Software Test Document

AJE Fire Detection System

Version 1.0

11/10/2018

AJE

Bellevue College Department of Computer Science

Anis Beyzaee, Joey Colombi, Ephraim Scarf

Alfred Nehme

Revisions Page

Date	Version	Changes
11/20/2018	1.0	Creation of document
11/30/2018	1.1	Adding new test cases

1. Introduction

1.1 Purpose

This document describes the tests needed to ensure the functionality of the AJE fire detection system. It is intended to be read by the testing staff and developers of the AJE system.

1.2 Scope

This document goes over the test plan for the AJE, covers several test cases about the functionality of the system and the requirements traceability matrix. Test cases which are related to the performance and functional requirement are explained in this document.

1.3 Test Approach

For This project two methods of testing are used:

Proactive: some designed cases, mostly functionalities of methods and functions, are to be prepared ahead. Test process will be performed by developers as coding process is done. Unit testing in this phase will be applied. Each test case will be created and each module will be tested against the expected output. The goal in this phase is to make sure each unit is working properly as an individual and won't have any effect to the whole system.

Reactive: other test cases, functional and requirement related, which are about errors, are done after the code is written and project will be tested against that. The goal is for the whole project to meet the requirements. The goal for this test approach is to make sure the whole system is capable to output correct result in expected time.

1.4 References

- AJE Fire Detection SDD
- AJE Fire Detection SRS

2. Test Plan

2.1 Features to be tested

- Test whether or not the system can handle a variety of image sizes
- Test whether or not the system can determine if a given image has a fire within 5 minutes
- Test how accurately system classifies images from the test dataset.
- Test how the system behaves when it detects a fire

2.2 Features not to be tested

- We will not test whether or not the image can successfully classify images delivered from a satellite feed, since the API that would facilitate this delivery is not available to us yet.

2.3 Testing Tools and Environment

- JUnit, Eclipse, Intellij
- Microsoft Windows, Linux, macOS

3. Test Cases

3.1 Test Case - 1 Convert image sizes to one fixed image size

3.1.1 Purpose: testing the output of Image Adaptor unit

3.1.2 Input: Image of bigger size(bigger than defined size for the system)

3.1.3 Expected Output uniformly sized image

3.1.4 Pass/Fail Criteria: image is not the exact size will be considered as fail, pass otherwise

3.2 Test Case - 2 Image Detection Efficiency

3.2.1 Purpose: Testing the efficiency of the detection module

3.2.2 Input: An image from the testing dataset

3.2.3 Output: A notification of the image being declared as a fire or not a fire

3.2.4 Pass/Fail Criteria: If a notification is delivered within 5 minutes from being put into the system, it will pass. Otherwise, it will fail.

3.3 Test Case - 3 Accuracy of the detection system

3.3.1 Purpose: To evaluate how accurate the system is at correctly predicting fire/no fire images

3.3.2 Input: A testing dataset of forest images

3.3.3 Expected Output: A percentage of how many images were correctly identified

3.3.4 Pass/Fail Criteria: If the percentage from the output is greater or equal to 90%, it will pass. Otherwise, it will fail.

3.4 Test Case - 4 GUI Fire Alert

3.4.1 Purpose: To verify that the GUI displays a warning to the user if the application classifies an image as having a fire.

3.4.2 Input: An instance of Image that has a fire

3.4.3 Expected Output: The GUI module should display a warning message to the user, informing them of the fire, and presenting them with any information associated with the image.

3.4.4 Pass/Fail Criteria: Test passes if the GUI displays the appropriate warning message with information related to the image (if there is any), fails otherwise.

3.5 Test Case - 5 Automatic Restart

3.5.1 Purpose: To verify that the application starts automatically after the host operating system shuts down.

3.5.2 Input: Shutting down the host operating system, while the application is already running

3.5.3 Expected Output: When the host computer is restarted, the AJE system should appear under the operating system's list of running services.

3.5.4 Pass/Fail Criteria: Test passes if the GUI displays if the service associated with the AJE system appears under the operating system's list of running service's.

4. Requirements Traceability Matrix

Requirement -ID	Requirement Description	Design Component	Data Design Component	Interface Design Component	Test Case #
3.3.2	The system shall have an accuracy rating of at least 90% when detecting a fire	Network Module	Application Module	N/A	3.3
3.3.1	The system shall be able to determine whether a given image is a fire/no fire in 5 or less minutes	Detection Module	Application Module	N/A	3.2
3.2.4	The system shall be able to take images with a variety of sizes	Image Adaptor Unit	Application Module	N/A	3.3

	and convert it to 1 fixed image to be used throughout the system				
3.2.3	The system shall send a notification to the admin when it recognizes a fire in the image given	Network Module	Application Module	GUI	3.4
3.2.5	System runs as a service on host computer	AJE App	Application Module	N/A	3.5
3.2.6	System automatically starts when host computer restarts	AJE App	Application Module	N/A	3.5

5. Responsibilities:

Anis Beyzaee: Test Case 1.

Joey Colombi: Test Case 2 and 3.

Ephraim Scarf: Test Case 4 and 5.

6. Staffing and training needs:

Testing staff should have access to and familiarity with both a Java testing framework such as JUnit 5, and a Java IDE (integrated development environment) such as IntelliJ

7. Schedule:

Testing should be completed by February 11 2019, the 6th week of the Winter Quarter.