying.

iv. Countermeasures

We must have countermeasures in the event of a threat or breach in the system. In order to prevent confidential threats, integrity breach, and availability disruption. A possible countermeasure would be having a two factor authentication system to reduce the vulnerability. With the two factor authentication, this would prevent fully accessing an authorized person's privileges as they need to breach another wall that they need to break. Another possible countermeasure is creating a firewall. This will be extremely helpful as the system we are dealing with will have a network infrastructure which will increase the security measures in between connections with the entire network.

f. Class Diagram



The next following bullet points will introduce the purpose of the Camera class and explain the attributes and the expected operations. The attributes for this class include connection features in which the camera can connect to the network, local test facility, or central processing station. It does have the ability to be turned on and off as well as night vision capabilities. The camera can take images and has an interval timer that would take images intervally. Operations include its own default constructor and specific constructors with its parameters labeled on the diagram. It can take and send images to the local test facility which would return true if succeeded. As well as the connection and turning on and off capabilities would return true if succeeded. Lastly, viewLiveFeed would show a live-feed of what the cameras see.

- Class: Camera
    - Attributes

        - powerOn: bool
        - connectedToNetwork: bool
        - connectedToLTF: bool
        - connectedToCPS: bool
        - nightVision: bool
        - internalTimer: Timer
        - takenImage: Image
            - Operations
                - Camera(), Camera(...)
                - sendImage (Image): bool, takeImage(): Image
                - turnOffCamera(), turnOnCamera(): bool
                - connectToLTF(), connectToWifi(), connectToCPS(): bool
                - turnOnNightVision(), turnOffNightVision(): bool
                - viewLiveFeed(): void

The next following bullet points will introduce the purpose of the Image class and explain the attributes and the expected operations. Attributes include the value of the colored images to display color. geoLocation and timeStamp would provide the appropriate geolocation and a detailed timestamp to where the image was taken at that current time. It also has a bluffRating which would have a default value of zero as a base. Attributes consist of a default constructor and a specific constructor with parameters. It has getters that would grab color images, timestamps, geolocation, and bluff ratings. Also include setters that would set values for the color of the images such as red, blue, and green and have a value set for the bluffRatings. generateTimeStamp would mark the image with a printed or digital record while generateGeoLocation would be the use of location technologies such as IP addresses to identify and track connected electronic devices.

- Class: Image
    - Attributes
        - redVal: int
        - blueVal: int
        - greenVal: int
        - geoLocation: int
        - timeStamp: int
        - bluffRating: int = 0
            - Operations
                - Image(), Image(...)
                - takeColorImage(), takeGreyScaleImage(): Image
                - generateTimeStamp(), generateGeoLocation(): void
                - getTimeStamp(), getGeoLocation(): int
                - setRedVal(), setBlueVal(), setGreenVal(): void

                - setRedVal(), setBlueVal(), setGreenVal(): int
                - setBluffRating(bluffRating): void
                - getBluffRating(): int

The next following bullet points will introduce the purpose of the Timer class and explain the attributes and the expected operations. Attributes include a timeInterval. Operations include a default constructor and a specific constructor for the timeInterval parameter. Lastly a setTimeInterval to set uses the input string to update the interval of this timer and getTimeInterval would return the time interval of this timer.

- Class: Timer
    - Attributes
        - timeInterval: string
            - Operations
                - Timer(), Timer(timeInterval)
                - setTimeInterval(string): bool
                - getTimeInterval(): string

The next following bullet points will introduce the purpose of the Central Processing Station class and explain the attributes and the expected operations. Attributes include niConnection that would have a boolean response if it succeeded (true) in connecting or not (false). Operations include if changes occurred rateBluffChanges would return a boolean. The sendEmergencyMessage would return a boolean that would send out emergency messages when a bluff becomes highly unstable. seeLiveFeed would open a live-feed to the camera as it can also take images for the image class in the takePicture operation. It can also trigger or silence alarms which would return a boolean. It has to connect to the Network Infrastructure to send and receive signals and can disconnect to prevent any leaks which would return a boolean. The CPS is also able to send a timing interval through the network interval to the camera. It returns true if successful.

- Class: Central Processing Station (CPS)
    - Attributes
        - niConnection(): bool
            - Operations
                - rateBluffChanges(): bool
                - seeLiveFeed(): void
                - takePicture(): Image
                - sendEmergencyMessage(): bool
                - silenceAlarm(), triggerAlarm(): bool
                - connectToNI(): bool

- setTimeInterval(): string
- disconnectFromNI(): bool
- sendToNI(int, string): bool
- sendToNI(Image): bool

The next following bullet points will introduce the purpose of the Network Infrastructure class and explain the attributes and the expected operations. Attributes include control of the local test facilities and cameras. Operations include connection features such as connecting to Central Processing Station or the Local Test Facility. It can also send and receive images that would return a boolean depending if it was successful or not. The resetSystemConnection would return a boolean whenever the system is rebooted for any case scenario. Lastly, it shows the details or status of the connection between the systems.

- Class: Network Infrastructure
    - Attributes
        - localTestFacilities: LocalTestFacility [6]
        - cameras : Camera [6]
            - Operations
                - connectToCPS(), connectToLTF(int): bool
                - sendImageToCPS(image), acceptImage(image): bool
                - resetSystemConnection(): bool
                - showConnectionStatus(): void

The next following bullet points will introduce the purpose of the Local Test Facility class and explain the attributes and the expected operations. This would include connection features that would return a boolean for the cameras and network. It also stores an image from its corresponding camera and can send it through the network infrastructure to the CPS It also would include turning on and off on the network infrastructure.

- Class: Local Test Facility
    - Attributes
        - cameraConnection: bool
        - networkConnection: bool
        - storedImage: Image
            - Operations
                - turnOffNIConnection(): bool
                - connectToNI(), connectToCamera(int): bool
                - sendToNI(image): bool

g. Verification

i. Unit Test

Unit Test 1 Create LTF Object

Description:

This unit test uses white box testing to determine if an overloaded local test facility constructor can be created. The LTF constructor and unit test both take in a boolean value representing the camera connection, an Image object representing the stored image, and a boolean value representing the ltf's network connection status. The unit test attempts to make an overloaded ltf object and will return

Test:

```
bool createLTF (bool camConnection, Image pic, bool niConnection)
{
  try
  {
       LTF ltf = new LTF(camConnection, pic, niConnection)
  }
  catch(InputMismatchException e)
  {
    System.out.println("Error! Parameter Input mismatch");
    return false;
  }

  return true;
}
```

Inputs & Outputs:

Input	**Expected Output**
createLTF(true, colorImage, true)	true
createLTF("false", colorImage, true)	Print statement: "Error! Input mismatch" Boolean: false
createLTF(true, greyImage, false)	true

Unit Test 2 Create Camera Object

Description:

This unit test uses white box testing to determine if an overloaded Camera constructor can be created. The parameters for the constructor and test case are boolean values relating to the camera's power status, connection to a local test facility, connection to the central processing station, and night vision status. The test attempts to make an overloaded camera object and will return a boolean value based on the outcome and an error message if applicable.

Test:

```
bool createCamera (bool powerOn, bool connectedToWifi, bool connectedToLTF, bool
                    connectedToCPS, bool nightVision)
{
  try
  {
       Camera testCamera = new Camera(powerOn, connectedToWifi, connectedToLTF,
                                        connectedToCPS, nightVision);
  }
  catch(InputMismatchException e)
  {
    System.out.println("Error! Parameter Input mismatch");
    return false;
  }
  return true;
}
```

Inputs & Outputs:

Input	**Expected Output**
createCamera(true, true, true,true, true)	true
createCamera("true", true, true, true, true)	Print statement: "Error! Input mismatch" Boolean: false
createCamera(false, false, false, false, false)	true

ii. System Test

System Test 1 takePicture()

Description:

This system test uses black box testing to evaluate if a camera can send a picture to the central processing system. The parameters represent a specific camera number, a color or grayscale image request and connection statuses for the camera, NI, LTF, and CPS. To simulate the Detection System, objects are made for the central processing system, network infrastructure, and camera. If any one of the connection statuses are false, which indicates a connection failure, then the test will throw a connection error and terminate. A test is also made to check for a valid camera index. If there are no connection errors then a camera object takes an image and sends it to its corresponding LTF. Then the LTF sends the image though the network infrastructure to the CPS. If the system is unable to send the image between its components to the CPS, then it displays an error message and terminates the test. The CPS checks if the image has the correct rgb combination for the requested color or grayscale image as well as the other parameters. If there are any unexpected values then the test throws a corruption error and quits. If not, then the test passes.

Test:

```
takePicture(int camNum, string color, bool cameraCon, bool niCon, bool cpsCon, bool ltfCon)
{
  // Create objects to simulate the detection system
  Camera cam = cameras[camNum];
  NI network = new NI();
  CPS cps = new CPS();
  LTF ltf = localTestFacilities[camNum];

  if(camera, network infrastructure, ltf, or cps aren't connected to wifi or eachother)
       Throw connection error;

  if(camNum > 0 || camNum < 5)
      Throw index out of bounds error

  // take a color or grayscale image depending on color parameter
  if(color equals ‘color’)
      Image pic = cam.takeColorImage();
   else
       Image pic = cam.takeGrayImage();

   send pic to ltf using sendImage(image)
        if returns false then display error message and terminate test

   from ltf send pic to network using sendToNI(image)
        if returns false then display error message and terminate test

  From network send pic to cps using sendImageToCPS(image)
         if returns false then display error message and terminate test

  if(color = “color”)
     Check if pic’s rgb combination values are all greater than 50, if not throw color error
  else if(color = “gray”)
      Check if pic’s rgb combination values are all equal to 50, if not throw gray error

 Print(“takePicture system test passed”);
}
```

Inputs & Outputs:

Input	**Expected Output**
takePicture(2, “color”, false, true, true, true)	throw connection error
takePicture(5, “color”, true, true, true, true)	takePicture system test passed
takePicture(3, “gray”, true, true, true, true)	takePicture system test passed
takePicture(100, “gray”, true, true, true, true)	throw index out of bounds error

System Test 2 testTimeInterval()

Description:

This system test uses black box testing to evaluate if the central processing system can successfully send a new timer interval to a specific camera. The parameters represent a specific camera number, connection statuses for the camera, central processing system, local test facility, and minutes, hours, and seconds. The last three parameters will be used to construct a timing interval which will be sent from the central processing system to the camera. To simulate the Detection System a central processing system, network infrastructure, and camera object are each made. If any one of the connection statuses are false, which indicates connection failure, then the test will throw a connection error and terminate. Tests are also made to check for a valid camera index number and valid time ranges. Next, a String time interval is constructed and sent from the central processing system to a specific local test facility. There, the specific LTF will send the time interval to its paired camera and assign its new values. Finally, the camera’s minutes, hours, and seconds will be evaluated against the original parameters. If they all match then the test passes, if not then the test fails.

Test:

```
testTimeInterval(int cameraNum, bool cameraCon, bool niCon, bool cpsCon, int mins, int hours,
int seconds)
{
  // Create class objects to simulate detection system
  CPS cps = new CPS();
  NI networkInfrastructure = new NI();
  Camera testCamera = cameras[cameraNum];
  if (camera, network infrastructure, or cps aren't connected to wifi or eachother)
       throw connection error
   if (cameraNum < 0 || cameraNum > 5)
       throw index out of bounds error
   if( (mins < 0 || mins > 60) || (hours < 0 || hours > 24) || (seconds < 0 || seconds > 60))
       throw invalid interval error
  // Create a time interval string and send it to the network infrastructure
  String interval = hours + "-" + mins + "-" + seconds;

  String timeInterval = cps.setTimeInterval(interval);

  // Send to NI, which has an array of cameras and LTFs
  cps.sendToNI(cameraNum, timeInterval);

  // Update testCamera's internal timer
 testCamera.Timer(timeInterval);

 // Test if updates
 if(testCamera.hours != hours || testCamera.mins != mins || testCamera.seconds != seconds)
 {
   throw TimerError;
 }
 else
 {
    System.out.println("Test Successful");
  }
}
```

Inputs & Outputs:

Input	**Expected Output**
testTimeInterval(2, false, true, true, 0, 10, 10)	throw connection error
testTimeInterval(5, true, true, true, 0, 0, 0)	testTimeInterval passed
testTimeInterval(3, true, true, true 0, -10, 0)	throw invalid interval error
testTimeInterval(100, true true, 1, 1, 1,)	throw index out of bounds error

iii. Functional Test

Functional Test 1 connectToCamera

Description:

In this functional test, we will be using black box testing to test if an LTF can connect to a specific camera. The parameters for the test are an integer representing an index in the camera array, a boolean representing the LTF's connection status, and a boolean representing the camera's connection status. The test creates an LTF object and an array of 6 cameras to simulate the system architecture. Next, the test checks if the integer parameter is within array bounds. If not, then an error message is displayed and a false is returned. Afterwards, the LTF object is assigned the parameter cameraConnectionStatus. The LTF object and camera object will then attempt to connect to each other. Each of their results are stored in boolean variables. Lastly, the aforementioned boolean variables are compared to the boolean test case parameters for equality. The LTF and camera should only be able to connect to each other when tfConnectionStatus and cameraConnectionStatus are both true. Furthermore, the boolean results should not have opposite boolean values from the boolean parameters. If so, that constitutes an error and ends the test case with an error message and a false boolean result. If the test successfully passes then a true boolean value is returned with a "Test Passed" cout statement.

Test:

```
bool testConnectToCamera(int num, bool ltfConnectionStatus, bool cameraConnectionStatus)
{
        // create objects
        LTF ltf = new LTF();

        // camera array simulates the architecture array of cameras
        Camera cameras[6];

        // populate camera array with values
        for(int i  = 0; i < cameras.length(); i++)
        {
                cameras[i].connectedToLTF = ltfConnectionStatus;
        }

        // test for invalid index
        if(num > cameras.length() -1 || num < 0)
        {
                cout << "Error! Invalid index" << endl;
                return false; // invalid index
        }
// assign ltf's camera connection with parameter value
ltf.cameraConnection = cameraConnectionStatus;

//attempt to connect
bool camResult = camera[num].connectToLTF();
bool ltfResult =   ltf.cameraConnection;

// evaluate result
if(camResult != ltfConnectionStatus | | ltfResult != cameraConnectionStatus)
{
cout << "Test failed" << end;
 return false;
}
cout << "Test passed" << end;
return true;
} // end of test
```

Inputs & Outputs:

Input	**Expected Output**

connectToCamera(10, true, true)	cout << "Error! Invalid index" << endl;
	return false

connectToCamera(5, false, false)	cout << "Test passed" << end;
	return true

connectToCamera(4, true, true)	cout << "Test passed" << end;
	return true

<u>Functional Test 2 acceptImage</u>

Description:

In this functional test, we will be using black box testing to see if the NI can accept images taken from a LTF. The parameters for the test are Image from the LTF that will accept it and a boolean representing the LTF's connection to the network infrastructure. The test will create LTF and Network infrastructure objects and set the LTF connection status to the parameter value. It will attempt to send an image directly to the Network Infrastructure (NI) and store the outcome into the results. Lastly it will go through a final test if the image was properly received or accepted. If the final test shows a false boolean, that will show as an error and ends the test case with an error message "Test failed" and a false boolean result. If the test successfully passes then a true boolean value is returned with a "Test Passed" print statement.

Test:

```
bool testacceptImage(Image pic, bool niConnection)
{
        // create objects
        LTF ltf = new LTF();
        NI network = new NI();

        // set ltf connection status to parameter
        ltf.networkConnection = niConnection;
        // attempts to send an image directly to the NI
        ltf.sendToNI(pic);

        // store outcome
        bool result = network.acceptImage();

        // evaluate image acceptance
        if(result != true)
        {
        cout << “Test failed” << end;
        return false;
        }
        cout << “Test passed” << end;
        return true;
        }
```

Inputs & Outputs:

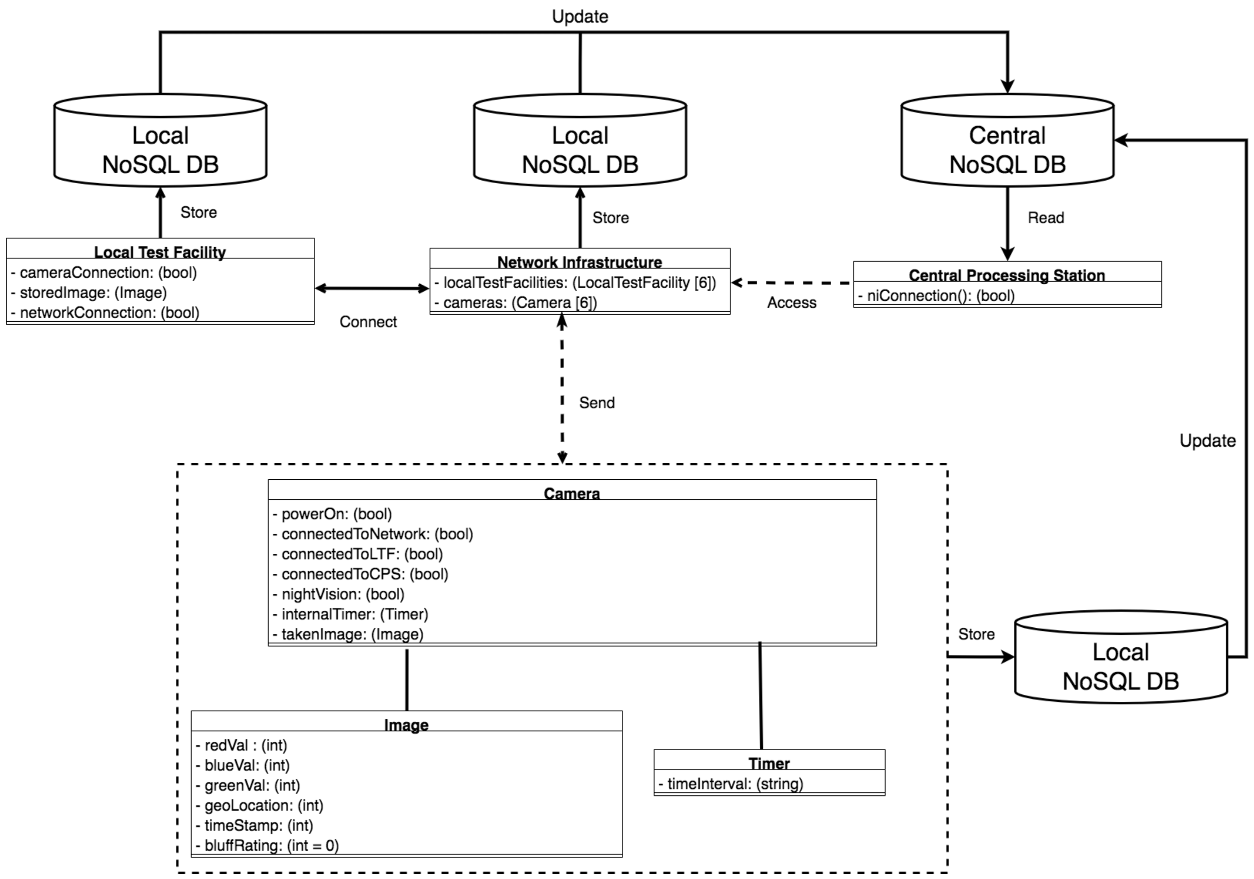
Input	**Expected Output**

acceptImage(Bluff, true)	cout << “Test passed” << end;
	return true

acceptImage(UnstableBluff, true)	cout << “Test passed” << end;
	return true

acceptImage(Bluff, false)	cout << “Test failed” << end;
	return false

h. Data Management Diagram

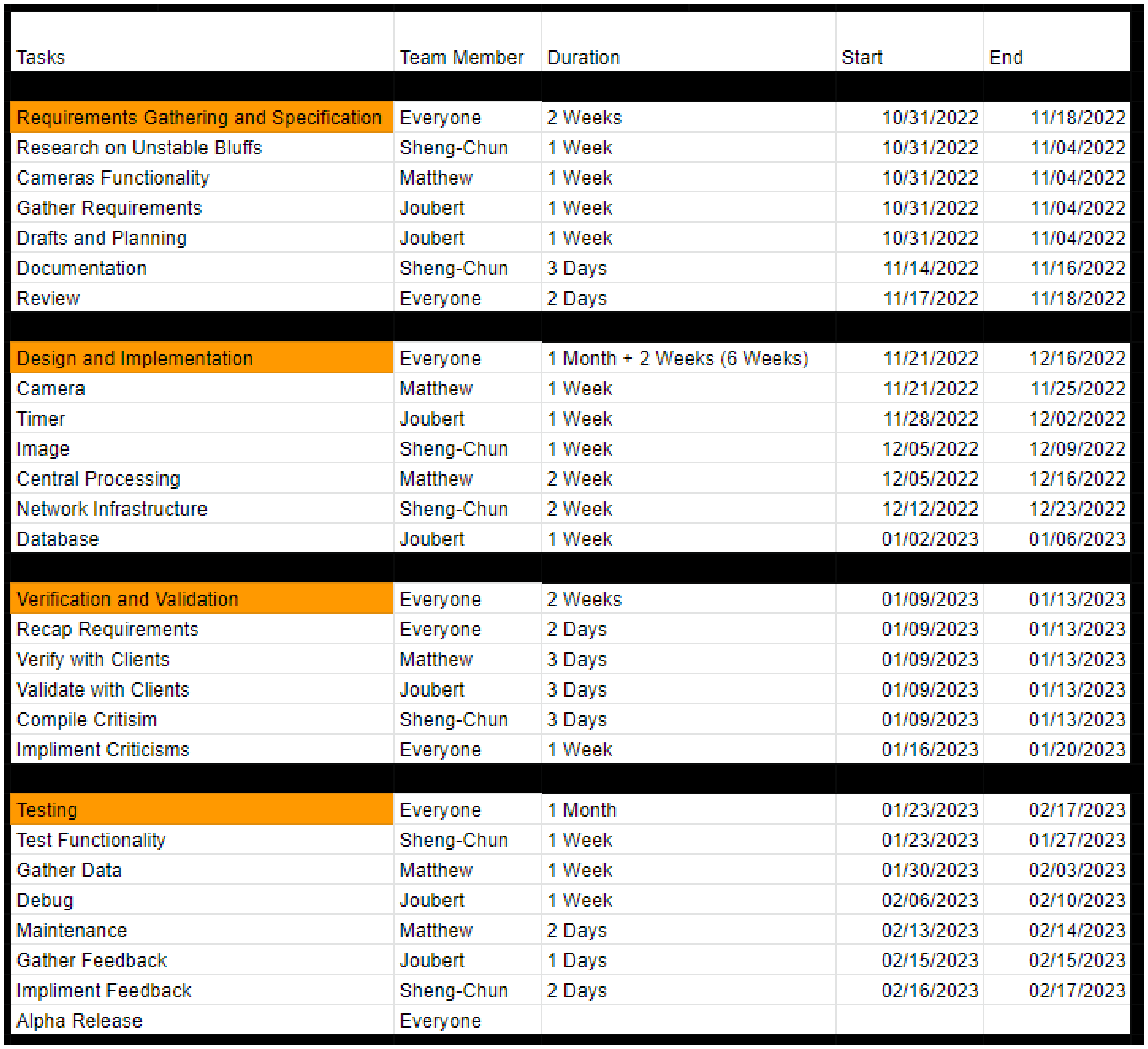


In our system, we distribute our NoSQL database to 3 local databases and 1 central database to improve the availability. The local database is responsible for temporarily storing the data generated from Camera, Local Test Facility, and Network Infrastructure. This data will periodically update to the central database, which is directly connected to the Central Processing Station. We establish this local-central database system to periodically backup the data collected from each component in the central database.

i. Future Features

Future changes may include compatibility for mobile devices which makes it more easier for users or administrators to access. It will affect the entire system which may require downtime or maintenance for a time. The administrator will have the ability to work remotely by controlling the system from a mobile device such as an iPad. We can assume that cameras might get damaged in the process of an unstable bluff abrupt or collapse at any time or any foul or malicious intent before authorities could come. We can add a feature that will alert administrators and authorities about the sudden damage to the cameras.

j. Timeline

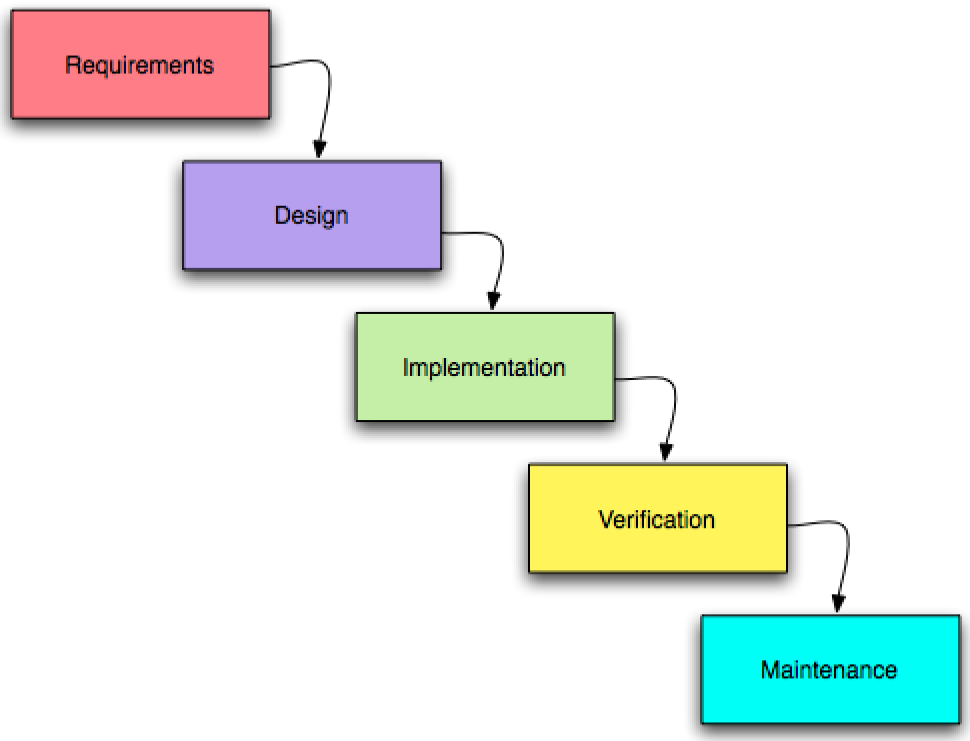


Tasks	Team Member	Duration	Start	End
Requirements Gathering and Specification	Everyone	2 Weeks	10/31/2022	11/18/2022
Research on Unstable Bluffs	Sheng-Chun	1 Week	10/31/2022	11/04/2022
Cameras Functionality	Matthew	1 Week	10/31/2022	11/04/2022
Gather Requirements	Joubert	1 Week	10/31/2022	11/04/2022
Drafts and Planning	Joubert	1 Week	10/31/2022	11/04/2022
Documentation	Sheng-Chun	3 Days	11/14/2022	11/16/2022
Review	Everyone	2 Days	11/17/2022	11/18/2022
Design and Implementation	Everyone	1 Month + 2 Weeks (6 Weeks)	11/21/2022	12/16/2022
Camera	Matthew	1 Week	11/21/2022	11/25/2022
Timer	Joubert	1 Week	11/28/2022	12/02/2022
Image	Sheng-Chun	1 Week	12/05/2022	12/09/2022
Central Processing	Matthew	2 Week	12/05/2022	12/16/2022
Network Infrastructure	Sheng-Chun	2 Week	12/12/2022	12/23/2022
Database	Joubert	1 Week	01/02/2023	01/06/2023
Verification and Validation	Everyone	2 Weeks	01/09/2023	01/13/2023
Recap Requirements	Everyone	2 Days	01/09/2023	01/13/2023
Verify with Clients	Matthew	3 Days	01/09/2023	01/13/2023
Validate with Clients	Joubert	3 Days	01/09/2023	01/13/2023
Compile Critisim	Sheng-Chun	3 Days	01/09/2023	01/13/2023
Impliment Criticisms	Everyone	1 Week	01/16/2023	01/20/2023
Testing	Everyone	1 Month	01/23/2023	02/17/2023
Test Functionality	Sheng-Chun	1 Week	01/23/2023	01/27/2023
Gather Data	Matthew	1 Week	01/30/2023	02/03/2023
Debug	Joubert	1 Week	02/06/2023	02/10/2023
Maintenance	Matthew	2 Days	02/13/2023	02/14/2023
Gather Feedback	Joubert	1 Days	02/15/2023	02/15/2023
Impliment Feedback	Sheng-Chun	2 Days	02/16/2023	02/17/2023
Alpha Release	Everyone			

This spreadsheet will keep track of the timeline for the dates and responsibilities of each team member. This will allow for the entire team and clients to be in track of how the team is going to progress throughout the following days.

3. Life-Cycle Model

a. Life Cycle Used



The Life Cycle that will be used for this project is the Waterfall Life Cycle Model. The Waterfall model is a linear sequence in which it starts from the top and steadily goes down as if it was an actual waterfall. This is the closet model we have done that reflected on our development throughout the project. We went from getting the requirements clearly from the clients in which it is described for the waterfall model. It

b. Pros

It emphasizes on a clear structure and requirements. The project we have is very clear and does not need any complex expertise or management for anyone or the project itself. There is a lot of documentation but this will aid in having a tracked and concise progress throughout the timeline. It will help to keep track of the timeline and get the most detailed information we can get from the clients. It is a straightforward model for this system as the requirements have been clearly stated. The system is not utterly complex and the requirements have been clearly stated.

c. Cons

No backtracking is one of the cons for this development. It will be difficult to change the system mid-way whenever the client wants to change somewhere or something for any of the phases above. Fixed requirements is also a con, as we mentioned earlier it will become difficult to change the system mid-way or even add something, having to add, restart, or redo the entire system from scratch will result in negatively for the team.

d. Improvements on Life Cycle

Add backtracking or a verification for each phase before tackling the next phase to ensure it will work. It will definitely be hard when the client changes their mind midway through the development phase which may result in the possibility of going back to square one. If we were able to get the ability to backtrack to add or remove something in any of the phases that would make the life cycle much better, especially for other projects. We can also have a much stronger testing phase to ensure that the system works perfectly. This model does not entirely offer a solid testing phase so adding more resources in that phase would make the quality of the system better.