Assignment 01

Course Name: Software Testing and Quality Assurance

Course ID: CSE 4495

Section: A

Group Name: ARKv2.0

Group Members:

Md. Abdur Rahman (011202260)

Md. Kabirul Hossain (011202026)

Submitted on: 17th Sept, 2024

Test cases, generated and experimented by Abdur Rahman, and Kabirul Hossain, are denoted as **AR**, and **KH**, respectively.

**1. API 1: (/student):**

Our goal was for this API test was to check the information from *valid IDs.*The information included students’ IDs, list of course passed by the students, and their respective names.

|  |  |  |
| --- | --- | --- |
| **#Choice** | **Rationale** | **#Specification** |
| 1 | For this API, the number of testable choice is 1 since it does not any more testable choice. | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request Type** | **Body** | **URL** | **Assertions** | **Description** |
| GET | This request does not have a body | http://127.0.0.1:5000/student | pm.**test**("Response time is less than 100ms", **function** () {  pm.expect(pm.response.responseTime).to.be.below(100);  });  pm.**test**("Status code is 200", **function** () {  pm.response.to.have.status(200);  }); | Our goal for this test was to check whether all student info is available, and to check if the API response time is below 100ms with a status code of 200;  For this test, the test result was successfully executed |

Test case check for API 1 (done by **AR** & **KH**):

**2. API 2: (/student/<id>):**

The goal for this test was to check the information of a student by ID. The this api provided the student’s name, ID, personnummer, and list of courses passed by the student. For this test, our choice was 1: ID. List of specification and respective test scenarios are added to the table below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Done By** | **Choice 1: “**ID” | **http requests** | **Output** | **Decision** |
| AR | 0 <ID<9 | http://127.0.0.1:5000/student/3 | {"courses\_passed": [], "id": 3, "name": "Joel Stevensson ", "personnummer": "011030-9999"} | valid ID |
| AR | 0 > | http://127.0.0.1:5000/student/-1 | "error": "Student ID -1 does not exist" | negative ID doesn’t exist |
| KH | 0 | http://127.0.0.1:5000/student/0 | "error": "Student ID 0 does not exist" | ID 0 doesn’t exist |
| KH | 9 | http://127.0.0.1:5000/student/9 | {"courses\_passed": ["CSE2213"],"id":9,"name": "Natasha Romanov", "personnummer": "861231-9999"} | max valid ID |
| AR | > 9 | http://127.0.0.1:5000/student/10 | "error": "Student ID 10 does not exist | greater than max ID |
| KH | float | http://127.0.0.1:5000/student/1.1 | "error": "Student ID 1.1 is not formatted correctly." | ID isn’t in float datatype |
| KH | C | http://127.0.0.1:5000/student/C | "error": "Student ID C is not formatted correctly." | character is invalid |
| Total #Specifications: 7 | | | | |

From our findings, we noticed that the API doesn’t hold any ID with value more than 9, and the datatype of the ID is always in integer data type.

A demo for some test cases is provided below:

|  |  |  |
| --- | --- | --- |
| **Valid** | **Doesn’t Exist** | **Formatted Error** |
|  |  |  |

**3. API 3: (/create):**

Task for this test case is to create a valid student. To create a new student, we had to provide 3 information: name, personnummer, and list of courses passed (course\_passed). The choices for this task are choose to be 3. Following are the test cases along with results:

|  |  |  |  |
| --- | --- | --- | --- |
| **Done By** | **Choice 1:** Name | **Choice 2:** Personnummer | **Choice 3:** CoursesPassed |
| KH | Empty | [yy: 00-99][mm: 01-12][dd: 01-28]-[0000-9999] | [A-Z][00000-10000] |
| KH | Special Characters | [yy: 00-99][mm: 00][dd: --]-[0000-9999] | [A-Z][000-999] |
| AR | Numeric Values | [yy: 00-99][mm: >12][dd: --]-[0000-9999] | [A-Z][0000-9999] |
| AR | String [A-Z a-z] | [yy: 00-99][mm: 01-12][dd: 00]-[0000-9999] |  |
| KH |  | [yy: 00-99][mm: 01-12][dd: >31]-[0000-9999] |  |
| AR |  | [yy: 00-99][mm: 01-12][dd: >31]-[0000-9999] |  |
| AR |  | [yy: 00-99][mm: 02][dd: >28]-[0000-9999] |  |
| #Total Specification: 4 \* 7 \* 3 = 84 | | | |

After deducing some specifications:

|  |  |  |
| --- | --- | --- |
| **Choice 1:** Name | **Choice 2:** Personnummer | **Choice 3:** CoursesPassed |
| Empty | dont care (-) | **-** |
| Special Characters | **-** | **-** |
| Numeric Values | **-** | **-** |
| String [A-Z a-z] | [yy: 00-99][mm: 01-12][dd: 01-28]-[0000-9999] | [A-Z][0000-9999] |
| **-** | [yy: 00-99][mm: 00][dd: --]-[0000-9999] | **-** |
| **-** | [yy: 00-99][mm: >12][dd: --]-[0000-9999] | **-** |
| **-** | [yy: 00-99][mm: 01-12][dd: 00]-[0000-9999] | **-** |
| **-** | [yy: 00-99][mm: 01-12][dd: >31]-[0000-9999] | **-** |
| **-** | [yy: 00-99][mm: 01-12][dd: >31]-[0000-9999] | **-** |
| **-** | [yy: 00-99][mm: 02][dd: >28]-[0000-9999] | **-** |
| **-** | - | [A-Z][9999<] |
| **-** | - | [A-Z][999>] |
| **#Specification:** 3 + 2 + 6 + (1 \* 1 \* 1) = 12 | | |

A demo for some test cases is provided below:

|  |  |  |
| --- | --- | --- |
| **Validity Check** | **Screenshots (SS) from Postman** | **Decision** |
| **Valid**  (but should’ve been invalid) |  | Names shouldn’t have contained special chars/numeric values; February month cannot have 30 days |
| **Invalid Course Format** |  | courses have to contain 3 characters followed by 4 digits |
| **Corrected** |  | Passed |

**4. API 4: (/update/{student\_id}):**

Task for this test case is to update information of valid students. A typical student has: name, personnummer, ID, and course\_passed properties. However, personnummer is not changeable. So, the choices for this task are choose to be 3. Following are the test cases along with results:

|  |  |  |  |
| --- | --- | --- | --- |
| **Done By** | **Choice 1:** ID | **Choice 2:** Name | **Choice 3:** CoursesPassed |
| KH | Exists | Empty | [A-Z][9999<] |
| AR | Doesn’t Exist | Special Characters | [A-Z][999>] |
| KH |  | Numeric Values | [A-Z][0000-9999] |
| AR |  | String [A-Z a-z] |  |
| #Total Specification: 2 \* 4 \* 3 = 24 | | | |

After deducing specifications:

|  |  |  |
| --- | --- | --- |
| **Choice 1:** ID | **Choice 2:** Name | **Choice 3:** CoursesPassed |
| Doesn’t Exist | **-** | **-** |
| - | Empty | **-** |
| - | Special Characters | **-** |
| - | Numeric Values |  |
| - | - | [A-Z][9999<] |
| - | - | [A-Z][999>] |
| Exists | String [A-Z a-z] | [A-Z][0000-9999] |
| #Total Specification: 1 + 3 + 2 + (1 \* 1 \* 1) = 7 | | |

A demo for some test cases is provided below:

|  |  |  |
| --- | --- | --- |
| **Validity** | **SS from postman** | **Decisions** |
| **Valid** (Name Change) |  | Name changed from “Romano**va**” to “Romano**ff**”, and added an additional course; |
| **Invalid Course Format** |  | Issue with ID 8:  The info cannot be updated for this one, as the personnummer is already invalid |

It is seen that, regardless of the scenario, info of the student with ID 8 cannot be changed as the api hold misformat for this one (see personnummer format).

**5. API 5: (/delete/{student\_id}):**

The task for this one is to delete students by their respective IDs. For this one, the choice is set to 1 (only the student ID).

|  |  |  |  |
| --- | --- | --- | --- |
| **\*Choice 1:** “student\_id” | **http requests** | **Output** | **Decision** |
| 11 | http://127.0.0.1:5000/student/11 | "deleted": "11" | valid ID |
| 0 > | http://127.0.0.1:5000/student/-1 | "error": "Student ID -1 does not exist" | negative ID doesn’t exist |
| 0 | http://127.0.0.1:5000/student/0 | "error": "Student ID 0 does not exist" | ID 0 doesn’t exist |
| > 14 | http://127.0.0.1:5000/student/15 | "error": "Student ID 15 does not exist | greater than max ID |
| float | http://127.0.0.1:5000/student/1.1 | "error": "Student ID 1.1 is not formatted correctly." | ID isn’t in float datatype |
| Chars | http://127.0.0.1:5000/student/C | "error": "Student ID C is not formatted correctly." | character is invalid |
| Total #Specifications: 6 | | |  |

\*[done by AR & KH together]

1. *B. We have added some more students, as a result our current maximum student ID is 14]*

A demo for some test cases is provided below:

|  |  |
| --- | --- |
| **Before** | **After** |
|  |  |

**6. API 6: (/program):**

This is similar to the API 1. Here, our goal was is to check the information from program. The scenario is given below:

|  |  |  |
| --- | --- | --- |
| **Choice 1:** program | **Rationale** | **#Specification** |
| 1 | For this API, the number of testable choice is 1 since it does not any more testable choice. | 1 |

\*[done by AR & KH]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request Type** | **Body** | **URL** | **Assertions** | **Description** |
| GET | This request does not have a body | http://127.0.0.1:5000/program | pm.**test**("Successful POST request", **function** () {  pm.expect(pm.response.code).to.be.oneOf([201, 202]);});  pm.**test**("Response time is less than 200ms", **function** (){pm.expect(pm.response.responseTime).to.be.below(200);});  pm.**test**("Status code is 200", **function** () {  pm.response.to.have.status(200);}); | Our goal for this test was to get all programs with their respective IDs. The test case was checked to see:   1. if the response time is less than 200 ms 2. if the status code is *200* 3. if the POST request was successful (with either *201* or *202* response code).   from the test, 2/3 were successful. |

Test case check for API 6:

**7. API 7: (/program/{program\_id}):**

This is similar to the API 2. Here, we had to check the program with via their IDs. Thus the choice is set to one. Our approach is show below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Done By** | **Choice 1:** “program\_id” | **http requests** | **Output** | **Decision** |
| AR | 1 | http://127.0.0.1:5000/program/1 | "courses\_required": [  "CSE2213","CSE1110","CSE1111","CSE1114","CSE1115","CSE1116","CSE2117","CSE2118","CSE3119","CSE4010"] | valid ID |
| AR | 0 > | http://127.0.0.1:5000/program/-1 | "error": "Program ID -1 does not exist" | negative ID doesn’t exist |
| KH | 0 | http://127.0.0.1:5000/program/0 | "error": "Program ID 0 does not exist" | ID 0 doesn’t exist |
| AR | > 2 | http://127.0.0.1:5000/program/3 | "error": "Program ID 3 does not exist" | greater than max program ID |
| KH | Float | http://127.0.0.1:5000/program/1.1 | "error": "Program ID 1.1 is not formatted correctly." | program ID isn’t in float datatype |
| KH | Chars | http://127.0.0.1:5000/program/C | "error": "Program ID C is not formatted correctly."  formatted correctly." | character is invalid |
| Total #Specifications: 6 | | | |  |

Demo is shown below:

|  |  |  |
| --- | --- | --- |
| **Validity** | **SS from postman** | **Decisions** |
| **Valid** |  | Program exists for ID 1, and 2. |
| **Invalid** |  | Maximum ID for program is 2 |

**8. API 8: (/finished/{student\_id}/{program\_id}):**

For this API test, task is to check if a student with valid *student\_id* has passed from a program (there are many) with *program\_id.* The choices for this one is set to 2 as there are 2 changeable params. Approach is show below:

|  |  |  |
| --- | --- | --- |
| **Done By** | **Choice 1:** student\_id | **Choice 2:** program\_id |
| AR | 1 | 1 |
| AR | < 1 | < 1 |
| AR | > 14 | > 2 |
| KH | Float | Float |
| KH | Chars | Chars |
|  | #Total Specification: 5 \* 5 = 25 | |

*[N.B. for the previous testing (especially in “create student” API, we have created more some students which totaled to 14]*

Demo test cases are provided below:

|  |  |  |
| --- | --- | --- |
| **URL** | **Outputs** | **Test** |
| http://127.0.0.1:5000/finished/1/1 | "completed\_courses": 10,  "status": **true** | Pass |
| http://127.0.0.1:5000/finished/1/0 | "error": "Program ID 0 does not exist" | Fail |
| http://127.0.0.1:5000/finished/2/2 | "completed\_courses": 0,  "status": **false** | Pass |
| http://127.0.0.1:5000/finished/2/-1 | "error": "Program ID -1 does not exist" | Fail |
| http://127.0.0.1:5000/finished/2/3 | "error": "Program ID 3 does not exist" | Fail |
| http://127.0.0.1:5000/finished/\*/3 | "error": "Student ID \* is not formatted correctly." | Fail |