

CODE TESTING PLAN AND RESULTS

Copy-Waste

General Code Testing Process

The dashboard and Copy-Waste Data Augmentation code bases are tested using unit testing. When creating unit tests for these deliverables we followed a three step process: arrange, act, compare. We first arranged our test environment through initializing test variables, pytest fixtures, and beforeAll statements. Then, we pass the test parameters to the function we wish to test. Finally, we compare the result from this action with our expected result.

Our rare contaminant and bin detection machine learning models are tested through a manual process where results from testing and validation are either approved or denied. Based on the results, the model is either deployed to be used or requires further training.

Green Screen Dashboard Tests

The dashboard will be tested using unit testing through the Jest which is a JavaScript testing framework. As the code for the front-end is written in TypeScript, this tool makes the testing process simple and efficient.

Prior to running the test cases, a “beforeAll” function is executed to initialize constant variables which will be used in the following tests. The variables that are initialized and used are:

1. serviceEvents - Contains all events which will be displayed on the dashboard. Each event contains various historical information of each collection event.
2. contaminantsOne - a list of contaminants found in a service event
3. contaminantsTwo - a list of contaminants found in a service event

It is to be noted that these tests only verify calculations made on the front-end. It assumes that the result from StreamSight API are accurate and return data based on interfaces which are consistent with the dashboard

Test Name	Test Description	Expected Result
Test Get Dates Collected Function	Provided '2022/04/01', '2022/04/03' dates convert Dates to Days of Week. return an array of days	[Friday, Sunday]

Test Get Recent Collection Dates	Provided service events, return an array of unique dates	'2022-04-03'
Test Get Total Contaminants	Provided service events, return the total count of contaminants within all service events	4
Test Get Unique Offenders	Provided service events, return how many unique households had contaminants	2
Test Get Bins Collected	Provided service events, return how many total bins were collected	3
Test Get Rare Contaminants	Provided service events, get rare contaminants which have the type RareContaminant. RareContaminant: {name: string, date: string, time: string}	name: 'Leaves', date: 'Sat Apr 02 2022', time: '6:00:00 PM'
Test Get Contamination Rate	Provided the contaminant total, and bins collected, return contamination rate	29
Test Get Bin Set Out Rate	Provided the bins collected total, and total number of houses, return bin set out rate	52
Test Get Pie Data	Provided service events, return an array of contaminants formatted with the type PieData.	<pre>[{ name: 'Leaves', value: 1, }, { name: 'Black garbage bag', value: 2, }, { name: 'Pizza box', value: 1, },];</pre>
Test Get Collection Date Function	Given a day, get the previous collection date for that day. Given "wednesday" return date	March 30th 2022

Dashboard Testing Results

After running the dashboard tests, results as seen below will be outputted on a successful execution. It provides information on how many tests were successful, where the failures occurred, and additional test coverage details.

File	% Stmts	% Branch	% Funcs	% Lines
All files	89.47	78.88	100	89.47
components/utils	89.33	78.88	100	89.33
calculationHelper.ts	89.33	78.88	100	89.33
lib/models	100	100	100	100
rareContaminant.ts	100	100	100	100

Test Suites: 1 passed, 1 total
Tests: 10 passed, 10 total
Snapshots: 0 total
Time: 3.802 s
Ran all test suites matching /.\\apps\\copywaste\\src\\i.
Done in 5.43s.

```
$ yarn run jest-copywaste-test
yarn run v1.22.17
$ npx jest ".\\apps\\copywaste\\src\\" -c ".\\apps\\copywaste\\jest.config.js" --runInBand
PASS copywaste apps/copywaste/src/test/calculations.spec.ts
  Calculation Tests
    ✓ should get collection date for day (3 ms)
    ✓ should get recent collection dates (1 ms)
    ✓ should convert Dates to Days of Week (23 ms)
    ✓ should get total contaminants as 4 (1 ms)
    ✓ should get get unique offenders as 2
    ✓ should get get bins collected as 3
    ✓ should return rare contaminant object for leaves (3 ms)
    ✓ should return contamination rate as 29%
    ✓ should return bin set out rate as 52%
    ✓ should return pie data (1 ms)
```

Copy-Paste Data Augmentation Pipeline Tests

The Copy-Paste Data Augmentation Pipeline will be tested using the Python library Pytest. Pytest is a testing framework which provides useful features such as fixtures and assert statements which are used often in these tests. Using fixtures, a set of test environment variables can be created and passed to any test function, these can all be found in the conftest.py file. Assert statements are found at the end of each test function which consists of the word 'assert' followed by a statement that results in a boolean such as result == 0. If the result is true then the test passed and vice-versa.


Test Name	Test Description	Expected Result
test_get_next_image_id_returns_zero	Pass a path which points to an empty annotations file. Given an empty file, the get_next_image_id function is	0

o	expected to return an id of zero.	
test_get_next_image_id_returns_next	Pass a path which points to an annotation file. Given an annotation file, the get_next_image_id function is expected to return the last image id + 1.	3
test_get_next_annotation_id_returns_next	Pass a path which points to an annotation file. Given an annotation file, the get_next_annotation_id function is expected to return the last annotation id + 1.	6
test_get_next_annotation_id_returns_zero	Pass a path which points to an empty annotations file. Given an empty file, the get_next_annotation_id function is expected to return an id of zero.	0
test_build_image_json_with_raw_directory	The build_image_json function has two paths, one where the path parameter contains "Raw Images" and one where it does not. In this test a path with "Raw Images" is passed and the expected result is that the function will remove this from the file_name. The passed parameters id, width, and height should all be set properly.	File name does not contain "Raw Images", id is 0, width is 200, height is 100.
test_build_image_json_without_raw_directory	The build_image_json function has two paths, one where the path parameter contains "Raw Images" and one where it does not. In this test a path with "Raw Images" is passed and the expected result is that the function will remove this from the file_name. The passed parameters id, width, and height should all be set properly.	File name does not contain "Raw Images", id is 0, width is 200, height is 100.
test_new_annotations	A clipped polygon mask, segmentation, mask index, category id, and image index are sent to new_annotation_json and the expected result is that all passed parameters can be found in the returned variable.	new_annotation_json[id] == mask_index, new_annotation_json[image_id] == image_index, new_annotation_json[category_id] == category_id, new_annotation_json['segmentation'] == segmentation
test_mask_to_polygons_returns_a_polygon	Assures that the returned variable from mask_to_polygons is of type Polygon.	Type of returned variable is Polygon
test_polygon_to_list_returns_list	Assure that the returned variable from polygon_to_list is of type list.	Type of returned variable is list

test_check_for_multiple_contours_returns_true	Pass the function a set of objects containing multiple contours, the function should return true.	True
test_check_for_multiple_contours_returns_false	Pass the function a set of objects containing multiple contours, the function should return true.	False
test_clip_polygon_masks	Pass a set of masks which overlap and assure that after being passed to the clip_polygon_masks, the clipped masks pass the is_polygon_inside_polygon test.	Clipped masks are not inside each other
test_verify_flags_bisected	The verify flags function accepts a set of flags and whether or not they are set or not. A set of flags where the reject_bisected flag is true is passed to verify_flags and this function should return false because a flag has been set off. The returned message should also include bisected	result is False, message contains "bisected"
test_verify_flags_polygon_in_polygon	The verify flags function accepts a set of flags and whether or not they are set or not. A set of flags where the reject_bisected flag is false and the reject_polygon_in_polygon flag is true is passed to verify_flags and this function should return false because a flag has been set off. The returned message should also include "Polygon inside Polygon".	Result is false, the message is 'Polygon inside Polygon'
test_verify_flags_passed	Pass a flag variable where both previously mentioned flags are set to false. Verify flags should return True and "Passed"	Result is true, the message is "Passed"
test_is_polygon_bisected_true	Pass an array of masks which are bisected to is_polygon_bisected.	Returns true
test_is_polygon_bisected_false	Pass an array of masks which are not bisected to is_polygon_bisected.	Returns false.
test_is_polygon_inside_polygon_true	Pass an array of masks which overlap to is_polygon_inside_polygon.	Returns true
test_is_polygon_inside_polygon_false	Pass an array of masks which do not overlap to is_polygon_inside_polygon.	Returns false


test_augment_data_raises_error_loading_original	Pass the augment_data function data where the function will fail to load images.	Result == 'Rejected', raises an error named 'Error loading image'
test_augment_data_raises_error_loading_artificial	Pass the augment_data function data where the function will fail to load artificial images.	Result == 'Rejected', Raises an error named 'Error loading Artificial image'
test_augment_data_raises_error_extracting_masks	Pass the augment_data function data where the function will fail to extract masks.	Result == 'Rejected' Raises an error named 'Error extracting masks'
test_augment_data_returns_successfully	Pass the augment_data function data where the function will return successfully.	Result != 'Rejected'
test_add_annotation_json	Add an annotation to an annotations_json_base variable which previously had no annotations.	annotations_json_base['annotations'] is not None
test_add_annotation_json_empty_base_raises_error	Attempt to add an annotation to the annotations_json_base before initializing it.	Raises an error named 'JSON not initialised'
test_add_image_json	Add an image to an annotations_json_base variable which previously had no image.	annotations_json_base['images'] is not None and the passed image can be found in annotations_json_base
test_add_image_json_empty_base_raises_error	Attempt to add an image to the annotations_json_base before initializing it.	Raises an error named 'JSON not initialised'

Copy Paste Augmentation Pipeline Testing Results

Run:  pytest in test_copypaste_coco.py ×

Test Results 7 ms

- test_copypaste_coco 7 ms
 - test_augment_data_raises_error_loading_original 3 ms
 - test_augment_data_raises_error_loading_artificial 2 ms
 - test_augment_data_raises_error_extracting_masks 2 ms
 - test_add_annotation_json 0 ms
 - test_add_annotation_json_empty_base_raises_error 0 ms
 - test_add_image_json 0 ms
 - test_add_image_json_empty_base_raises_error 0 ms

Run:  pytest in test_copypaste_utils.py ×

Test Results 38 ms

- test_copypaste_utils 38 ms
 - test_new_annotations 11 ms
 - test_mask_to_polygons_returns_a_polygon 1 ms
 - test_polygon_to_list_returns_list 1 ms
 - test_check_for_multiple_contours_returns_false 5 ms
 - test_clip_polygon_masks 4 ms
 - test_verify_flags_bisected 0 ms
 - test_verify_flags_polygon_in_polygon 2 ms
 - test_verify_flags_passed 4 ms
 - test_is_polygon_bisected_true 0 ms
 - test_is_polygon_bisected_false 3 ms
 - test_is_polygon_inside_polygon_true 2 ms
 - test_is_polygon_inside_polygon_false 5 ms

