**Monitoring Methane**

**From Space**

**Test Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Position** | **email** | **phone** |
| **Zachary Qi Jie Teng** | Team Leader / Supervisor Liaison / Usability 2nd Lead | [102416353@student.swin.edu.au](mailto:102416353@student.swin.edu.au) | 0424393852 |
| **Md Radif Rafayet Chowdhury** | Documentation Lead / Quality Lead / Client Liaison | [103539316@student.swin.edu.au](mailto:103539316@student.swin.edu.au) | 0431747026 |
| **Uddhav Grover** | Research Lead / Planning Lead | [103513802@student.swin.edu.au](mailto:103513802@student.swin.edu.au) | 0479080083 |
| **Ho Man Lai** | Documentation 2nd Lead / Usability Lead | [103495104@student.swin.edu.au](mailto:103495104@student.swin.edu.au) | 0404620685 |
| **Saborni Barua** | Git Lead / Trello Lead / Developer 2nd Lead | [103512168@student.swin.edu.au](mailto:103512168@student.swin.edu.au) | 0448445167 |
| **Ang Fu** | Developer Lead / Research 2nd Lead | [103001255@student.swin.edu.au](mailto:103001255@student.swin.edu.au) | 0418376938 |

*SWE40002, Software Engineering Project B, Semester 2 2023*

**Table 1. Document Change Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Authors** | **Summary of Changes** |
| 1.00 | 13/08/2023 | All | Initial Draft |
| 1.10 | 25/10/2023 | All | Completed version |

**Table 2. Document Sign Off**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Position** | **Signature** | **Date** |
| **Zachary Qi Jie Teng** | Team Leader / Supervisor Liaison / Usability 2nd Lead | Zachary Teng | 23/08/2023 |
| **Md Radif Rafayet Chowdhury** | Documentation Lead / Quality Lead / Client Lead | Radif Rafayet | 23/08/2023 |
| **Uddhav Grover** | Research Lead / Planning Lead | Uddhav Grover | 23/08/2023 |
| **Ho Man Lai** | Documentation 2nd Lead / Usability Lead | Ho Man Lai | 23/08/2023 |
| **Saborni Barua** | Git Lead / Trello Lead / Developer 2nd Lead | Saborni Barua | 23/08/2023 |
| **Ang Fu** | Developer Lead / Research 2nd Lead | Ang Fu | 23/08/2023 |

# Table of Contents

[Table of Contents 3](#_Toc149162609)

[1. Introduction 4](#_Toc149162610)

[1.1. Purpose of the test plan 4](#_Toc149162611)

[1.2. Scope 4](#_Toc149162612)

[1.3) References Material 5](#_Toc149162613)

[1.4) Objective 5](#_Toc149162614)

[1.5) Resources Required 5](#_Toc149162615)

[1.6) Environment Requirements 5](#_Toc149162616)

[2. Test Items 6](#_Toc149162617)

[2.1 Features to be Tested 6](#_Toc149162618)

[2.2 Test Cases 6](#_Toc149162619)

[2.2.1 Functional Test Cases 6](#_Toc149162620)

[3.Strategy 22](#_Toc149162621)

[3.1. Roles and Responsibilities 22](#_Toc149162622)

[3.2. Test Deliverables 23](#_Toc149162623)

[3.3. Schedule 23](#_Toc149162624)

[3.4. Risk and Contingency 23](#_Toc149162625)

[3.5. Testing Tasks 23](#_Toc149162626)

[4. Pass/Fail Criteria 24](#_Toc149162627)

[**4.1 Product Level** 24](#_Toc149162628)

[**4.2. Testing Stages** 25](#_Toc149162630)

[4.3. Suspension criteria and resumption requirements 25](#_Toc149162631)

[4.3.1. Suspension Criteria 25](#_Toc149162632)

[4.3.2. Resumption Requirements 25](#_Toc149162633)

[4.4. Approvals 26](#_Toc149162634)

# 1. Introduction

## 1.1. Purpose of the test plan

The test plan is a document designed to outline the scope, resources, approach, strategy and criteria for all testing activities carried out during and after the development of the system. The plan will clearly identify the features that are to be tested and the stages at which tests are to be carried out. We intend to cover all the features and authenticate that the products work as planned. It allows the team to highlight and pre-plan the required steps during the testing phase.

## 1.2. Scope

Testing will be carried out at several stages of the life cycle as the system is developed. Testing will ensure system defects, bugs to be caught early in the development and helps the team reevaluate their approach to developing a particular aspect of the project. Testing will be carried out at several stages of the life cycle as the system is developed. The focus will be on both unit and integration testing. Specifically, unit testing will target Python functions like **‘search\_catalog’** and **‘convert\_format\_date’**, while integration testing will evaluate how these functions interact with other system components.

**Constraints/Assumptions**

* The tests will assume that the user has a basic understanding of python programming language.
* The system will be tested under the assumption that the users have access to the internet.
* The tests will assume that the requirements.txt file have been established in the user’s machine.

The team will be following a two-level testing strategy. This will allow the project to be tested at different development stages at maximum efficiency.

* Unit Testing
* Integration Testing

## 1.3) References Material

The following document has been created according to the details specified in the documents below. The documents below will provide further information on the project.

* Project Plan
* Software Quality Assurance Plan
* Software Requirements Specification
* IEEE 829 Standard template (Test Plan)

## 1.4) Objective

Carry out appropriate tests at appropriate stages of development.

The test plan shall act as a guideline for performing these different types of tests required for maintaining the quality and efficiency of the system and to detect flaws at any step of the development process.

## 1.5) Resources Required

* Internet
* Jupyter Notebook
* Python IDE
* IDE: Visual Studio Code
* Unittest library
* Microsoft Planetary Hub

## 1.6) Environment Requirements

* Computer system to be compatible with the software.
* Requirements.txt file established in the computer system.
* Internet
* Notes application (new discoveries of bugs and anomalies should be listed)

# 2. Test Items

The product to be tested is the methane monitor pipeline.

## 2.1 Features to be Tested

The following features are the primary focus of the testing activities:

**Unit Testing**

* **search\_catalog function:** This function is responsible for validating bounding box and date period parameters. It is critical for querying satellite data based on geographical and temporal parameters.

**Test Case 1:** Valid Bounding Box

**Test Case 2:** Invalid Bounding Box

* **convert\_format\_date function:** This function is responsible for validating various date formats. It is essential for handling date conversions within the system.

**Test Cases:** Various formats including "YYYY-MM-DD", "DD MM YYYY", etc.

**Integration Testing**

Testing how the search\_catalog and convert\_format\_date functions interact with other system components. This will involve multiple modules of the system being tested together.

## 2.2 Test Cases

### 2.2.1 Functional Test Cases

1. **Unit Testing**
   1. **Individual Components**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Component(s)** | **Description** | **Expected Results** | **Actual Results** | **Passed/Failed** |
| **search\_catalog** Function | Validate bounding box and date period | Correct search parameters for valid inputs | Same as expected result | Passed |
| **convert\_format\_date** Function | Validate date formats | Date in "YYYY-MM-DD" format for valid inputs | Same as expected result | Passed |

* 1. **Unit Testing for ‘search\_catalog’**

**Test Case 1: Valid Bounding Box (search\_catalog)**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Was it Expected?** |
| ‘**region = [145.030035, -37.828963, 145.042158, -37.815471]**’  ‘**date\_period = "2022-09-01/2022-09-30"**’ | ‘{"bbox": [145.030035, -37.828963, 145.042158, -37.815471]}’ | **yes** |

**Test Case 2: Invalid Bounding Box (search\_catalog)**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Was it Expected?** |
| ‘**region = [190, -90, 200, -100]**’  ‘**date\_period = "2022-09-01/2022-09-30"**’ | Raises ‘ ValueError: "Invalid coordinate in bbox" ‘ | **Yes** |

**Test Case 3: Invalid longitude (search\_catalog)**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Was it Expected?** |
| **‘region = [245.030035, -37.828963, 145.042158, -37.815471]’**  ‘**date\_period = "2022-09-01/2022-09-30"**’ | Raises ‘ ValueError: "Invalid longitudes in bbox" ‘ | **Yes** |

**Test Case 4: Invalid latitude (search\_catalog)**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Was it Expected?** |
| **‘region = [145.030035, -97.828963, 145.042158, -37.815471]’**  ‘**date\_period = "2022-09-01/2022-09-30"**’ | Raises ‘ ValueError: "Invalid latitude in bbox" ‘ | **Yes** |

**Test Case 5: Valid country name (search\_catalog)**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Was it Expected?** |
| **‘region = “Japan”’**  ‘**date\_period = "2022-10-01/2022-10-30"**’ | ‘Number of items for input: 62’ | **Yes** |

**Test Case 6: Invalid country name (search\_catalog)**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Was it Expected?** |
| **‘region = “111”’**  ‘**date\_period = "2022-10-01/2022-10-30"**’ | Raises ‘IndexError: “list index out of range”’ | **Yes** |

* 1. **Unit Testing for ‘convert\_format\_date’**

**Test Case 1: Valid Date Format (convert\_format\_date)**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Was it Expected?** |
| ‘**input\_date = "2021 12 31"**’ | ‘"2021-12-31**"’** | **Yes** |

**Test Case 2: Valid Date Format with Different Delimiter (convert\_format\_date)**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Was it Expected?** |
| ‘**input\_date = "31 12 2021"**’ | Raises ‘"2021-12-31**"’** | **Yes** |

**Test Case 3: Invalid Date Variation or Format (convert\_format\_date)**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Was it Expected?** |
| ‘**input\_date = "31 12 21"**’ | Raises ‘ValueError: "Invalid data format" ‘ | **yes** |

**Test Case 4: Invalid Date Components (convert\_format\_date)**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Was it Expected?** |
| ‘**input\_date = "2021 13 01"**’ | Raises ‘ValueError: "Invalid data format"’ | **Yes** |

**Test Case 5: Non-Standard Delimiters (convert\_format\_date)**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Was it Expected?** |
| ‘**input\_date = "2021.12.31"**’ | Raises ‘ValueError: "Invalid data format"’ | **Yes** |

**Test Case 6: Date with Text (convert\_format\_date)**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Was it Expected?** |
| ‘**input\_date = "2021-12-31 ABC"**’ | Raises ‘ValueError: "Invalid data format" ‘ | **Yes** |

**Test Case 7: Empty Input (convert\_format\_date)**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Was it Expected?** |
| ‘**input\_date = ""**’ | Raises ‘ValueError: "Invalid start/end date format. Please check the acceptable formats" ‘ | **Yes** |

**Test Case 8: Non Date Input (convert\_format\_date)**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Was it Expected?** |
| ‘**input\_date = "This is not a date"’** | Raises ‘ValueError: "Invalid start/end date format. Please check the acceptable formats" ‘ | **Yes** |

**Test Case 9: Single Digit Month and Day (convert\_format\_date)**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Was it Expected?** |
| ‘**input\_date = "5 9 2022"’** | Raises ‘"05-09-2022**"’** | **Yes** |

**Test Case 10: Date Boundary (convert\_format\_date)**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Was it Expected?** |
| ‘**input\_date = "0001 01 01"’** | Raises ‘"0001-01-01**"’** | **Yes** |

* 1. **Inputs for Data Validation**

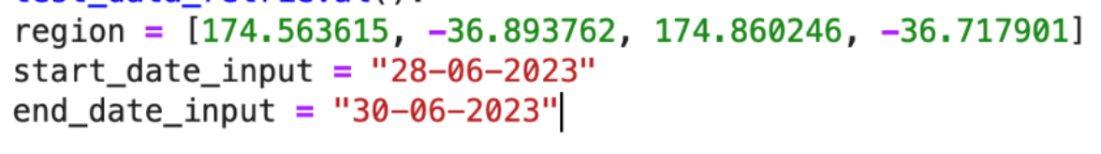
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case(s)** | **Description** | **Expected results** | **Actual Results** | **Passed/Failed** |
| Valid Region and Date | The system should accept valid region and date period | The system should return correct search parameters | Same as expected result | Passed |
| Invalid Date Format | The system should not accept invalid date formats | The system should raise a **ValueError** | Same as expected result | Passed |

**Integration Testing:**

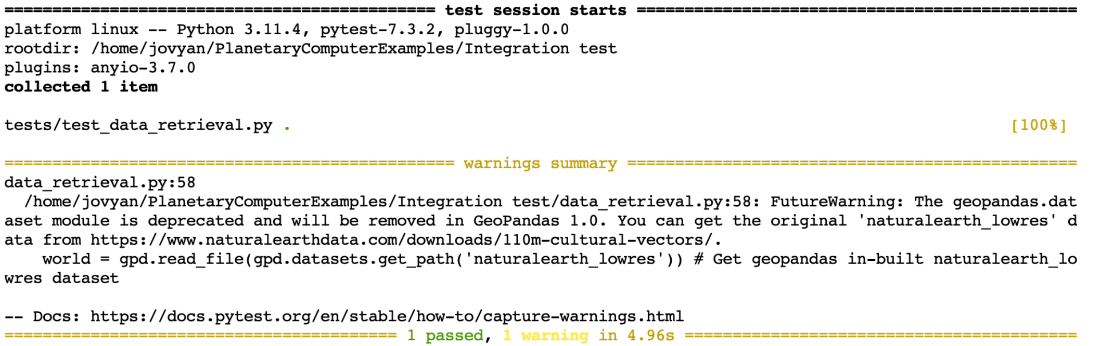
We have conducted integration testing for both user input processing (date format) and data search query functionalities. These two aspects are interdependent, meaning that if there is a failure in the user input processing, it will subsequently result in a failure when searching for data. The user input processing module is subjected to multiple test cases as described below. In cases where valid date formats are provided, the date format would be changed to the default format (YYYY-MM-DD) and the data search query component will successfully retrieve and print items corresponding to the valid dates. Conversely, if the date format is invalid, the search query will generate an error instead of returning data.

**TestCase1: Valid Input Format - Dashes (DD-MM-YYYY):**

**Input:**



**Output:**

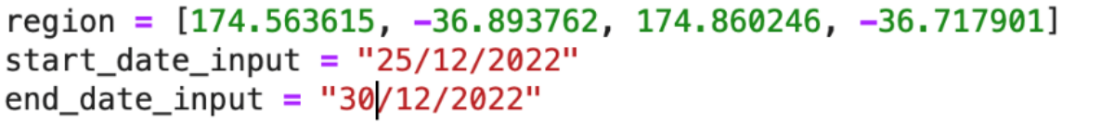


**Expected Output: Pass**

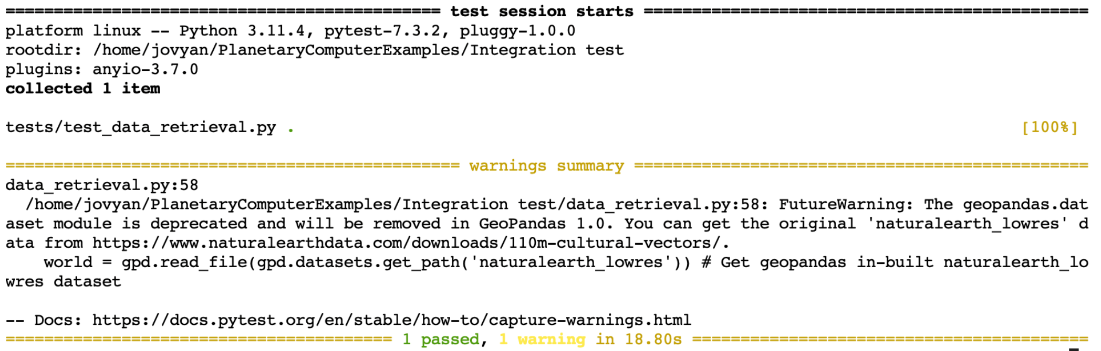
**Real Output: Pass**

**TestCase2: Valid Input Format - Slashes (DD/MM/YYYY):**

**Input:**



**Output:**

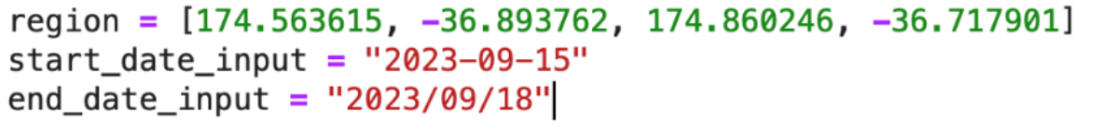


**Expected Output: Pass**

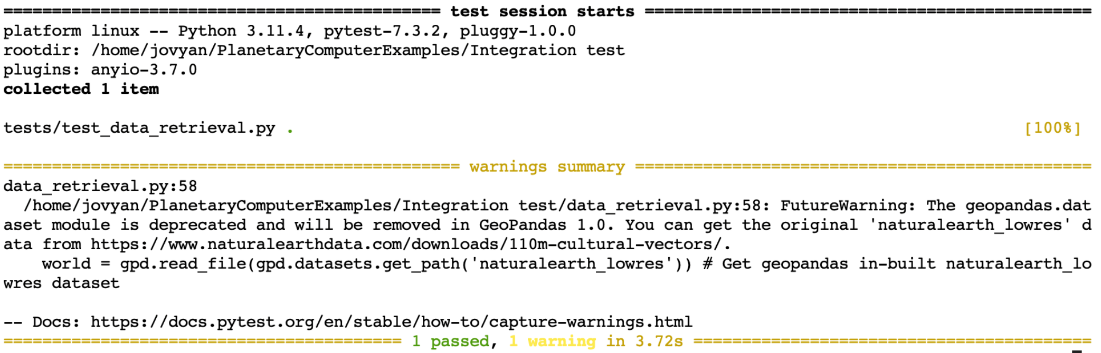
**Real Output: Pass**

**TestCase3: Valid Input Format – accepting different formats (DD-MM-YYYY/YYYY-MM-DD):**

**Input:**



**Output:**

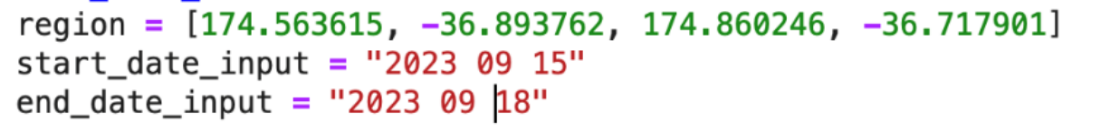


**Expected Output: Pass**

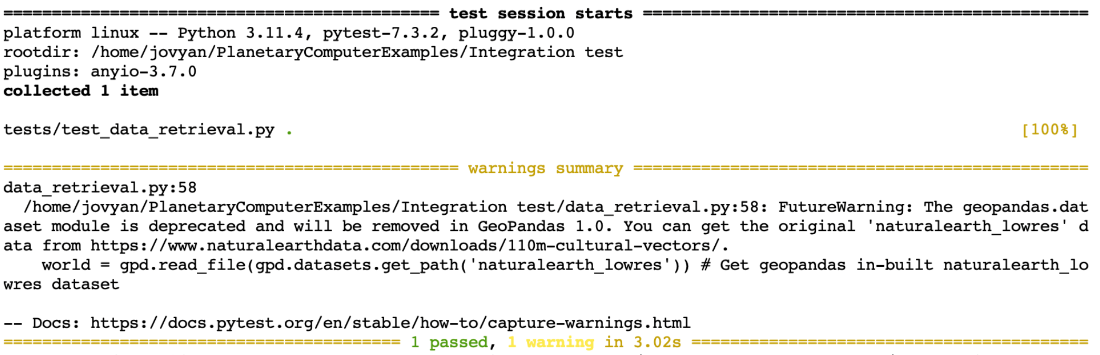
**Real Output: Pass**

**TestCase4: Valid Input Format - Spaces (DD MM YYYY):**

**Input:**



**Output:**

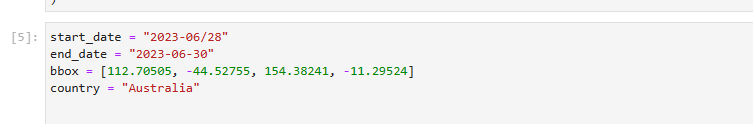


**Expected Output: Pass**

**Real Output: Pass**

**TestCase5: Invalid Input Format (Invalid format with both slashes and dashes)**

**Input:**



**Output:**

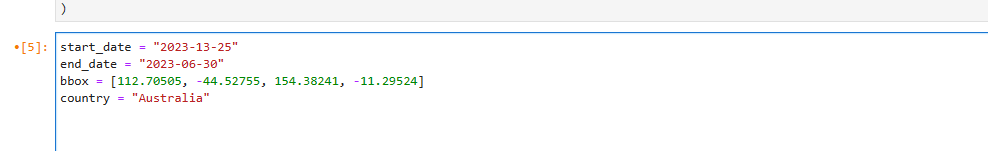
****

**Expected Output: Fail**

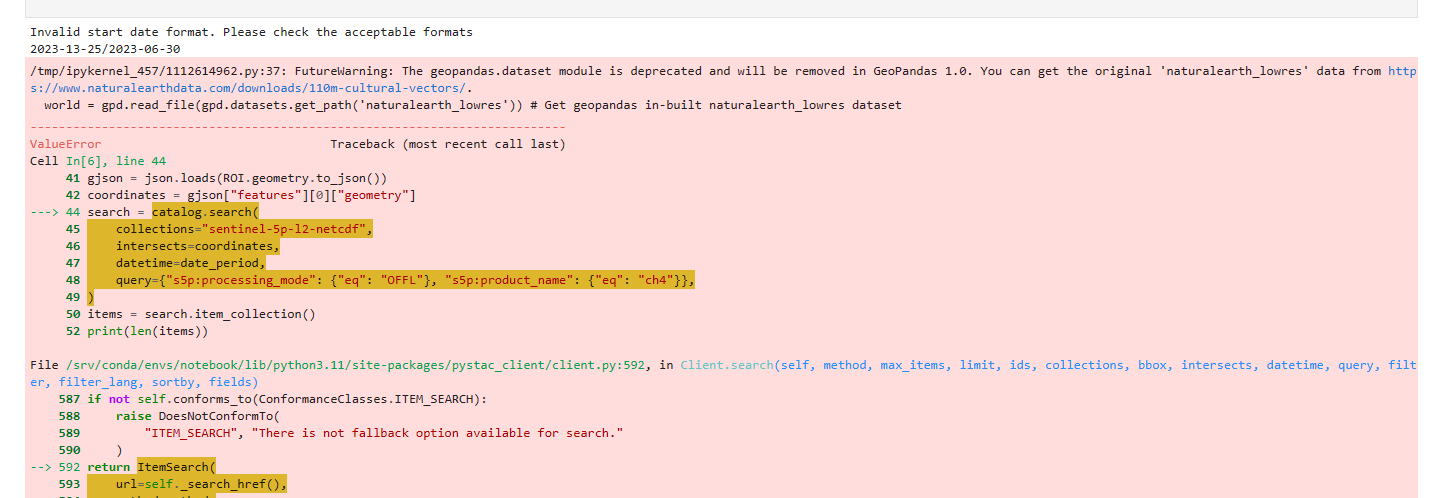
**Real Output: Fail**

**TestCase6: Invalid Input Format (Month 13 is out of range)**

**Input:**



**Output:**

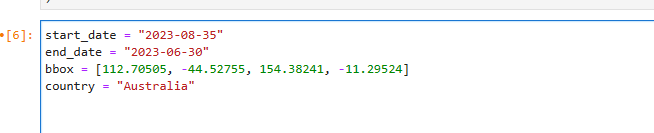


**Expected Output: Fail**

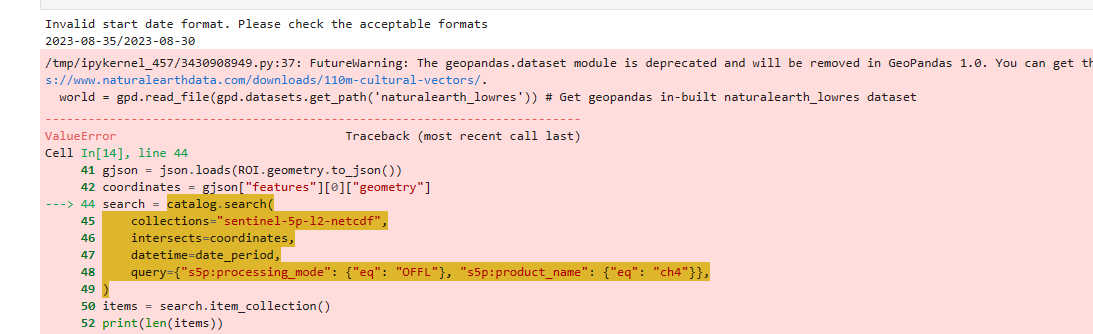
**Real Output: Fail**

**TestCase7: Invalid Input Format (Day 35 is out of range for August)**

**Input:**



**Output:**

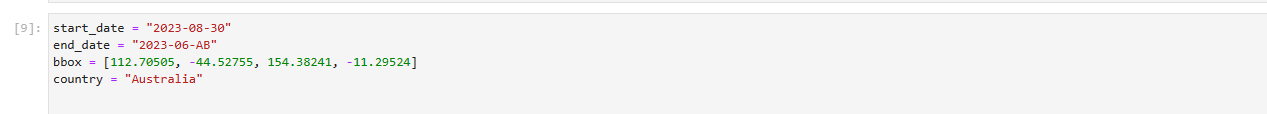


**Expected Output: Fail**

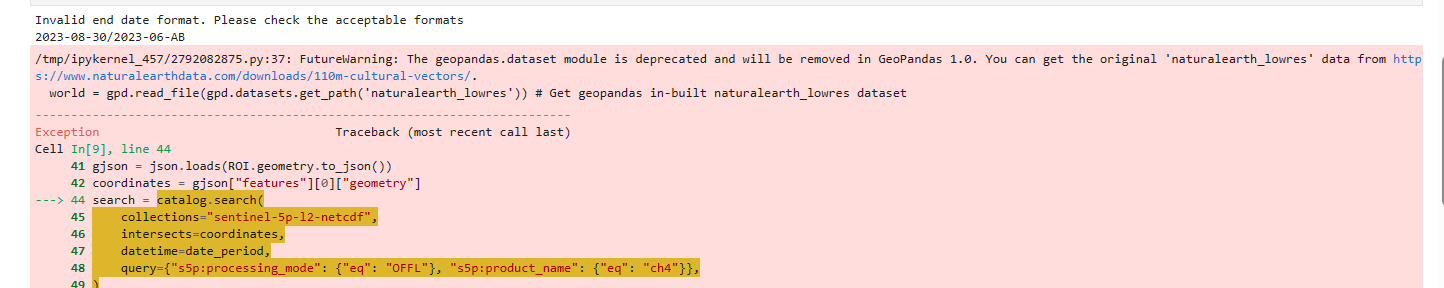
**Real Output: Fail**

**TestCase8: Invalid Input Format (Non-Numeric Characters)**

**Input:**



**Output:**

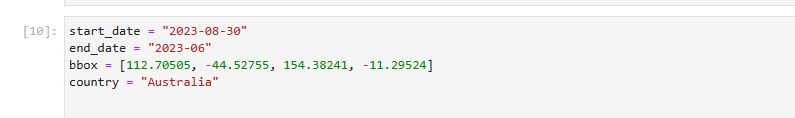


**Expected Output: Fail**

**Real Output: Fail**

**TestCase 9: Invalid Input Format (Incomplete Date)**

**Input:**



**Output:**

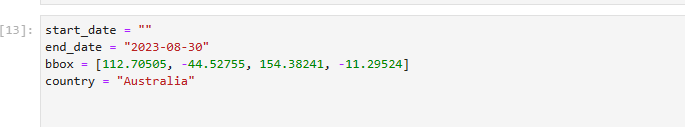


**Expected Output: Fail**

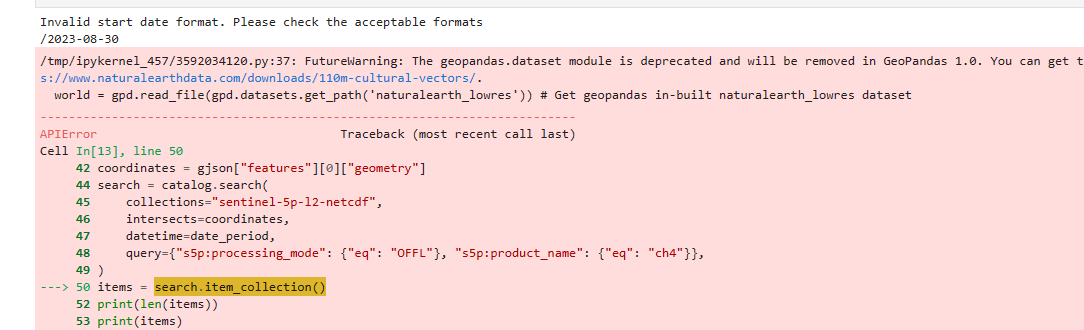
**Real Output: Fail**

**TestCase10: Invalid Input Format (Empty input date)**

**Input:**



**Output:**



**Expected Output: Fail**

**Real Output: Fail**

# 3.Strategy

As stated the project will be tested following a four-level testing strategy

* **Unit Testing –** This stage of testing will be carried out during the initial coding stage when there are multiple modules of code being developed by different members of the team. This will be carried out by generating multiple unit tests to test pieces of code. Will help detect errors early on.
* **Integration Testing –** This stage of testing will be carried out when the system reaches a low-level design and all prior unit tests have been passed. Multiple modules of the system will be tested together using appropriate test cases. This will confirm if various modules function effectively altogether.

## 3.1. Roles and Responsibilities

* **Unit Testing –** Must be carried out by individual developers on a continuous basis by creating unit tests throughout the development of the system.
* **Integration Testing –** May be carried out by the developers as well the team leader/testing managers by using appropriate test cases.

## 3.2. Test Deliverables

* Test Plan
* Test Cases
* Defect/Enhancement logs
* Test Reports
* Usability Test Guide

## 3.3. Schedule

Table 5. Semester 2 Schedule

|  |  |
| --- | --- |
| **Features to test** | **Estimated Testing** |
| Unit testing | Week 9, Semester 2 |
| Integration testing | Week 10, Semester 2 |

## 3.4. Risk and Contingency

Table 7. Risks associated with testing

|  |  |
| --- | --- |
| **Risks** | **Contingency** |
| Incomplete Test Coverage | Ensure that all critical functions ( **‘search\_catalog’** and **‘convert\_format\_date’** ) are covered in both unit and integration testing. |
| Hardware or Software Failure | Have backup hardware and software configurations as specified in the test report to ensure that testing can continue without significant delays. |
| Discrepancies in Expected and Actual Results | Immediate review and debugging to identify the root cause. Update test cases or system code as necessary. |

## 3.5. Testing Tasks

* Testing scenarios shall be taken from the testing team, and documented. The test scenarios will be required before creating the test cases.
* Test cases shall be created for all individual features of the system.
* Required resources shall be accounted for before carrying out the tests.
* Bugs will be logged and reported in the test report and defect/enhancement logs.

# 

# 4. Pass/Fail Criteria

## 4.1 Product Level

**Table 8. Test Case Pass/Fail Criteria**

* Unit Testing

|  |  |  |
| --- | --- | --- |
| **Testing Criteria** | **Pass Criteria** | **Fail Criteria** |
| Criteria for **‘search\_catalog’** Function | The function must correctly validate bounding box and date period parameters. All test cases must produce the expected output. | Any deviation from the expected output in any of the test cases. |
| Criteria for **‘convert\_format\_date’** Function | The function must validate various date formats correctly. All test cases must produce the expected output. | Any deviation from the expected output in any of the test cases. |

## Integration Testing

|  |  |  |
| --- | --- | --- |
| **Test Criteria** | **Pass Criteria** | **Fail Criteria** |
| Criteria for Integration of ‘**search\_catalog’** and ‘**convert\_format\_date’** | The integrated functions must work seamlessly with other system components, producing expected results in all test cases. | Any failure in the integration that leads to incorrect output or system behavior. |

## **4.2. Testing Stages**

During each development stage test will be conducted and judged according to the following criteria

Table 9. Testing level pass/fail criteria

|  |  |  |
| --- | --- | --- |
| **Testing Level** | **Pass criteria** | **Fail criteria** |
| Unit Testing | * All unit tests have been passed successfully * At least 70% of all code written has been covered during testing. * All bugs and errors found have been logged and been accounted for. | * Not all unit tests have been passed * Not enough code written has been included in testing. |
| Integration Testing | * 90% of all modules developed have been tested. * Modules perform their assigned function successfully when tested together. * All issues have been logged and corrected | * Modules fail to carry out their function when put together. * Too many critical issues found. |

## 4.3. Suspension criteria and resumption requirements

### 4.3.1. Suspension Criteria

* The design of the system is found to be defective.
* The client has changed his mind about the specified requirements.
* Code cells that running found to have critical issues and requires fixing before proceeding further

### 4.3.2. Resumption Requirements

* All critical issues found have been fixed.
* The modified code has passed the testing successful
* The new changes have gained the approval from the Test Champion as well as the Team Leader

## 4.4. Approvals

* Both the Team leader and the Test Champion must agree to the completion of a testing level before marked as completed or passed
* Any changes or addition to the system features would have to be approved by the client.