API Design A1.2

1)What were some of the alternative design options considered? Why did you choose the selected option?
Answer :

The design option I considered was to consider given capabilities as steps in bigger activities. The other option is to consider the capability itself as activity. By considering the capabilities as activities, we can't implement the necessary activities like create an event on particular date, delete the event on particular date etc.. To use capabilities as steps in bigger activities make our code more productive as with these steps we can build many activities. Hence the selected design option is productive, usable and more accessible in case of designing bigger and multiple activities.

2)What changes did you need to make to your tests (if any) to get them to pass. Why were those changes needed, and do they shed any light on your design?
Answer :

I used JSONObject parameters to create a request body while testing, then I had to convert it in to string, to get the test cases pass. I missed this point initially. I need to send my input as the request body, hence I used this way. I created JSONObject named parameters and added all the fields that are required to test my case. I used spark framework as it was well designed, supported and recent.

```
@Test
void testPutEventPass() {
    JSONObject parameters = new JSONObject();
    parameters.put("Day", "9");
    parameters.put("Month", "November");
    parameters.put("year", "2022");
    parameters.put("PatientId", "1");
    parameters.put("DOctorId", "3");
    parameters.put("EventID", "1");
    given().
            body(parameters.toString());
    when().
            get( path: "/event/eventID");
    then().
            statusCode( expectedStatusCode: 200).body( path: "message",equalTo( operand: "Success"));

}
```

I used Postman to test my put and post events, and using Postman has helped me lot to debug and fix my test cases.

3)Pick one design principle discussed in class and describe how your design adheres to this principle.

Answer :

The design principle that best adheres to my design is the Single Responsibility Principle.

Single Responsibility Principle:

A class or module should have one, and only one, reason to be changed.

I have designed my code in a way that all my methods are in one class. This class is focussed on only to handle events and this makes my class more robust. My code is easier to explain, understand and implement than the ones that provide a solution for everything because my class have only one responsibility to manage hospital API. Also, all my functions have only one individual responsibility. My code design perfectly adheres to Single Responsibility principle and well documented and implemented.