Software Development

Final Project

[Link to GitHub Repository]

Group 21

Aarón Espasandín Geselmann (G89 - 100451339)

Mark Joseph Aala Bernardo (G89 - 100451329)

23 May, 2022

Table of Contents

Abstract

Our goal during this project was to develop two functions get_vaccine_date and cancel_appointment using the class VaccineManager as it was an API and implementing the rest of the code outside of it.

The first function, get_vaccine_date, takes an input JSON file and a date as parameters and returns the date signature that has been generated from those parameters.

The second function, cancel_appointment, takes a JSON file whose format is checked through some tests number as a parameter and cancels the appointment, returning the date signature of the appointment that has been cancelled.

Also, a new JSON Store file has been created called cancellation_json_store.py has been created in order to store the cancellations properly. It follows the singleton pattern.

Equivalence classes and boundary values analysis

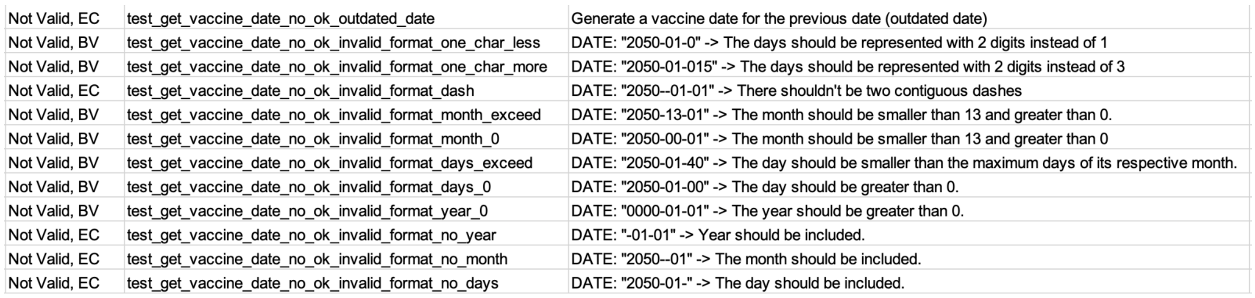
get_vaccine_date

Excel file

Theoretically, the boundary values should have 4 tests each. One that tests a value below the lower bound, one above the lower bound, one below the upper bound and one above the lower bound.

However, we decided to consider only the invalid tests (one test below the lower bound and one above) since the datetime implementation already checks those valid cases where the number of days depends on the corresponding month and year.

If we wanted to implement the valid cases, we should test with month 1, with month 12, with day 1 and with the maximum day of a month according to its month and year (leap or not).



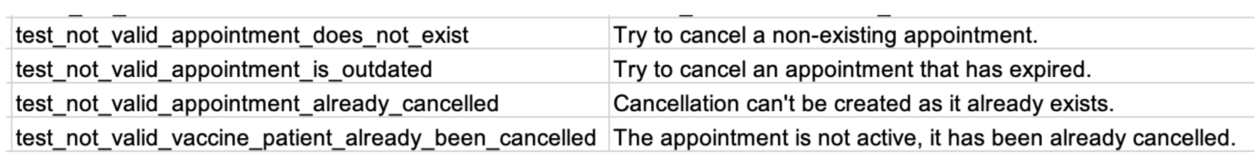
Not Valid, EC	test_get_vaccine_date_no_ok_outdated_date	Generate a vaccine date for the previous date (outdated date)
Not Valid, BV	test_get_vaccine_date_no_ok_invalid_format_one_char_less	DATE: "2050-01-0" -> The days should be represented with 2 digits instead of 1
Not Valid, BV	test_get_vaccine_date_no_ok_invalid_format_one_char_more	DATE: "2050-01-015" -> The days should be represented with 2 digits instead of 3
Not Valid, EC	test_get_vaccine_date_no_ok_invalid_format_dash	DATE: "2050--01-01" -> There shouldn't be two contiguous dashes
Not Valid, BV	test_get_vaccine_date_no_ok_invalid_format_month_exceed	DATE: "2050-13-01" -> The month should be smaller than 13 and greater than 0.
Not Valid, BV	test_get_vaccine_date_no_ok_invalid_format_month_0	DATE: "2050-00-01" -> The month should be smaller than 13 and greater than 0
Not Valid, BV	test_get_vaccine_date_no_ok_invalid_format_days_exceed	DATE: "2050-01-40" -> The day should be smaller than the maximum days of its respective month.
Not Valid, BV	test_get_vaccine_date_no_ok_invalid_format_days_0	DATE: "2050-01-00" -> The day should be greater than 0.
Not Valid, BV	test_get_vaccine_date_no_ok_invalid_format_year_0	DATE: "0000-01-01" -> The year should be greater than 0.
Not Valid, EC	test_get_vaccine_date_no_ok_invalid_format_no_year	DATE: "-01-01" -> Year should be included.
Not Valid, EC	test_get_vaccine_date_no_ok_invalid_format_no_month	DATE: "2050--01" -> The month should be included.
Not Valid, EC	test_get_vaccine_date_no_ok_invalid_format_no_days	DATE: "2050-01-" -> The day should be included.

All the descriptions of the boundary values (BV) and equivalence classes (EC) are included in the third column of the image. The type is indicated in the first column and the name of the test in the second column.

cancel_appointment

Excel file

Although all the tests for appointment cancellations are found in the python script test_cancellation_tests.py , the last test was done for Vaccine Patient to avoid a patient from cancelling an appointment that has been cancelled previously (link to that last test).



test_not_valid_appointment_does_not_exist	Try to cancel a non-existing appointment.
test_not_valid_appointment_is_outdated	Try to cancel an appointment that has expired.
test_not_valid_appointment_already_cancelled	Cancellation can't be created as it already exists.
test_not_valid_vaccine_patient_already_been_cancelled	The appointment is not active, it has been already cancelled.

These four last tests are equivalence classes. Their name is included in the first column and its description in the second column.

Grammar and Derivation Tree of the JSON

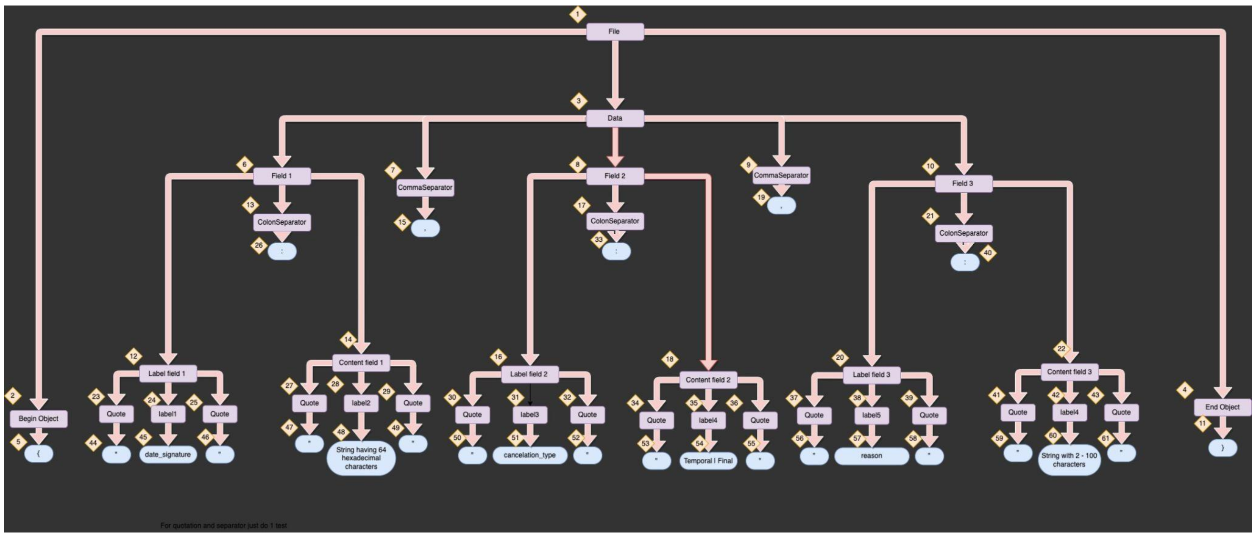
The function cancel_appointment makes use of a JSON with a particular form. In order to make sure that the JSON has the expected format, a grammar and a derivation tree have been created to check if it has the correct form.

All these tests have been implemented in Python in the test_cancellation_tests.py file and they also can be found in the PF_TestCases_Cancellation.xlsx file.

Grammar

File := Begin_object Data End_object
Begin_object ::= {
End_object ::= }
Data ::= Field1 CommaSeparator Field2 CommaSeparator Field3
Field1 ::= Label_Field1 ColonSeparator ContentField1
Field2 ::= Label_Field2 ColonSeparator ContentField2
Field3 ::= Label_Field3 ColonSeparator ContentField3
CommaSeparator ::= ,
ColonSeparator ::= :
Quote ::= "
Label_Field1 ::= Quote label1 Quote
label1 ::= date_signature
ContentField1 ::= Quote label2 Quote
label2 ::= a|b|c|d|e|f|0|1|2...|9 (64)
Label_Field2 ::= Quote label3 Quote
label3 ::= cancellation_type
ContentField2 ::= Quote label4 Quote
label4 ::= Temporal | Final
Label_Field3 ::= Quote label5 Quote
label5 ::= reason
ContentField3 ::= Quote label6 Quote
label6 ::= a|b|c|d|e|f|0|1|2...|9|+|-|...|$ (2-100)

Derivation Tree



GitHub Link to visualize it properly

Following the rules of grammar modeling, functional testing and syntax analysis; we numbered all terminal and non-terminal nodes from 1 to 61. Later on, we started to generate the JSON files relating to the deletion, duplication and modification of the nodes.

There are a total of 3 tests, covering each of the JSON file input. Two of those tests cover the valid cases where the JSON has the correct form and where the Reason value has been duplicated (in this case, the duplicated reason string has a length of less than 100, that's why it's valid).

The third test is a parameterized test that covers all the non-valid cases. It is composed of a total of 54 subtests. There are no JSONs including the terminal nodes as we decided to only use the name of the non-terminal node that leads to its respective terminal node. In this way, we say that the nodes that are modified are the non-terminal nodes although, in reality, the modified ones are the terminal nodes.

We have analyzed 4 different types of VaccineManagementExceptions:

- "JSON Decode Error - Wrong JSON Format": This exception is raised when the JSON doesn't have a valid format. For example, when a comma is missing.

- "The JSON file is empty": This exception is raised if the JSON file has no content.
- "The key of the JSON file is wrong.": This exception is raised when one or more of the three keys is not the one that we expected. This can happen if the key is date_no_sig instead of date_signature for example.
- "The value of the JSON file is wrong.": This happens when the value is not the one that we expected. For example, the string representing the reason has a length of only 1 character.

Conclusion

We have enjoyed being able to apply everything we have seen during the course in this project. While it is true that developing the code has slowed down the speed at which we were developing the code a little bit, it is now easier to understand.

Also, by using PyLint we maintained a standard code style during the project and avoided discrepancies.

In addition, through the Test Driven Development, we first developed the tests and these clarified the functionalities that we had to develop in our company.