Airborne sw test assignment

**Table of Contents**

[**I.** **Initial Findings** 5](#_Toc139027131)

[**II.** **Application Flow** 6](#_Toc139027132)

[1. Input file is .json file 7](#_Toc139027133)

[2. Main program allows the entry to process and formulate the reports. 7](#_Toc139027134)

[3. Base program builds structure of the json file with the classes and Constructs the structure at ground level with default values. 7](#_Toc139027135)

[4. Loader program reads input file and processes the values from the lists, dictionaries. And then it dumps related values to respective fields of respective reports. 7](#_Toc139027136)

[5. Csv\_reports program generates 6 csv validated different reports given below. 7](#_Toc139027137)

[**III.** **White box Test Plan** 9](#_Toc139027138)

[1. Test\_main\_final.py 9](#_Toc139027139)

[2. Test\_loader\_final.py 9](#_Toc139027140)

[3. Test\_base\_final.py 9](#_Toc139027141)

[4. Test\_csv\_reports\_final.py 9](#_Toc139027142)

[**IV.** **Software to Install** 9](#_Toc139027143)

[**V.** **Pytest Framework Installation and Configuration** 9](#_Toc139027144)

[**VI.** **Output Results.** 13](#_Toc139027145)

[**VII.** **Improvements and Thoughts** 17](#_Toc139027146)

[**VIII.** **Time Spent** 17](#_Toc139027147)

[Analysis & Initial thoughts 17](#_Toc139027148)

Project Scope

Airborne SW project on Reports Generation

This project takes .JSON file as an input. It processes and generates different .csv reports.

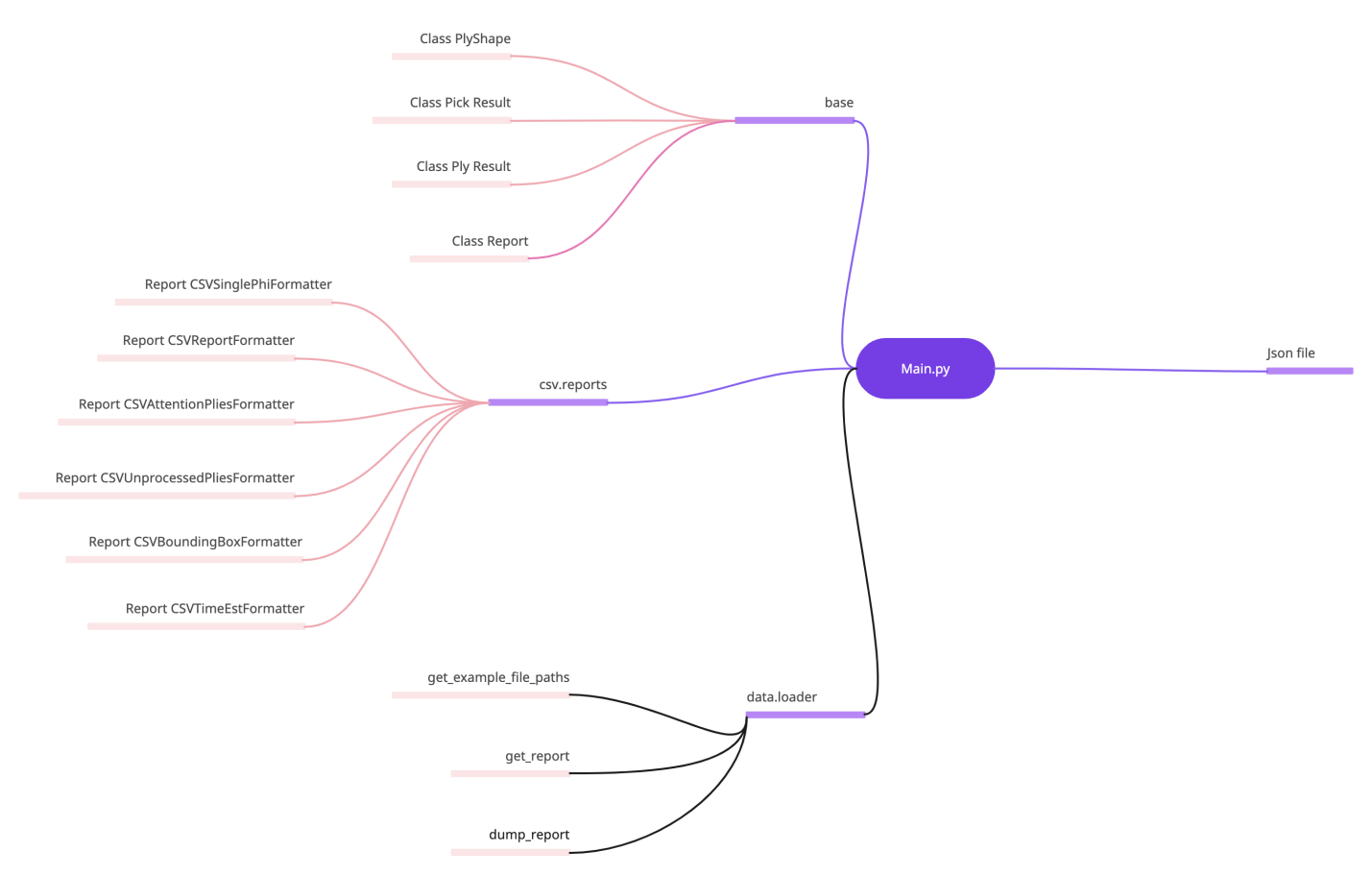
**Phases of the Project**

1. Initial findings
2. Application Flow
3. Whitebox test plan
4. Software to Install
5. Pytest framework installation with configuration
6. Results
7. Future Improvements

# **Initial Findings**

* Input file is JSON file with the different key values of reports, polygon, ply shape, ply result, pick result and ply shape.
* Different CSV Reports are generated through main program based on success and failure, ply shape, ply result, pick result and ply shape inputs from the json file.
* Five different csv reports are generated using loader & csv\_reports programs.

Below picture depicts the input json file with different functions used by programs to generate csv reports.



# **Application Flow**

The received .json input file is processed, formulated and generated with different .csv reports.

### Input file is .json file

### Main program allows the entry to process and formulate the reports.

### Base program builds structure of the json file with the classes and constructs the structure at ground level with default values.

### Loader program reads input file and processes the values from the lists, dictionaries. And then it dumps related values to respective fields of respective reports.

### Csv\_reports program generates 6 csv validated different reports as given below.

CSVReportFormatter (Detailed Data Output (CSV))

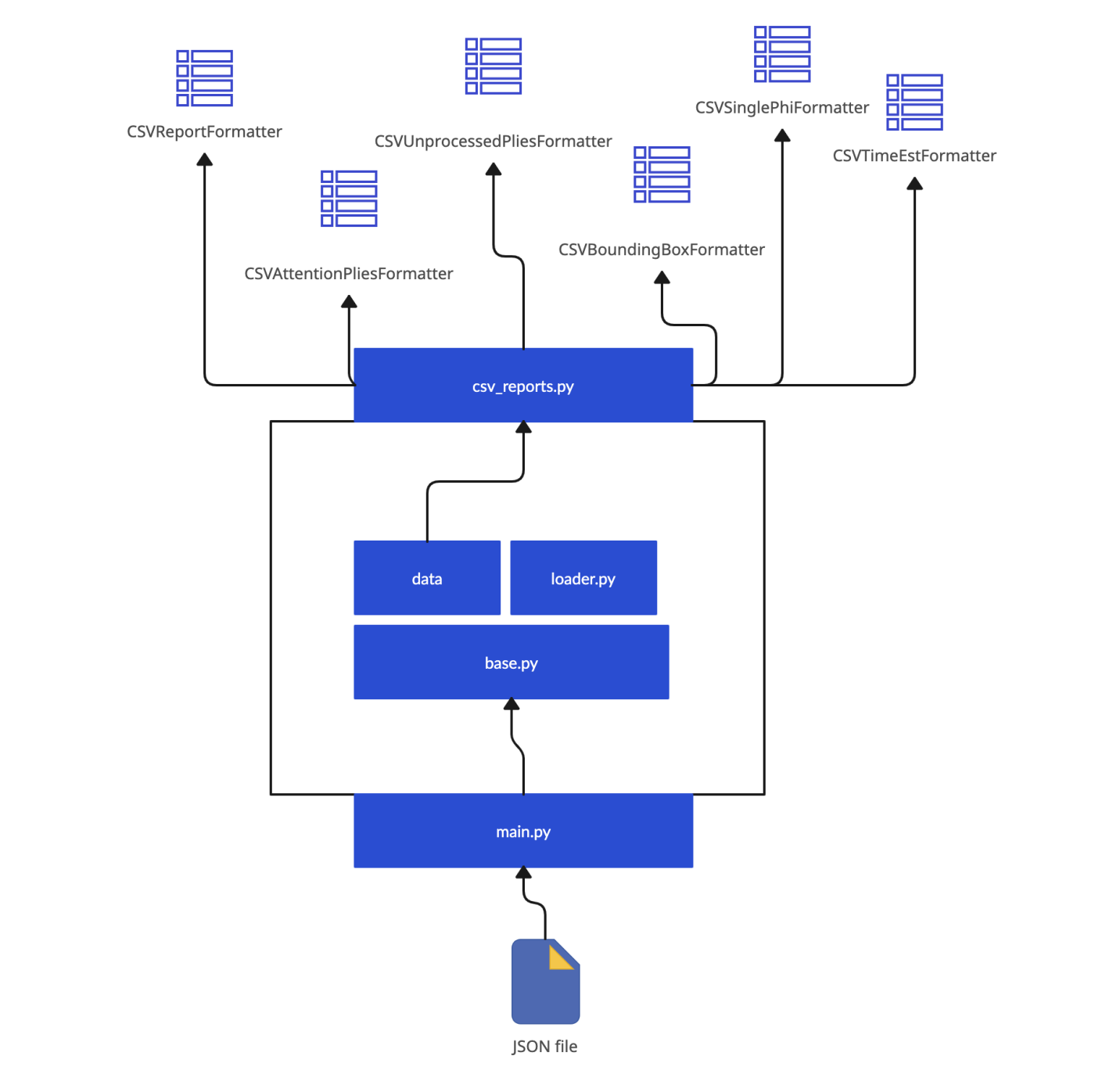
CSVAttentionPliesFormatter (Plies Requiring Attention Overview (CSV))

CSVUnprocessedPliesFormatter (Overview of DXF files with warnings (CSV))

CSVBoundingBoxFormatter (Detailed Data Output w Bounding Box (CSV))

CSVSinglePhiFormatter (Simple Data Output w Bounding Box (CSV))

CSVTimeEstFormatter (Time Estimate (CSV))



# **White box Test Plan**

Test cases are designed on the application flow with the internal structure of the programs. Considered on the logic, functions and conditions used in the programs. There are other approached can be implemented in future.

4 test programs are designed and tested based on POM model approach.

### Test\_main\_final.py

### Test\_loader\_final.py

### Test\_base\_final.py

### Test\_csv\_reports\_final.py

### 

# **Software to Install**

* 1. Python 3.9 version used.
  2. PyCharm Editor or VS Code Editor IDE
  3. Pytest Framework tool.
  4. Windows 10 Home Edition.

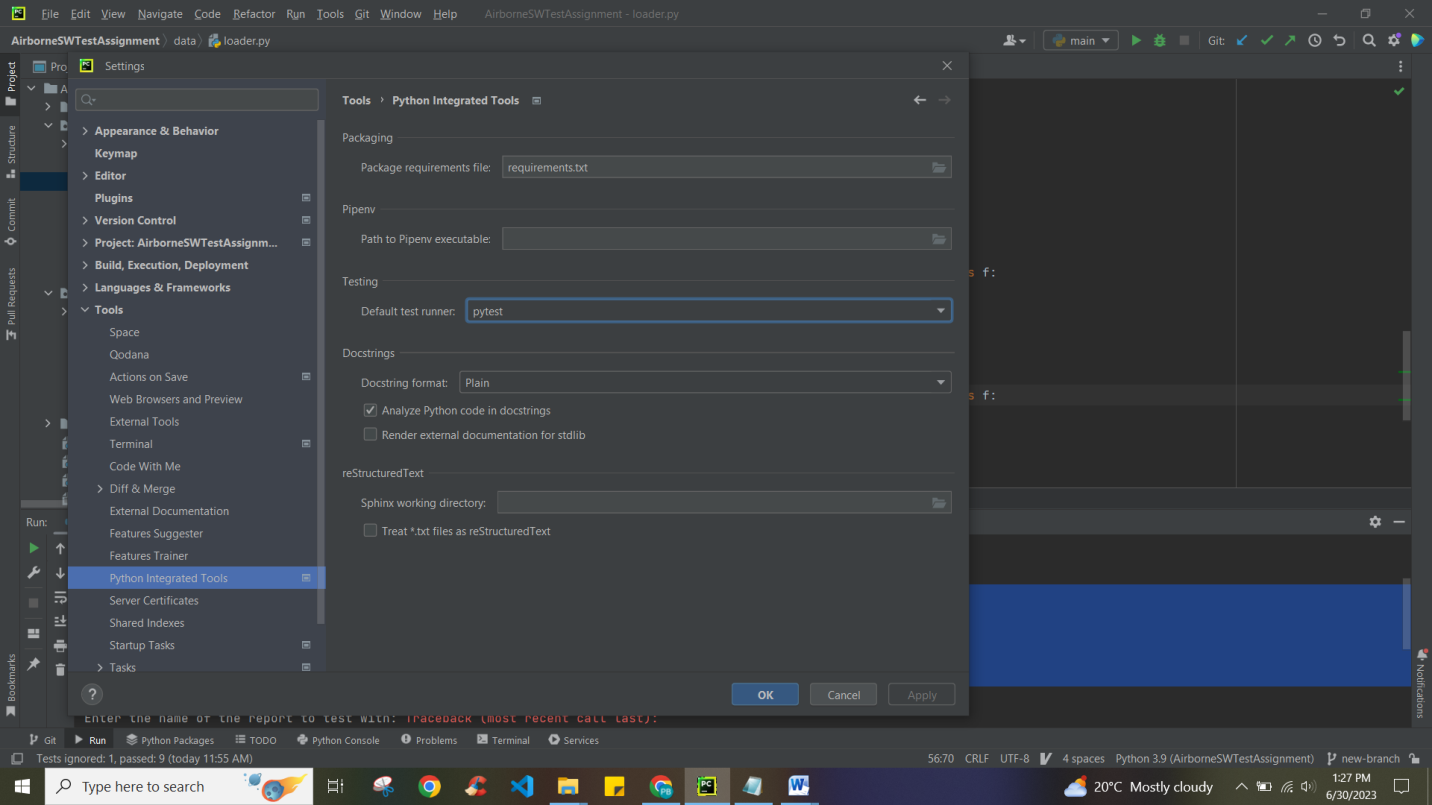
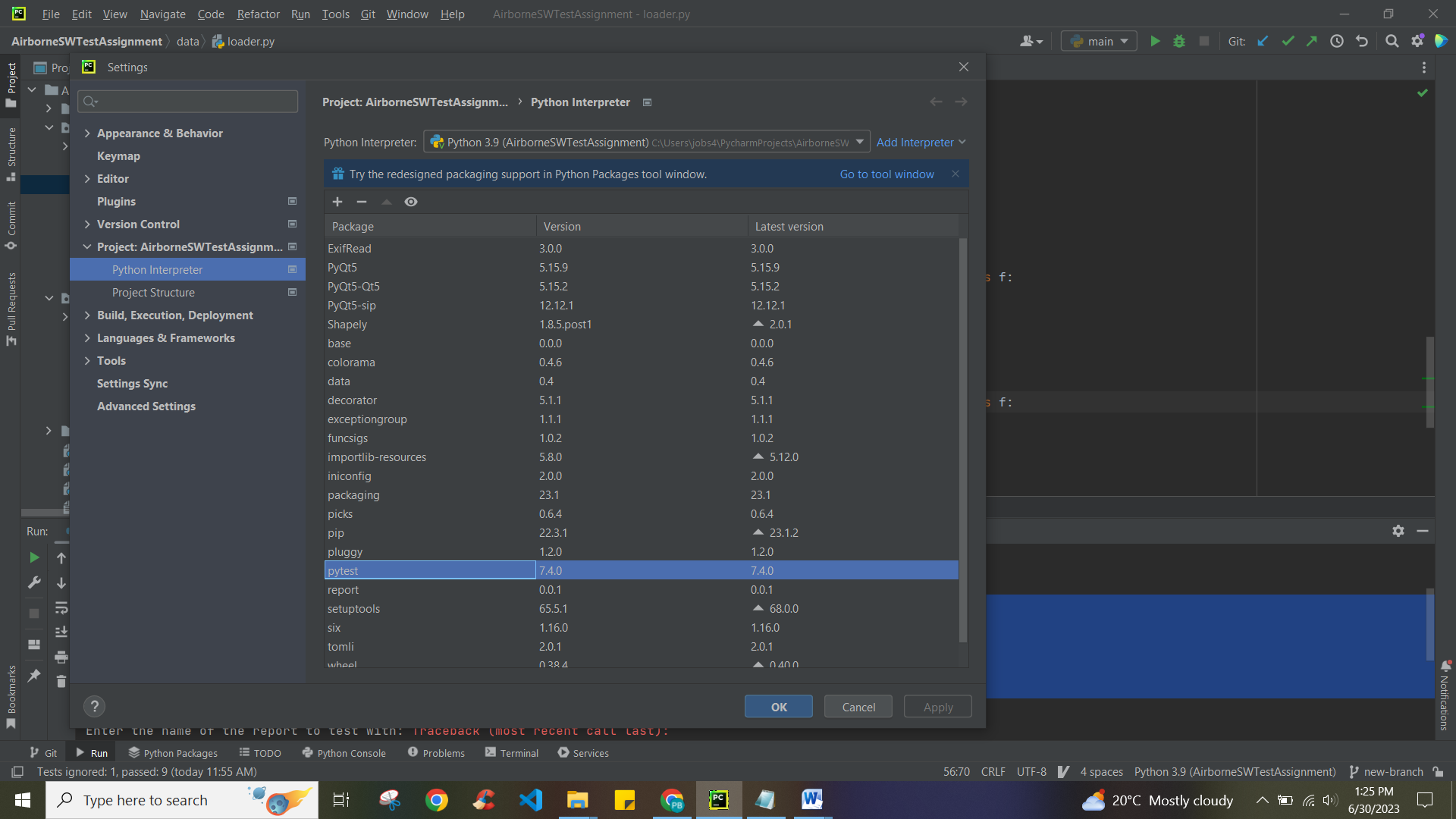
# **Pytest Framework Installation and Configuration**

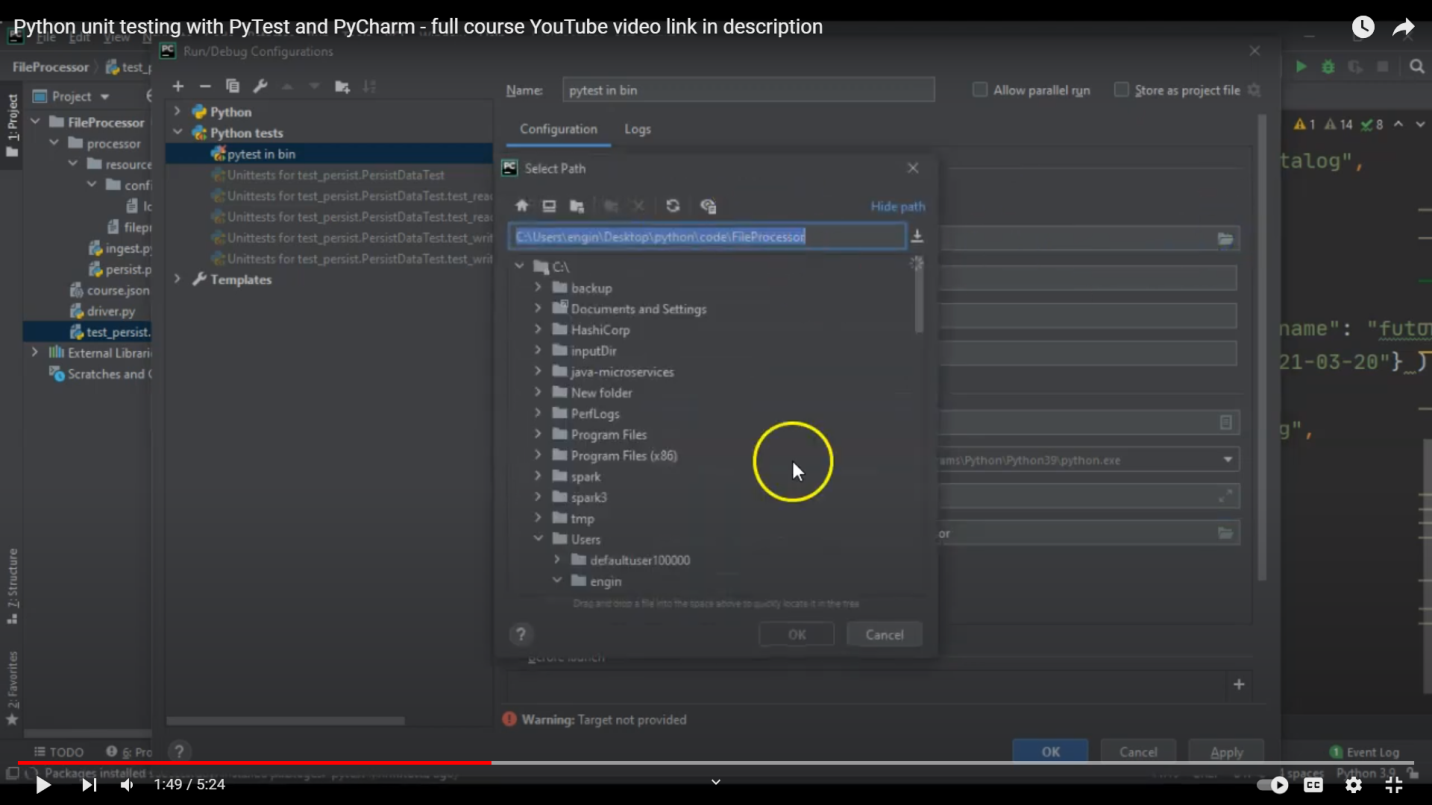
1. “Pip install pytest” Or install pytest package from python packages as shown below screenshots.

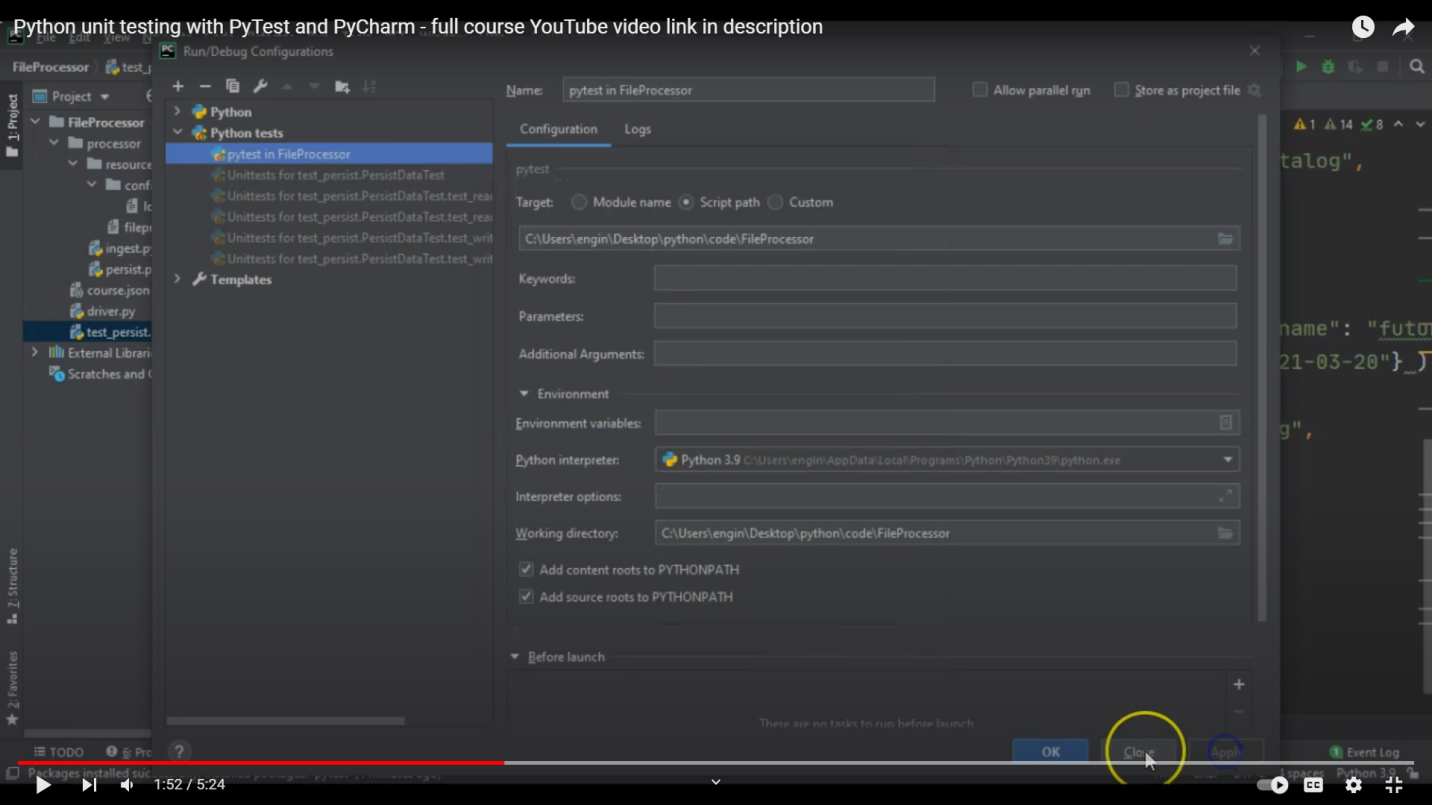
1. From python console after setting the current path we can run command

“pytest test\_loader\_final.py”

1. Or do the configuration settings on pycharm to select the runner as pytest and choose the working directory as shown below screenshots.



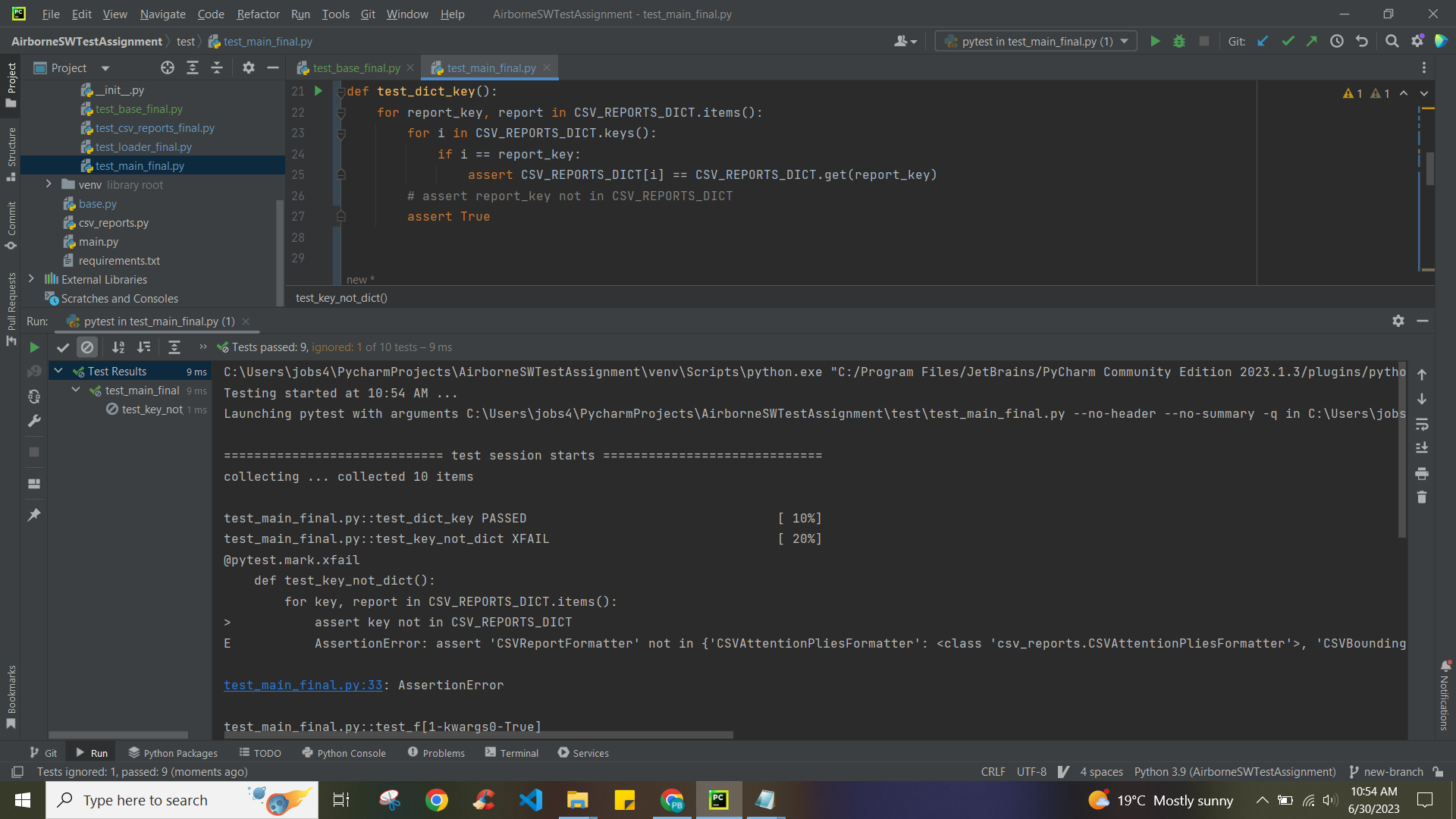




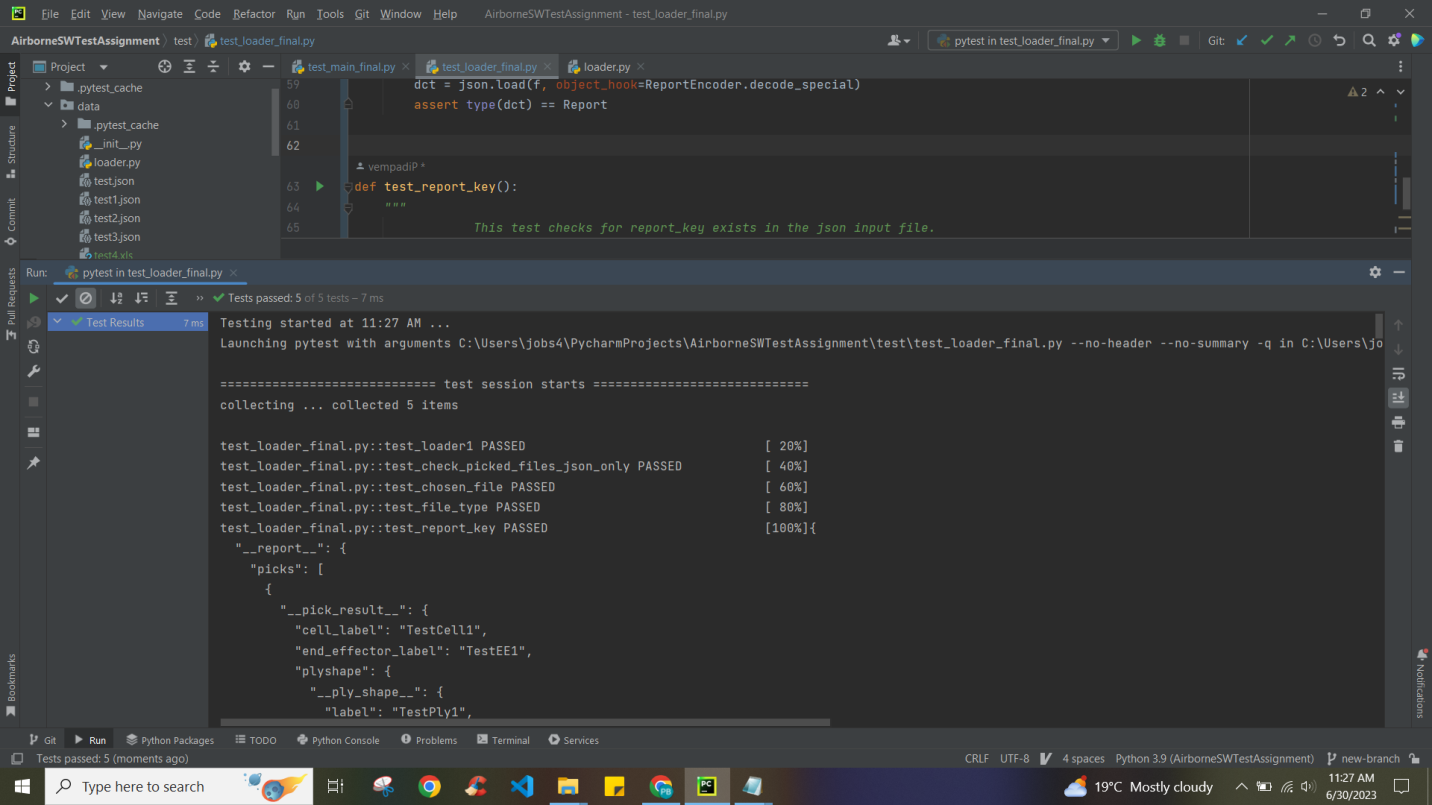
# **Output Results.**

* 1. Test\_main\_final.py : output results
  2. Test\_loader\_final.py : output results
  3. Test\_base\_final.py : output results
  4. Test\_csv\_reports.py : output results.

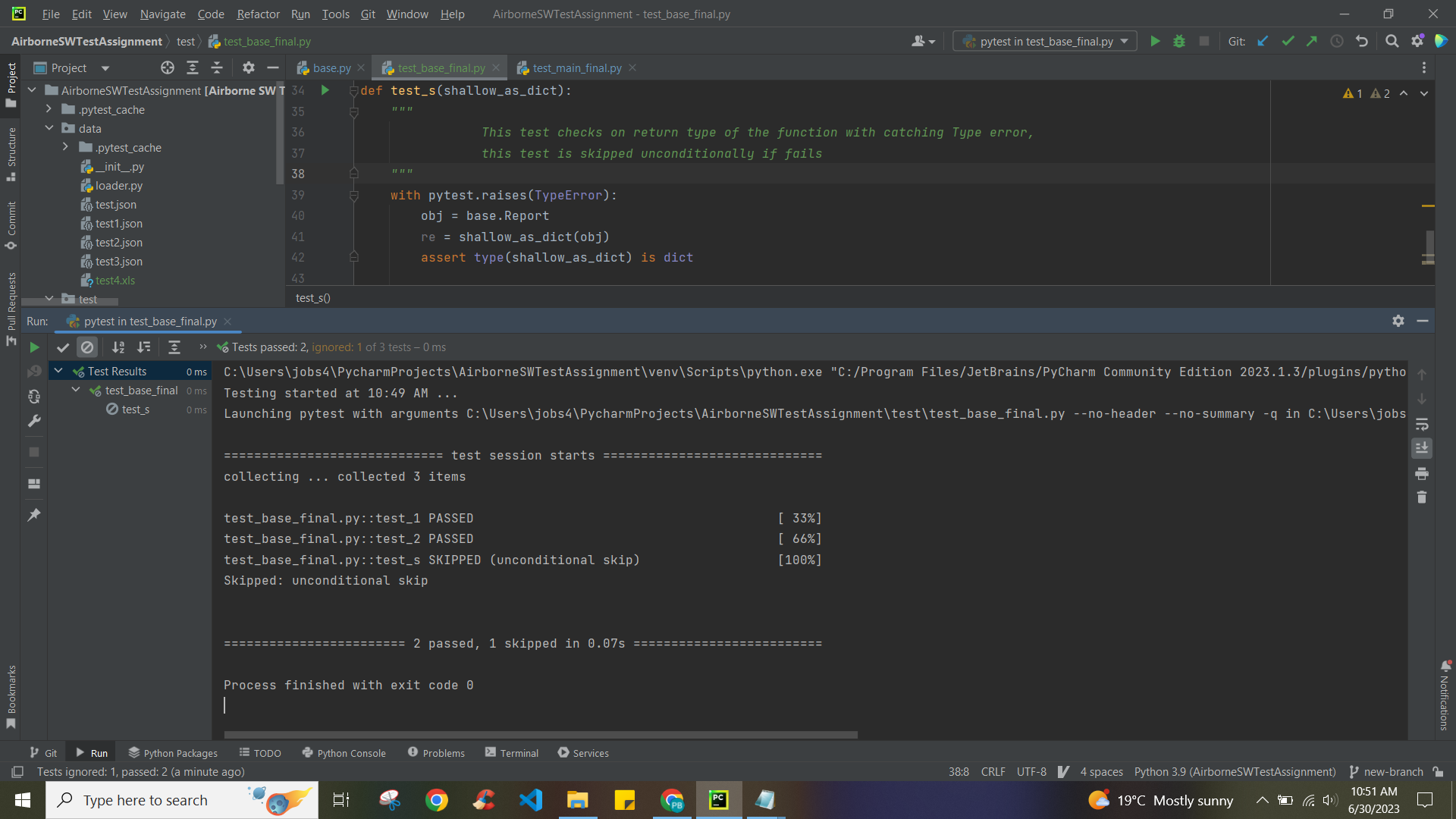
Test\_main\_final.py : output results



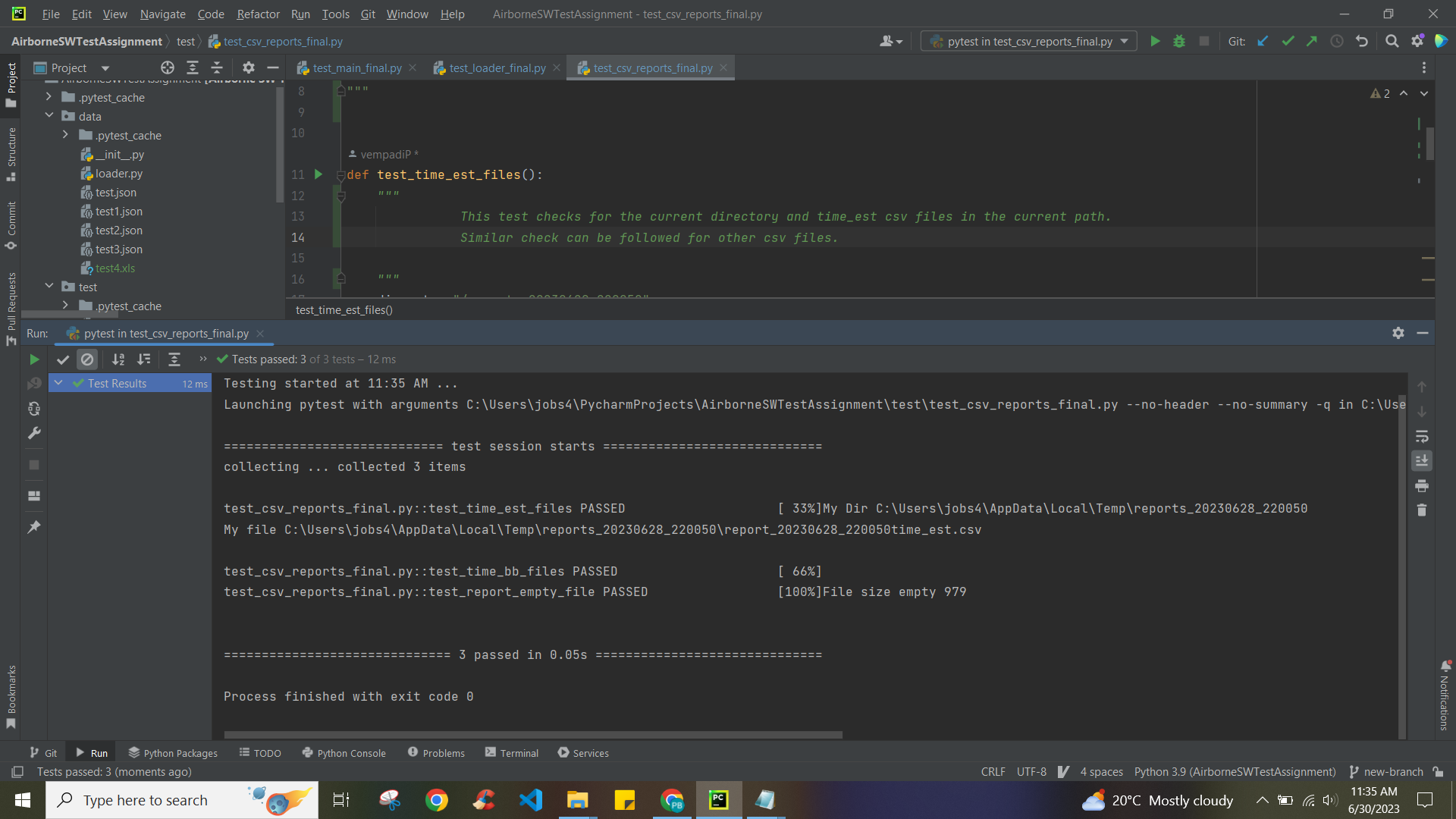
Test\_loader\_final.py : output results



Test\_base\_final.py : output results



Test\_csv\_reports\_final.py : output results



# **Improvements and Thoughts**

1. Output folder structure can be improved in the source code by having dedicated folder for Year. Eg: Year-> yymmdd folder -> everyday current timestamp files. This will enable us to introduce another check at day level.

2. Comments to be added in the source programs.

3. Exception Handling can be implemented where needed.

Eg: get\_example\_file\_paths() in this function for file fetch failures.

4. From testing perspective the common code used for fixtures can be separately done in conftest.py file and reuse the same code.

5. From testing perspective the functions with object arguments passed can be improved more using fixtures in the Improvements.

Eg: object arguments fixtures on these functions - compactness , decode\_special, shallow\_as\_dict can be implemented in future to enable testing.

# **Time Spent**

|  |  |
| --- | --- |
| **Name** | **Total Hours** |
| Analysis & Initial thoughts | 2 – 4 |
| Test, Code review and execute few test cases to build the Test plan and approach | 20 -24 |
| Pytest white box testing implementation | 16 - 20 |
| GitHub CI/CD deployment  Note - due to ssh key expired and new policy rules, my permissions denied. Haven’t used from 2.5years. | 12 |
| Documentation + Improvements | 6-8 hrs |