**TEST PLAN**

**(You can find and run autotest selection in Python-Project.ipynb)**

***get\_nearby\_places(api\_key, latitude, longitude, radius, place\_type)***

**Test Scenarios:**

Provide valid inputs and ensure it returns nearby places.

Data: Valid API key, latitude, longitude, radius, and place type.

Expected Outcome: List of nearby places within the specified radius.

Test with an invalid API key and ensure it handles the error gracefully.

Data: Invalid API key, valid latitude, longitude, radius, and place type.

Expected Outcome: Proper error handling for invalid API key.

Test with invalid latitude or longitude and ensure it handles the error gracefully.

Data: Valid API key, invalid latitude or longitude, valid radius, and place type.

Expected Outcome: Proper error handling for invalid latitude or longitude.

Test with a large radius and ensure it returns appropriate results.

Data: Valid API key, latitude, longitude, large radius, and place type.

Expected Outcome: List of nearby places within the specified large radius.

***check\_s3\_connection()***

**Test Scenarios:**

Test connection to S3 with valid credentials.

Data: Valid AWS credentials.

Expected Outcome: Successful connection to S3.

Test with invalid AWS credentials and ensure it handles the error gracefully.

Data: Invalid AWS credentials.

Expected Outcome: Proper error handling for invalid AWS credentials.

***save\_to\_s3(data, file\_name)***

**Test Scenarios:**

Test saving data to S3 with valid inputs.

Data: Valid data and file name.

Expected Outcome: Data saved to the S3 bucket with the specified file name.

Test saving data to S3 with an invalid file name and ensure it handles the error gracefully.

Data: Valid data, invalid file name.

Expected Outcome: Proper error handling for invalid file name.

Test saving data to S3 when S3 connection fails and ensure it handles the error gracefully.

Data: Valid data and file name, failed S3 connection.

Expected Outcome: Proper error handling for failed S3 connection.

***filter\_places\_by\_keyword(data, keyword)***

**Test Scenarios:**

Provide valid place data and keyword, ensure it filters places correctly.

Data: Valid place data and keyword.

Expected Outcome: Filtered list of places containing the keyword.

Test with no keyword provided and ensure it returns all places.

Data: Valid place data, no keyword provided.

Expected Outcome: Unfiltered list of places.

Test with an empty list of places and ensure it handles gracefully.

Data: Empty list of places, valid keyword.

Expected Outcome: Proper handling for an empty list of places.

***list\_files\_from\_s3()***

**Test Scenarios:**

Test listing files from S3 when files are present.

Data: Files present in the S3 bucket.

Expected Outcome: List of files present in the S3 bucket.

Test listing files from S3 when no files are present.

Data: No files present in the S3 bucket.

Expected Outcome: Proper handling for no files present in the S3 bucket.

Test listing files from S3 when S3 connection fails and ensure it handles the error gracefully.

Data: Failed S3 connection.

Expected Outcome: Proper error handling for failed S3 connection.

***show\_file\_info\_from\_s3(file\_name)***

**Test Scenarios:**

Test showing file info from S3 with a valid file name.

Data: Valid file name.

Expected Outcome: Contents of the specified file displayed.

Test showing file info from S3 with an invalid file name and ensure it handles the error gracefully.

Data: Invalid file name.

Expected Outcome: Proper error handling for invalid file name.

Test showing file info from S3 when S3 connection fails and ensure it handles the error gracefully.

Data: Failed S3 connection.

Expected Outcome: Proper error handling for failed S3 connection.

***delete\_file\_from\_s3(file\_name)***

**Test Scenarios:**

Test deleting a file from S3 with a valid file name.

Data: Valid file name.

Expected Outcome: The specified file is deleted from the S3 bucket.

Test deleting a file from S3 with an invalid file name and ensure it handles the error gracefully.

Data: Invalid file name.

Expected Outcome: Proper error handling for invalid file name.

Test deleting a file from S3 when S3 connection fails and ensure it handles the error gracefully.

Data: Failed S3 connection.

Expected Outcome: Proper error handling for failed S3 connection.

***delete\_files\_containing\_keyword(keyword***)

**Test Scenarios:**

Test deleting files containing a specific keyword.

Data: Keyword present in some file names.

Expected Outcome: Files containing the keyword are deleted from the S3 bucket.

Test deleting files when no files contain the specified keyword.

Data: Keyword not present in any file name.

Expected Outcome: Proper handling for no files containing the specified keyword.

Test deleting files when S3 connection fails and ensure it handles the error gracefully.

Data: Failed S3 connection.

Expected Outcome: Proper error handling for failed S3 connection.

***process\_data\_by\_keyword(keyword)***

**Test Scenarios:**

Test processing and saving data containing a specific keyword.

Data: Data containing the keyword.

Expected Outcome: Information from files containing the keyword is processed and saved to a new file.

Test processing data when no files contain the specified keyword.

Data: No files containing the keyword.

Expected Outcome: Proper handling for no files containing the specified keyword.

Test processing data when S3 connection fails and ensure it handles the error gracefully.

Data: Failed S3 connection.

Expected Outcome: Proper error handling for failed S3 connection.

***haversine\_distance(lat1, lon1, lat2, lon2)***

**Test Scenarios:**

Test calculating the distance between two valid coordinates.

Data: Valid latitude and longitude coordinates.

Expected Outcome: Distance between the coordinates calculated accurately.

Test with invalid latitude or longitude values and ensure it handles the error gracefully.

Data: Invalid latitude or longitude values.

Expected Outcome: Proper error handling for invalid latitude or longitude values.

Test with identical coordinates and ensure it returns a distance of 0.

Data: Identical coordinates.

Expected Outcome: Distance between identical coordinates is 0.

***parse\_json\_request(event)***

**Test Scenarios:**

Test parsing a valid JSON request.

Data: Valid JSON request.

Expected Outcome: JSON request parsed successfully.

Test parsing an invalid JSON request and ensure it handles the error gracefully.

Data: Invalid JSON request.

Expected Outcome: Proper error handling for invalid JSON request.

***lambda\_handler(event, context)***

**Test Scenarios:**

Test handling different actions such as list, parse, show, delete, delete by keyword, and save by keyword.

Data: Various actions in the event.

Expected Outcome: Proper handling of each action.

Test handling invalid request formats and ensuring it returns appropriate error responses.

Data: Invalid request formats.

Expected Outcome: Proper error handling for invalid request formats.

Test calling the main function with various combinations of input parameters and ensure it behaves as expected.

Data: Various combinations of input parameters.

Expected Outcome: Proper execution of the main function with each combination of input parameters.

***check\_positive\_number(number)***

**Test Scenarios:**

Test with a positive integer.

Test with a positive float.

Test with zero and negative numbers.

Test with non-numeric input.

***check\_latitude\_longitude\_digit(latitude, longitude)***

**Test Scenarios:**

Test with valid latitude and longitude digits.

Test with non-digit latitude or longitude.

Test with non-numeric input.

***check\_latitude\_longitude\_range(latitude, longitude)***

**Test Scenarios:**

Test with latitude and longitude within the valid range.

Test with latitude or longitude exceeding the valid range.

Test with non-numeric latitude or longitude.

***Stress Test for get\_nearby\_places(api\_key, latitude, longitude, radius, place\_type)***

**Test Scenario:** Send a large number of concurrent requests to fetch nearby places.

***Stress Test for save\_to\_s3(data, file\_name)***

**Test Scenario:** Attempt to save a large amount of data to S3 simultaneously.

***Stress Test for list\_files\_from\_s3()***

**Test Scenario:** List files from S3 when the bucket contains a large number of files.

***Stress Test for show\_file\_info\_from\_s3(file\_name)***

**Test Scenario:** Retrieve information from a large number of files stored in S3.

***Stress Test for delete\_file\_from\_s3(file\_name)***

**Test Scenario:** Delete a large number of files from S3 concurrently.

***Stress Test for delete\_files\_containing\_keyword(keyword)***

**Test Scenario:** Delete files containing a specific keyword from S3 when there is a large number of such files.

***Stress Test for process\_data\_by\_keyword(keyword)***

**Test Scenario:** Process data from a large number of files stored in S3 and generate a new file based on a keyword.

***Stress Test for main(radius, latitude, longitude, place\_type, keyword=None, no\_keyword=None)***

**Test Scenario:** Simulate a high load scenario by invoking the main function with a large number of concurrent requests containing various combinations of parameters.

***Integration Testing:***

**Test Scenarios:**

Test the entire workflow from receiving a request to processing and storing data in S3.

Test the Lambda function's behavior under load and ensure it scales appropriately.

Test the Lambda function's error handling and ensure it provides informative error messages.