# **Testing documentation**

# Introduction

The Flight Data Lookup and Merge system is designed to provide detailed information about flights and generate comprehensive reports by merging datasets. This testing documentation outlines various test scenarios to ensure the robustness and reliability of the system. It covers functional and non-functional testing aspects to verify the correctness of flight lookup operations and the accuracy of merged data.

# Test Environment Setup

Python Environment:

* Ensure Python 3.x is installed on machine.

Clone the Repository:

* Clone the project repository from the GitHub repository:

|  |
| --- |
| git clone <repository\_url>  cd <repository\_directory> |

Azure Blob Storage Configuration:

* Obtain access to the Azure Blob Storage account specified in the config.json file.
* Update connection\_string and container\_name in config.json with the correct values.

Data Files:

* Ensure that necessary data files (schedule.json, airports.csv, fleet.csv) are available in Azure Blob Storage or locally in the data\_files directory.

Configuration:

* Review and modify config.json to align with the testing environment.

# Execution:

To execute tests, run the ´flight\_data\_app.py` script with appropriate command line arguments.

Example:

python flight\_data\_app.py lookup <flight\_numbers>

python flight\_data\_app.py merge

# Test Cases

**Test Case 1**: Basic Lookup Operation

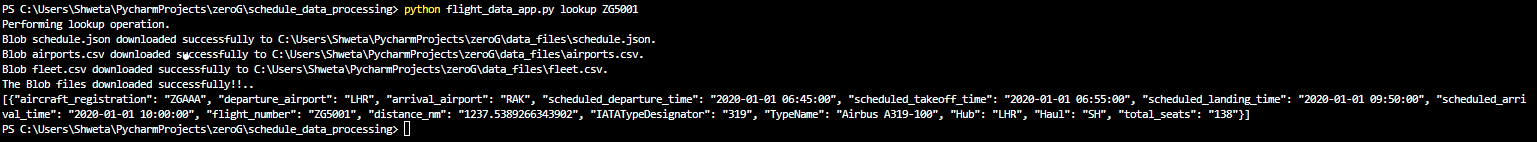
Description: Test the basic functionality of looking up flight information.

Steps:

Execute the script with the command: python flight\_data\_app.py lookup ZG5001.

Expected Outcome: JSON representation of flight details for ZG5001.

Screenshot:



**Test Case 2**: Basic Merge Operation

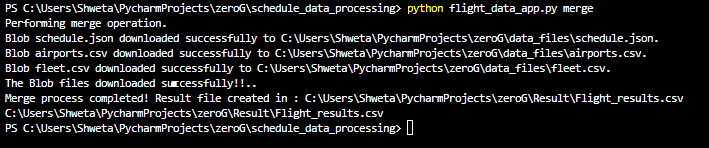
Description: Test the basic functionality of merging data and generating reports.

Steps:

Execute the script with the command: python flight\_data\_app.py merge.

Expected Outcome: Result file generated in the specified directory.

Screenshot:



**Test Case 3:** Advanced Lookup Operation with Multiple Flight Numbers

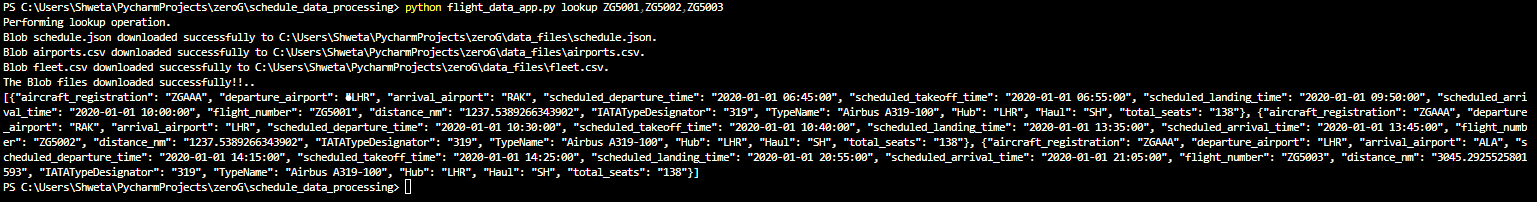
Description: Test the ability to look up information for multiple flight numbers.

Steps:

Execute the script with the command: python flight\_data\_app.py lookup ZG5001, ZG5002, ZG5003.

Expected Outcome: JSON representation of flight details for ZG5001, ZG5002, and ZG5003.

Screenshot:



**Test Case 4:** Advanced Merge Operation with Specific Result Directory

Description: Test the ability to specify a custom result directory for the merged file.

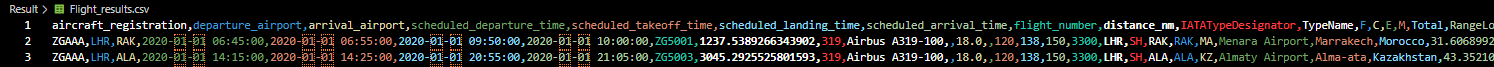
Steps:

Execute the script with the command: python flight\_data\_app.py merge.

Expected Outcome: Result file generated in the specified custom directory.

Screenshot:





**Test Case 5:** Error Handling for Invalid Flight Number

Description: Test the error handling mechanism for an invalid flight number.

Steps:

Execute the script with the command: python flight\_data\_app.py lookup INVALID\_FLIGHT\_NUMBER.

Expected Outcome: Error message indicating that the flight number is not found.

Screenshot:

