

**System Test Report**

**Senior Project**

Tech Tutor

**Version**

1.0

**Product Owner**

Mathew Brimberry

San Juan Unified School District

**Team Byte Brigade**

Angelo Karam

Joshua Grindstaff

Ryan Naveira

Austin Nolte

Jacob Sherer  
 Collin Dunkle

Joaquim Pedroza

Chisom Iwunze

**TABLE OF CONTENTS**

[Testing Environment Setup 2](#_Toc602455645)

[Record 1.0 3](#_Toc1027172501)

[Record 2.0 4](#_Toc825121928)

[Record 3.0 5](#_Toc203966557)

[Record 4.0 7](#_Toc2085136858)

[Record 5.0 10](#_Toc1937990953)

[Record 6.0 13](#_Toc1747199016)

[Record 7.0 17](#_Toc636206343)

[Signatures 21](#_Toc24640839)

# Testing Environment Setup

To setup the test environment, make sure that you have access to the repository Tech Tutor Github repository, have a working IDE, have python installed, and have a google service account that has access to the Google Sheet API and Google Drive API.

1. Clone the repository (or download it locally) and ensure that the required modules are installed. These modules should be in the “pip requirements” file. If a module is not installed when running the application, please use the command line with “pip install <module>”
2. You can also download Miniconada either manually or with the “InstallMiniconda.bat” script. You then use conda env create -f cpu\_environment.yml or cuda\_environment.
3. There are already a few classes made for automated testing for certain things. For example, test\_google\_sheet.py tests the functionality of the Google Sheet calls and the class that encapsulates it.
4. Other testing requires manual interaction and observation. For example, some of the models might be returning incorrect information, but based on the small sample size that was fed to the models it is difficult for them to understand whether an image is technically “correct”.
5. To begin testing, you can either run the application with the “main.py” file, or you can create an automated test script to confirm unit tests are being run correctly. Again, there are already a couple of automated unit test scripts that can be run to ensure functionality.
6. Testing for this was run on both iOS and Windows and are therefore the suggested OS for running this application when debugging and running test cases.
7. For test\_data\_loader.py, test\_image\_blur\_detection.py, test\_main.py, test\_model\_api.py, and test\_object\_detection\_model.py. You should use the cmd “pytest -v <file path of .py file>. For example, pytest –v src/test\_object\_detection\_model.py. Testui\_main.py is ran like src/testui\_main.py

# Record 1.0

Test Model API

**1.1 Test loading model**

**Test Description:** Test that our model API can find existing models and return the appropriate error when the model cannot be found.

**Tester:** Ryan Naveira

**Recorder:** Collin Dunkle

**Date:** 11/29/2024

**Specify Test input:** No user input. Tests model\_path as inputs to the function.

**Expected Output:** None. Function will return stating the test either passed or failed.

**Result:** Test Passed Successfully

**1.2 Test model prediction**

**Test Description:** Test that our model API can call the model prediction functions and return appropriate data for correct input and error when given bad inputs.

**Tester:** Ryan Naveira

**Recorder:** Collin Dunkle

**Date:** 11/29/2024

**Specify Test input:** No user input. Tests models and folder\_path as inputs to the function.

**Expected Output:** None. Function will return stating the test either passed or failed.

**Result:** Test Passed Successfully

Figure 1.1 Test load model passed successfully



Figure 1.2 Test model passed successfully



# ****Record 2.0****

Test for Main.py (test\_main.py)  
**2.1 Test loading model**

**Test Description:** Test that the stop button indeed stops the AI from continuing its tasks.

**Tester:** Angelo Karam

**Recorder:** Ryan Naveira

**Date:** 11/29/2024

**Specify Test input:** Mouse Click

**Expected Output:** Program stops upon clicking stop button in UI

**Result:** Test Passed Successfully

**Test Description:** Test that the running of Main.py generates a GUI with buttons in the correct format and visual appearance.

**Tester:** Angelo Karam

**Recorder:** Ryan Naveira

**Date:** 11/29/2024

**Specify Test input:** None

**Expected Output:** GUI launches successfully.

**Result:** Test Passed Successfully  
  
**Test Description:** Test that the only input file allowed in the GUI is a .json, not other types.

**Tester:** Angelo Karam

**Recorder:** Ryan Naveira

**Date:** 11/29/2024

**Specify Test input:** File selection

**Expected Output:** Does not allow non-.json files

**Result:** Test Passed Successfully

**Test Description:** Test that the GUI shuts down gracefully when stopping the app.

**Tester:** Angelo Karam

**Recorder:** Ryan Naveira

**Date:** 11/29/2024

**Specify Test input:** None

**Expected Output:** The app shuts down.

**Result:** Test Passed Successfully

# ****Record 3.0****

Test for data\_loader.py

**Test Description:** This test validates that the inputs are correct.

**Tester:** Joaquim Pedroza

**Recorder:** Angelo Karam

**Date:** 11/30/2024

**Specify Test Input:**

1. targ\_dir must be a string or a path.
2. targ\_dir must be valid.
3. Type must be of expected “desk” or “assignments” types.
4. otransform' must be of type torchvision.transforms.V2r a list of torchvision.transforms.V2
5. transform' must be of type torchvision.transforms.V2 or a list of torchvision.transforms.V2
6. Transform must contain valid data.

**Expected Output:** no output expected, all inputs must be accepted successfully.

**Result:** 

**Test Description:** This test validates that the bounding boxes were properly parsed.

**Tester:** Joaquim Pedroza

**Recorder:** Angelo Karam

**Date:** 11/30/2024

**Specify Test Input:**

1. There must be a file present in the path.
2. The return type must be of type tuple.
3. Return length must be (2).

**Expected Output:** no output expected.

**Result:** 

**Test Description:** This test validates the programs ability to draw a bounding box around the requested data.

**Tester:** Joaquim Pedroza

**Recorder:** Angelo Karam

**Date:** 11/30/2024

**Specify Test Input:**

1. The image taken must be of type torch.tensor.
2. The drawn box must be of type torch.tensor.
3. The class must be of type torch.tensor.
4. The return must be of type torch.tensor.

**Expected Output:** All types must be of type torch.tensor.

**Result:** 

**Test Description:** This test validates that the batch must be of type list[torch.Tensor,torch.Tensor,dict[int,str]].

**Tester:** Joaquim Pedroza

**Recorder:** Angelo Karam

**Date:** 11/30/2024

**Specify Test Input:**

1. Batch must be of type list[torch.Tensor,torch.Tensor,dict[int,str]].

**Expected Output:** None**.** All types must be of type list[torch.Tensor,torch.Tensor,dict[int,str]].

**Result:** 

# **Record 4.0**

Test for Image Blur Detection (test\_image\_blur\_detction)

All these test cases were run using the command:

pytest ./src/test\_image\_blur\_detection.py -v

**4.1 test\_detect\_image\_blur\_with\_blurry\_image**

**Test Description:** This test case goes over the mocking of a blurry image to see if the return checks are correct before running anything.

**Tester:** Austin Nolte

**Recorder:** Jacob Sherer

**Date:** 11/30/24

**Test Input:** mock\_laplacian, mock\_image (pytest fixtures that allow for easy mocking of inputs)

**Expected Output:** no output, test should assert the function returns True indicating the image is blurry

**Result:** Passed



**4.2 test\_detect\_image\_blur\_not\_blurry\_image**

**Test Description:** This test case goes over the mocking of a not blurry image to see if the return checks are correct before running anymore.

**Tester:** Austin Nolte

**Recorder:** Jacob Sherer

**Date:** 11/30/24

**Test Input:** mock\_laplacian, mock\_image (pytest fixtures that allow for easy mocking of inputs)

**Expected Output:** no output, test should assert the function returns False indicating the image is not blurry

**Result:** Passed



**4.3 test\_detect\_image\_blur\_input**

**Test Description:** This test case goes over the case where the input to image\_path is not a str or a pathlib.Path type.

**Tester:** Austin Nolte

**Recorder:** Jacob Sherer

**Date:** 11/30/24

**Test Input:** no input, the test is testing bad input parameters.

**Expected Output:** Test should raise a TypeError with error of “image path must be type str or pathlib.Path

**Result:** Passed



**4.4 test\_detect\_image\_blur\_none\_input**

**Test Description:** This test case goes over the case where the input to image\_path is not a str or a pathlib.Path type, specifically this one is a None type check.

**Tester:** Austin Nolte

**Recorder:** Jacob Sherer

**Date:** 11/30/24

**Test Input:** no input, the test is testing bad input parameters.

**Expected Output:** Test should raise a TypeError with error of “image path must be type str or pathlib.Path

**Result:** Passed



**4.5 test\_detect\_image\_blur\_bad\_path**

**Test Description:** This test case goes over the case where image\_path does not exist in the computer.

**Tester:** Austin Nolte

**Recorder:** Jacob Sherer

**Date:** 11/30/24

**Test Input:** no input, the test is testing bad input parameters.

**Expected Output:** Test should raise a FileNotFoundError

**Result:** Passed



**4.6 test\_detect\_image\_blur\_not\_image**

**Test Description:** This test case goes over the case where image\_path leads to something that is not an image.

**Tester:** Austin Nolte

**Recorder:** Jacob Sherer

**Date:** 11/30/24

**Test Input:** no input, the test is testing bad input parameters.

**Expected Output:** Test should raise a FileNotFoundError

**Result:** Passed



**4.7 test\_detect\_image\_blur\_good\_image**

**Test Description:** This test case goes over a positive case where the image is existing and not blurry.

**Tester:** Austin Nolte

**Recorder:** Jacob Sherer

**Date:** 11/30/24

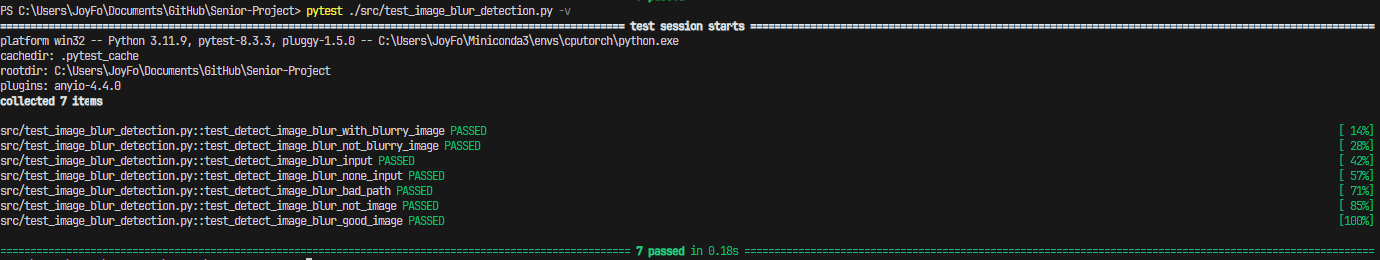
**Test Input:** No input takes images from test\_files.

**Expected Output: No output from this test, it should assert that the return is False to indicate a non-blurry image.**

**Result:** Passed



Figure 1.0



# **Record 5.0**

Test for UI (testui\_main.py)

These tests evaluate the program for core functionality, ensuring the application initializes correctly, assets are present, GUI components exist and function as intended, and button actions trigger the expected responses. Tests are run manually by calling python ./src/testui\_main.py from project directory.

**5.1 test\_appExists**

**Test Description:** This test verifies that the TechTutorApp instance is successfully created and is not None.

**Tester:** Jacob Sherer

**Recorder:** Austin Nolte

**Date:** 11/30/24

**Test Input:** Create an instance of TechTutorApp.

**Expected Output:** The app instance is successfully created and not None.

**Result:** Passed

**5.2 test\_guiExists**

**Test Description:** This test ensures that the TechTutorApp successfully builds the GUI and that the resulting GUI object is not None.

**Tester:** Jacob Sherer

**Recorder:** Austin Nolte

**Date:** 11/30/24

**Test Input:** Call build() on TechTutorApp.

**Expected Output:** The GUI object is successfully initialized.

**Result:** Passed

**5.3 test\_childrenExist**

**Test Description:** This test ensures that the GUI's root widget contains child elements.

**Tester:** Jacob Sherer

**Recorder:** Austin Nolte

**Date:** 11/30/24

**Test Input:** Children attribute of root widget

**Expected Output:** The children attribute of the root widget is not None.

**Result:** Passed

**5.4 test\_assetsExist**

**Test Description:** This test checks that image files required for the GUI exist in the expected file paths.

**Tester:** Jacob Sherer

**Recorder:** Austin Nolte

**Date:** 11/30/24

**Test Input:** File paths to gray1.png, gray2.png, gray3.png, and gray4.png.

**Expected Output:** All files exist at the specified paths.

**Result:** Passed

**5.5 test\_buttonsExist**

**Test Description:** This test verifies the existence of specific buttons in the GUI and confirms their text labels match the expected values.

**Tester:** Jacob Sherer

**Recorder:** Austin Nolte

**Date:** 11/30/24

**Test Input:** Retrieve button labels from the GUI.

**Expected Output:** The labels match: "Save Sheet ID", "0% Done", "Stop", "Start", "Change Account Credentials"

**Result:** Passed

**5.6 test\_start\_button**

**Test Description:** This test simulates pressing the start\_button and verifies that the associated start\_press method is called.

**Tester:** Jacob Sherer

**Recorder:** Austin Nolte

**Date:** 11/30/24

**Test Input:** Simulate Start button presses.

**Expected Output:** The associated method (start\_press) is triggered exactly once.

**Result:** Passed

**5.7 test\_stop\_button**

**Test Description:** This test simulates pressing the stop\_button and verifies that the associated stop\_press method is called.

**Tester:** Jacob Sherer

**Recorder:** Austin Nolte

**Date:** 11/30/24

**Test Input:** Simulate Stop button presses.

**Expected Output:** The associated method (stop\_press) is triggered exactly once.

**Result:** Passed

**5.8 test\_assetsMissing**

**Test Description:** This test verifies that missing\_image.png does not exist in the expected directory

**Tester:** Jacob Sherer

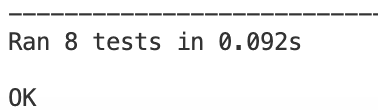
**Recorder:** Austin Nolte

**Date:** 11/30/24

**Test Input:** Invalid file path to missing\_image.png.

**Expected Output:** No file found.

**Result:** Passed



# Record 6.0

Test for Object Detection Model (test\_object\_detection\_model.py)

**6.1 Creating Model**

**Test Description:** Test item for the create model function. 5 test cases in total. The first four test different invalid input parameters and the last test case checks to see if the return object is the correct type.

**Tester:** Joshua Grindstaff

**Recorder:** Chisom Iwunze

**Date:** 11/29/24

**Test Input:** num\_objects\_to\_predict parameter for create\_model()

**Expected Output:** Correct error is raised, and function returns correct type

**Result:** Passed



**6.2 Training Model**

**Test Description:** Test item for the train\_model function. There are 7 test cases. The first two tests to see if the correct error shows up for incorrect model. The next four cases test the num\_epochs parameter. The last one checks if the training loop maintains the model’s type.

**Tester:** Joshua Grindstaff

**Recorder:** Chisom Iwunze

**Date:** 11/29/24

**Test Input:** input parameter of train\_model() function which are; model object, num\_epochs, and training

**Expected Output:** Correct error is raised, or function maintains correct type for input

**Result**: Passed



**6.3 Testing Model**

**Test Description:** This test item has 3 test cases. The first two checks if the right error is raised when passing invalid models. The last case checks to see if test\_model function won’t corrupt the model into another type.

**Tester:** Joshua Grindstaff

**Recorder:** Chisom Iwunze

**Date:** 11/29/24

**Test Input:** Inputs are function parameters of for model object

**Expected Output:** Correct error is raised, or function maintains correct type for input

**Result:** Passed



**6.4 Saving Model**

**Test Description:** Test the save\_model() function to ensure that the trained object detection model is correctly saved to the specified file path.

**Tester:** Chisom Iwunze

**Recorder:** Joshua Grindstaff

**Date:** 11/29/24

**Test Input:** Parameters for save\_model() function: model: An object of type FasterRCNN created using object\_detection\_model.create\_model(num\_objects\_to\_predict=2, type="packet"). path: A valid string path where the model will be saved, such as ./src/test\_files/obj\_detect\_test/test.p

**Expected Output:** A file named test.pt is created in the specified directory.

* If model is not of type FasterRCNN, it raises a TypeError with a clear message.
* If path is not a valid string, it raises a TypeError with a clear message.
* If path does not exist, it raises a FileNotFoundError

**Result:** Passed



**6.5 Saving Checkpoints**

**Test Description:** Test the save\_checkpoint() function to validate that model states, optimizer states, and training parameters are saved during training.

**Tester:** Chisom Iwunze

**Recorder:** Joshua Grindstaff

**Date:** 11/29/24

**Test Input:** Parameters for save\_checkpoint() function:model: An object of type FasterRCNN created using object\_detection\_model.create\_model(num\_objects\_to\_predict=2).

path: A valid string path where the checkpoint will be saved, such as ./src/test\_files/obj\_detect\_test/checkpoints/test.pth

**Expected Output:** A checkpoint file named test.pth is created in the specified directory.

* If model is not of type FasterRCNN, it raises a TypeError with a clear message.
* If path is not a valid string, it raises a TypeError with a clear message.
* If the directory for path does not exist, it raises a FileNotFoundError.

**Result:** Passed



**6.6 Loading Checkpoints**

**Test Description:** Test the load\_checkpoint() function to ensure that a previously saved checkpoint can restore the model's state correctly.

**Tester:** Chisom Iwunze

**Recorder:** Joshua Grindstaff

**Date:** 11/29/24

**Test Input:** Parameters for load\_checkpoint() function: model: An object of type FasterRCNN created using object\_detection\_model.create\_model(num\_objects\_to\_predict=2).

path: A valid string path of an existing checkpoint, such as ./src/test\_files/obj\_detect\_test/checkpoints/test.pth.

**Expected Output:** The function successfully restores the model's state from the checkpoint file.

* If model is not of type FasterRCNN, it raises a TypeError with a clear message.
* If path is not a valid string, it raises a TypeError with a clear message.
* If the checkpoint file does not exist, it raises a FileNotFoundError.

**Result:** Passed



**6.7 Creating and Training Model**

**Test Description:** Tests the function create\_and\_train\_model() using 8 test cases. The first 7 cases test to see if the right error is raised due to improper inputs. The last one tests to see if the function will correctly save model to test folder

**Tester:** Joshua Grindstaff

**Recorder:** Chisom Iwunze

**Date:** 11/29/24

**Test Input:** Function parameters; num\_epochs, model\_path, num\_object\_to\_predict, type

**Expected Output:** Correct Raised Errors and path in OS exists after function call completes

**Result:** Passed



**6.8 Predicting with model**

**Test Description:** Test the predict\_with\_model() function to validate its ability to generate predictions from a trained model and an input image.

**Tester:** Chisom Iwunze

**Recorder:** Joshua Grindstaff

**Date:** 11/29/24

**Test Input:** Parameters for predict\_with\_model() function: image: Path to the test image, such as ./src/test\_files/obj\_detect\_test/test\_image.png.

model: A trained object detection model of type FasterRCNN.

type: One of the valid class types, such as "packet"

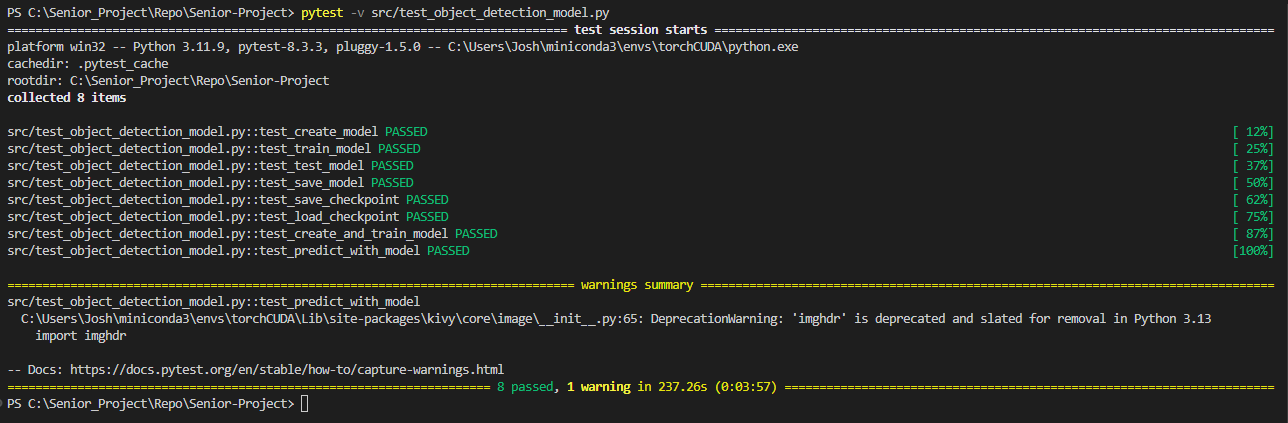
**Expected Output:** The function returns predictions, including bounding boxes, labels, and scores.

* If the image is not of type str or PIL.Image, it raises a TypeError with a clear message.
* If model is not of type FasterRCNN, it raises a TypeError with a clear message.
* If the type is not one of packet, desk, or caddy, it raises a KeyError with a clear message.

**Result:** Passed



Figure 1.0



# Record 7.0

Google Sheet and Google Drive API

**7.1 Read from Google Sheet**

**Test Description**: Confirm that our API calls can access and read correct information from the requested Google Sheet. We want to correctly read the information from the cell (1,1), aka A1, which is 23, using the get\_cell function.

**Tester**: Collin Dunkle

**Recorder**: Joaquim Pedroza

**Date**: 10/15/2024

**Specify** Test Input: No Input

**Expected Output**: Figure 7.1.1 Cell A1

**Result**: Success

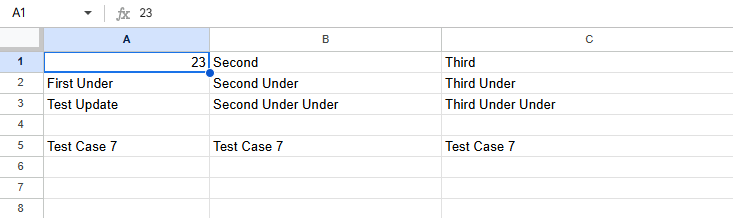
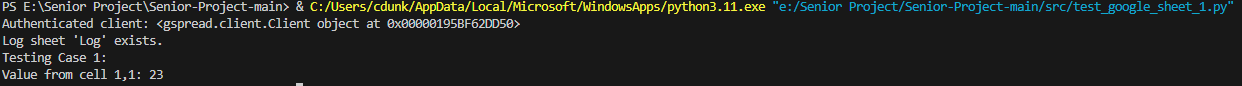
Figure 7.1.1 

Figure 7.1.2 Output from test case



**7.2 Read entire Column from Sheet**

**Test Description**: Successfully access an entire column and return the correct information using the get\_col function. Specifically access column 1 (A) from Figure 7.1.1.

**Tester**: Collin Dunkle

**Recorder**: Joaquim Pedroza

**Date**: 10/15/2024

**Specify** Test Input: No Input

**Expected Output**: Figure 7.1.1 Column A

**Result**: Success

Figure 7.2.1 Success Case



**7.3 Read entire row from Sheet**

**Test Description**: Successfully access an entire row and return the correct information using the get\_row function. Specifically access row 3 from figure 7.1.1.

**Tester**: Collin Dunkle

**Recorder**: Joaquim Pedroza

**Date**: 10/15/2024

**Specify** Test Input: No Input

**Expected Output**: Figure 7.1.1 Row 3

**Result**: Success

Figure 7.3.1 Success case



**7.4 Write to Cell**

**Test Description**: Write to Cell (1,3) (A3) with the string “Test Update” using the update\_cell function.

**Tester**: Collin Dunkle

**Recorder**: Joaquim Pedroza

**Date**: 10/15/2024

**Specify** Test Input: No Input

**Expected Output**: Cell A3 is updated from a blank cell to a cell that contains the string “Test Update”

**Result**: Success

Figure 7.2.1 Before Test Execution

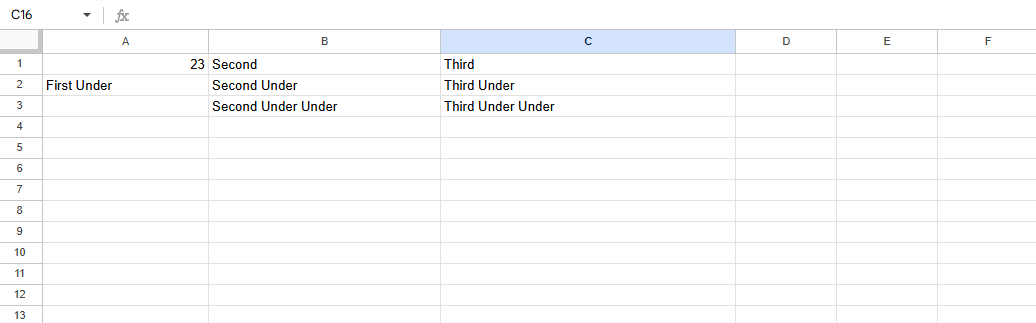


Figure 7.2.2 After Test Execution

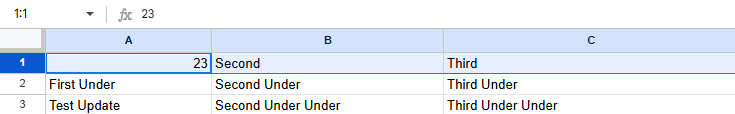


Figure 7.2.3



**7.5 Compare Cell to string**

**Test Description**: Compares the value of a cell to the value of a string specified in the arguments of the compare\_cell function. In this case we are comparing cell A1 with the string of “False”. Therefore, our output should be ‘False’ as 23 != “False”

**Tester**: Collin Dunkle

**Recorder**: Joaquim Pedroza

**Date**: 10/15/2024

**Specify** Test Input: No Input

**Expected Output**:

**Result**: Success

Figure 7.5.1



**7.6 Test Quota limit**

**Test Description**: Google Sheet API has a limit of the amount of requests that can be sent per minute, but we don’t want to lose information trying to write and then not be able to. Therefore, the update\_cell function has a sort of spin lock that when unable to write to the sheet due to too many requests it will sit and wait and query the sheet to see if it can write to it yet without losing any data. This test tries to write to the cell 150 times and should show that it waits. It will update the cell A1 with the number that it is at in the loop, and when complete we should see that the number is 149 (0 based math).

**Tester**: Collin Dunkle

**Recorder**: Joaquim Pedroza

**Date**: 10/15/2024

**Specify** Test Input: No Input

**Expected Output**:

**Result**: Success

Figure 7.6.1

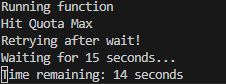


Figure 7.6.2 Current Cell Value at time of Figure 7.6.1

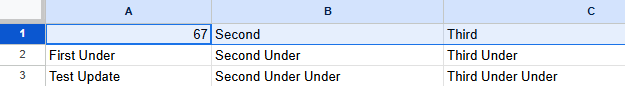
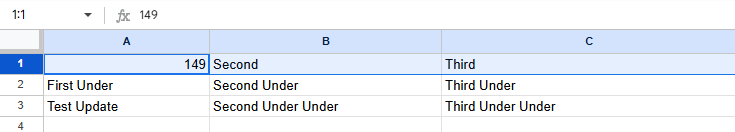


Figure 7.6.3 Completed Execution



# Signatures

**Team Members:** **System Test** **Verification Completed: Date: \_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |
| --- | --- |
| **Name**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Signature**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  |  |