ChangeLog

Version	Change Date	By	Description
V 1.1	11/01/2021	Chavey Villegas	- Storage in database - Interval timer going off, capturing, and running algorithm, saving image to DB
V 1.2	11/02/2021	Eddie Coda	- Image sent to algorithm for editing, geolocation, time, Score #, then sent to database, also may alert authorities.
V 1.3	11/02/2021	Eric Phan	- Presentation of UI at central monitoring
V 1.4	11/02/2021	Samara Marrow	- Comprehensive Networking to and from

CONTENTS

1 Introduction

Brief introduction of the test strategies, process, workflow and methodologies used for the project

1.1 Scope

1.1.1 In Scope

The BDPS *Test Plan* defines the unit, integration, system, regression, and Client Acceptance testing approach. The test scope includes the following:

- Testing of all functional, application performance, security and use cases requirements listed in the *SDS* document.
- Quality requirements and fit metrics for the BDPS
- End-to-end testing and testing of interfaces of all systems that interact with the BDPS.

1.1.2 Out of Scope

The following are considered out of scope for BDPS system Test Plan and testing scope:

- Functional requirements testing for systems outside BDPS
  - RedFlag Software
  - DynamoDB
- Testing of Business SOPs, disaster recovery, Business Continuity Plan, and Network failures

1.2 Quality Objective

- Ensure the Application Under Test conforms to functional and non-functional requirements
- Ensure the AUT meets the quality specifications defined by the client
- Bugs/issues are identified and fixed before go live

1.3 Roles and Responsibilities

Detail description of the Roles and responsibilities of testing team

- For User Acceptance testing, the Developer team has completed unit, system and integration testing and met all the Requirement's (including quality requirements) based on Requirement Traceability Matrix
- Test User Acceptance testing will be conducted by End-users

- Test results will be reported on daily basis using Gforge. Failed scripts and defect list from Gforge with evidence will be sent to Developer directly
- Use cases have been developed by Adopters for User Acceptance testing. Use cases are approved by test lead.
- Test scripts are developed and approved.
- Test Team will support and provide appropriate guidance to Adopters and Developers to conduct testing
- Major dependencies should be reported immediately after the testing kickoff meeting

Amongst others

2 Test Methodology

2.1 Overview

The test methodology selected for the project:

1. All tests will be buiilt in a deployable container, ran with example code and fence cases.
2. Contanier with tests should meet expexted confidence level based on report status.
3. Once confidence level is met, all systems tests inside of containers will be placed in a beta testing enviornment with sample system running.
4. Once Beta tests are met with passing confidence level, then tests will be deployed into a live environment (Gama tests) with system running to deliver final testing confidence levels.
5. Confidence levels will be based on this tier testing process and will ensure the highest quality of success and locate all chance for failures.
6. Building tests into containers will be the most important and effiecient feature of this test plan.

Understanding Requirements:

• Requirement specifications will be sent by client.
• Understanding of requirements will be done by QA

- Preparing Test Cases:

QA will be preparing test cases based on the exploratory testing. This will cover all scenarios for requirements.

- Preparing Test Matrix:

QA will be preparing test matrix which maps test cases to respective requirement. This will ensure the coverage for requirements.

- Reviewing test cases and matrix:

• Peer review will be conducted for test cases and test matrix by QA Lead
• Any comments or suggestions on test cases and test coverage will be provided by reviewer respective Author of Test Case and Test Matrix
• Suggestions or improvements will be re-worked by author and will be send for approval
• Re-worked improvements will be reviewed and approved by reviewer

Test Plan:

- Creating Test Data:

Test data will be created by respective QA on client's developments/test site based on

scenarios and Test cases.

- Executing Test Cases:

• Test cases will be executed by respective QA on client's development/test site based on designed scenarios, test cases and Test data.

• Test result (Actual Result, Pass/Fail) will updated in test case document Defect Logging Reporting:

QA will be logging the defect/bugs in Word document, found during execution of test cases. After this, QA will inform respective developer about the defect/bugs.

- Retesting and Regression Testing:

Retesting for fixed bugs will be done by respective QA once it is resolved by respective developer and bug/defect status will be updated accordingly. In certain cases, regression testing will be done if required.

- Deployment/Delivery:

• Once all bugs/defect reported after complete testing is fixed and no other bugs are found, report will be deployed to client's test site by PM.

• Once round of testing will be done by QA on client's test site if required Report will be delivered along with sample output by email to respective lead and Report group.

• QA will be submitting the filled hard copy of delivery slip to respective developer.

• Once lead gets the hard copy of delivery slip filled by QA and developer, he will send the report delivery email to client.

2.2 Bug Triage

Bug Triages will be held throughout all phases of the development cycle. Bug triages will be the responsibility of the Test Lead. Triages will be held on a regular basis with the time frame being determined by the bug find rate and project schedules.

Thus, it would be typical to hold few triages during the Planning phase, then maybe one triage per week during the Design phase, ramping up to twice per week during the latter stages of the Development phase. Then, the Stabilization phase should see a substantial reduction in the number of new bugs found, thus a few triages per week would be the maximum (to deal with status on existing bugs).

2.3 Suspension Criteria and Resumption Requirements

Suspension criteria define the criteria to be used to suspend all or part of the testing procedure while Resumption criteria determine when testing can resume after it has been suspended

2.4 Test Severity List

Severity ID	Severity	Severity Description
1	Critical	The module/product crashes or the bug causes non-recoverable conditions. System crashes, GP Faults, or database or file corruption, or potential data loss, program hangs requiring reboot are all examples of a Sev. 1 bug.
2	High	Major system component unusable due to failure or incorrect functionality. Sev. 2 bugs cause serious problems such as a lack of

		functionality, or insufficient or unclear error messages that can have a major impact to the user, prevents other areas of the app from being tested, etc. Sev. 2 bugs can have a work around, but the work around is inconvenient or difficult.
3	Medium	Incorrect functionality of component or process. There is a simple work around for the bug if it is Sev. 3.
4	Minor	Documentation errors or signed off severity 3 bugs.

2.5 Test Completeness

- 100% test coverage
- All Manual & Automated Test cases executed
- All open bugs are fixed or will be fixed in next release
- Confidence level above a 95% level.

3 Test Deliverables

- GUI Acceptance Tests
- Test Cases
- Functional/ Non-Function unit tests
- Bug Reports
- Test Metrics

4 Resource & Environment Needs

4.1 Testing Tools

No.	Resources	Descriptions
1.	Server	Need a DynamoDB server which install AWS EC2 server Web server which install Ubuntu Server
2.	Test tool	Develop a Test tool for container deployment which can auto generate the test result to the predefined form and automated test execution
3.	Network	Setup a LAN Gigabit and 1 internet line with the speed at least 1 GB/s. Setup Wifi in cameras
4.	Computer	At least 4 computer run Windows 10, Ram 2GB, CPU 3.4GHZ

-
- GForge bug tracker is used by caBIG to enter and track all bugs and project issues. The Test Lead is responsible for maintaining the GForge database.
- Requirements Tracking Tool

- Automation Tools

4.2 Test Environment

It mentions the minimum **hardware** requirements that will be used to test the Application.

Following **software's** are required in addition to client-specific software.

- Windows 10 and above
- Office 2016 and above
- MS Exchange, etc.

5 Testing Examples

5.1 GUI Tests

Summary: Ideal login scenario where the user email account is valid within the system's user directory. The email and password used to log in are valid and the user is granted access to the Bluff Detection System application.

Test Scenario ID	Auth-login-1	**Test Case ID**	Auth-login-1A
Test Case Description	User is authenticated and able to log in.	**Test Priority**	High
Pre-Requisite	Valid user account within system directory.	**Post-Requisite**	N/A

S.No	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Browser**	**Test Result**
1	Launch application	Desktop "Bluff Detection System" Application	Web Application Login	Web Application Login	Chrome Version 95	Pass
2	Enter email and password, press login button	Email: systemuser@bdps.com Password: Test123!	Login Success	Login Success	Chrome Version 95	Pass
3						

Summary: Invalid login scenarios where the user account does not exist within the system's user directory, or an incorrect password is used for a user account. Number of attempts are limited to a total of 6, system administrator is notified with an audit log when the number of attempts reach 0. System administrator will have to re-enable ability to log in as a security measure.

Test Scenario ID	Auth-login-1	**Test Case ID**	Auth-login-1B
Test Case Description	User is not able to be authenticated to the system.	**Test Priority**	High
Pre-Requisite	N/A	**Post-Requisite**	N/A

S.No	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Browser**	**Test Result**
1	Launch application	Desktop "Bluff Detection System" Application	Web Application Login	Web Application Login	Chrome Version 95	Pass

2	Enter invalid email and password, press login button	Email: unknown@gmail.com Password: myPass123	User account does not exist in directory. **X** attempts remaining.	User account does not exist in directory. 5 attempts remaining.	Chrome Version 95	Pass
3	Enter valid email and incorrect password	Email: systemuser@bdps.com Password: incorrectPass123!	The password that you've entered is incorrect. **X** attempts remaining.	The password that you've entered is incorrect. 4 attempts remaining.	Chrome Version 95	Pass

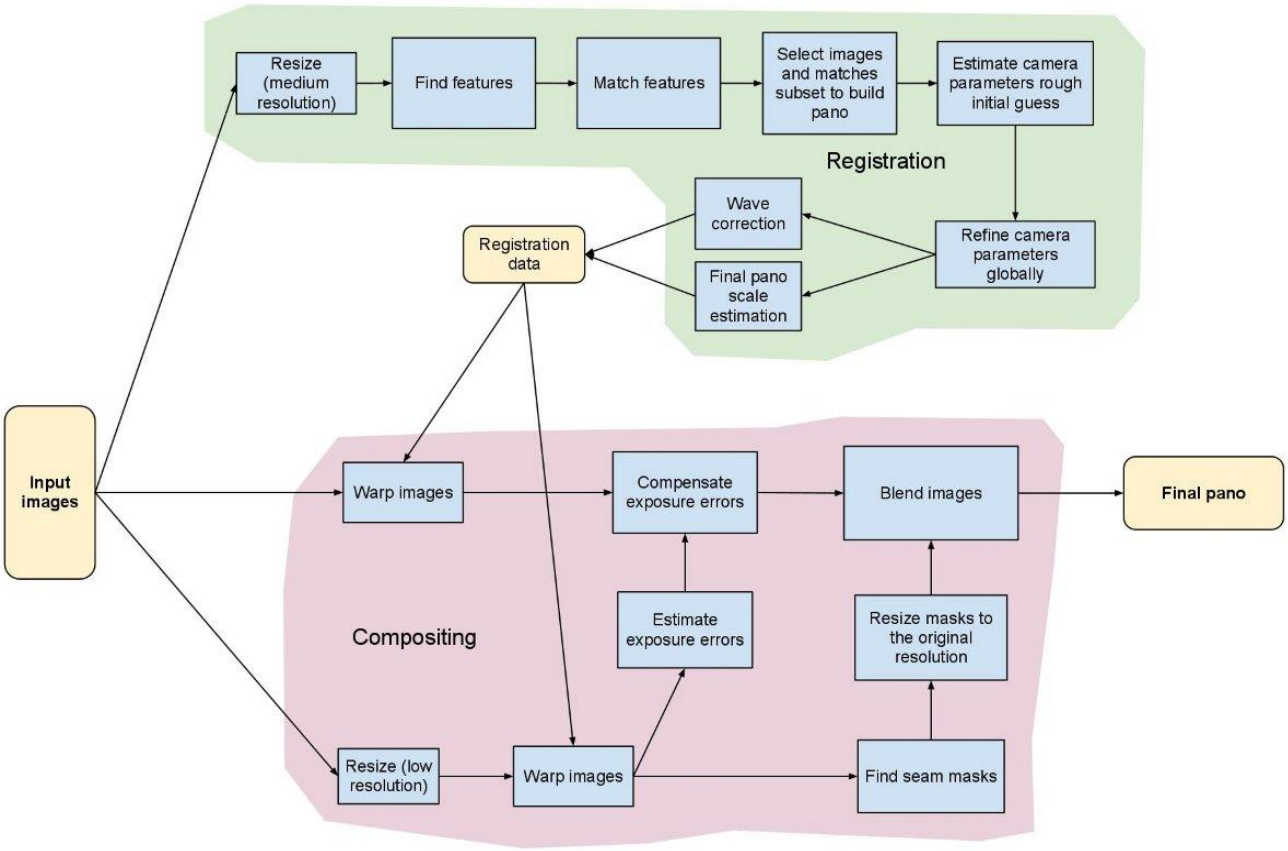
Summary: Authenticated user accessing the system can edit the interval of each camera if it is connected to the system. In this case, Camera 3 is used as an example.

Test Scenario ID	Overview-1	**Test Case ID**	Overview-1A
Test Case Description	Authenticated user intends to modify a camera's interval.	**Test Priority**	Medium
Pre-Requisite	Valid user within system directory, intended camera connected.	**Post-Requisite**	N/A

S.No	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Browser**	**Test Result**
1	Select camera on "Overview" page.	Click "Camera **X**"	Camera **X** menu expands with options	Camera 3 menu expands with options	Chrome Version 95	Pass
2	Press "Edit Interval" button	Click "Edit Interval"	Popup window with input field for Camera **X**'s new interval timer	Popup window with input field for Camera 3's new interval timer	Chrome Version 95	Pass
3	Enter new timer interval, press submit button	Interval input: 00h 30m 00s Press "Submit" button	New interval on Camera **X**'s menu set to "00h 30m 00s"	New interval on Camera 3's menu set to "00h 30m 00s	Chrome Version 95	Pass

5.2 Smoke Tests – encapsulate functionality of algorithms

This figure illustrates the stitching module pipeline implemented in the **Stitcher** class. Using that class it's possible to configure/remove some steps, i.e. adjust the stitching pipeline according to the particular needs. All building blocks from the pipeline are available in the detail namespace, one can combine and use them separately.



Picture Algorithm Visual Regression Test Code examples: Image Present or not?

Sometimes we must take care of the occurrences of a particular kind of image on the website. So, in such case, we must validate whether that image is present there or not.
Let's have a look at its implementation.
At first, we will take WebElement of the image and then we will create a Boolean form of the JavaScript. In the following sample program, we just verify whether the logo image is present on Inviul or not.

```
public class ImagePresent {
     WebDriver driver = null;

  @Test
  public void imagePresentProcess() throws InterruptedException {
       System.setProperty("webdriver.chrome.driver",
"C:\\Selenium\\chromedriver.exe");
       driver = new ChromeDriver();
       driver.get("https://www.inviul.com");
```

```
        driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);
        driver.manage().window().maximize();

        //Get WebElement reference of logo
        WebElement logoElement =
driver.findElement(By.xpath("//h1[@id='logo']//a//img"));

        Thread.sleep(3000);

        JavascriptExecutor js = ((JavascriptExecutor)driver);
        Boolean imgPresent =  (Boolean) (js.executeScript("return
arguments[0].complete && typeof arguments[0].naturalWidth !=
\"undefined\" && arguments[0].naturalWidth > 0", logoElement));

        if(imgPresent==true){
             System.out.println("Image is present");
        }else{
             System.out.println("Image is not present");
        }

        driver.close();
        driver.quit();
  }
}
```

Picture Algorithm – Set Picture min accuracy %
Description:
Specify the minimum level of accuracy required in matching baseline image objects with objects in captured test images for a match to be considered valid. Applies only to keypoint detection comparisons.
Arguments

- **value**

Minimum acceptable accuracy (units: percentage; valid range: 0 to 100; default = 50).
Valid contexts
This action may be used within the following project items: test modules and user-defined actions.
Notes

- Minimum accuracy in picture matching may also be set through the built-in minimum accuracy setting.
- This built-in action is only applicable to a keypoint detection technique for image recognition. Therefore, it is only valid when the built-in setting picture algorithm is set to key point, and applies to all picture handling built-in actions except check picture.
- When this built-in action executed, its value overrides the **Min Accuracy (%)** value specified in the Key Points Modification Tool dialog box.

- Android: This built-in action, when applied to a connected Android device, requires that the Test Architect be running. Also note that, when an Android device is restarted, Test Architect Agent is then stopped. Should this be the case, it is essential that you reactivate the service by observing the following steps:
    1. Connect the Android device to the test controller through a USB cable (not Wi-Fi), if not already so connected.
    2. Open the dialog box.
    3. Click the **Refresh devices list** button.
- This action supports the <ignore> modifier. If the string <ignore> is present as the value of the argument, or the argument contains an expression that evaluates to <ignore>, the action is skipped during execution.

Example

Action Lines

```
var imgCompare = new AWS.Compare();
Compare.compareimg(params, function (err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

6						
7	INITIAL	Setting up				
8						
9	//Perform a Key Point Detection technique comparison.					
10		setting	value			
11	setting	picture algorithm	key points			
12						
13	TEST CASE	TC 01	Check the existence of a picture			
14						
15		value				
16	set picture min accuracy	60				
17						
18		picture	window	control	rect	index
19	check picture exists	/rubik	home page			
20						

Picture Algorithm – Key point detection

Description:

Specify which algorithm is applied to perform comparisons between stored baseline images and images under test.

Allowable values:

- **exact**

Perform a pixel-by-pixel comparison.

- **key points**

Perform a keypoint detection comparison.

Default value:

- exact

Notes

- The exact value for the picture algorithm built-in setting (specifying the pixel-by-pixel comparison technique) applies to nearly every picture handling built-in action. The sole exception is set picture min accuracy, which does not interact with the AUT.
- The key points value for the picture algorithm built-in setting (specifying the keypoint detection technique) applies to every picture handling built-in action but two: check picture is exempt, since it always applies a pixel-by-pixel comparison, and set picture min accuracy, as mentioned, has no interaction with the AUT.

6 Example

Test Lines

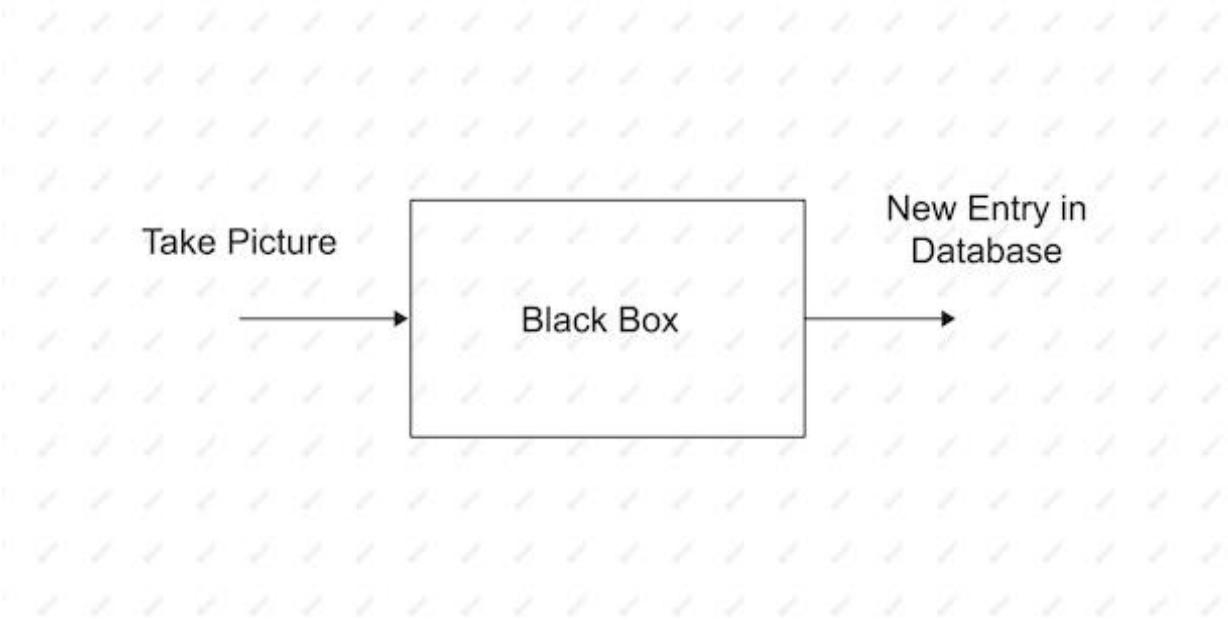
6						
7	INITIAL	Setting up				
8						
9	//Perform a Key Point Detection technique comparison.					
10		setting	value			
11	setting	picture algorithm	key points			
12						
13	TEST CASE	TC 01	Check the existence of a picture			
14						
15		picture	window	control	rect	index
16	check picture exists	/rubik	home page			
17						

6.1 Database Stress tests

Sending image to DB

Black Box

System testing



This system level test validates that the software system takes a picture which results in a new entry into the database with the correct information attached. The picture should include the timestamp, camera number, and the geological location of where the picture was taken. This test should be performed at least twice, one with gray scale turned on and one with gray scale turned off. The result should have a picture in color if gray scale is off and in gray scale if it is turned on.

DB Functions
White box
Unit testing

Create

```
bool testCreateTable (string camNum, string timeStamp, string
geoLocation, string imageURL, string photoKey){
    if(createTable(camNum, timeStamp, geoLocation, imageURL,
photoKey);){
        return true;
    }
    else{
        return false;
    }
}
```

This unit test is designed to test the creation function in the database. It takes the camera number, timestamp, geological location, image URL, and a photo key as inputs. It then passes those arguments into the createTable function which returns a boolean. If the creation is successful, a new table will be created with the arguments passed in as its attributes and the test will return a true statement. If the creation is not successful, the test will return a false statement.

Read

```
string testReadTable(string photoKey){

    return readTable(photoKey);

}
```

This unit test is designed to test the readTable function of the database. It takes the photo key as an input which then passes it to readTable. The expected result is the contents of the table in JSON format.

Update

```
bool testUpdateTable(string camNum, string timeStamp, string
geoLocation, string imageURL, string photoKey){

    if(updateTable(camNum, timeStamp, geoLocation, imageURL,
photoKey)){

        return true;

    }

    else{
```

```
        return false;

    }

}
```

This unit test is designed to test the updateTable function of the database. This test takes the camera number, timestamp, geological location, image URL, and a photo key as inputs. It then passes those arguments into the updateTable function. The expected outcome is the attributes will change to the arguments being passed in the table selected by the photo key. If this is successful, the test function will return true. If the update is not successful, the test will return false.

Delete

```
bool testDeleteTable(string photoKey){

    if(deleteTable(photoKey)){

        return true;

    }

    else{

        return false;

    }

}
```

This unit test is designed to test the deleteTable function of the database. This test takes a photo key as an input. The expected result is that the table with the corresponding photo key should be removed from the database. If this is successful, the test function will return true. If it is not successful, the test will return false.

A significant amount of overhead could be involved to determine the state of the database transactions

- **Solution:** The overall process planning, and timing should be organized so that no time and cost-based issues appear.

New test data must be designed after cleaning up of the old test data.

- Solution: A prior plan and methodology for test data generation should be at hand.

An SQL generator is required to transform SQL validators to ensure the SQL queries are apt for handling the required database test cases.

 Solution: Maintenance of the SQL queries and their continuous updating is a significant part of the overall testing process which should be part of the overall test strategy.

The above-mentioned prerequisite ensure that the set-up of the database testing procedure could be costly as well as time consuming.

 Solution: There should be a fine balance between quality and overall project schedule duration.

6.2 Send/ Receive Integration tests

```
alias: Detect Del Mar Bluff Change
description: interval timer goes off
mode: single
Trigger:image sent to database
  - platform: event
    event_type: SurveillanceCamera_capture
condition: []
action:
  - service: notify.notify
    data:
      message: Alert,a change has occurred (Rate:..)
      title: Bluff change has been detected
Rating of % has been detected, this function is sending notification to user
```

TC ID	TC Name	Description	Steps	Expected Result
TC1	Timer goes off	Surveillance camera detects bluff change, captures image of change along with a rating to go with it	1. Motion detected 2. Images captured and rated 3. Surveillance cameras will transmit Images through wifi to AWS cloud through the Amazon API Gateway and AWS Lambda function 4. SNS will be sent out 5. Lambda function interacts with DynamoDB and returns a response to API Gateway	Bluff detection notification with full access to details is sent to user

error	Output
Not allowed	Returns the message, a short explanation of the issue.
Set time interval to 2600	Error message returned to pick a valid time interval
Image sent unsuccessful	Forward back a message to central monitoring
Invalid number	Returns the code that is mapped to the error message
Capture not taken	Post a warning message
No response from camera	{ "status" : 403, "errorCode" : "901", "message" : "invalid connection", "errorType" : "connection" }

Request is being sent to capture the image, some possible invalid responses that might occur include the image not being sent successfully, a wrong time interval might be requested, capture not correctly taken, lost response from surveillance camera. By default, if the property operations cannot find or access the property to be compared, it waits for it for some period of time. Once time is up, the operation fails to complete.

```
function handler(Bluff Detection) {
   var request = event.request
   return request;
}
```

7 Terms/Acronyms

Make a mention of any terms or acronyms used in the project

TERM/ACRONYM	DEFINITION
API	Application Program Interface
AUT	Application Under Test
CAT	Client Acceptance Testing
End-to End Testing	Tests user scenarios and various path conditions by verifying that the system runs and performs tasks accurately with the

TERM/ACRONYM	DEFINITION
	same set of data from beginning to end, as intended.
ER;ES	Electronic Records; Electronic Signatures
N/A	Not Applicable
QA	Quality Assurance
RTM	Requirements Traceability Matrix
SME	Subject Matter Expert
SOP	Standard Operating Procedure
TBD	To Be Determined
TSR	Test Summary Report