TLDWorkerBee

Team: Austen Christensen, Morgan Lovato, Wei Song
Sponsor: Harlan Mitchell - Honeywell
Mentor: Austin Sanders

Software Testing Document - Version 1

April 5, 2019

Table of Contents

1. Introduction

Planes are the most popular way to travel quickly to anywhere around the world. For instance, a traveler can get from Phoenix to Los Angeles in a little over an hour by plane when it takes 8+ hours to drive. In order to travel such long distances, these machines must maintain a cruising altitude of 33,000 to 42,000 feet. Every day, there are over one hundred-thousand flights scheduled across the globe with up to one million people in the air at any given point in time. This makes it vital for every plane's engine to constantly be working properly since there is nothing stopping a plane from falling out of the sky other than it's own momentum.

Our team is TLD Worker Bee, and we are working on the project Prototype Time Limited Dispatch (TLD) Application for our sponsor, Harlan Mitchell from Honeywell Aerospace. Honeywell Aerospace is a leading manufacturer of all sorts of aircraft engines ranging from helicopters to commercial airliners. They are the leading manufacturer in engine control systems for a variety of private and commercial jets. These engines and their connected systems generate data every flight that is important for the functionality of their product. While in flight, an engine is constantly reading sensor data and storing it on the onboard computer called the Engine Control Unit(ECU). The computer will read the data and create a time-limited dispatch; time-limited dispatch allows the degraded redundancy dispatch of aircraft. Aircraft can be dispatched with certain control system faults and fault combinations for specified periods of time if the failure rates from those configurations meet certification requirements. The various system faults and fault combinations are assigned to dispatch categories according to these failure rates. This gives the dispatch criteria for the system.

Currently, to gather this data, a technician will physically download the data from the ECU through a wire connection. They do not check it after every flight, but will connect periodically to the ECU to retrieve the data. The cumbersome process of physically connecting to a computer and downloading this data on location greatly limits the amount of flight data to collect.

Our prototype is a webapp that uses an internet connection to connect to the data stored in the cloud for a completely wireless experience. It verifies data integrity before showing the user any data to avoid reading false data. This ensures the mechanic knows exactly what maintenance to perform on the engine from anywhere in the world.

This document lays out our testing plan for our software.

2. Unit Testing

Unit testing is a software testing method. It test the individual units or components, which is the smallest part of the software, in order to validate that each unit of the software performs as designed. It is the first level of software testing during the whole implementation cycle, and it's performed prior to integration testing or other testings.

There are many benefits for unit testing. The developers will be able to catch any defects or bugs promptly. The codes need to be modular in order to perform the unit testing, which makes the code more reusable.

Since our project is written in Python, we will be using pytest for unit testing. The pytest framework is a mature full-featured Python testing tool, which support both simple unit testing or complex functional testing for applications and libraries. It also support parameterization to manage the test cases.

Below are the detailed testing plan, which include a list of units for our programs, equivalence partitions, boundary values, erroneous values, and selected inputs for those partitions.

Unit 1: signUp(String Username, String Password, String ComfirmPW, String Email)

Test Item	Test Data	Test Type	Result
Username	David	Normal	Success (when other test items are correct)
	David	Erroneous (already existed in the database)	Error: username existed!
	A_very_long_username_which_exceed_20_characters	Erroneous (username too long, 20 maximum)	Error: username too long, 20 maximum!
	112233	Erroneous (should contain at least 1 character)	Error: username should contain at least 1 character
	Null	Erroneous (username should not be empty)	Error: username should not be empty

Password	SFhafwo124*@#	Normal	Success (when other test items are correct)
	So124*@	Erroneous (too short)	Error: password must contain at least 8 character
	12345678	Erroneous (too simple)	Error: password must contain at least 1 upper case letter, 1 lower case letter and one special character
	Null	Erroneous (password should not be empty)	Error: password should not be empty
Confirm Password	SFhafwo124*@#	Normal	Success (when other test items are correct)
	SFhafwo248*@#	Erroneous (Not match)	Error: The two password fields didn't match.
Email Address	aa123@nau.edu	Normal	Success (when other test items are correct)
	aa123	Erroneous (Not an email address)	Error: Not an email address
	aa123@nau.edu	Erroneous (Already existed in the database)	Error: Email address already existed
	Null	Erroneous (Email address should not be empty)	Error: Email address should not be empty

Unit 2: signIn(String Username, String Password)

Test Item	Test Data	Test Type	Result
Username	David	Normal	csfr token (when other test items are correct)

	David2	Normal	Username or Password not correct
	Null	Erroneous (username should not be empty)	Error: username should not be empty
Password	SFhafwo124*@#	Normal	csfr token (when other test items are correct)
	SFhafwo248*@#	Normal	Username or Password not correct
	Null	Erroneous (password should not be empty)	Error: password should not be empty

Unit 3: getAircraft(String Aircraft_ID)

Test Item	Test Data	Test Type	Result
Aircraft ID	12345	Normal	Aircraft Panel
	12346	Erroneous (aircraft not existed)	Error: Aircraft not existed
	12347	Erroneous (current user do not own this aircraft)	Error: Current user do not own this aircraft
	Ahiwhfeo	Erroneous (not a legal aircraft ID)	Error: Not a legal aircraft ID
	Null	Erroneous (aircraft ID should not be empty)	Error: Aircraft ID should not be empty

Unit 4: getChartView(String Aircraft_ID, String Search_Field)

Test Item	Test Data	Test Type	Result
Aircraft ID	The same as the Unit 3 as above		
Search Field	Event 1 Oil Pressure	Normal	Oil Pressure Data

	Plane Color	Erroneous (Do not have this data field)	Error: Do not have this data field
	Null	Erroneous (Search Field should not be empty)	Error: Search Field should not be empty

Unit 5: getTableViewData(String Aircraft_ID)

Test Item	Test Data	Test Type	Result
Aircraft ID	The same as the Unit 3 as above		

Unit 6: MD5Generator(String TLD_Data)

Test Item	Test Data	Test Type	Result
TLD Data	(String)	Normal	MD5 value
	Null	Erroneous (TLD Data should not be empty)	Error: TLD Data should not be empty

Unit 7: MD5Checker(String localMD5, String cloudMD5)

Test Item	Test Data	Test Type	Result
MD5 Pairs (Local MD5, Cloud MD5)	(MD5, MD5)	Normal	True / False
	(Null, MD5) / (MD5, Null) / (Null, Null)	Erroneous (MD5 should not be empty)	Error: MD5 should not be empty

Unit 8: parsingTool(File RawDataFile.txt)

Test Item	Test Data	Test Type	Result
Raw Data File	Datafile 1	Normal	Success
	Datafile 2 (without block num)	Erroneous (without block num)	Error: Raw datafile incomplete
	Datafile 3 (TLD data record incomplete)	Erroneous (TLD data record incomplete)	Error: Raw datafile incomplete

	Datafile 4 (Damage, or not txt file)	Erroneous (File damage)	Error: Raw datafile incomplete

3. Integration Testing

Integration testing is the testing of how individual parts of a program work together; the components tested in the unit testing are tested to see how they perform as a group. Integration testing is performed after the completion of unit testing and before the start of usability testing. The goal of integration testing is to find any problems with how units interact when they are integrated as a whole. Our team wants to ensure our front end and back end communicate properly in our final program. We are testing all module connections and that our data accuracy does not get compromised when the program is tested as a whole.

To ensure our front end and back end communicate properly, we would test to ensure Bootstrap (front end) and Django (back end) work together. To test the connection between the cloud and the back end, we test that the request function that gets the raw data from the cloud is working properly. To ensure data accuracy does not get compromised we compare the local and cloud MD5 values.

4. Usability Testing

Aircraft Technicians will utilize the frontend to visualize patterns and trends in engine activity from the Engine Control Unit to determine what type of maintenance to perform. This requires that our frontend is vetted and conforms to UI/UX best practices. Usability testing will strengthen our design by providing insight from a larger pool of users.

Population

The population we are meant to satisfy is certified aircraft technicians. This means we can assume that our audience has some level of technical expertise. The population will know the data being displayed very well and will know if something is misplaced.

Methods

Our testing plan will comprise two separate parts:

- Categorical Acceptance: User will be given flashcards with two different categories written on them. They will be asked to match them according to what they find most logical.

- Live Usability: We will record users interacting with our application given a testing script. The user will be asked to talk out loud about how they think as they are interacting with the application.

Plan

We plan on selecting a sample of 10-15 individuals, all who are certified aircraft technicians, to test our program. If they aren't certified, it will be difficult for the users to give us useful information and would be inaccurate to the intended audience of our frontend. We will perform this test on users individually and start them with the categorical acceptance test. The categorical acceptance test will help us determine how the user will think about our application prior to ever seeing it. We will use it to determine the most logical layout of our elements as well as an appropriate color palette. For this test, we will ask the user to match items from the left column with items in the right column:

Category

Table View	Top
Graph View	Middle
Raw Data Download	Bottom

Color (color cards can be matched to multiple items in right column)

Blue	Aircraft Number
Green	PASS
Red	FAIL
Black	Refresh Button
Purple	Graph Background
White	Line on Graph

After completing this test, users will be provided a testing script so they can complete a live usability test. The script will include the following steps:

1. Create an account using access code.
2. Go to aircraft 15672 and view the table view.

3. While in table view, go to block 27 and read the data. How easy was it to find on a scale to 1-10? What would you change and what did you like? Was it easy to read? Was it accurate?
4. Repeat step 3 looking at blocks 16 and 11.
5. Now go to the graph view for aircraft 16723.
6. Look at the oil temperature over time in block 27. Does the graph line up with data you saw in step 3? If no, how is it off?
7. Download the raw data file for aircraft 15672.
8. Access that file using the EEI software. Does the data for block 27 match up for both programs?
9. Logout.

While completing this script, we will record the user so that we can see how they interact with the application. This will help us determine how easily our items were to find, as well as show us where users thought these items should be. This will give us a clear indication as to whether or not our application will require any reorganization or additional visual cues.

Results

Once our focus group has completed their individual testing sessions, we will compile the data and see if we can identify any important patterns. We will do so by creating a table representing commonality among all users. For example, if 70% of users misidentified item 3 in our testing script, we will reorganize our layout to resolve this. Similarly, if 70% of users chose Green as the color that best represented the PASS category, we will adjust our color palette to match. These tests will help lay out the groundwork for any reorganization (both design and layout-wise) that will improve our application's overall usability.

5. Conclusion

With aircrafts being a necessary source of travel for some and a leisure for others, it is vital to human safety that they are always functioning properly. Our client, Honeywell, produces many aircraft engines and it is their job to make sure they are always in working order. If something does go wrong, technicians need to be able to quickly identify and fix a problem. Our prototype web app is our client's way of quickly and easily identifying a problem with and aircraft and this web app should be working without bugs or any data inaccuracies. To ensure there are no bugs or data inaccuracies in our web app, our team did multiple types of testing to ensure the integrity of our program. Our unit testing ensures different components of our program

work individually and do not fail with unknown circumstances. The integration testing ensures these components perform together without error or unknown results and that our front end and back end are communicating properly. Usability testing for this program ensure the aircraft technicians that need to use this web app can do so without confusion or a heavy lesson in how to navigate first. Our sponsor, Harlan Mitchell, is happy with our current program and is pleased with the accuracy it provides. He feels we have adequately done the job of allowing aircraft technicians to view accurate, real time data about aircraft engines.