System Test Report

Project name: TrashClassifier

Team name: Big Data Energy

Team Members
Travis Hammond
Kenta Miyahara
Julian Hernandez
Jeffrey de Jesus
Bryan Burch
Daniel Smagly
Santiago Bermudez
Christopher Allen

Date: 05/12/2023

TABLE OF CONTENTS

Introduction

The TrashClassifier System Test Report documents the comprehensive testing conducted on the TrashClassifier program, a deep neural network-based application designed to enhance waste management practices and reduce the environmental impact of improper waste disposal. The primary objective of these tests was to ensure the program's reliability, accuracy, and usability in real-world scenarios, as well as to identify and address potential issues before deployment. The testing process was carried out in multiple stages, covering the program's backend model inference and annotation database servers, as well as the front-end terminal app, terminal script, and website. To achieve maximum coverage and accuracy, a combination of automated unit tests and manual tests were conducted, along with manual integration testing to evaluate the seamless functioning of different components. This document presents a detailed account of the test environment setup, test suite, run records, and artifacts.

Test Environment Setup

Test environment section containing detailed instructions on how to set up the test environment to include pulling the code & test scripts from the repository. Version numbers for all 3rd party test applications. Operating system used by the team for testing.

Python Version 3.9 available at https://www.python.org/downloads/release/python-390/. Then run python3 -m pip install -r requirements.txt

OR

Conda: install Anaconda, create an environment conda create --name trashsorting, activate environment conda activate trashsorting, install requirements pip install -r requirements.txt

For GPU torch, instead of installing torch in requirements.txt do the below:

Windows/Linux: conda install pytorch torchvision torchaudio pytorch-cuda=11.7 -c pytorch -c nvidia

NodeJS
Download and install the latest version of Node.js from the official website: https://nodejs.org/en/download/. Follow the installation instructions for your operating system.

Yarn
Once NPM is installed with NodeJS you can run npm install --global yarn

Run the server
Change to react-website directory and run following commands: cd project\front-end_apps\website\react-website\

yarn install (may also need to run yarn add tabler-react tabler-icons-react)

yarn build generates a build folder in the current directory

yarn start to run on http://localhost:3000/

start model_inference_app.py in separate terminal to test submitting images

Test Suite

1.0 Annotation Database Server - test_db_server_app.py

1.1 Description:

This is a group of tests that validate the functionality of the REST API for the annotation database. These tests validate uploading and getting images and data from the database. There are a total of 20 individual tests for 7 different classes of tests, with each class of tests meant for a different function in db_server_app.py.

1.2 Valid Test Inputs:

Valid Data that can be submitted to the annotation database.

1.3 In-Valid Test Inputs:

Invalid inputs include requests with missing or bad data, with many testing for an invalid key.

1.4 Tester: Jeffrey de Jesus

1.5 Date Tested: 05/05/2023

1.6 Results: ALL PASSING

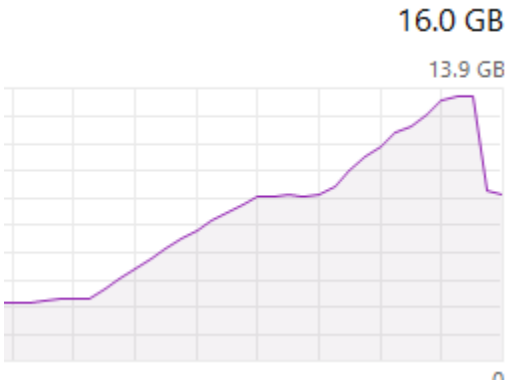
```
jeffdejesus@Macbook annotation_database % pytest -vv
============================= test session starts ==============================
platform darwin -- Python 3.9.2, pytest-7.2.1, pluggy-1.0.0 -- /Library/Frameworks/Python.framework/Versions/3.9/bin/python3
cachedir: .pytest_cache
rootdir: /Users/jeffdejesus/Trash-Sorting/project/backend/annotation_database
plugins: anyio-3.6.2
collected 20 items

test_db_server_app.py::TestReadEntry::test_invalid_key PASSED            [  5%]
test_db_server_app.py::TestReadEntry::test_image_missing PASSED          [ 10%]
test_db_server_app.py::TestReadEntry::test_missing_annotation PASSED     [ 15%]
test_db_server_app.py::TestReadEntry::test_missing_num_annotation PASSED [ 20%]
test_db_server_app.py::TestReadEntry::test_missing_dataset PASSED        [ 25%]
test_db_server_app.py::TestReadEntry::test_missing_metadata PASSED       [ 30%]
test_db_server_app.py::TestReadEntry::test_success PASSED                [ 35%]
test_db_server_app.py::TestGetImage::test_image_success PASSED           [ 40%]
test_db_server_app.py::TestGetImage::test_image_out_of_bounds PASSED     [ 45%]
test_db_server_app.py::TestGetMetadata::test_empty_result PASSED         [ 50%]
test_db_server_app.py::TestGetMetadata::test_success PASSED              [ 55%]
test_db_server_app.py::TestReadEntries::test_get_min_max_annotation PASSED [ 60%]
test_db_server_app.py::TestHandleGetEntryMaxnnotations::test_max_annotation_success PASSED [ 65%]
test_db_server_app.py::TestHandleAnnotationApproved::test_approved_invalid_key PASSED [ 70%]
test_db_server_app.py::TestHandleAnnotationApproved::test_approved_success PASSED [ 75%]
test_db_server_app.py::TestHandleEntryUpdate::test_update_invalid_key PASSED [ 80%]
test_db_server_app.py::TestHandleEntryUpdate::test_update_entry_success PASSED [ 85%]
test_db_server_app.py::TestDeleteImage::test_delete_invalid_key PASSED   [ 90%]
test_db_server_app.py::TestDeleteImage::test_delete_success PASSED       [ 95%]
test_db_server_app.py::TestDeleteImage::test_delete_out_of_bounds PASSED [100%]

============================== 20 passed in 0.73s ==============================
jeffdejesus@Macbook annotation_database %
```

2.0 Terminal App - test_trashshorting.py

2.1 This test confirms that our program can be run from the command line and arguments can be passed and parsed correctly. This was a requirement from our client, that he

wanted the ability to automated processing through the command line and auto launch tasks.

2.2 Valid Test Inputs: Server needs to be running and available RAM space for the local model to load (~6GB). Tests using a valid image, local model



Local model loaded
Image exists and server running

2.3 In-Valid Test Inputs:
Bad file path to image
No server running
Local model not available

2.4 Tester Name: Julian Hernandez

2.5 Date Tested: 5/5/2023

2.6 Results: ALL PASSING

```
front-end_apps> pytest .\test_trashsorting.py
============================= test session starts =============================
platform win32 -- Python 3.9.7, pytest-6.2.4, py-1.10.0, pluggy-0.13.1
rootdir: C:\Users\julian\repos\Senior-Project\project\front-end_apps
plugins: anyio-2.2.0
collected 4 items

test_trashsorting.py ....                                                [100%]

============================= warnings summary ==============================
..\..\..\..\anaconda3\lib\site-packages\pyreadline\py3k_compat.py:8
  C:\Users\julian\anaconda3\lib\site-packages\pyreadline\py3k_compat.py:8: DeprecationWarning: Using or importing the ABCs from 'collections' instead of from 'collections.abc' is deprecated since Python 3.3, and in 3.1
0 it will stop working
    return isinstance(x, collections.Callable)

-- Docs: https://docs.pytest.org/en/stable/warnings.html
======================== 4 passed, 1 warning in 57.42s ========================
```

3.0 Classification UI - test_classify.ipynb

3.1 Brief Description of Test:

This involves manually testing the classify functions on the terminal for the direct app. Basically, from the terminal, we would run the ssd model first and then we would run the program to test the classification functions. This test was conducted during an early phase of the project, so not all features were complete yet, but we were still able to test things to make sure that we are on track.

3.2 Valid Test Inputs: description and value or sequence of values entered

For starters, we will open the terminal on VS Code and use the ‘ls’ and ‘cd’ commands to go to the filepath where our front-end_apps set of code is located.

```
PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project> cd front-end_apps
PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps> ls


    Directory: C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         4/24/2023   1:26 PM                .pytest_cache
d-----         4/12/2023   4:35 PM                website
d-----         4/24/2023   1:29 PM                __pycache__
-a----         4/12/2023   4:35 PM           2425 about.py
-a----         4/12/2023   4:35 PM          11721 annotate.py
-a----         4/12/2023   4:35 PM          60695 annotate_test.ipynb
-a----         4/12/2023   4:35 PM           3458 camera.py
-a----         4/12/2023   4:45 PM           4350 classify.py
-a----         4/12/2023   4:35 PM           1817 download_model.py
-a----         4/24/2023   1:40 PM           1880 main.py
-a----         4/12/2023   4:35 PM           2480 misc.py
-a----         4/12/2023   4:35 PM             72 README.md
-a----         4/24/2023   1:33 PM             40 settings.cfg
-a----         4/12/2023   4:35 PM           4307 settings.py
-a----         4/12/2023   4:42 PM           7955 ssd.py
-a----         4/24/2023   1:26 PM             41 temp_settings.cfg
-a----         4/12/2023   4:35 PM           1208 test_about.py
-a----         4/12/2023   4:35 PM           1753 test_annotate.py
-a----         4/12/2023   4:35 PM          72079 test_classify.ipynb
-a----         4/12/2023   4:35 PM           5544 test_settings.py
-a----         4/12/2023   4:35 PM           2427 test_trashsorting.py
-a----         4/12/2023   4:46 PM           3033 trashsorting.py
-a----         4/12/2023   4:35 PM           1981 trash_info.py
```

Once we get to where we need to be, we will first run the ssd.py file to get our ssd model code running. We need to do this before we can test our classify functions, otherwise things won't work without the server model.

```
-a----         4/24/2023   1:33 PM             40 settings.cfg
-a----         4/12/2023   4:35 PM           4307 settings.py
-a----         4/12/2023   4:42 PM           7955 ssd.py
-a----         4/24/2023   1:26 PM             41 temp_settings.cfg
-a----         4/12/2023   4:35 PM           1208 test_about.py
-a----         4/12/2023   4:35 PM           1753 test_annotate.py
-a----         4/12/2023   4:35 PM          72079 test_classify.ipynb
-a----         4/12/2023   4:35 PM           5544 test_settings.py
-a----         4/12/2023   4:35 PM           2427 test_trashsorting.py
-a----         4/12/2023   4:46 PM           3033 trashsorting.py
-a----         4/12/2023   4:35 PM           1981 trash_info.py


PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps> python3 ssd.py
PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps>
```

After, we will run the "main.py" file to test only the classify functionality.

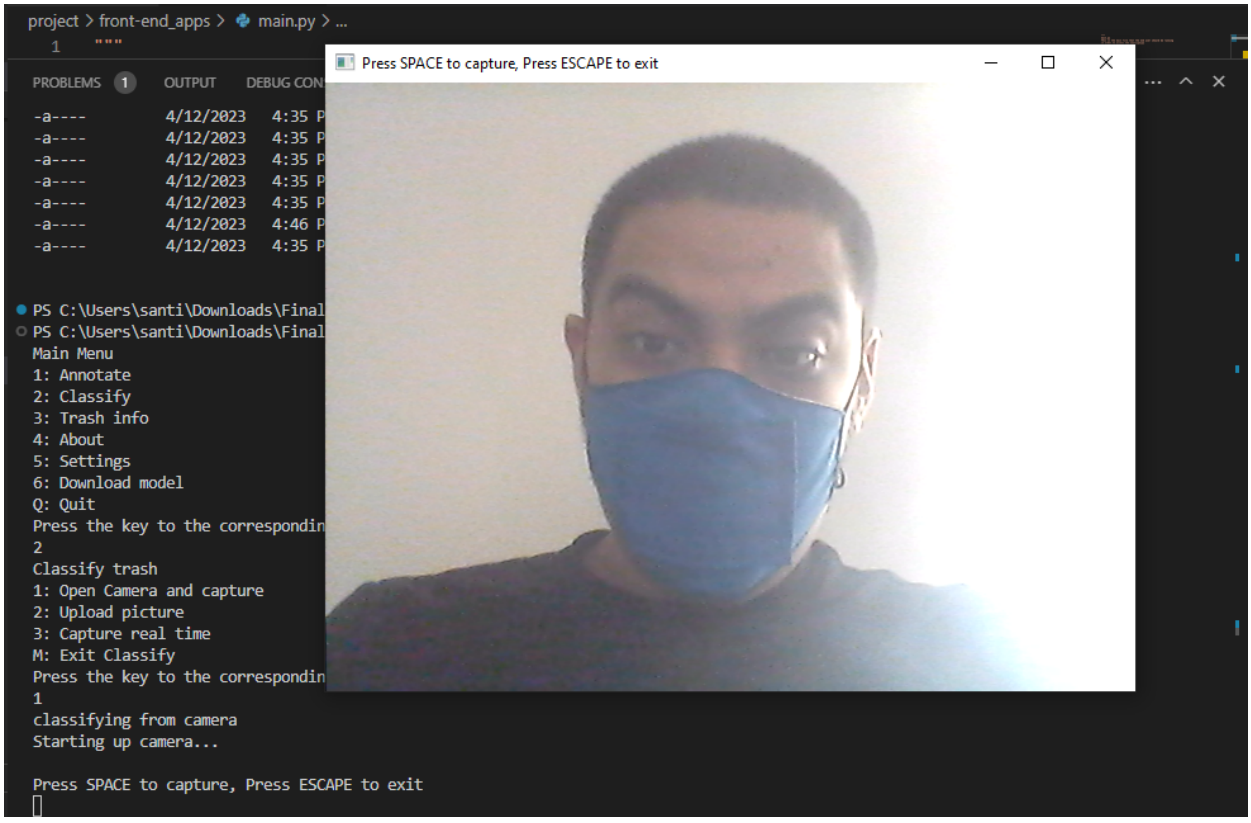
```
-a----         4/24/2023   1:33 PM             40 settings.cfg
-a----         4/12/2023   4:35 PM           4307 settings.py
-a----         4/12/2023   4:42 PM           7955 ssd.py
-a----         4/24/2023   1:26 PM             41 temp_settings.cfg
-a----         4/12/2023   4:35 PM           1208 test_about.py
-a----         4/12/2023   4:35 PM           1753 test_annotate.py
-a----         4/12/2023   4:35 PM          72079 test_classify.ipynb
-a----         4/12/2023   4:35 PM           5544 test_settings.py
-a----         4/12/2023   4:35 PM           2427 test_trashsorting.py
-a----         4/12/2023   4:46 PM           3033 trashsorting.py
-a----         4/12/2023   4:35 PM           1981 trash_info.py


PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps> python3 ssd.py
PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps> python3 main.py
Main Menu
1: Annotate
2: Classify
3: Trash info
4: About
5: Settings
6: Download model
Q: Quit
Press the key to the corresponding action.
```

We will enter 2 to go to classify features and then we will test all the commands one by one.

```
-a----         4/24/2023   1:33 PM             40 settings.cfg
-a----         4/12/2023   4:35 PM           4307 settings.py
-a----         4/12/2023   4:42 PM           7955 ssd.py
-a----         4/24/2023   1:26 PM             41 temp_settings.cfg
-a----         4/12/2023   4:35 PM           1208 test_about.py
-a----         4/12/2023   4:35 PM           1753 test_annotate.py
-a----         4/12/2023   4:35 PM          72079 test_classify.ipynb
-a----         4/12/2023   4:35 PM           5544 test_settings.py
-a----         4/12/2023   4:35 PM           2427 test_trashsorting.py
-a----         4/12/2023   4:46 PM           3033 trashsorting.py
-a----         4/12/2023   4:35 PM           1981 trash_info.py


PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps> python3 ssd.py
PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps> python3 main.py
Main Menu
1: Annotate
2: Classify
3: Trash info
4: About
5: Settings
6: Download model
Q: Quit
Press the key to the corresponding action.
2
Classify trash
1: Open Camera and capture
2: Upload picture
3: Capture real time
M: Exit Classify
Press the key to the corresponding action
```

We will then enter 1 to open the camera and capture.

```
-a----         4/12/2023   4:42 PM           7955 ssd.py
-a----         4/24/2023   1:26 PM             41 temp_settings.cfg
-a----         4/12/2023   4:35 PM           1208 test_about.py
-a----         4/12/2023   4:35 PM           1753 test_annotate.py
-a----         4/12/2023   4:35 PM          72079 test_classify.ipynb
-a----         4/12/2023   4:35 PM           5544 test_settings.py
-a----         4/12/2023   4:35 PM           2427 test_trashsorting.py
-a----         4/12/2023   4:46 PM           3033 trashsorting.py
-a----         4/12/2023   4:35 PM           1981 trash_info.py


PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps> python3 ssd.py
PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps> python3 main.py
Main Menu
1: Annotate
2: Classify
3: Trash info
4: About
5: Settings
6: Download model
Q: Quit
Press the key to the corresponding action.
2
Classify trash
1: Open Camera and capture
2: Upload picture
3: Capture real time
M: Exit Classify
Press the key to the corresponding action
1
classifying from camera
Starting up camera...
```



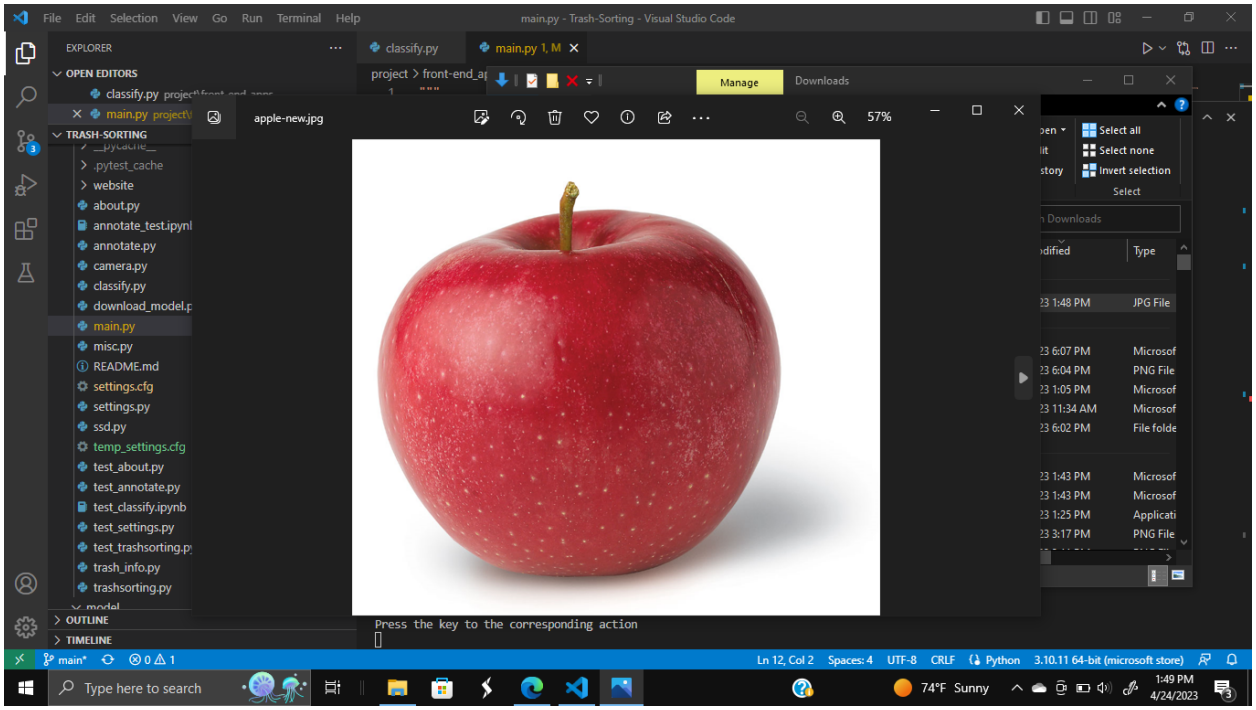
After we press space to capture, we should see a message like the one below.

```
PROBLEMS 1  OUTPUT  DEBUG CONSOLE  TERMINAL                    powershell - front-end_apps
1: Annotate
2: Classify
3: Trash info
4: About
5: Settings
6: Download model
Q: Quit
Press the key to the corresponding action.
2
Classify trash
1: Open Camera and capture
2: Upload picture
3: Capture real time
M: Exit Classify
Press the key to the corresponding action
1
classifying from camera
Starting up camera...

Press SPACE to capture, Press ESCAPE to exit

Captured
Traceback (most recent call last):
  File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\main.py", line 65, in <module>
    main()
  File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\main.py", line 45, in main
    classify.main(settings.PROCESS_ONLINE, settings.FPS_RATE)
  File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\classify.py", line 45, in main
    camera_classify(process_online)
  File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\classify.py", line 83, in camera_classify
    pred = preds(img, process_online)
TypeError: preds() missing 1 required positional argument: 'single_classification'
PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps>
```

We then enter 2 to upload images.

```
Captured
Traceback (most recent call last):
  File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\main.py", line 65, in <module>
    main()
  File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\main.py", line 45, in main
    classify.main(settings.PROCESS_ONLINE, settings.FPS_RATE)
  File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\classify.py", line 45, in main
    camera_classify(process_online)
  File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\classify.py", line 83, in camera_classify
    pred = preds(img, process_online)
TypeError: preds() missing 1 required positional argument: 'single_classification'
PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps> python3 main.py
Main Menu
1: Annotate
2: Classify
3: Trash info
4: About
5: Settings
6: Download model
Q: Quit
Press the key to the corresponding action.
2
Classify trash
1: Open Camera and capture
2: Upload picture
3: Capture real time
M: Exit Classify
Press the key to the corresponding action
2
Classifying from file
Enter image file path to be classifyed:
```

The image we will be using for testing is a JPG of an apple, but you can use anything.



Below is the result you should expect after entering the file path incorrectly to the apple image.

```
  Q: Quit
Press the key to the corresponding action.
  2
  Classify trash
  1: Open Camera and capture
  2: Upload picture
  3: Capture real time
  M: Exit Classify
  Press the key to the corresponding action
  2
  Classifying from file
  Enter image file path to be classifyed:

  [ WARN:0@87.622] global loadsave.cpp:244 cv::findDecoder imread_('''): can't open/read file: check file path/integrity
  Image could not be read
  1: Open Camera and capture
  2: Upload picture
  3: Capture real time
  M: Exit Classify
  Press the key to the corresponding action
```

Below is the result to be expected for JPGs.

```
Main Menu
 1: Annotate
 2: Classify
 3: Trash info
 4: About
 5: Settings
 6: Download model
 Q: Quit
 Press the key to the corresponding action.
 2
 Classify trash
 1: Open Camera and capture
 2: Upload picture
 3: Capture real time
 M: Exit Classify
 Press the key to the corresponding action
 2
 Classifying from file
 Enter image file path to be classifyed:
 C:\Users\santi\Downloads\apple-new.jpg
 Traceback (most recent call last):
   File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\main.py", line 65, in <module>
     main()
   File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\main.py", line 45, in main
     classify.main(settings.PROCESS_ONLINE, settings.FPS_RATE)
   File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\classify.py", line 47, in main
     file_classify(process_online)
   File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\classify.py", line 115, in file_classify
     pred = preds(img, process_online)
 TypeError: preds() missing 1 required positional argument: 'single_classification'
PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps>
```

Below is the result to be expected for PNGs.

```
 TypeError: preds() missing 1 required positional argument: 'single_classification'
 PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps> python3 main.py
Main Menu
 1: Annotate
 2: Classify
 3: Trash info
 4: About
 5: Settings
 6: Download model
 Q: Quit
 Press the key to the corresponding action.
 2
 Classify trash
 1: Open Camera and capture
 2: Upload picture
 3: Capture real time
 M: Exit Classify
 Press the key to the corresponding action
 2
 Classifying from file
 Enter image file path to be classifyed:
 C:\Users\santi\Downloads\apple-new.png
 Traceback (most recent call last):
   File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\main.py", line 65, in <module>
     main()
   File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\main.py", line 45, in main
     classify.main(settings.PROCESS_ONLINE, settings.FPS_RATE)
   File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\classify.py", line 47, in main
     file_classify(process_online)
   File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\classify.py", line 115, in file_classify
     pred = preds(img, process_online)
 TypeError: preds() missing 1 required positional argument: 'single_classification'
PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps>
```

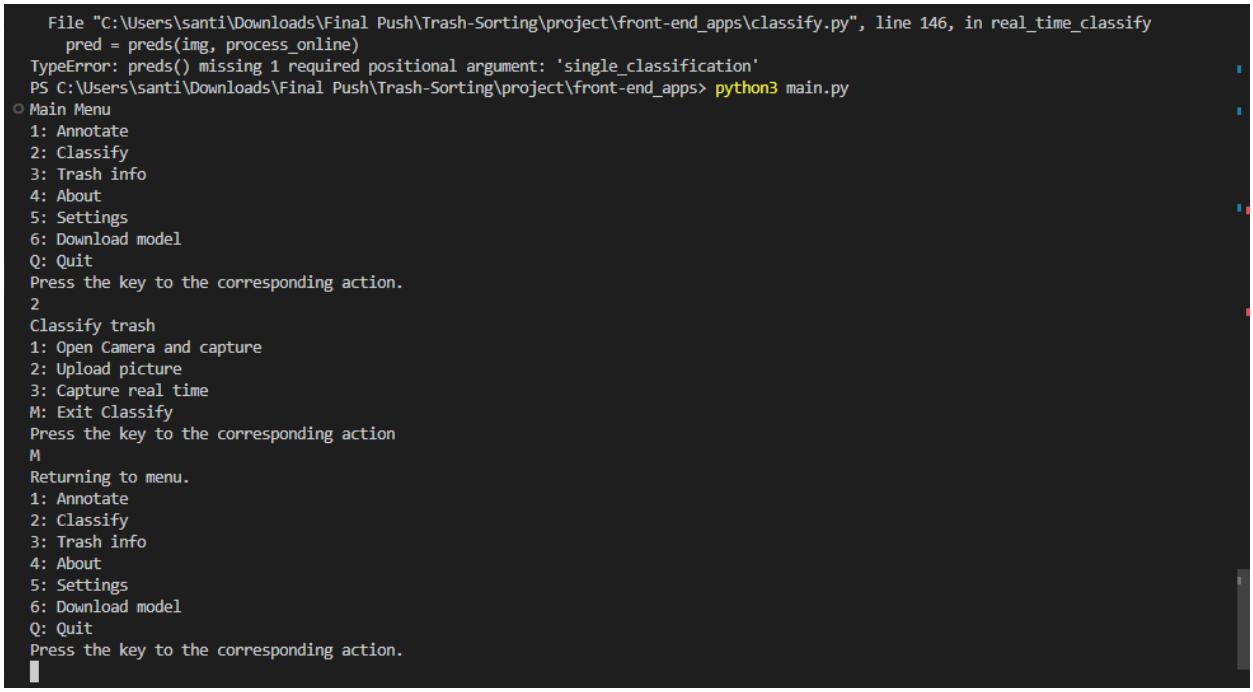
Next, we will enter 3 to capture an object live (*this feature will quickly take an image for you).

```
2: Upload picture
3: Capture real time
M: Exit Classify
Press the key to the corresponding action
2
Classifying from file
Enter image file path to be classifyed:
C:\Users\santi\Downloads\apple-new2
[ WARN:0@590.603] global loadsave.cpp:244 cv::findDecoder imread_('C:\Users\santi\Downloads\apple-new2'): can't open/read file: check
file path/integrity
Image could not be read
1: Open Camera and capture
2: Upload picture
3: Capture real time
M: Exit Classify
Press the key to the corresponding action
3
Classifying in real time
Starting up camera...
Active. Press Q to stop
```

You can expect to see a result like the one below.

```
ERROR: Invalid option (Enter key to the left of menu options).

1: Open Camera and capture
2: Upload picture
3: Capture real time
M: Exit Classify
Press the key to the corresponding action
2
Classifying from file
Enter image file path to be classifyed:
C:\Users\santi\Downloads\apple-new2
[ WARN:0@590.603] global loadsave.cpp:244 cv::findDecoder imread_('C:\Users\santi\Downloads\apple-new2'): can't open/read file: check
file path/integrity
Image could not be read
1: Open Camera and capture
2: Upload picture
3: Capture real time
M: Exit Classify
Press the key to the corresponding action
3
Classifying in real time
Starting up camera...
Traceback (most recent call last):
  File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\main.py", line 65, in <module>
    main()
  File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\main.py", line 45, in main
    classify.main(settings.PROCESS_ONLINE, settings.FPS_RATE)
  File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\classify.py", line 49, in main
    real_time_classify(process_online, fps_rate)
  File "C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\classify.py", line 146, in real_time_classify
    pred = preds(img, process_online)
TypeError: preds() missing 1 required positional argument: 'single_classification'
PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps> python3 main.py
```

Finally, we test the exit feature by simply entering ‘M’.



3.3 In-Valid Test Inputs: To test error handling/recovery

N/A.

3.4 Tester Name: Santiago Bermudez

3.5 Date Tested: 03/01/2023

3.6 Results: ALL PASSING

4.0 React Website Tests - classifyForm.test.js

4.1 Description:

These set of tests confirm that the functionality of the React website works and runs correctly. There are six individual tests with each testing a separate functionality of the website like selecting a model or inserting a file. There is one test which renders the content of the website to ensure that the content can be rendered without any error and makes sure the other tests work.

4.2 Valid Test Inputs:

A list of fake models to be used and selected for confirming features work and to be used in other tests.

4.3 In-Valid Test Inputs:

There are no invalid inputs that are being tested for.

4.4 Tester Name: Jeffrey de Jesus

4.5 Date Tested: 05/05/2023

4.6 Results: ALL PASSING

```
 PASS  src/classifyForm.test.js
  ClassifyForm
    ✓ should render without throwing an error (8ms)
    componentDidMount
      ✓ should set state.options with model names on successful fetch (3ms)
    handleDropdownChange
      ✓ should set selectedModel state (3ms)
    handleAddFileInput
      ✓ should increment fileInputCount state and disable submit/add file buttons (3ms)
    handleRemoveFileInput
      ✓ should decrement fileInputCount state and remove the last file from filesToSubmit state (4ms)
      ✓ should not remove the last file input element and corresponding file object if there is only one file input element (2ms)

Test Suites: 1 passed, 1 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        6.122s
Ran all test suites.

Watch Usage
 › Press f to run only failed tests.
 › Press o to only run tests related to changed files.
 › Press p to filter by a filename regex pattern.
 › Press q to quit watch mode.
 › Press t to filter by a test name regex pattern.
 › Press Enter to trigger a test run.

✨  Done in 22.08s.
jeffdejesus@Macbook-2 react-website %
```

5.0 Annotation UI - annotate_test.ipynb

5.1 Description:

This is a notebook which is a set of tests for the features of annotate.py which cannot be tested through automated tests. This tests the graphical and GUI features that the annotate part of the program is capable of doing. There are a total of 4 tests which each give a list of steps for the tester to do. There are acceptance criteria written for each test and if everything works then success will be printed at the end of the cell with no error messages printed. Each test involves the tester interacting with the UI in some way.

5.2 Valid Test Inputs: A simple image was used

5.3 In-Valid Test Inputs: To test error handling/recovery

5.4 Tester Name: Jeffrey de Jesus

5.5 Date Tested: 05/05/2023

5.6 Results: ALL PASSING

In [2]:
```
img = annotate.open_from_camera()
assert img == None
print("Success")
```

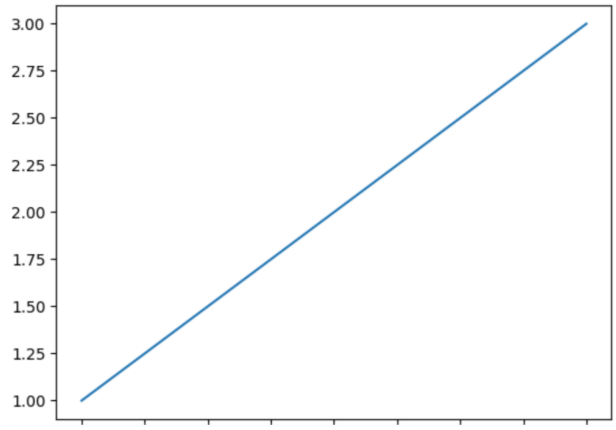
```
Annotating from camera
Starting up camera...

Press SPACE to capture, Press ESCAPE to exit
Success
```

In [3]:
```
img = annotate.open_from_camera()
assert isinstance(img, np.ndarray)
print("Success")
```

```
Annotating from camera
Starting up camera...

Press SPACE to capture, Press ESCAPE to exit

Captured
Success
```

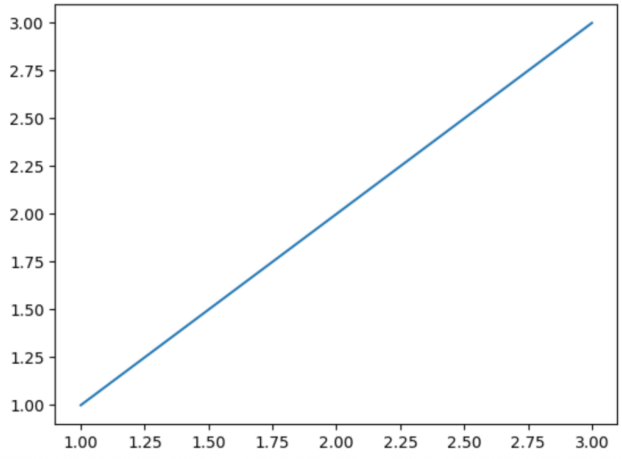
In [4]:
```
img = cv2.imread(get_img())
annotations = annotate.handle_annotation_ui(img)

assert annotations.annotations == []
print("success")
del_img()
```

```
 Draw: Left click drag
 Reset: Double Right Click
 Done: Ctrl + Right Click
 Exit: Esc
success
```



In [5]:
```
img = cv2.imread(get_img())
annotations = annotate.handle_annotation_ui(img)

assert len(annotations.annotations) == 2
print("success")
del_img()
```

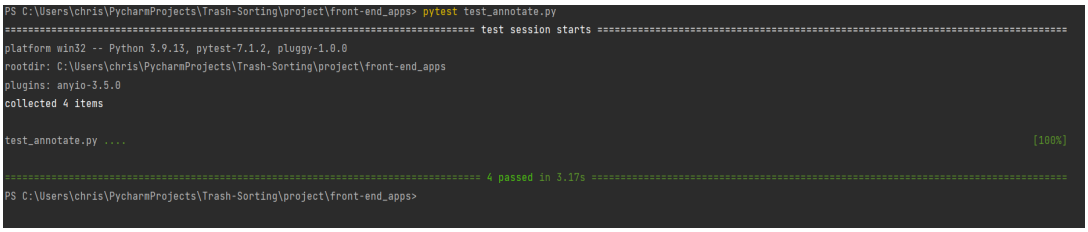
```
 Draw: Left click drag
 Reset: Double Right Click
 Done: Ctrl + Right Click
 Exit: Esc
success
```



6.0 Annotation - test_annotate.py

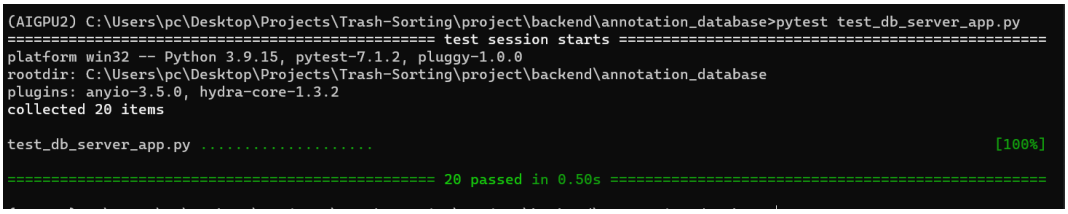
6.1 Unit tests that verifies valid, and invalid pathing, as well as finished and unfinished annotations.

6.2 Valid Test Inputs: Inputs are done automatically via monkeypatch for test_invalid_path, and test_valid_path, so no user input is required, “fake_image.jpg”, and “temp_path” are also used for test_valid_path and test_invalid_path.
6.3 In-Valid Test Inputs: In test_invalid_path, “fake_image.jpg” produces an invalid path.
6.4 Tester Name: Christopher Allen
6.5 Date Tested: 5/9/2023
6.6 Results: ALL PASSING



7.0 Annotation Database - test_db_server_app.py

7.1 This test is an automated unit test that confirms that our annotation database works correctly. The annotation database is used for storing user annotations of images. The client did not require this feature Explicitly but we provided it to allow the model inference to be improved by a custom data set.
7.2 Valid Test Inputs: This is an automated test that inputs many different values to test a half dozen functionalities, in short, it takes a key, annotation string, num_annotations, dataset source, metadata, and an image.
7.3 In-Valid Test Inputs: Missing image, corrupted image format, and missing or invalid key.
7.4 Tester Name: Travis Hammond
7.5 Date Tested: 5/8/2023
7.6 Results: ALL PASSING



8.0 OpenCLIP model download - test_download_model.py

8.1 This is an automated unit test that verifies that all four OpenCLIP models can be downloaded to the user's device without issue. The test checks to see if a model already exists on a user's device, and if not, verifies that the user will be able to successfully install each model.
8.2 Valid Test Inputs: One of five options is selected, either one of the four models or the ‘return to menu’ option. Each valid option is tested to verify that a model can be downloaded as well as check to make sure already downloaded models are not reinstalled.

8.3 In-Valid Test Inputs: There are no invalid inputs to be tested, if the user selects an option not available in the download models menu, they will be prompted to enter a valid option (1, 2, 3, 4, m).

8.4 Tester Name: Daniel Smagly

8.5 Date Tested: 5/9/2023

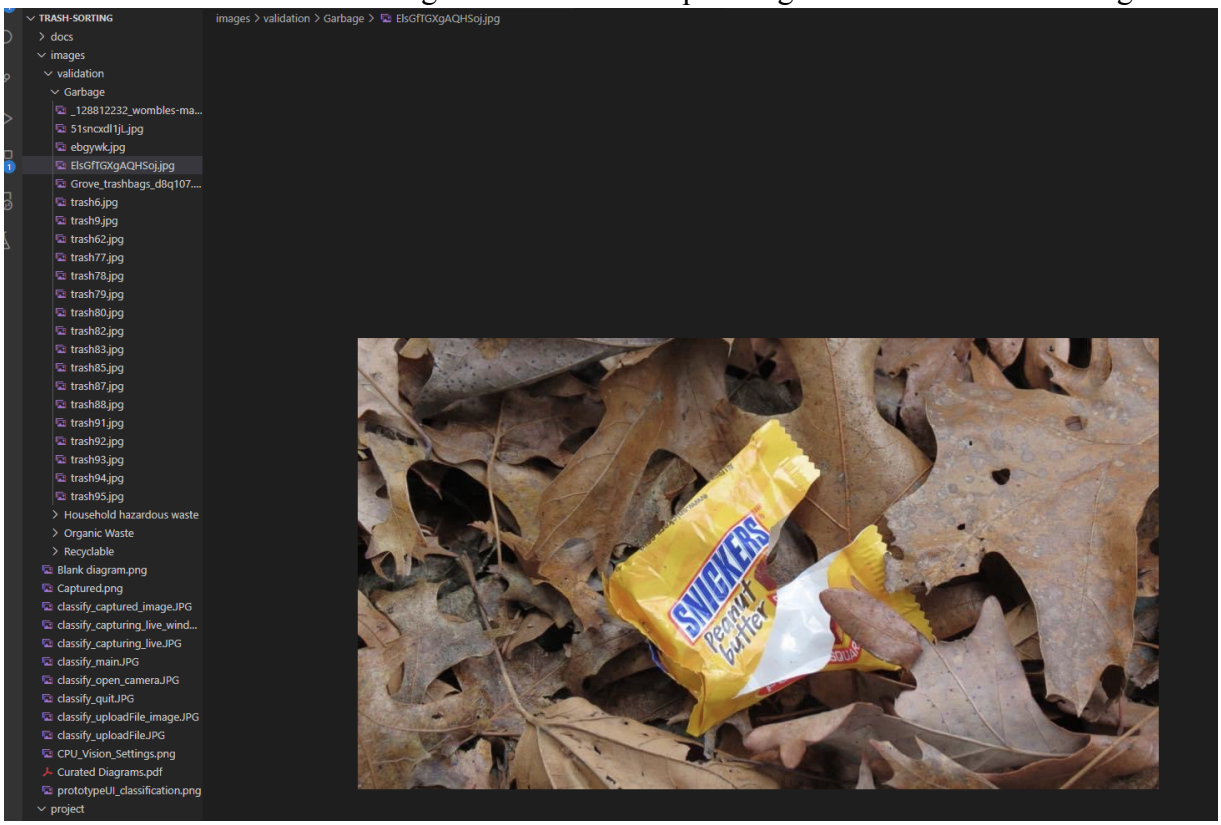
8.6 Results: ALL PASSING

```
danielsmagly@Daniels-MacBook-Pro-2 front-end_apps % pytest test_download_model.py
========================================== test session starts ==========================================
platform darwin -- Python 3.9.8, pytest-7.3.1, pluggy-1.0.0
rootdir: /Users/danielsmagly/Desktop/CSC190_1/TrashSorting/Trash-Sorting/project/front-end_apps
collected 5 items

test_download_model.py .....                                                                    [100%]

========================================== 5 passed in 35.71s ===========================================
```

9.0 Model Accuracy Validation - Validate.py

9.1 Test all current models accuracy. The script will run through a set of images that is provided in the path Trash-Sorting\images**validation.** The result will display execution time, mean average precision, and confusion matrix.

9.2 Valid Test Inputs: When testing model_inference_app.py should be running, which is the back-end server that will handle classification. When running through this validation, the script will match the image folder name, which is the correct classification. The image below is an example image from the validation images



9.3 In-Valid Test Inputs: There are no invalid inputs because the user won't be using this for the functionality of the program.
9.4 Tester Name: Kenta Miyahara
9.5 Date Tested: 4/12/2023
9.6 Results
9.6.1 Did it pass : Yes. Efficientnet_v2_1 passed with mAP with 0.78 and one of the larger ViT-g-14 model scored mAP of 0.72.

```
Model: efficientnet_v2_l
labels_bin shape: (97, 4)
predicted_classes_bin shape: (97, 4)
Mean Average Precision: 0.78
-----------------------------------------------
Model: ViT-g-14
labels_bin shape: (97, 4)
predicted_classes_bin shape: (97, 4)
Mean Average Precision: 0.72
-----------------------------------------------
Model: ViT-L-14
labels_bin shape: (97, 4)
predicted_classes_bin shape: (97, 4)
Mean Average Precision: 0.67
-----------------------------------------------
Model: ViT-B-16
labels_bin shape: (97, 4)
predicted_classes_bin shape: (97, 4)
Mean Average Precision: 0.50
-----------------------------------------------
efficientnet_v2_l 0.7825558977105369
ViT-g-14 0.7217932677288348
ViT-L-14 0.6734351010998718
ViT-B-16 0.501293227418851
```

10.0 Inference Server - test_model_inference_app.py

10.1 Unit tests for each endpoint (creating, reading, updating, deleting) relating to the model inference server that is responsible for handling classification requests. Each set of tests is grouped into its own class for clarity and discussed below.
10.2 Valid Test Inputs:
10.2.1 TestCreate: json object has the correct key, metadata, and model data
10.2.2 TestReadInference: json object includes the correct image data
10.2.3 TestReadBatchInf: same as previous
10.2.4 TestReadModelList: check that previously created model "test2" was added
10.2.5 TestReadModel: 200 response for added model
10.2.6 TestReadMetadata: same as previous
10.2.7 TestUpdate: updating a pre-existing model with the correct key
10.2.8 TestDelete: same as previous
10.3 In-Valid Test Inputs: To test error handling/recovery

10.3.1 TestCreate: using the wrong key; trying to create the same model; missing the model or metadata fields
10.3.2 TestReadInference: no image passed
10.3.3 TestReadBatchInf: same as previous
10.3.4 TestReadModelList: no test
10.3.5 TestReadModel: 200 response with error_code 3 for model that doesn't exist
10.3.6 TestReadMetadata: same as previous
10.3.7 TestUpdate: wrong key
10.3.8 TestDelete: same

10.4 Tester: Bryan Burch
10.5 Date Tested: 5/10/23
10.6 Results: Passed 19/20 tests

```
=========================== short test summary info ============================
FAILED test_model_inference_app.py::TestReadMetadata::test_known - assert {'error_code'...': '{"a": 5}'} == {'error_code'...': '{"a": 7}'}
========================= 1 failed, 19 passed in 6.89s =========================
```

11.0 Terminal App Settings - test_settings.py

11.1 Unit tests for toggling fps and computation mode.
11.2 Valid Test Inputs:
11.2.1 TestToggleFps: changing fps to 15 which is in range
11.2.2 TestToggleCompMode: inputting 1 for online; 2 for offline. Corresponding changes saved.
11.2.3 TestSettingsConfig: loading of and saving settings values works as expected
11.2.4 TestHandle: toggle fps/computation and exit work properly
11.3 In-Valid Test Inputs: To test error handling/recovery
11.3.1 TestToggleFps: changing to 1 which is out of range
11.3.2 TestToggleCompMode: invalid option 3 outputs proper error message
11.3.3 TestSettingsConfig: no test
11.3.4 TestHandle: invalid option -1 outputs proper error message
11.4 Tester: Bryan Burch
11.5 Date Tested: 5/10/23
11.6 Results: ALL PASSING

```
test_settings.py ...........                                             [100%]

============================== 11 passed in 0.05s ==============================
```

Artifacts

Artifacts section in the test report explaining the location of the test artifacts. All test scripts used are provided via a git link in the system test report or submitted as a zip file via Canvas. Any output files generated during testing are collocated with the code in a separate testing folder in git or submitted as a zip file via Canvas.

Signatures

Team Members: **System Test Verification Completed: Date:** 05/10/2023

Name	Signature
Travis Hammond	Travis Hammond
Kenta Miyahara	Kenta
Julian Hernandez	
Jeffrey de Jesus	
Bryan Burch	Bryan Burch
Daniel Smagly	
Santiago Bermudez	Santiago Bermudez
Christopher Allen	Christopher Allen