Imaduddin Sheikh
9/9/2023

G-Research Take-Home Assessment Report

Project Overview

Task Description

A car rental application logs a start event when a rental car is collected by the customer, and a corresponding end event when the car is returned. Each rental session is limited to 24 hours. The application writes the events into an output file as JSON records, where each record represents either the start or end of a rental session. The file contains start and end records for multiple rental sessions.

Objectives

Using the logged information in an output file, the objective of this project is to develop an application that parses the output file and generates a single summary record for each session in the format prescribed in the project documentation.

Approach

Technical Choices

The project will use Python 3.11 as the programming language. For this project, PyCharm will be used as an Integrated Development Environment (IDE) due to its seamless integration with GitHub through its Git functionality.

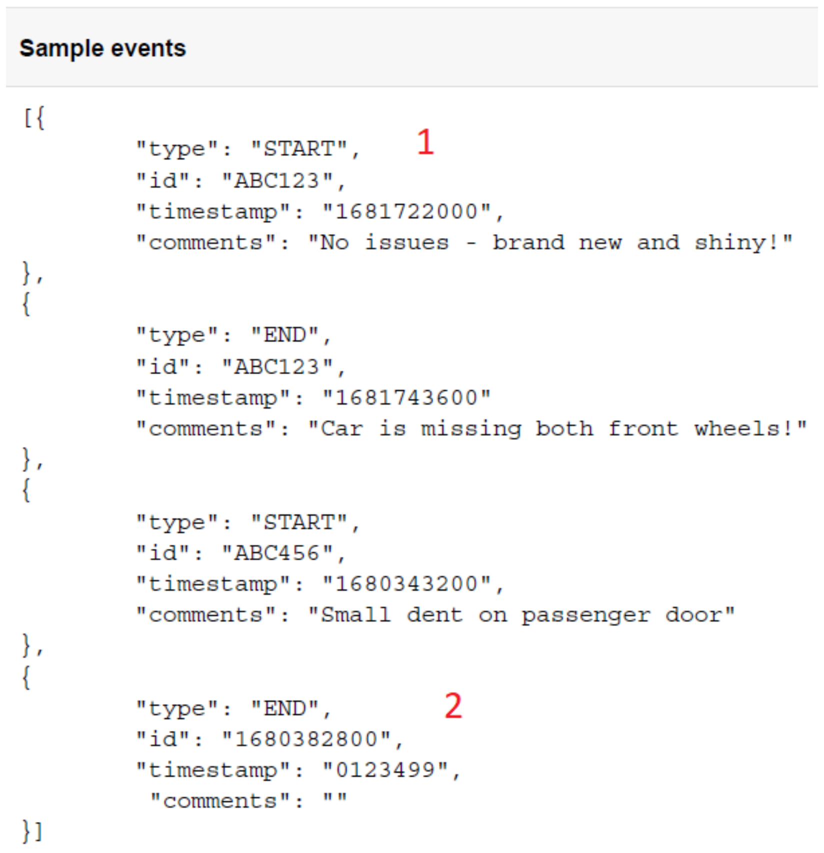
Algorithm and Data Structures

The code reads JSON data from a text file to populate a dictionary called ‘db’, keyed by an ‘id’ and containing ‘Summary’ objects as values. The ‘Summary’ objects hold information like start and end timestamps, duration, and flags for late receipt and damaged return'. The main logic uses a straightforward iteration over items in the dictionary to write them into a tab-separated text file, using and evaluating Unix timestamps in the process. Local timezone-formatted dates could also be used in place of timestamps. The use of a dictionary for quick look-up and the in-place modification of ‘Summary’ objects make the approach efficient.

Issues and Challenges

Problems Encountered

The primary challenge I encountered was interpreting the project requirements based on the provided test cases. While the documentation effectively laid out the context of the task, inconsistencies in the example output files made parsing the data confusing.



Sample events

```
[{
        "type": "START",
        "id": "ABC123",
        "timestamp": "1681722000",
        "comments": "No issues - brand new and shiny!"
},
{
        "type": "END",
        "id": "ABC123",
        "timestamp": "1681743600"
        "comments": "Car is missing both front wheels!"
},
{
        "type": "START",
        "id": "ABC456",
        "timestamp": "1680343200",
        "comments": "Small dent on passenger door"
},
{
        "type": "END",
        "id": "1680382800",
        "timestamp": "0123499",
         "comments": ""
}]
```

For instance, the first record, indicated by a red number 1, adheres to the expected data format for its corresponding 'id'.

However, the record near the red number 2 displays an 'id' that appears to be in a timestamp format rather than a standard 'id'. Moreover, the timestamp for that particular record is not coherent or easily understandable.

Solutions and Workarounds

I operated under the assumption that the timestamp in the final entry was erroneously placed. Consequently, I switched it with the 'id' and assigned the 'id' of the third record as the rental session id. I also generated new test data files to account for consistency and various edge cases. I am also under the impression that as the logs come in regardless of their ‘START’ and ‘END’ type, their timestamp is at an increasing order(can be equal in rare cases).

Testing Strategy

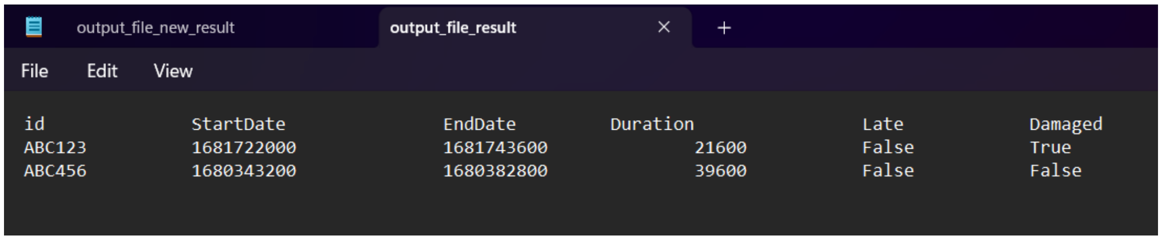
Test Cases

I generated a fresh test file named **‘output_file_new.txt’** manually and a synthetically generated **‘synthetic_output.txt’** test file to accommodate various edge cases. The **‘synthetic_output.txt’** generated from the **‘synthetic_data_generation.py’** file contains test data of 100,000 entries or rental sessions which tests the code’s performance of large input data. Other tests include scenarios such as multiple rental sessions initiating simultaneously, overlapping rental sessions, and sessions that exceed the 24-hour limit. I’ll be designing test cases against the car rental application logging and making sure data arriving is logically consistent—meaning rental session IDs will be unique, records will arrive sorted by timestamp, and no session will lack an ending record, among other things. Test files can be found in the tests folder of the root directory. The three test files are renamed from the test data files in the root directory.

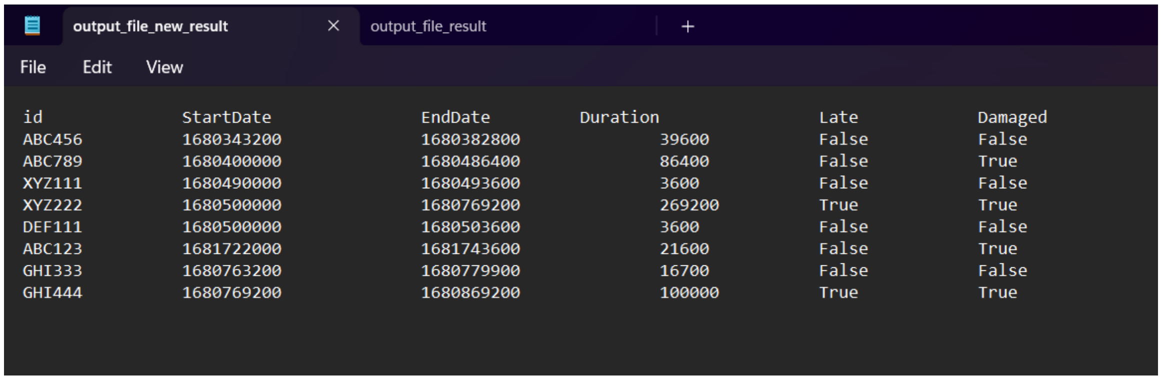
<u>Test Results</u>

There are three test data files in the project folder. One file was provided in the documentation. The other files were generated. These are the results my code gives upon processing the data in the three files. The output generated files can be identified by their corresponding test files with '_result' added at the end in the tests folder but you can also run the 'main.py' file yourself and the files will be generated and named with the same prefix added after their names.

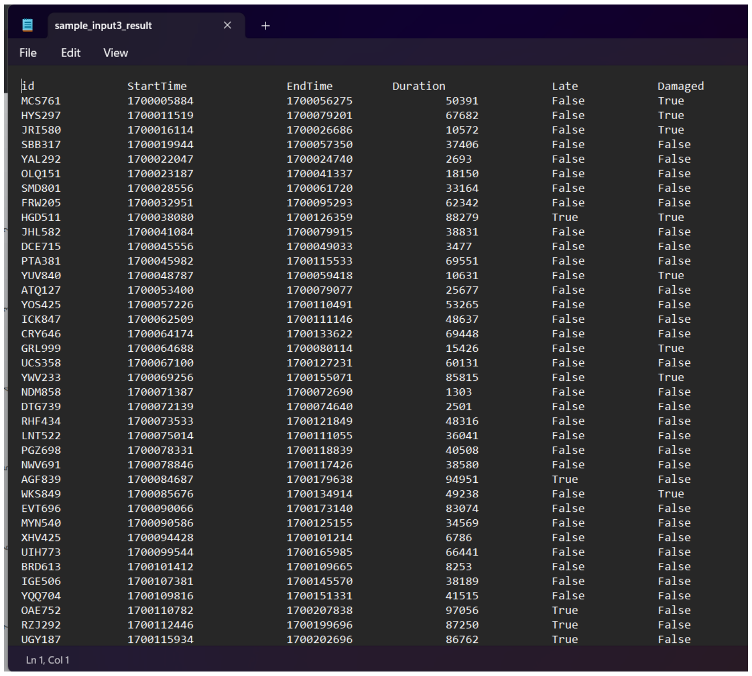
You can also test the files in the test folder by typing on the terminal: **python -m pytest**



```
id          StartDate       EndDate     Duration        Late        Damaged
ABC123      1681722000      1681743600      21600       False       True
ABC456      1680343200      1680382800      39600       False       False
```



```
id          StartDate       EndDate     Duration        Late        Damaged
ABC456      1680343200      1680382800      39600       False       False
ABC789      1680400000      1680486400      86400       False       True
XYZ111      1680490000      1680493600      3600        False       False
XYZ222      1680500000      1680769200      269200      True        True
DEF111      1680500000      1680503600      3600        False       False
ABC123      1681722000      1681743600      21600       False       True
GHI333      1680763200      1680779900      16700       False       False
GHI444      1680769200      1680869200      100000      True        True
```



```
id          StartTime       EndTime     Duration        Late        Damaged
MCS761      1700005884      1700056275      50391       False       True
HYS297      1700011519      1700079201      67682       False       True
JRI580      1700016114      1700026686      10572       False       True
SBB317      1700019944      1700057350      37406       False       False
YAL292      1700022047      1700024740      2693        False       False
OLQ151      1700023187      1700041337      18150       False       False
SMD801      1700028556      1700061720      33164       False       False
FRW205      1700032951      1700095293      62342       False       False
HGD511      1700038080      1700126359      88279       True        True
JHL582      1700041084      1700079915      38831       False       False
DCE715      1700045556      1700049033      3477        False       False
PTA381      1700045982      1700115533      69551       False       False
YUV840      1700048787      1700059418      10631       False       True
ATQ127      1700053400      1700079077      25677       False       False
YOS425      1700057226      1700110491      53265       False       False
ICK847      1700062509      1700111146      48637       False       False
CRY646      1700064174      1700133622      69448       False       False
GRL999      1700064688      1700080114      15426       False       True
UCS358      1700067100      1700127231      60131       False       False
YWV233      1700069256      1700155071      85815       False       True
NDM858      1700071387      1700072690      1303        False       False
DTG739      1700072139      1700074640      2501        False       False
RHF434      1700073533      1700121849      48316       False       False
LNT522      1700075014      1700111055      36041       False       False
PGZ698      1700078331      1700118839      40508       False       False
NWV691      1700078846      1700117426      38580       False       False
AGF839      1700084687      1700179638      94951       True        False
WKS849      1700085676      1700134914      49238       False       True
EVT696      1700090066      1700173140      83074       False       False
MYN540      1700090586      1700125155      34569       False       False
XHV425      1700094428      1700101214      6786        False       False
UIH773      1700099544      1700165985      66441       False       False
BRD613      1700101412      1700109665      8253        False       False
IGE506      1700107381      1700145570      38189       False       False
YQQ704      1700109816      1700151331      41515       False       False
OAE752      1700110782      1700207838      97056       True        False
RZJ292      1700112446      1700199696      87250       True        False
UGY187      1700115934      1700202696      86762       True        False
```

Potential Improvements

Instead of simply generating a text file with the data, it is also easy to set up code to store the output data to a MySQL/SQLite3 database but I didn’t feel the need to do so because of its complexity in testing.

Reflection

This project served as an invaluable refresher for my coding and problem-solving abilities. It provided me with the opportunity to delve back into Python testing frameworks, reinforcing my understanding of test-driven development in Python. Through tackling various challenges in the project, I was able to hone my skills and revisit key programming and debugging techniques, effectively revitalizing my expertise in these areas. Overall, the experience was both educational and rewarding, allowing me to keep my coding skills up-to-date while reinforcing important testing methodologies.