Spoonacular (Harold):

Spoonacular					
mapIngredients					
No.	**Description**	**Input**	**Expected Output**	**Actual Output**	**Result**
1	Good	["flour", "bacon", "oil"]	['UPC#1','UPC#2','UPC#n']	['UPC#1','UPC#2','UPC#n']	TRUE
2	Bad	[]	[]	[]	TRUE
3	Bad	None	"Not enough input arguments."	requests.exceptions.JSONDecodeError: Expecting value: line 1 column 1 (char 0)	FALSE
4	Bad	["safjkdkbdvbl"]	[]	[]	TRUE
searchRestaurant					
No.	**Description**	**Input**	**Expected Output**	**Actual Output**	**Result**
5	Good	[43.00441722228767, -81.27617531329763, "Italian"]	King Richie's Pizzeria	King Richie's Pizzeria	TRUE
6	Bad	[missing word (cuisine)]	King Richie's Pizzeria	7-Eleven	FALSE
7	Bad	[invalid word (cuisine)]	Error	IndexError: list index out of range	FALSE
8	Bad	[missing first (lat) AND/OR second number(lng)]	valueError	"API was unable to find a restaurant nearby or timed out. Try again?", TypeError: 'NoneType' object is not subscriptable	FALSE
9	Bad	[]	valueError	"API was unable to find a restaurant nearby or timed out. Try again?", TypeError: 'NoneType' object is not subscriptable	FALSE
10	Bad	None	valueError	"Searching API", KeyError: 'restaurants'	FALSE
11	Boundary	[lat < -90 OR lng <-180]	valueError	IndexError: list index out of range	FALSE
12	Boundary	[lat >90 OR lng >180]	valueError	IndexError: list index out of range	FALSE

MapIngredients errors: Most tests were true/successful. However, the error of inputting None into the function in test no. 3 did not return the expected output of having not enough input arguments. The function executed normally, then returned a result where the built-in

JSON decoder could not decode. A mitigation of this error would be to check that input is not equal to None type. However, this is not likely to happen as entering an empty list will not cause an app-breaking error and that an empty list is declared in the main .py.

SearchRestaurant errors: Most tests were false/unsuccessful towards an expected error. For example, number 7 returns an empty list but the expected output was any error. Number 8 and 9 were caught using the try-except block, but still returned an error as pytest tried to iterate through a variable that was not returned or declared (NoneType). A mitigation could be to not iterate through the results file if its datatype is None. Errors 10 to 12 describe similar behaviour.

EdamamAPI_v2 (Andrew):

EdamamAPI_v2					
testQuery					
No.	**Description**	**Input**	**Expected Output**	**Actual Output**	**Result**
1	Good	"chicken"	TRUE	TRUE	TRUE
2	Bad	1234	TRUE	TRUE	TRUE
3	Bad	None	TRUE	TRUE	TRUE
4	Bad	1274y27ehquirh283ygiqw	TRUE	TRUE	TRUE
testOutputCount					
No.	**Description**	**Input**	**Expected Output**	**Actual Output**	**Result**
5	Good	3	TRUE	TRUE	TRUE
6	Bad	"123abc"	FALSE	FALSE	TRUE
7	Bad	None	FALSE	FALSE	TRUE
8	Bad	{'test':123}	FALSE	FALSE	TRUE
9	Boundary	-10	FALSE	FALSE	TRUE
10	Boundary	20	FALSE	FALSE	TRUE
11	Boundary	0	TRUE	TRUE	TRUE
12	Boundary	10	TRUE	TRUE	TRUE

testQuery: 1 failure was found for when the input was not a string, the expected outcome was False when it was True in actuality. After some investigation, I found that because of the way I formatted the request URL, the integers were automatically converted to strings which allowed it to fail the first if statement check as expected. However, after further investigation, I found that not all integers will result in an unsuccessful search, which means this specific test case was just luck. This bug was addressed by checking the query type before each search, and immediately returning an empty dictionary if the input is not a string. This change also allowed me to remove some redundant functions.

testOutputCount: 1 failure was found for when the test cases were out of bounds, where the upper-bound test failed. The problem appears to be hitting the maximum rate limit allowable for the API per minute, stopping the program after 17 recipes, preventing it from reaching the original maximum expected of 20. There is nothing that can be done to “fix” this, aside from purchasing a higher rate limit on our end. However, this does not actually pose any real issue as it is unrealistic for us to do more than 10 recipe requests (the new limit). The alternative solution we opted for is to hard cap the number of recipe requests that can be made regardless of how large the limit number is.

Geopify (KwanNok):

Geopify					
TestQueryInput					
No.	**Description**	**Input**	**Expected Output**	**Actual Output**	**Result**
1	Good	"N6G5R6"	TRUE	TRUE	TRUE
2	Bad	1234	TRUE	TRUE	TRUE
3	Bad	None	TRUE	TRUE	TRUE
4	Bad	31j2;lkh12	TRUE	TRUE	TRUE
checkOutput					
No.	**Description**	**Input**	**Expected Output**	**Actual Output**	**Result**
5	Good	“n6g5r6”	TRUE	TRUE	TRUE
6	Bad	None	FALSE	FALSE	TRUE
7	Bad	{'test':123}	FALSE	FALSE	TRUE
8	Good	“1600 Amphitheatre Parkway, Mountain View, CA”	37 <= latitude <= 38 and -123 <= longtitude <= -122	TRUE	TRUE
9	Bad	"123abc"	Status code: 400	FALSE	TRUE

TestQueryInput: Three out of four tests were successful. The 1 failure discovered was caused by the difference in expected and actual output. Where it expects FALSE but detected TRUE. It was found to be an error in formatting the input query. It was then solved by re-ensuring the format of query inputs to avoid this bug from happening again.

TestOutput: 2 Failure was spotted. Where the terminal displays error code: TestOutput::test_get_coordinates_empty_address - assert (44.933143, 7.540121) == False

TestOutput::test_get_coordinates_invalid_address - IndexError: list index out of range

The first failure was caused by the inaccuracy of the result values. The resulting longitude and latitude were far away from the expected range. Therefore, we changed the detecting range for expected output to address the issue. The second failure was discoverd to be the same issue as the one received in TestQueryInput. The same solution was applied to address it.

OpenFoodFacts(Muhammad Ahmad)

OpenFoodFacts					
TestQuery					
No.	Description	Input	Expected Output	Actual Output	Result
1	Good	[]	0	N/A	N/A
2	Good	[3017624010701, 9300650271135, 2324343434]	41	41	TRUE
3	Good	None	0	N/A	N/A
4	Good	["jkhklj"]	0	0	TRUE

TestQuery: Two outputs were successful, and the other two didn’t yield any result, even though the expected output was zero. However, when tried later in the terminal, the results of #1 and #2 appeared as zero. Since no other bugs were found, the code seems to be running according to expectations.